Apuntes Matemáticas Discretas

18 de agosto de 2026

Índice general

Capítulo 1

Lógica

1.1. Introducción a la lógica

La lógica es la disciplina que estudia los principios y métodos para distinguir razonamientos correctos de los incorrectos. Es una herramienta fundamental en la ciencia de la computación, ya que nos permite formalizar y analizar rigurosamente la estructura de los argumentos, la validez de las inferencias y la correctitud de los sistemas computacionales. En esta sección, introduciremos algunos conceptos básicos de la lógica y su relevancia en la ciencia de la computación, así como una visión general de losdiferentes tipos de lógica que existen y su aplicabilidad en distintos contextos.

1.1.1. Lógica y ciencia de la computación

En esta sección, estaremos interesados en el estudio de la lógica y sus aplicaciones. Notemos que la lógica es una herramienta fundamental y fundacional en diversas áreas de la ciencia de la computación. Algunos de los campos donde su aplicación es directa y esencial incluyen:

SAT Solvers y Razonamiento Automatizado Los algoritmos de resolución del problema de satisfacibilidad booleana (SAT solvers) utilizan lógica proposicional de manera intensiva para explorar sistemáticamente espacios de búsqueda inmensos y resolver problemas de optimización de gran escala en tiempos prácticos.

Ciberseguridad La lógica permite modelar sistemas complejos y protocolos de comunicación, facilitando la demostración formal rigurosa de propiedades de seguridad, así como la detección algorítmica de vulnerabilidades antes de que sean explotadas.

Razonamiento sobre algoritmos, programas y estructuras de datos Utilizamos lógica formal para razonar matemáticamente sobre el comportamiento temporal y la correctitud de los algoritmos, asegurando que operan exactamente según lo esperado para cualquier entrada válida, sin depender exclusivamente de pruebas empíricas.

Verificación de software La verificación formal emplea herramientas lógicas avanzadas para demostrar matemáticamente que un sistema de software o hardware cumple a cabalidad con sus especificaciones formales, un requisito crítico en sistemas donde los fallos conllevan riesgos vitales.

Bases de datos Los lenguajes de consulta relacionales, como SQL, se fundamentan teóricamente de manera directa en el álgebra relacional y la lógica de primer orden, permitiendo recuperar información de forma declarativa y precisa.

Lenguajes de programación La semántica formal y los sistemas de tipos estáticos de los lenguajes de programación modernos están construidos íntegramente sobre bases lógicas, lo que permite a los compiladores detectar errores lógicos antes de la ejecución del código.

1.1.2. Más allá de la proposición: Tipos de lógica

La *lógica proposicional* (a veces llamada lógica de orden cero) es el sistema formal más básico y directo, donde la unidad indivisible de análisis es la *proposición atómica* —un enunciado que asume estrictamente el valor de verdadero o falso— y el razonamiento se construye exclusivamente mediante la combinación de estas unidades a través de conectivos lógicos.

Si bien este marco constituye el punto de partida de nuestro curso y posee la robustez necesaria para modelar y resolver problemas computacionales fundamentales, el universo analítico es considerablemente más amplio. Cuando los sistemas requieren modelar propiedades estructurales complejas, relaciones dinámicas o flujos continuos, necesitamos lenguajes más expresivos. Otros tipos de lógica que se han desarrollado para abordar estas necesidades incluyen, pero no se limitan a:

- **Lógica de primer orden:** También conocida como lógica de predicados, la *lógica de primer orden* introduce la capacidad de hablar sobre propiedades de los objetos y relaciones entre ellos. Añade los cuantificadores universal ($\forall$, "para todo") y existencial ($\exists$, "existe"), lo que nos permite formular expresiones sobre elementos individuales de un dominio determinado.
- **Lógica de segundo orden:** La *lógica de segundo orden* da un paso más allá de la *lógica de primer orden* al permitirnos cuantificar no solo sobre objetos individuales, sino sobre conjuntos de objetos, relaciones o funciones. Incrementa enormemente el poder expresivo, permitiendo capturar propiedades como la conectividad de un grafo.
- **Lógicas de órdenes superiores:** Siguiendo esta misma jerarquía matemática, las *lógicas de órdenes superiores* nos permiten cuantificar sobre conjuntos de conjuntos, conjuntos de funciones, y así sucesivamente. Su análisis algorítmico se vuelve sustancialmente más complejo.
- **Lógica difusa:** A diferencia de los sistemas donde el valor de verdad es estrictamente 1 o 0, la *lógica difusa* (o *fuzzy logic*) permite grados de verdad continuos en el intervalo $[0, 1]$. Es particularmente útil en sistemas de control que lidian con información imprecisa.

Además de las listadas anteriormente, existen familias de lógicas diseñadas para modelar otros escenarios específicos, como la lógica modal (para razonar sobre posibilidades y necesidades), la lógica temporal (para razonar sobre eventos a lo largo del tiempo), entre otras. En este curso, nos centraremos en la lógica proposicional y en la lógica de primer orden.

1.2. Lógica proposicional

1.2.1. Proposiciones

Definición. Una *proposición* es un enunciado declarativo que sin ambigüedad es verdadero o falso, pero no ambos a la vez.

Como ejemplo podemos recordar las siguientes proposiciones:

- Santiago es la capital de Chile.
- $2 + 2 = 4$.
- Hoy está lloviendo en la ciudad de Concepción.
- $2 + 1 < 3$.

Acá, las primeras dos son verdaderas, mientras que la última es falsa. El valor de verdad de la tercera dependerá del clima actual, pero indudablemente asume exactamente un valor de verdad.

En el estudio formal de la lógica, es crucial distinguir el nivel de complejidad estructural de estos enunciados. Llamaremos *proposición atómica* (o proposición simple) a aquella que constituye la unidad básica e indivisible de información; es decir, un enunciado que no puede descomponerse en proposiciones más pequeñas. Todos los casos presentados en el ejemplo anterior corresponden a proposiciones atómicas.

Por otro lado, las siguientes oraciones no corresponden a proposiciones, ya que no declaran un estado de verdad evaluable:

- ¿Cuál es la capital de Chile?
- Esta oración es falsa.
- Silencio por favor.
- $x + 2 = 4$.
- $x + y$ es divisible por z.

Para formalizar y abstraer el contenido específico de los enunciados en el mundo real, usaremos letras para denotar *variables proposicionales*; es decir, variables abstractas que representan proposiciones atómicas. Convencionalmente en la literatura, utilizaremos letras minúsculas como p, q, r, s.

1.2.2. Operadores lógicos

Dadas proposiciones atómicas, podemos construir proposiciones de mayor complejidad, conocidas como *proposiciones compuestas*, usando conectivos u operadores lógicos.

Definición. Dadas p y q proposiciones, definimos las siguientes proposiciones compuestas:

- *Negación*: $\overline{p}$ es la proposición que es verdadera si y sólo si p es falsa.

- *Conjunción*: $p \wedge q$ es la proposición que es verdadera si y sólo si p y q son ambas verdaderas.
- *Disyunción*: $p \vee q$ es la proposición que es verdadera si y sólo si al menos una entre p y q es verdadera (incluyendo el caso donde ambas lo son).
- *Implicancia*: $p \Rightarrow q$ es la proposición que es falsa si y sólo si p es verdadera y q es falsa.
- *Equivalencia*: $p \Leftrightarrow q$ es la proposición que es verdadera si y sólo si p y q comparten exactamente el mismo valor de verdad.

Podemos construir nuevas proposiciones a partir de otras usando los operadores lógicos. Por ejemplo, si definimos las variables proposicionales p como la proposición atómica "Santiago es la capital de Chile" y q como la proposición atómica "$2+2 = 5$", entonces podemos formar las siguientes *proposiciones compuestas*:

- $\overline{p}$ es la proposición "Santiago no es la capital de Chile", que es falsa.
- $p \wedge q$ es la proposición "Santiago es la capital de Chile y $2 + 2 = 5$", que es falsa.
- $p \vee \overline{q}$ es la proposición "Santiago es la capital de Chile o $2 + 2 \neq 5$", que es verdadera.
- $p \Rightarrow q$ es la proposición "Si Santiago es la capital de Chile, entonces $2 + 2 = 5$", que es falsa.

1.2.3. Argumentos

Definición. Un *argumento* es una secuencia estructurada de proposiciones que termina en una proposición específica llamada *conclusión*. Las proposiciones que preceden lógicamente a la conclusión y sirven de fundamento se llaman *premisas*.

Un ejemplo clásico de argumento estudiado desde la antigüedad es el siguiente:

Premisa 1: Todas las personas son mortales.
Premisa 2: Sócrates es una persona.

Conclusión: Sócrates es mortal.

Diremos de manera informal que un argumento es *válido* si la conclusión es verdadera de manera obligatoria siempre que las premisas sean verdaderas. En el ejemplo anterior, el argumento es lógicamente válido. Notemos que la validez de un argumento bajo ninguna circunstancia depende de la verdad empírica de las premisas en el mundo real, sino exclusivamente de la estructura y relación lógica que une las premisas con la conclusión. Por ejemplo, consideremos el siguiente argumento:

Premisa 1: Los mamíferos no ponen huevos.
Premisa 2: El ornitorrinco es un mamífero.

Conclusión: El ornitorrinco no pone huevos.

A pesar de que empíricamente la primera premisa es falsa (y por consiguiente la conclusión también lo es en el mundo biológico), el argumento posee una estructura lógicamente válida.

Por el contrario, el siguiente argumento no es válido, ya que es perfectamente posible que las premisas sean verdaderas sin que la conclusión lo sea:

Premisa 1: Si llueve, entonces el suelo está mojado.
Premisa 2: El suelo está mojado.

Conclusión: Llueve.

1.3. Formalizando la lógica proposicional

Para analizar los argumentos sin ambigüedades derivadas del lenguaje natural, necesitamos un marco matemático preciso.

1.3.1. Sintaxis

La sintaxis de la lógica proposicional se refiere a las reglas gramaticales estrictas que determinan cómo se pueden construir proposiciones bien formadas a partir de variables proposicionales y operadores lógicos.

Definición. La sintaxis de la lógica proposicional se define inductivamente de la siguiente manera:

- Si p es una variable proposicional, entonces p es una fórmula válida.
- Si α y β son fórmulas válidas, entonces $(\overline{\alpha})$, $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$ y $(\alpha \Rightarrow \beta)$ son también fórmulas válidas.

Para simplificar la notación y evitar el exceso de paréntesis que dificultan la lectura, estableceremos reglas de precedencia estándar para los operadores lógicos. En el curso usaremos el siguiente orden descendente de prioridad: negación, $\wedge$, $\vee$, $\Rightarrow$.

La fórmula $((\overline{p}) \Rightarrow q)$ es una fórmula válida. Usando nuestras reglas de precedencia, podemos escribir esta fórmula directamente sin paréntesis como $\overline{p} \Rightarrow q$. Otros ejemplos de simplificación sintáctica son presentados en la siguiente tabla:

Proposición	Forma abreviada
$((p \vee q) \wedge r)$	$(p \vee q) \wedge r$
$((p \Rightarrow q) \vee (r \Rightarrow s))$	$p \Rightarrow q \vee r \Rightarrow s$
$((p \wedge q) \Rightarrow r)$	$p \wedge q \Rightarrow r$
$((p \vee q) \Rightarrow (r \vee s))$	$p \vee q \Rightarrow r \vee s$

1.3.2. Semántica

La semántica de la lógica proposicional se refiere a las reglas que determinan el valor de verdad (el significado matemático) de una proposición, dada una asignación específica de valores de verdad a las variables proposicionales que la componen. Esto usualmente se formaliza rigurosamente mediante una *función de verdad*.

Definición. Dada una formula α que contiene n variables proposicionales distintas $p_1, p_2, \ldots, p_n$, una *función de verdad* es una asignación $v(\alpha) : \{0,1\}^n \to \{0,1\}$ que asocia un valor booleano a cada variable proposicional.

El valor de verdad de una fórmula válida depende únicamente de los valores de verdad de sus variables proposicionales base. Dada la fórmula compuesta $p \wedge q$, su *función de verdad* viene dada exhaustivamente por la siguiente tabla:

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

Las representaciones tabulares anteriores se conocen comúnmente como *tablas de verdad*. Si una proposición evalúa a 1 para todas las combinaciones posibles en su *tabla de verdad*, la denominamos *tautología*. De manera análoga, las *funciones de verdad* de los demás operadores lógicos elementales son dadas por las siguientes tablas:

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

p	q	$p \Rightarrow q$
0	0	1
0	1	1
1	0	0
1	1	1

p	$\overline{p}$
0	1
1	0

Naturalmente podemos construir tablas de verdad para fórmulas compuestas más complejas, como por ejemplo $p \wedge (q \vee r)$ o $(p \Rightarrow q) \vee (r \Rightarrow s)$, evaluando sistemáticamente cada subfórmula y combinando sus resultados según las reglas de los operadores lógicos.

Otra forma de hacer esto es mediante la evaluación aritmética de las fórmulas compuestas, utilizando las siguientes reglas inductivas:

Definición. Sean α y β fórmulas válidas. El valor de verdad de las fórmulas compuestas se define inductivamente de la siguiente manera:

- $v(\overline{\alpha}) = 1 - v(\alpha)$
- $v(\alpha \wedge \beta) = \text{mín}\{v(\alpha), v(\beta)\}$
- $v(\alpha \vee \beta) = \text{máx}\{v(\alpha), v(\beta)\}$
- $v(\alpha \Rightarrow \beta) = \text{máx}\{1 - v(\alpha), v(\beta)\}$

Estas reglas de evaluación son completamente equivalentes a las tablas de verdad presentadas anteriormente. Por ejemplo, para evaluar $v((p \wedge q) \Rightarrow r)$, primero evaluamos $v(p \wedge q) = \text{mín}\{v(p), v(q)\}$ y luego sustituimos este valor en la evaluación de la implicancia, obteniendo $v((p \wedge q) \Rightarrow r) = \text{máx}\{1 - \text{mín}\{v(p), v(q)\}, v(r)\}$. Para que esta fórmula compuesta evalúe a 0, es necesario que $v(p) = 1$, $v(q) = 1$ y $v(r) = 0$. En cualquier otro caso, la fórmula compuesta evaluará a 1. Obtendríamos la siguiente tabla de verdad para esta fórmula compuesta:

p	q	r	$(p \wedge q) \Rightarrow r$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Mientras que al evaluar por tablas de verdad, obtenemos exactamente el mismo resultado:

p	q	r	$(p \wedge q)$	$(p \wedge q) \Rightarrow r$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1

1.3.3. Formalización de la validez de un argumento

Habiendo establecido la sintaxis y semántica de la lógica proposicional, podemos ahora formalizar matemáticamente el concepto introducido en la sección anterior sobre cuándo un argumento es válido.

Definiremos esto vía dos máneras equivalentes. La primera se basa en la idea de *consecuencia lógica*

Definición. Una conclusión C es una *consecuencia lógica* de un conjunto de premisas $P_1, P_2, \ldots, P_k$ si para toda función de verdad v que asigna 1 a todas las premisas simultáneamente ($v(P_1) = 1, v(P_2) = 1, \ldots, v(P_k) = 1$), se tiene que obligatoriamente $v(C) = 1$. Denotamos esto como $\{P_1, P_2, \ldots, P_k\} \models C$.

Similarmente, podemos definir la validez de un argumento a través del concepto de *tautología*.

Definición. Una *tautología* es una fórmula proposicional que evalúa a 1 bajo cualquier *valor de verdad* asignado a sus variables proposicionales.

Un argumento será válido si la conclusión es una consecuencia lógica de las premisas o si la implicación que conecta secuencialmente las premisas con la conclusión es una tautología.

Definición. Un argumento compuesto por un conjunto de *premisas* $P_1, P_2, \ldots, P_k$ y una *conclusión* C es *formalmente válido* si $\{P_1, P_2, \ldots, P_k\} \models C$ o, de manera equivalente, si la fórmula proposicional $(P_1 \wedge P_2 \wedge \ldots \wedge P_k) \Rightarrow C$ es una tautología.

En efecto, ambas definiciones son lógicamente equivalentes pues la implicancia sólo evalúa a 0 cuando el lado izquierdo (las premisas asumidas como ciertas) es 1 y el lado derecho (la conclusión derivada) es 0. Por lo tanto, si existe alguna función de verdad v que asigna 1 a todas las *premisas* pero asigna 0 a la *conclusión*, entonces la fórmula $(P_1 \wedge P_2 \wedge \ldots \wedge P_k) \Rightarrow C$ no sería una tautología, ya que bajo esa función de verdad específica evaluaría a 0.

A continuación, analizaremos algunos esquemas de argumentación clásicos utilizando esta definición formal, demostrando su validez tanto por evaluación analítica como mediante *tablas de verdad*.

Ejemplo. Analicemos la validez formal del siguiente argumento:

Premisa 1: El mayordomo y el cocinero no son ambos inocentes.
Premisa 2: O el mayordomo está mintiendo o el cocinero es inocente.

Conclusión: El mayordomo miente o es inocente.

Para formalizar este argumento, asignemos las siguientes variables proposicionales:

- M_i: El mayordomo es inocente.
- C_i: El cocinero es inocente.
- M_m: El mayordomo está mintiendo.

Luego, el argumento tiene la siguiente estructura formal:

Premisa 1: $\overline{(M_i \wedge C_i)}$
Premisa 2: $M_m \vee C_i$

Conclusión: $M_m \vee M_i$

Equivalentemente podemos escribirlo empleando la notación de consecuencia lógica:

$$\{\overline{(M_i \wedge C_i)}, M_m \vee C_i\} \models M_m \vee M_i$$

o como una tautología:

$$((\overline{(M_i \wedge C_i)}) \wedge (M_m \vee C_i)) \Rightarrow (M_m \vee M_i)$$

Comprobemos que este argumento es formalmente válido vía tabla de verdad:

M_i	C_i	M_m	$\overline{(M_i \wedge C_i)}$	$M_m \vee C_i$	$M_m \vee M_i$
0	0	0	1	0	0
0	0	1	1	1	1
0	1	0	1	1	0
0	1	1	1	1	1
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	0	1	1

La tercera fila de la tabla de verdad ($M_i = 0, C_i = 1, M_m = 0$) muestra un escenario posible donde ambas premisas se cumplen (evalúan a 1), pero la conclusión es falsa (evalúa a 0). Por lo tanto, el argumento es inválido.

Ejemplo. Veamos ahora un argumento clásico que sí es válido: el clásico *Modus Ponens*. Es una de las reglas de inferencia deductiva más elementales y fundamentales en la lógica matemática. Su estructura formal es la siguiente:

$$\begin{array}{l} \text{Premisa 1: } p \Rightarrow q \\ \text{Premisa 2: } p \\ \hline \text{Conclusión: } q \end{array}$$

Verifiquemos evaluando exhaustivamente todas las asignaciones posibles mediante una *tabla de verdad*:

p	q	$p \Rightarrow q$	p	q
0	0	1	0	0
0	1	1	0	1
1	0	0	1	0
1	1	1	1	1

Como podemos observar en la *tabla de verdad*, la única fila donde todas las *premisas* son simultáneamente verdaderas es la última. En ese escenario particular, notamos que la *conclusión* q también evalúa a 1. Por tanto el argumento es válido.

1.4. Deducción natural y reglas de inferencia

Hasta ahora, nuestra herramienta principal para analizar la validez de los argumentos ha sido la construcción de *tablas de verdad*. Si bien esta técnica es exhaustiva y garantiza una evaluación precisa, se vuelve rápidamente impráctica a medida que el número de variables proposicionales aumenta. Por ejemplo, con sólo 5 variables proposicionales, ya tendríamos que evaluar $2^5 = 32$ filas en la tabla de verdad.

Para superar esto, hablaremos del sistema de *deducción natural*, el cual nos permite construir demostraciones formales de manera puramente sintáctica y estructurada. La idea central emula el proceso de razonamiento deductivo que aplicamos intuitivamente en matemáticas. En este sistema, partimos de un conjunto de *premisas* y aplicamos mecánicamente una serie de *reglas de inferencia* para derivar nuevas fórmulas, construyendo un camino paso a paso hasta alcanzar la *conclusión* deseada.

> **Definición** (Deducción natural)**.** La *deducción natural* es un sistema formal basado en reglas sintácticas que mantiene una lista de fórmulas demostradas. Comenzamos con las premisas iniciales en la lista. En cada paso de la demostración, aplicamos una regla de inferencia sobre fórmulas ya establecidas en la lista para generar una nueva fórmula que se agrega a la lista. La demostración concluye cuando la conclusión aparece en las formulas demostradas.

Usaremos el símbolo $\vdash$ para denotar que una fórmula se ha demostrado vía deducción natural a partir de un conjunto de fórmulas iniciales. Es decir, $\{\alpha_1, \alpha_2, \ldots, \alpha_k\} \vdash \beta$ significa estructuralmente que β se sigue lógicamente de $\alpha_1, \alpha_2, \ldots, \alpha_k$ manipulando los símbolos bajo las reglas de inferencia permitidas.[1]

[1] De forma más general, la notación $\Gamma \vdash \beta$ se suele usar para indicar que β es un teorema derivable en el sistema de demostración a partir Γ.

1.4.1. Reglas de inferencia básicas

Conjunción De manera intuitiva, imagina que empíricamente constatamos que "Hoy es jueves" (α) y, por otro lado, sabemos que "Está soleado" (β). Resulta completamente natural deducir la frase conjunta "Hoy es jueves y está soleado" ($\alpha \wedge \beta$). Al revés, si nuestro punto de partida es la frase compuesta "Hoy es jueves y está soleado", es lógico poder extraer cualquiera de los dos hechos aislados como una verdad por sí misma.

> **Inferencia** ($\wedge$-introducción)**.** Sean α y β fórmulas proposicionales. Si hemos demostrado α y β por separado, entonces podemos concluir $\alpha \wedge \beta$.

Formalmente esto se representa como $\{\alpha, \beta\} \vdash \alpha \wedge \beta$, o

$$\frac{\begin{array}{c}\alpha \\ \beta\end{array}}{\alpha \wedge \beta}$$

> **Inferencia** ($\wedge$-eliminación)**.** Si hemos demostrado $\alpha \wedge \beta$, entonces podemos concluir α y también podemos concluir β.

Para representar esta regla de forma compacta escribimos: $\{\alpha \wedge \beta\} \vdash \alpha$ y $\{\alpha \wedge \beta\} \vdash \beta$. De forma gráfica, la eliminación de la conjunción se ve como

$$\frac{\alpha \wedge \beta}{\alpha} \qquad \frac{\alpha \wedge \beta}{\beta}$$

Negación Respecto a la manipulación de polaridad lógica, tenemos lo siguiente:

> **Inferencia** (Doble negación)**.** Si hemos demostrado α, entonces podemos concluir $\overline{\overline{\alpha}}$. De manera complementaria, si hemos demostrado $\overline{\overline{\alpha}}$, podemos desarmar la estructura para concluir directamente α.

En notación formal utilizamos: $\{\alpha\} \vdash \overline{\overline{\alpha}}$ y $\{\overline{\overline{\alpha}}\} \vdash \alpha$, mientras que gráficamente se representa como

$$\frac{\alpha}{\overline{\overline{\alpha}}} \qquad \frac{\overline{\overline{\alpha}}}{\alpha}$$

Ejemplo. Usemos las reglas que conocemos hasta ahora para demostrar formalmente que el siguiente argumento es válido:

Premisa 1: El perro ladra.
Premisa 2: El gato maulla.

Conclusión: Es falso que el gato no maulla y es falso que el perro no ladra.

Asignemos las siguientes variables proposicionales para limpiar el lenguaje natural:

- P: El perro ladra.
- G: El gato maulla.

Luego, el argumento cobra la siguiente estructura formal:

$$\frac{\begin{array}{l}\text{Premisa 1: } P \\ \text{Premisa 2: } G\end{array}}{\text{Conclusión: } \overline{\overline{G}} \wedge \overline{\overline{P}}}$$

Demostremos la validez del argumento utilizando deducción natural. Acumularemos las fórmulas enumeradas paso a paso, justificando la operación empleada. Una vez que construyamos sintácticamente la conclusión, habremos presentado un argumento legítimo, sin necesidad de tablas de verdad.

1. P (Premisa 1)
2. G (Premisa 2)
3. $\overline{\overline{P}}$ (Doble negación, aplicada a la línea 1)
4. $\overline{\overline{G}}$ (Doble negación, aplicada a la línea 2)
5. $\overline{\overline{G}} \wedge \overline{\overline{P}}$ ($\wedge$-introducción, aplicada combinando las líneas 4 y 3)

Nótese que en el paso 5 fuimos cuidadosos de invertir el orden de los elementos según dictaba la conclusión, construyendo primero el lado de G y luego el de P.

Implicancia Para comprender la intuición detrás de la implicancia, pensemos en la introducción como el acto de razonar hipotéticamente. Si propongo el experimento mental "Supongamos que me gano la lotería" (α), y logro deducir "Me compraré un auto" (β), he demostrado un hecho incondicional "Si me gano la lotería, entonces me compraré un auto" ($\alpha \Rightarrow \beta$). El Modus Ponens o eliminación, en cambio, es la ejecución de la garantía: sé empíricamente que "Si gano, compraré el auto", y si el hecho "Gané la lotería" se concreta, concluyo que "Compraré el auto".

> **Inferencia** ($\Rightarrow$-eliminación o Modus Ponens)**.** Si hemos demostrado $\alpha \Rightarrow \beta$ y también hemos demostrado la condición α, entonces podemos derivar el consecuente β.

Formalmente, la eliminación de la implicancia se representa como: $\{\alpha \Rightarrow \beta, \alpha\} \vdash \beta$. De manera gráfica:

$$\frac{\begin{array}{l}\alpha \Rightarrow \beta \\ \alpha\end{array}}{\beta}.$$

La regla de introducción de la implicancia es más sútil pues requiere del concepto de *descarga de premisas*. La *descarga de premisas* nos permite hacer un supuesto temporal en el proceso de demostración, lo cual es fundamental para poder introducir la implicancia.

De manera intuitiva, hacer un supuesto en deducción natural es como abrir una subdemostración. Durante ese bloque identado, asumimos que una fórmula α es verdadera a modo de hipótesis de trabajo. Todo lo que deduzcamos allí vive estrictamente condicionado a α. Sin embargo, esta subdemostración no queda abierta indefinidamente y hay que cerrarlo cuando concluimos β obteniendo

la fórmula incondicional $\alpha \Rightarrow \beta$. Al cerrar el bloque, el supuesto α queda formalmente descargado, es decir, ya no es una hipótesis activa para el resto de la demostración.

Formalmente, tenemos la regla de introducción de la implicancia.

> **Inferencia** ($\Rightarrow$-introducción)**.** Si, asumiendo una hipótesis temporal α, logramos demostrar sintácticamente β, entonces podemos descargar dicha hipótesis para concluir la fórmula incondicional $\alpha \Rightarrow \beta$.

Formalmente, tenemos que si $\{\alpha \vdash \beta\} \vdash \alpha \Rightarrow \beta$. Graficamente, la introducción de la implicancia se ve como

$$\begin{array}{ll} \alpha & \text{(supuesto temporal)} \\ \vdots & \\ \beta & \text{(conclusión temporal)} \\ \hline \alpha \Rightarrow \beta & \end{array}$$

Ejemplo. Veamos la validez de la transitividad de la implicancia, es decir, que si $\alpha \Rightarrow \beta$ y $\beta \Rightarrow \gamma$, entonces $\alpha \Rightarrow \gamma$. Formalmente, queremos demostrar que $\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \vdash \alpha \Rightarrow \gamma$.

1. $\alpha \Rightarrow \beta$ (Premisa)
2. $\beta \Rightarrow \gamma$ (Premisa)
    3. α (Supuesto temporal para preparar la $\Rightarrow$-introducción)
    4. β ($\Rightarrow$-eliminación, utilizando 1 y 3)
    5. γ ($\Rightarrow$-eliminación, utilizando 2 y 4)
6. $\alpha \Rightarrow \gamma$ ($\Rightarrow$-introducción, utilizando el bloque 3-5)

Al salir de la indentación en el paso 6, el supuesto α queda formalmente descargado e inactivo para el resto de la demostración (si es que hubiera).

Disyunción La introducción de la disyunción nos permite combinar afirmaciones demostradas con algunas sin demostrar. Por ejemplo si "Tengo un perro" es una afirmación demostrada, entonces añadir alternativas mediante la frase "Tengo un perro o tengo un dragón" genera una proposición demostrada.

La eliminación de la disyunción codifica el razonamiento por casos. Supongamos que sé de antemano que "El ladrón escapó por la puerta o por la ventana". Examino el caso de la puerta (Supuesto 1) y concluyo lógicamente que "Dejó huellas de barro". Luego, descargo ese caso, examino el caso de la ventana (Supuesto 2) y derivo también que "Dejó huellas de barro". Como agotamos todas las opciones de laa disyunción y ambas rutas convergen a lo mismo, podemos concluir incondicionalmente que "El ladrón dejó huellas de barro".

> **Inferencia** ($\vee$-introducción)**.** Si hemos demostrado α, entonces podemos concluir la disyunción $\alpha \vee \beta$ adjuntando cualquier fórmula arbitraria β. De manera simétrica, si hemos demostrado β, podemos concluir $\alpha \vee \beta$.

La introducción de la disyunción se anota: $\{\alpha\} \vdash \alpha \vee \beta$ y $\{\beta\} \vdash \alpha \vee \beta$. De manera gráfica, tenemos:

$$\frac{\alpha}{\alpha \vee \beta} \qquad \frac{\beta}{\alpha \vee \beta}.$$

La $\vee$-eliminación es un poco más compleja, pues nuevamente requiere la descarga de premisas.

Inferencia ($\vee$-eliminación)**.** Si hemos demostrado $\alpha \vee \beta$, y mediante subdemostraciones independientes probamos que una conclusión objetivo γ se sigue tanto si asumimos temporalmente α como si asumimos temporalmente β, entonces podemos concluir γ.

La eliminación de disyunción es entonces $\{\alpha \vee \beta, \alpha \vdash \gamma, \beta \vdash \gamma\} \vdash \gamma$. De manera gráfica, se representa como

$$\begin{array}{l} \alpha \vee \beta \\ \quad \alpha \qquad \text{(supuesto temporal)} \\ \quad \vdots \\ \quad \gamma \qquad \text{(conclusión temporal)} \\ \quad \beta \qquad \text{(supuesto temporal)} \\ \quad \vdots \\ \quad \gamma \qquad \text{(conclusión temporal)} \\ \hline \gamma \end{array}.$$

Ejemplo. Demostremos una familiar desde la lógica $\{p \wedge (q \vee r)\} \vdash (p \wedge q) \vee (p \wedge r)$ (la propiedad distributiva).

1. $p \wedge (q \vee r)$ (Premisa principal)
2. p ($\wedge$-eliminación aplicada a 1)
3. $q \vee r$ ($\wedge$-eliminación aplicada a 1)
    4. q (Supuesto temporal: Caso 1 de la disyunción)
    5. $p \wedge q$ ($\wedge$-introducción con 2 y 4)
    6. $(p \wedge q) \vee (p \wedge r)$ ($\vee$-introducción en 5)
    7. r (Supuesto temporal: Caso 2 de la disyunción)
    8. $p \wedge r$ ($\wedge$-introducción combinando 2 y 7)
    9. $(p \wedge q) \vee (p \wedge r)$ ($\vee$-introducción en 8)
10. $(p \wedge q) \vee (p \wedge r)$ ($\vee$-eliminación, utilizando los bloques 4-6 y 7-9)

1.4.2. Reglas de inferencia adicionales

Hay varias reglas de inferencia adicionales.

Inferencia (Silogismo hipotético)**.** Si hemos demostrado $\alpha \Rightarrow \beta$ y también hemos demostrado $\beta \Rightarrow \gamma$, entonces podemos concluir $\alpha \Rightarrow \gamma$.

Formalmente, el silogismo hipotético se representa como: $\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \vdash \alpha \Rightarrow \gamma$

$$\frac{\begin{array}{c}\alpha \Rightarrow \beta \\ \beta \Rightarrow \gamma\end{array}}{\alpha \Rightarrow \gamma}$$

> **Inferencia** (Silogismo disyuntivo)**.** Si hemos demostrado $\alpha \vee \beta$, y también hemos demostrado $\overline{\alpha}$, entonces podemos concluir β.

Formalmente, el silogismo disyuntivo se representa como: $\{\alpha \vee \beta, \overline{\alpha}\} \vdash \beta$

$$\frac{\begin{array}{c}\alpha \vee \beta \\ \overline{\alpha}\end{array}}{\beta}$$

> **Inferencia** (Resolución)**.** Si hemos demostrado $\alpha \vee \beta$ y también hemos demostrado $\overline{\alpha} \vee \gamma$, entonces podemos concluir $\beta \vee \gamma$.

Formalmente, la resolución se representa como: $\{\alpha \vee \beta, \overline{\alpha} \vee \gamma\} \vdash \beta \vee \gamma$

$$\frac{\begin{array}{c}\alpha \vee \beta \\ \overline{\alpha} \vee \gamma\end{array}}{\beta \vee \gamma}$$

En particular resolución es una regla de inferencia muy importante en lógica computacional, pues es la base de muchos algoritmos de demostración automática de teoremas y sistemas de razonamiento basado en reglas.[2]

1.4.3. Contradicciones

Contradicción y el principio de explosión Nuevas reglas de inferencia vienen del concepto de contradicción.

> **Definición.** Una fórmula proposicional α es una *contradicción* si evalúa a 0 bajo cualquier función de verdad que asigna valores a sus variables proposicionales.

Por ejemplo $p \wedge \overline{p}$ es una contradicción, pues no importa si p es 0 o 1, la fórmula siempre evalúa a 0. Anotaremos la contradicción con el símbolo $\bot$ y las tauutologías con el símbolo $\top$. Asociadas a las contradicciones tenemos tres reglas de inferencia adicionales.

La primera regla de inferencia es la *$\bot$-introducción*, intuitivamente esta regla nos que si hemos derivado una formula y su negación, entonces hemos derivado una contradicción.

> **Inferencia** ($\bot$-introducción)**.** Si hemos demostrado $\alpha \wedge \overline{\alpha}$, entonces podemos concluir $\bot$.

[2] `prolog` es un ejemplo de tal sistema.

La segunda regla de inferencia es el $\bot$-*eliminación* o *principio de explosión*, esta regla nos dice que si hemos demostrado una contradicción, entonces podemos concluir cualquier fórmula.

> **Inferencia** ($\bot$-eliminación o principio de explosión)**.** Si hemos demostrado $\bot$, entonces podemos concluir cualquier fórmula α.

La última regla de inferencia es lo que hacemos habitualmente para demostraciones *por contradicción*. Esta regla nos dice que si al suponer α llegamos a una contradicción, entonces podemos concluir $\overline{\alpha}$.

> **Inferencia** (Reducción al absurdo)**.** Si al suponer α llegamos a una contradicción, entonces podemos concluir $\overline{\alpha}$.

Estas tres reglas de inferencia se denotan como $\{\alpha \wedge \overline{\alpha}\} \vdash \bot$, $\{\bot\} \vdash \alpha$ y $\{\{\alpha\} \vdash \bot\} \vdash \overline{\alpha}$. Gráficamente estas reglas de inferencia se representan de la siguiente manera:

$$\frac{\alpha \wedge \overline{\alpha}}{\bot} \qquad \frac{\bot}{\alpha} \qquad \frac{\begin{array}{c}\alpha \\ \vdots \\ \bot\end{array}}{\overline{\alpha}}$$

Ejemplo. Probemos el silogismo disyuntivo $\{\alpha \vee \beta, \overline{\alpha}\} \vdash \beta$ utilizando el principio de explosión.

1. $\alpha \vee \beta$ (Premisa)
2. $\overline{\alpha}$ (Premisa)
    3. α (Supuesto temporal: Caso 1 de la disyunción)
    4. $\alpha \wedge \overline{\alpha}$ ($\wedge$-introducción con 3 y 2)
    5. $\bot$ ($\bot$-introducción, en 4)
    6. β (Principio de explosión, en 5)
    7. β (Supuesto temporal: Caso 2 de la disyunción)
8. β ($\vee$-eliminación, utilizando los bloques 3-6 y 7)

1.4.4. Corrección y completitud del sistema

Hay una pregunta natural con sistemas de demostración como la deducción natural ¿cómo se relaciona nuestra capacidad de derivar símbolos (denotada por $\vdash$) con la semantica (denotada por $\models$)? Esto tiene que ver con la corección y completitud de la deducción natural para la lógica proposicional.

La *correctitud* de un sistema de demostración establece que el sistema no prueba nada que no sea cierto en términos semánticos. Esto es verdad para el sistema de deducción natural en la lógica proposicional, como menciona el siguiente teorema.

Teorema (Correctitud)**.** Si $\Gamma \vdash \phi$, entonces $\Gamma \models \phi$.

Podemos demostrar este hecho de manera inductiva sobre la estructura de las demostraciones en deducción natural. Para esto basta con ver que las reglas de inferencia que hemos introducido son todas semánticamente válidas.[3]

La *completitud* por otra parte es una propiedad bastante más compleja. Un sistema de demostración es completo si es capaz de probar todas las fórmulas que son semánticamente verdaderas. En el caso de la deducción natural para la lógica proposicional, también se cumple esta propiedad.

Teorema (Completitud)**.** Si $\Gamma \models \phi$, entonces $\Gamma \vdash \phi$.

La demostración es más técnica, pero también es basada en una construcción inductiva.[4]

1.4.5. Equivalencias

Recordemos la noción de equivalencia entre fórmulas proposicionales.

Definición. Dos fórmulas proposicionales α y β son *equivalentes* si para cualquier función de verdad que asigna valores a sus variables proposicionales, α y β evalúan al mismo valor de verdad.

De manera equivalente, α y β son equivalentes si $\alpha \Leftrightarrow \beta$ es una tautología o si $\alpha \models \beta$ y $\beta \models \alpha$. Recordemos algunas equivalencias útiles:

Proposición (Equivalencias clásicas)**.** Para cualquier fórmula proposicional α, β y γ se cumplen las siguientes equivalencias.

- $\alpha \wedge \top \equiv \alpha$ (Identidad del $\wedge$)
- $\alpha \vee \bot \equiv \alpha$ (Identidad del $\vee$)
- $\alpha \wedge \bot \equiv \bot$ (Anulación del $\wedge$)
- $\alpha \vee \top \equiv \top$ (Anulación del $\vee$)
- $\alpha \wedge \beta \equiv \beta \wedge \alpha$ (Conmutatividad del $\wedge$)
- $\alpha \vee \beta \equiv \beta \vee \alpha$ (Conmutatividad del $\vee$)
- $(\alpha \wedge \beta) \wedge \gamma \equiv \alpha \wedge (\beta \wedge \gamma)$ (Asociatividad del $\wedge$)
- $(\alpha \vee \beta) \vee \gamma \equiv \alpha \vee (\beta \vee \gamma)$ (Asociatividad del $\vee$)
- $\alpha \wedge (\beta \vee \gamma) \equiv (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$ (Distributividad del $\wedge$ sobre el $\vee$)
- $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ (Distributividad del $\vee$ sobre el $\wedge$)

[3] Por ejemplo, vimos que modus ponens era válida hace un par de páginas.
[4] Puede ser interesante googlear la demostración de Kálmar.

- $\overline{(\alpha \wedge \beta)} \equiv \overline{\alpha} \vee \overline{\beta}$ (De Morgan)
- $\overline{(\alpha \vee \beta)} \equiv \overline{\alpha} \wedge \overline{\beta}$ (De Morgan)
- $\alpha \Rightarrow \beta \equiv \overline{\alpha} \vee \beta$ (Caracterización de la implicancia)

Notemos que cada equivalencia nos da dos reglas de inferencia adicionales, una para cada dirección de la equivalencia. Por ejemplo, la equivalencia de De Morgan $\overline{(\alpha \wedge \beta)} \equiv \overline{\alpha} \vee \overline{\beta}$ nos da las siguientes reglas de inferencia:

Inferencia. Si hemos demostrado $\overline{(\alpha \wedge \beta)}$, entonces podemos concluir $\overline{\alpha} \vee \overline{\beta}$.

Inferencia. Si hemos demostrado $\overline{\alpha} \vee \overline{\beta}$, entonces podemos concluir $\overline{(\alpha \wedge \beta)}$.

1.5. Satisfacibilidad

Una de las propiedades más interesantes de las fórmulas proposicionales es la de satisfacibilidad.

Definición. Una fórmula proposicional α es *satisfacible* si existe una función de verdad que asigna valores a sus variables proposicionales y hace que α evalúe a 1.

En otras palabras, una fórmula es satisfacible si no es una contradicción.

Ejemplo. La siguiente fórmula es satisfacible:

$$(p \vee q) \wedge (\overline{p} \vee r) \wedge (\overline{q} \vee \overline{r}).$$

Para ver esto, basta con asignar $p = 0$, $q = 1$ y $r = 0$, con lo que la fórmula evalúa a 1.

Por otro lado, la siguiente fórmula no es satisfacible:

$$(p \vee q) \wedge (\overline{p} \vee r) \wedge (\overline{q} \vee \overline{r}) \wedge (\overline{p} \vee \overline{q} \vee r).$$

Lo que podemos ver via su tabla de verdad

p	q	r	Fórmula
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

La satisfacibilidad es una de las propiedades más importantes de las fórmulas proposicionales. Por ejemplo, la satisfacibilidad es uno de los problemas centrales de la teoría de la complejidad

computacional, pues es un problema canónico que creemos es difícil de resolver, pero fácil de certificar.

A pesar de lo anterior, en la actualidad existen SAT-solvers bastante eficientes que pueden resolver problemas de satisfacibilidad con bastantes variables proposicionales de manera razonable. Dichos programas utilizan una variedad de técnicas para determinar si una fórmula es satisfacible o no, reportando la asignación de variables proposicionales que hace que la fórmula sea satisfacible en caso de que lo sea. Esto es bastante útil para aplicaciones, pues la flexibilidad de la lógica proposicional hace que podamos modelar una gran variedad de problemas en términos de fórmulas proposicionales, y luego resolverlos utilizando SAT-solvers.

Ejemplo. Supongamos que tenemos que programar tres examenes en dos franjas horarias: mañana y tarde. Tenemos tres ramos: lógica, computación y matemáticas. Tenemos las siguientes restricciones:

- Cada examen debe ser programado en exactamente una franja horaria.
- Los examenes de lógica y computación no pueden ser programados en la misma franja horaria.
- El examen de matemáticas debe ir antes que el de computación.

Modelemos este problema via la lógica proposicional. Consideremos las siguientes variables proposicionales:

- L_m: El examen de lógica es programado en la mañana.
- L_t: El examen de lógica es programado en la tarde.
- C_m: El examen de computación es programado en la mañana.
- C_t: El examen de computación es programado en la tarde.
- M_m: El examen de matemáticas es programado en la mañana.
- M_t: El examen de matemáticas es programado en la tarde.

La gran gracia de modelar via lógica proposicional es que podemos modelar cada restricción por separado, y luego combinarlas via $\wedge$ para obtener una fórmula proposicional que modela el problema completo. Luego, podemos traducir las restricciones a fórmulas proposicionales de la siguiente manera:

- Cada examen debe ser programado en exactamente una franja horaria:
  - $L_m \vee L_t$ (El examen de lógica debe ser programado en al menos una franja horaria)
  - $\overline{(L_m \wedge L_t)}$ (El examen de lógica no puede ser programado en ambas franjas horarias)
  - $C_m \vee C_t$ (El examen de computación debe ser programado en al menos una franja horaria)
  - $\overline{(C_m \wedge C_t)}$ (El examen de computación no puede ser programado en ambas franjas horarias)
  - $M_m \vee M_t$ (El examen de matemáticas debe ser programado en al menos una franja horaria)

  - $\overline{(M_m \wedge M_t)}$ (El examen de matemáticas no puede ser programado en ambas franjas horarias)

- Los examenes de lógica y computación no pueden ser programados en la misma franja horaria:
  - $\overline{(L_m \wedge C_m)}$ (Los examenes de lógica y computación no pueden ser programados en la mañana)
  - $\overline{(L_t \wedge C_t)}$ (Los examenes de lógica y computación no pueden ser programados en la tarde)
- El examen de matemáticas debe ir antes que el de computación: $C_t \Rightarrow M_m$ (Si el examen de computación es programado en la tarde, entonces el examen de matemáticas debe ser programado en la mañana)

Podemos combinar todas estas fórmulas proposicionales utilizando $\wedge$ para obtener una fórmula proposicional que modela el problema completo:

$$\phi = (L_m \vee L_t) \wedge \overline{(L_m \wedge L_t)} \wedge (C_m \vee C_t) \wedge \overline{(C_m \wedge C_t)} \wedge (M_m \vee M_t) \wedge \overline{(M_m \wedge M_t)} \wedge \overline{(L_m \wedge C_m)} \wedge \overline{(L_t \wedge C_t)} \wedge (C_t \Rightarrow M_m).$$

Si usaramos ϕ como entrada de un SAT-solver, este nos diría que ϕ es satisfacible, y nos daría la siguiente asignación de variables proposicionales que hace que ϕ sea satisfacible:

- $L_m = 1$ (El examen de lógica es programado en la mañana)
- $L_t = 0$ (El examen de lógica no es programado en la tarde)
- $C_m = 0$ (El examen de computación no es programado en la mañana)
- $C_t = 1$ (El examen de computación es programado en la tarde)
- $M_m = 1$ (El examen de matemáticas es programado en la mañana)
- $M_t = 0$ (El examen de matemáticas no es programado en la tarde)

Ejemplo. Veamos un segundo ejemplo. Supongamos que tenemos un conjunto de n estaciones de transmisión $E = \{e_1, e_2, \ldots, e_n\}$. Cada estación de transmisión puede transmitir en una de tres frecuencias: baja, media y alta. Queremos asignar una frecuencia a cada estación de transmisión de tal manera que no haya interferencia entre estaciones de transmisión cercanas. La interferencia ocurre cuando dos estaciones de transmisión a menos de 10 km de distancia transmiten en la misma frecuencia. Supongamos que conocemos los pares de estación cercanos dados como una lista C. ¿Es posible asignar frecuencias a las estaciones de transmisión de tal manera que no haya interferencia entre estaciones cercanas?

Introduzcamos las siguientes variables proposicionales:

- $e_{i,b}$: La estación de transmisión e_i transmite en la frecuencia baja.
- $e_{i,m}$: La estación de transmisión e_i transmite en la frecuencia media.
- $e_{i,a}$: La estación de transmisión e_i transmite en la frecuencia alta.

La primera restricción a considerar es que cada estación de transmisión debe transmitir en exactamente una frecuencia:

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Figura 1.1: Un puzzle de Sudoku a la izquierda, y su solución a la derecha.

- $e_{i,b} \vee e_{i,m} \vee e_{i,a}$ (La estación de transmisión e_i debe transmitir en al menos una frecuencia)
- $\overline{(e_{i,b} \wedge e_{i,m})} \wedge \overline{(e_{i,b} \wedge e_{i,a})} \wedge \overline{(e_{i,m} \wedge e_{i,a})}$ (La estación de transmisión e_i no puede transmitir en más de una frecuencia)

La segunda restricción es que no haya interferencia entre estaciones cercanas:

- Para cada par de estaciones cercanas $(e_i, e_j) \in C$, debemos tener la restricción:

$$\overline{(e_{i,b} \wedge e_{j,b})} \wedge \overline{(e_{i,m} \wedge e_{j,m})} \wedge \overline{(e_{i,a} \wedge e_{j,a})}$$

Podemos combinar todas estas fórmulas proposicionales utilizando $\wedge$ para obtener una fórmula proposicional que modela el problema completo. Luego, podemos usar un SAT-solver para determinar si esta fórmula proposicional es satisfacible o no, lo que nos dirá si es posible asignar frecuencias a las estaciones de transmisión de tal manera que no haya interferencia entre estaciones cercanas.

Ejemplo. Otra cosa que podemos hacer via SAT-solvers es diseñar un solver de Sudoku. El *Sudoku* es un juego clásico que dada una cuadrícula de 9×9 con algunos números del 1 al 9 ya escritos, buscamos llenar el resto de la cuadrícula con números del 1 al 9 de tal manera que cada fila, cada columna y cada subcuadrícula de 3×3 contenga todos los números del 1 al 9 sin repetir ninguno.

Para diseñar un solver de Sudoku utilizando un SAT-solver, podemos modelar el problema del Sudoku como un problema de satisfacibilidad. Para esto, introducimos las siguientes variables proposicionales:

- $x_{i,j,k}$: La celda en la fila i, columna j contiene el número k.

Luego, podemos traducir las reglas del Sudoku a fórmulas proposicionales de la siguiente manera:

- Cada celda debe contener al menos un número del 1 al 9:
  - $x_{i,j,1} \vee x_{i,j,2} \vee \cdots \vee x_{i,j,9}$ para cada i y j.

- Cada celda no puede contener más de un número del 1 al 9:
  - $\overline{(x_{i,j,k} \wedge x_{i,j,l})}$ para $k \neq l$ y cada i y j.
- Cada número del 1 al 9 debe aparecer exactamente una vez en cada fila:
  - $x_{i,1,k} \vee x_{i,2,k} \vee \cdots \vee x_{i,9,k}$ para cada i y k
  - $\overline{(x_{i,j,k} \wedge x_{i,l,k})}$ para $j \neq l$ y cada i y k
- Cada número del 1 al 9 debe aparecer exactamente una vez en cada columna:
  - $x_{1,j,k} \vee x_{2,j,k} \vee \cdots \vee x_{9,j,k}$ para cada j y k
  - $\overline{(x_{i,j,k} \wedge x_{l,j,k})}$ para $i \neq l$ y cada j y k
- Cada número del 1 al 9 debe aparecer exactamente una vez en cada subcuadrícula de 3×3:
  - $x_{i,j,k} \vee x_{i+1,j,k} \vee x_{i+2,j,k} \vee x_{i,j+1,k} \vee x_{i+1,j+1,k} \vee x_{i+2,j+1,k} \vee x_{i,j+2,k} \vee x_{i+1,j+2,k} \vee x_{i+2,j+2,k}$ con $i \in \{1,4,7\}$ y $j \in \{1,4,7\}$
  - $\overline{(x_{i,j,k} \wedge x_{l,m,k})}$ para $(i,j) \neq (l,m)$ dentro de la misma subcuadrícula y cada k

Podemos combinar todas estas fórmulas proposicionales utilizando $\wedge$ para obtener una fórmula proposicional que modela el problema completo del Sudoku. Luego, podemos usar un SAT-solver para determinar si esta fórmula proposicional es satisfacible o no, lo que nos dirá si el Sudoku tiene solución o no, y en caso de que tenga solución, el SAT-solver nos dará la asignación de variables proposicionales que hace que la fórmula sea satisfacible, lo que nos permitirá reconstruir la solución del Sudoku.

1.6. Expresividad

La lógica proposicional tiene una intima relación con las *funciones booleanas*. Una función booleana es una función $f : \{0,1\}^n \to \{0,1\}$, es decir, una función que toma n bits de entrada y devuelve un bit de salida. Por ejemplo, la función $\wedge\colon \{0,1\}^2 \to \{0,1\}$ que representa la conjunción es una función booleana. De hecho, cada fórmula proposicional con n variables proposicionales puede ser vista como una función booleana de n variables. Por ejemplo, la fórmula $(p \wedge q) \vee r$ puede ser vista como la función booleana $f(p,q,r) = (p \wedge q) \vee r$.

De manera interesante cada función booleana puede ser representada por una fórmula proposicional. Esto se debe a que cada función booleana puede ser representada por una tabla de verdad y cada tabla de verdad puede ser representada por una fórmula proposicional en *formal normal disyuntiva*. Por ejemplo, consideremos la función booleana $f\colon \{0,1\}^3 \to \{0,1\}$ dada por la siguiente tabla de verdad:

p	q	r	$f(p,q,r)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Notemos que esta función booleana devuelve 1 para las siguientes combinaciones de entrada: $(0,0,1)$, $(1,0,0)$ y $(1,1,0)$. Por lo tanto, podemos representar esta función booleana de la siguiente manera:

$$f(p,q,r) = (\overline{p} \wedge \overline{q} \wedge r) \vee (p \wedge \overline{q} \wedge \overline{r}) \vee (p \wedge q \wedge \overline{r}).$$

La fórmula que resulta de este proceso siempre es una disyunción de conjunciones de literales, es decir, una fórmula en *forma normal disyuntiva*.

> **Definición.** Una fórmula proposicional está en *forma normal disyuntiva* si es una disyunción de conjunciones de variables proposicionales o sus negaciones.

(Otra forma normal común es la *forma normal conjuntiva*, que es una conjunción de disyunciones.)

Notemos que la fórmula obtenida a partir de la tabla de verdad no es única, ya que podemos aplicar leyes de la lógica proposicional para obtener fórmulas equivalentes. A pesar de esto obtenemos una consecuencia interesante: cada función booleana puede ser representada por una fórmula proposicional en forma normal disyuntiva. En particular, cada función booleana puede ser representada usando únicamente los operadores $\wedge$, $\vee$ y $\neg$. Dicho fenómeno se conoce como la *completitud funcional* de los operadores $\wedge$, $\vee$ y $\neg$.

> **Definición.** Un conjunto de operadores es *funcionalmente completo* si cada formula es equivalente a una fórmula que solo utiliza operadores de ese conjunto.

Concluimos que el conjunto de operadores $\{\wedge, \vee, \neg\}$ es funcionalmente completo. De hecho, existen otros conjuntos de operadores que también son funcionalmente completos. Por ejemplo, el conjunto de operadores $\{\wedge, \neg\}$ es funcionalmente completo, ya que podemos expresar la disyunción utilizando la conjunción y las leyes de De Morgan.

1.7. Lógica de primer orden

A pesar de que la lógica proposicional puede representar cualquier función booleana, no es capaz de expresar ciertas propiedades que involucran objetos y sus relaciones. Incluso, no es capaz de expresar argumentos sencillos como el que consideramos al inicio del Capítulo:

Todos las personas son mortales.
Sócrates es persona.

Por lo tanto, Sócrates es mortal.

A pesar de que este argumento nos pareció "naturalmente válido" no lo podemos expresar via la lógica proposicional. Esto se debe a que nos falta la capacidad de hablar sobre objetos y sus propiedades, así como de cuantificar sobre ellos. En particular, necesitamos:

- Describir objetos y sus propiedades, por ejemplo denotar que x es una persona se puede expresar como $\texttt{persona}(x)$, donde `persona` es un símbolo de predicado que representa la propiedad de ser una persona. Similarmente, denotar que x es mortal se puede expresar como $\texttt{mortal}(x)$, donde `mortal` es un símbolo de predicado que representa la propiedad de ser mortal. Luego, el hecho de que Socrátes es persona se puede expresar como $\texttt{persona}(\texttt{Sócrates})$.
- Cuantificar sobre objetos, por ejemplo la afirmación "todas las personas son mortales" se puede expresar como $\forall x(\texttt{persona}(x) \to \texttt{mortal}(x))$.

Luego, el argumento completo se puede expresar como:

$$\frac{\begin{array}{l}\forall x(\texttt{persona}(x) \to \texttt{mortal}(x)) \\ \texttt{persona}(\texttt{Sócrates})\end{array}}{\texttt{mortal}(\texttt{Sócrates})}$$

Notemos que no sólo nos interesa razonar sobre objetos individuales, sino también sobre relaciones entre objetos. Por ejemplo, consideremos el siguiente argumento:

> Si un usuario sigue a otro en una red social, entonces el primero puede ver las publicaciones del segundo. Alicia sigue a Bob. Por lo tanto, Alicia puede ver las publicaciones de Bob.

Este argumento también es "naturalmente válido" pero no lo podemos expresar usando la lógica proposicional. Sin embargo, podemos expresarlo usando la lógica de primer orden de la siguiente manera:

$$\frac{\begin{array}{l}\forall x \forall y(\texttt{Sigue}(x, y) \to \texttt{PuedeVer}(x, y)) \\ \texttt{Sigue}(\texttt{Alicia}, \texttt{Bob})\end{array}}{\texttt{PuedeVer}(\texttt{Alicia}, \texttt{Bob})}$$

1.7.1. Predicados

Tal como la lógica proposicional estudiaba las proposiciones, la lógica de primer orden estudia los *predicados*.

Un *predicado* es una función que toma un número finito de argumentos y devuelve un valor de verdad. En la lógica de primer orden, los predicados se utilizan para representar propiedades de objetos y relaciones entre objetos.

Ejemplo. El predicado $\texttt{persona}(x)$ representa la propiedad de ser una persona, mientras que el predicado $\texttt{Sigue}(x, y)$ representa la relación de seguir entre dos objetos. Otro ejemplo de predicado es el predicado $\texttt{Mayor}(x, y)$ que representa la relación de ser mayor entre dos objetos, es decir, $\texttt{Mayor}(x, y)$ es verdadero si x es mayor que y y falso en caso contrario.

Notemos que podemos cuantificar sobre los predicados para obtener otros predicados.

Ejemplo. El predicado $\texttt{esMinimo}(x)$ que representa la propiedad de ser el minimo se puede definir como $\texttt{esMinimo}(x) = \forall y(\texttt{Mayor}(y, x))$, es decir, x es el minimo si para todo y, x es mayor que y.

Las variables que aparecen sin cuantificar se llaman *variables libres*, mientras que las variables que aparecen dentro del alcance de un cuantificador se llaman *variables ligadas*. Una fórmula que no tiene variables libres se llama *fórmula cerrada* y una fórmula que tiene al menos una variable libre se llama *fórmula abierta*.

Ejemplo. La formula $\texttt{esMinimo}(x)$ es una fórmula abierta, ya que x es una variable libre. Por otro lado, la fórmula $\exists x(\texttt{esMinimo}(x))$ es una fórmula cerrada, ya que x está cuantificado.

Este último ejemplo nos muestra que la semántica de la lógica de primer orden es más compleja que la semántica de la lógica proposicional, ya que dependen no sólo de la semántica de los predicados, sino también del dominio de discurso sobre el cual cuantificamos.

En particular, $\exists x.\texttt{esMinimo}(x)$ es verdadero si el dominio de discurso es, por ejemplo, el conjunto de los números naturales, y existe un número natural que es el minimo, es decir, el 0. Pero, si el dominio de discurso es el conjunto de los números enteros, entonces $\exists x.\texttt{esMinimo}(x)$ es falso, ya que no existe un número entero que sea el minimo.

1.7.2. Sintaxis

Nuevamente, para definir la lógica de primer orden, necesitamos definir su sintaxis y semántica. Para esto definiremos términos y fórmulas de la lógica de primer orden.

Definición. Un *término* es una expresión que representa un objeto en el dominio de discurso. Los términos pueden ser variables, constantes o funciones aplicadas a otros términos.

Definición. Una *fórmula* de la lógica de primer orden se define inductivamente de la siguiente manera:

- Si P es un predicado de aridad n y $t_1, t_2, \ldots, t_n$ son términos, entonces $P(t_1, t_2, \ldots, t_n)$ es una fórmula.
- Si ϕ y ψ son fórmulas, entonces $\overline{\phi}$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$ y $(\phi \leftrightarrow \psi)$ son fórmulas.
- Si ϕ es una fórmula y x es una variable, entonces $\forall x(\phi)$ y $\exists x(\phi)$ son fórmulas.

Ejemplo. Tenemos que $\forall x(\forall y(P(f(x)) \to (Q(x, y) \wedge \overline{R}(x, f(y), z))))$ es una fórmula de la lógica de primer orden, donde P es un predicado de aridad 1, Q es un predicado de aridad 2, R es un predicado de aridad 3 y f es una función de aridad 1.

Nuevamente omitiremos los parentesis haciendo uso de la precedencia de los operadores. Una convención común es que la negación tiene la mayor precedencia, seguida de la conjunción, disyunción, cuantificadores, implicancia y biimplicancia. A pesar de esto, es recomendable usar paréntesis cuando ayuden a mejorar la legibilidad de las fórmulas.

Es común también agrupar los cuantificadores, por ejemplo, en lugar de escribir $\forall x \forall y \forall z\ P(x, y, z)$, podemos escribir $\forall x, y, z\ P(x, y, z)$.

Ejemplo. Podemos usar nuestro nuevo lenguaje para representar propiedades que conocemos de las relaciones.

- La *reflexividad* de una relación R se puede expresar como $\forall x R(x, x)$.

- La *simetría* de una relación R se puede expresar como $\forall x \forall y(R(x,y) \to R(y,x))$.
- La *antisimetría* de una relación R se puede expresar como $\forall x, y.(R(x,y) \land R(y,x) \to x = y)$, donde $=$ es un símbolo de relación que representa la igualdad entre objetos.
- La *transitividad* de una relación R se puede expresar como $\forall x, y, z.((R(x,y) \land R(y,z)) \implies R(x,z))$.

1.7.3. Semántica

La semántica de la lógica de primer orden es más compleja que la semántica de la lógica proposicional, ya que depende no sólo de la semántica de los predicados, sino también del dominio de discurso sobre el cual cuantificamos. Acá el concepto clave es el de *estructura*.

Definición. Sea ϕ una fórmula de la lógica de primer orden. Una *estructura* para ϕ es una tupla $\mathcal{M} = (D^{\mathcal{M}}, I^{\mathcal{M}})$ donde $D^{\mathcal{M}}$ es un conjunto no vacío llamado *dominio de discurso* y $I^{\mathcal{M}}$ es una función de interpretación que asigna a cada símbolo de predicado de aridad n una función $I^{\mathcal{M}}(P) : (D^{\mathcal{M}})^n \to \{0,1\}$, a cada símbolo de función de aridad n una función $I^{\mathcal{M}}(f) : (D^{\mathcal{M}})^n \to D^{\mathcal{M}}$ y a cada símbolo de constante un elemento de $D^{\mathcal{M}}$.

Si una fórmula tiene variables libres, entonces también necesitamos una asignación de variables para evaluar la fórmula.

Definición. Sea ϕ una fórmula de la lógica de primer orden y $\mathcal{M} = (D^{\mathcal{M}}, I^{\mathcal{M}})$ una estructura para ϕ. Una *asignación de variables* para ϕ es una función σ que asigna a cada variable libre de ϕ un elemento de $D^{\mathcal{M}}$.

Por ejemplo consideremos la fórmula dada por

$$\forall x \forall y(P(s(x), s(y))) \to (P(x,y))$$

Y consideramos la estructura $\mathcal{M} = (\mathbb{N}, I)$, donde $I(s) : \mathbb{N} \to \mathbb{N}$ es la función sucesor, $I(P) : \mathbb{N}^2 \to \{0,1\}$ es la función que representa la relación de ser mayor entre números naturales. En esta estructura, la fórmula es verdadera, ya que si el sucesor de x es mayor que el sucesor de y, entonces x es mayor que y. Por otro lado, si interpretamos la formula de la misma manera excepto que P represente la relación de ser menor, entonces la fórmula sería falsa.

Veamos otra fórmula, esta vez con variables libres:

$$\exists x(P(x,y)).$$

Consideremos la estructura $\mathcal{M} = (\mathbb{N}, I)$, donde $I(P) : \mathbb{N}^2 \to \{0,1\}$ es el predicado $x^2 = y$. En esta estructura, la fórmula es verdadera para la asgnación de variables $\sigma(y) = 4$, ya que existe un número natural x tal que $x^2 = 4$, por ejemplo $x = 2$. Sin embargo, la fórmula es falsa para la asignación de variables $\sigma(y) = 3$, ya que no existe un número natural x tal que $x^2 = 3$.

1.7.4. Satisfacibilidad

Definición. Diremos que una fórmula ϕ es *satisfacible* si existe una estructura $\mathcal{M}$ y una asignación de variables σ tal que $(\mathcal{M}, \sigma) \models \phi$, es decir, ϕ es verdadera en la estructura $\mathcal{M}$ bajo la asignación de variables σ. En dicho caso, diremos que $\mathcal{M}$ es un *modelo* de ϕ.

Nuevamente podemos definirlo de manera inductiva sobre la estructura de las fórmulas. Veamos algunos casos:

- $(\mathcal{M}, \sigma) \models P(t_1, t_2, \ldots, t_n)$, cuando $I^{\mathcal{M}}(P)(\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_n)) = 1$.
- $(\mathcal{M}, \sigma) \models \phi \wedge \psi$ cuando $(\mathcal{M}, \sigma) \models \phi$ y $(\mathcal{M}, \sigma) \models \psi$, y así sucesivamente para los demás operadores.

Al contrario de la lógica proposicional, no existen algoritmos para determinar si una fórmula de la lógica de primer orden es satisfacible o no (de hecho, no pueden existir tales algoritmos).

1.7.5. Consequencia lógica y Equivalencia

Definición. Diremos que una fórmula ψ es una *consecuencia lógica* de un conjunto de fórmulas Γ si para toda estructura $\mathcal{M}$ y toda asignación de variables σ, si $(\mathcal{M}, \sigma) \models \Gamma$ entonces $(\mathcal{M}, \sigma) \models \psi$. En dicho caso, escribimos $\Gamma \models \psi$.

Una vez que tenemos el concepto de consecuencia lógica, podemos definir el concepto de equivalencia entre fórmulas.

Definición. Dos fórmulas ϕ y ψ son *equivalentes* si $\{\phi\} \models \psi$ y $\{\psi\} \models \phi$.

Es decir, ϕ y ψ son equivalentes si para toda estructura $\mathcal{M}$ y toda asignación de variables σ, $(\mathcal{M}, \sigma) \models \phi$ si y sólo si $(\mathcal{M}, \sigma) \models \psi$.

Conocemos algunas equivalencias importantes, por ejemplo:

- $\overline{\forall x \phi} \equiv \exists x \overline{\phi}$.
- $\overline{\exists x \phi} \equiv \forall x \overline{\phi}$.
- $\forall x(\phi \wedge \psi) \equiv \forall x \phi \wedge \forall x \psi$.
- $\exists x(\phi \vee \psi) \equiv \exists x \phi \vee \exists x \psi$.

¿Cómo podemos demostrar dichas equivalencias? Notemos que no podemos usar tablas de verdad, ya que las fórmulas de la lógica de primer orden pueden tener variables libres y cuantificadores. En su lugar, razonaremos via nuevas reglas de inferencia que se basan en la semántica de la lógica de primer orden.

1.8. Inferencia en lógica de predicados

Al igual que antes, el framework de la deducción natural nos permitirá razonar sobre la lógica de primer orden. Simplemente, consideraremos reglas de inferencia adicionales para los cuantificadores.

Inferencia (∀-eliminación)**.** Si $\forall x\ \phi(x)$ es una fórmula, entonces podemos inferir $\phi(t)$ para cualquier término t.

Gráficamente, esta regla se puede representar de la siguiente manera:

$$\frac{\forall x\ \phi(x)}{\phi(t)},$$

mientras que de manera formal tenemos $\{\forall x\ \phi(x)\} \vdash \phi(t)$.

Ejemplo. Recordemos el argumento que consideramos al inicio del capítulo:

$$\frac{\begin{array}{l}\forall x(\texttt{persona}(x) \to \texttt{mortal}(x)) \\ \texttt{persona}(\texttt{Sócrates})\end{array}}{\texttt{mortal}(\texttt{Sócrates})}$$

Podemos demostrar la validez de este argumento usando la regla de ∀-eliminación de la siguiente manera:

1. $\forall x(\texttt{persona}(x) \to \texttt{mortal}(x))$ (premisa)
2. $\texttt{persona}(\texttt{Sócrates})$ (premisa)
3. $\texttt{persona}(\texttt{Sócrates}) \to \texttt{mortal}(\texttt{Sócrates})$ (∀-eliminación en 1.)
4. $\texttt{mortal}(\texttt{Sócrates})$ (⇒-eliminación en 2. y 3.)

Inferencia (∀-introducción)**.** Si posemos inferir $\phi(t)$ para cualquier término t, entonces podemos inferir $\forall x\ \phi(x)$.

Gráficamente, esta regla se puede representar de la siguiente manera:

$$\frac{\begin{array}{l}\text{Introducción de } t \text{ arbitrario.} \\ \vdots \\ \phi(t)\end{array}}{\forall x\ \phi(x)}.$$

Esta regla es similar a las reglas con descarga de premisas que vimos para la implicancia y la disyunción. Es decir, vamos a suponer temporalmente que t es un término arbitrario y vamos a demostrar $\phi(t)$ sin hacer ninguna suposición adicional sobre t.

Ejemplo. Consideremos el siguiente argumento:

$$\frac{\begin{array}{l}\forall x(P(x) \to Q(x)) \\ \forall x(Q(x) \to R(x))\end{array}}{\forall x(P(x) \to R(x))}$$

Podemos demostrar la validez de este argumento usando la regla de ∀-introducción de la siguiente manera:

1. $\forall x(P(x) \to Q(x))$ (premisa)

2. $\forall x(Q(x) \to R(x))$ (premisa)

   - Introducción de t arbitrario.

     3. $P(t)$ ($\Rightarrow$-introducción)
     4. $P(t) \Rightarrow Q(t)$ ($\forall$-eliminación en 1.)
     5. $Q(t)$ ($\Rightarrow$-eliminación en 3. y 4.)
     6. $Q(t) \Rightarrow R(t)$ ($\forall$-eliminación en 2.)
     7. $R(t)$ ($\Rightarrow$-eliminación en 5. y 6.)

   8. $P(t) \Rightarrow R(t)$ ($\Rightarrow$-introducción en 3.-7.)

9. $\forall x(P(x) \to R(x))$ ($\forall$-introducción en 8.)

Inferencia ($\exists$-introducción)**.** Si $\phi(t)$ es una fórmula, entonces podemos inferir $\exists x\ \phi(x)$.

Gráficamente, esta regla se puede representar de la siguiente manera:

$$\frac{\phi(t)}{\exists x\ \phi(x)}.$$

Inferencia ($\exists$-eliminación)**.** Si $\exists x\ \phi(x)$ y a partir de un elemento arbitrario t tal que $\phi(t)$ podemos inferir ψ, entonces podemos inferir ψ.

Gráficamente, esta regla se puede representar de la siguiente manera:

$$\frac{\begin{array}{l}\exists x\ \phi(x) \\ \quad \text{Introducción de } t \text{ arbitrario.} \\ \quad \phi(t) \\ \quad \vdots \\ \quad \psi\end{array}}{\psi}.$$

Ejemplo. Veamos un ejemplo de uso de la regla de $\exists$-eliminación. Consideremos el siguiente argumento:

$$\frac{\begin{array}{l}\forall x(P(x) \Rightarrow Q(x)) \\ \exists x(P(x) \wedge R(x))\end{array}}{\exists x Q(x)}$$

Podemos demostrar la validez de este argumento usando la regla de $\exists$-eliminación de la siguiente manera:

1. $\forall x(P(x) \Rightarrow Q(x))$ (premisa)
2. $\exists x(P(x) \wedge R(x))$ (premisa)

   - Introducción de t arbitrario. ($\exists$-eliminación en 2.)

   3. $P(t) \wedge R(t)$ ($\exists$-eliminación en 2.)

4. $P(t)$ ($\wedge$-eliminación en 3.)
5. $P(t) \Rightarrow Q(t)$ ($\forall$-eliminación en 1.)
6. $Q(t)$ ($\Rightarrow$-eliminación en 4.)
7. $\exists x Q(x)$ ($\exists$-introducción en 6.)
8. $\exists x Q(x)$ ($\exists$-eliminación en 2. y 3.-7.)

Por último, tendremos una relación que tendrá un significado semántico especial, la relación de igualdad.

Inferencia (Reglas de igualdad)**.** La igualdad es una relación binaria que se denota por el símbolo $=$ y que es reflexiva, simétrica y transitiva. Además, la igualdad permite sustituir términos por otros términos iguales en cualquier fórmula.

Gráficamente, las reglas de igualdad se pueden representar de la siguiente manera:

$$\frac{}{t = t} \quad \frac{t_1 = t_2}{t_2 = t_1} \quad \frac{t_1 = t_2 \quad t_2 = t_3}{t_1 = t_3} \quad \frac{t_1 = t_2 \quad \phi(t_1)}{\phi(t_2)}.$$

1.8.1. Corrección y completitud

Nuevamente, una pregunta natural es si el sistema de deducción natural para la lógica de primer orden es correcto y completo. La correctitud se sigue de la corrección de las reglas de inferencia para los cuantificadores. Por otro lado, la completitud es un resultado famoso de Gödel que establece que si una fórmula es válida, entonces existe una demostración de esa fórmula usando las reglas de inferencia de la deducción natural.

Teorema (Completitud de Gödel 1930)**.** Para cualquier conjunto de fórmulas Γ y cualquier fórmula ϕ, tenemos que $\Gamma \vdash \phi$ si y solo si $\Gamma \models \phi$.

Capítulo 2

Técnicas de demostración

2.1. Técnicas inspiradas en deducción natural

En clases anteriores hemos hablado de la "deducción natural" como sistema formal de demostraciones. En esta clase hablaremos de técnicas de demostración, muchas veces inspiradas en deducción natural, pero esta vez, nos interesa más que el lector sea capaz de entender y validar las demostraciones, que el hecho de que sean formales.

Presentaremos varias técnicas de demostración, inspiradas en las reglas de deducción natural, y veremos ejemplos de cada una de ellas.

2.1.1. Técnicas de demostración: implicancia

Demostración directa La demostración directa se usa cuando queremos probar una implicancia de la forma "si P, entonces Q" avanzando desde las hipótesis hacia la conclusión. La idea es suponer verdadera la premisa y, mediante definiciones o propiedades conocidas, deducir paso a paso propiedades intermedias, hasta llegar a la conclusión.

> **Proposición.** Si $n \in \mathbb{N}$ es par, entonces n^2 es par.

Demostración. Como n es par, existe $k \in \mathbb{N}$ tal que $n = 2k$. Por lo tanto,

$$n^2 = (2k)^2 = 4k^2 = 2(2k^2).$$

Como $n = 2(2k^2)$, tenemos que n^2 es par. □

Demostración por contrarrecíproca Esta técnica consiste en probar la afirmación lógicamente equivalente: en vez de demostrar "$P \implies Q$", demostramos "$\neg Q \implies \neg P$". Suele ser útil cuando trabajar con la negación de la conclusión permite un argumento más natural o da más información.

> **Proposición.** Si $n \in \mathbb{N}$ es tal que n^2 es par, entonces n es par.

Demostración. Por contrarrecíproca, si n es impar, entonces existe $k \in \mathbb{N}$ tal que $n = 2k + 1$. Por lo tanto, $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$, y de esto tenemos que n^2 es impar. □

2.1.2. Técnicas de demostración: disyunción

Demostración por casos: La demostración por casos se aplica cuando la información disponible se expresa como una disyunción, o cuando podemos particionar el problema en escenarios. Se demuestra la misma conclusión en cada caso y, como no quedan alternativas, la conclusión vale en general.

> **Proposición.** Si n no es multiplo de 3, entonces 3 divide a $n^2 - 1$.

Demostración. Tenemos dos casos: o $n = 3k + 1$ para algún $k \in \mathbb{N}$, o $n = 3k + 2$ para algún $k \in \mathbb{N}$. En el primer caso,

$$n^2 - 1 = (3k + 1)^2 - 1 = (3k + 2)3k.$$

De esto, tenemos que 3 divide a $n^2 - 1$. En el segundo caso,

$$n^2 - 1 = (3k + 2)^2 - 1 = (3k + 3)(3k + 1) = 3(k + 1)(3k + 1).$$

De esto, tenemos que 3 divide a $n^2 - 1$. □

2.1.3. Técnicas de demostración: negación

Reescritura de la negación. Cuando el objetivo es una negación, muchas veces conviene reescribirla en una forma equivalente más manejable "en positivo". Esto pues en general, la negación de una afirmación es más difícil de trabajar que una "afirmación positiva".

> **Proposición.** Sean A, B, C conjuntos tal que $A \cap C \subseteq B$ y $a \in C$. Pruebe que $a \notin A \setminus B$.

Demostración. Notemos que $a \notin A \setminus B$ es equivalente a probar que

$$\overline{(a \notin A \setminus B)} \equiv a \notin A \vee a \in B \equiv a \in A \implies a \in B.$$

Por lo tanto, supongamos que $a \in A$. De esto se sigue que, $a \in A \cap C \subseteq B$, y por lo tanto, $a \in B$. □

Contradicción En una demostración por contradicción suponemos temporalmente que la afirmación que queremos probar es falsa y derivamos una contradicción. Del principio de explosión, podemos entonces concluir que la afirmación original es verdadera.

> **Teorema.** $\sqrt{2}$ es irracional.

Demostración. Supongamos, en busca de una contradicción, que $\sqrt{2}$ es racional. Entonces, existe $p, q \in \mathbb{N}$ con $q \neq 0$ y $\gcd(p, q) = 1$ tal que $\sqrt{2} = \frac{p}{q}$. Concluimos que, $2 = \frac{p^2}{q^2}$, y por lo tanto, $p^2 = 2q^2$. De esto obtenemos que p^2 es par y por lo demostrado anteriormente, p es par. Como p es par, existe $k \in \mathbb{N}$ tal que $p = 2k$. Luego, $2q^2 = p^2 = (2k)^2 = 4k^2$, y por lo tanto, $q^2 = 2k^2$. Obtenemos que q^2 es par, y por ende, q es par. Concluimos que p y q son pares, lo cual es una contradicción, ya que $\gcd(p, q) = 1$. □

2.1.4. Demostraciones con equivalencias

Separación de la equivalencia. Cuando el objetivo es una equivalencia $P \iff Q$, una estrategia común es separar la demostración en dos implicancias: "$P \implies Q$" y "$Q \implies P$".

> **Proposición.** Sean $x, y \in \mathbb{Z}$. Demuestre que x y y tienen la misma paridad si y sólo si $x + y$ es par.

Demostración. Primero, demostraremos que si x y y tienen la misma paridad, entonces $x + y$ es par. Si x y y son ambos pares, entonces existen $k, l \in \mathbb{N}$ tales que $x = 2k$ y $y = 2l$, y por lo tanto, $x + y = 2(k + l)$ es par. Si x y y son ambos impares, entonces existen $k, l \in \mathbb{N}$ tales que $x = 2k + 1$ y $y = 2l + 1$, y por lo tanto, $x + y = 2(k + l + 1)$ es par.

Ahora, demostraremos que si $x + y$ es par, entonces x y y tienen la misma paridad. Por contrarrecíproca, si x e y no tienen la misma paridad, entonces sin pérdida de generalidad, supongamos que x es par y y es impar. Entonces, existen $k, l \in \mathbb{N}$ tales que $x = 2k$ y $y = 2l + 1$, y por lo tanto, $x + y = 2(k + l) + 1$ es impar. □

Cadena de equivalencias Algo que sirve a veces para demostrar algunas equivalencias es escribir una cadena de equivalencias, es decir, una secuencia de enunciados $P_1, P_2, \ldots, P_n$ tal que P_1 es el enunciado que queremos demostrar, P_n es un enunciado que sabemos que es verdadero, y para cada i, $P_i \iff P_{i+1}$.

> **Proposición.** Sean A, B, C conjuntos, entonces $A \setminus (B \cup C) = (A \setminus B) \cap (A \setminus C)$.

Demostración. Notemos que nos piden probar que $x \in A \setminus (B \cup C)$ es equivalente a $x \in (A \setminus B) \cap (A \setminus C)$. Por lo tanto, escribamos una cadena de equivalencias:

$$\begin{aligned} x \in A \setminus (B \cup C) &\iff x \in A \wedge x \notin B \cup C \\ &\iff x \in A \wedge x \notin B \wedge x \notin C \\ &\iff x \in A \setminus B \wedge x \in A \setminus C \\ &\iff x \in (A \setminus B) \cap (A \setminus C). \end{aligned}$$

□

2.1.5. Demostraciones con cuantificadores

Cuantificador universal Cuando un enunciado tiene cuantificadores, la estrategia depende de su forma lógica. Para probar "$\forall x \in A,\ P(x)$", tomamos un elemento arbitrario $x \in A$ y demostramos $P(x)$.

Demostración de un enunciado universal

> **Proposición.** Para todo $x, y \in \mathbb{Q}$, tenemos que $x + y \in \mathbb{Q}$.

Demostración. Sea $x, y \in \mathbb{Q}$. Entonces, existen $a, b, c, d \in \mathbb{Z}$ con $b, d \neq 0$ tales que $x = \frac{a}{b}$ y $y = \frac{c}{d}$. Por lo tanto,

$$x + y = \frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}.$$

Como $ad + bc \in \mathbb{Z}$ y $bd \in \mathbb{Z} \setminus \{0\}$, tenemos que $x + y \in \mathbb{Q}$. □

Cuantificador existencial Para probar un enunciado de la forma "$\exists x \in A,\ P(x)$", basta con encontrar un elemento específico $x \in A$ que satisfaga $P(x)$ y demostrar que efectivamente lo hace.

> **Proposición.** Existe $m \in \mathbb{Z}$ tal que $\frac{m-7}{2m+4} = 5$.

Demostración. Tomamos $m = -3$. Entonces,

$$\frac{m-7}{2m+4} = \frac{-3-7}{2(-3)+4} = \frac{-10}{-2} = 5.$$

Por lo tanto, existe un entero m que satisface la ecuación. □

Notemos que usualmente obtenemos el candidato a ser el elemento que satisface la afirmación buscando propiedades que nos indiquen quien o como debe ser dicho elemento. Por ejemplo, en el caso anterior, podríamos haber despejado m de la ecuación dada para encontrar el candidato a ser el elemento que satisface la afirmación.

2.2. Inducción

El principio de inducción es una técnica de demostración que se utiliza para probar propiedades que dependen de un número natural n. La idea es demostrar que la propiedad es verdadera para un caso base, y luego demostrar que si la propiedad es verdadera para un número natural n, entonces también es verdadera para el número natural siguiente, es decir, para $n+1$. De esta forma, se concluye que la propiedad es verdadera para todo número natural n.

> **Definición.** El *principio de inducción* establece que si $P(n)$ es una propiedad que depende de un número natural n, y se cumplen las siguientes condiciones:
>
> - $P(0)$ es verdadera (caso base).
> - Para todo $n \in \mathbb{N}$, si $P(n)$ es verdadera, entonces $P(n+1)$ también es verdadera (paso inductivo).
>
> Entonces, $P(n)$ es verdadera para todo $n \in \mathbb{N}$.

No nos preocuparemos aún de por qué el principio de inducción es válido, sino que veamos algunos ejemplos de cómo aplicarlo para demostrar afirmaciones.

> **Proposición.** Para todo $n \in \mathbb{N}$, se cumple que $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$.

Demostración.

- **Caso base:** Para $n = 0$, tenemos que $1 + 2 + \ldots + 0 = 0$ y $\frac{0(0+1)}{2} = 0$, por lo que la afirmación es verdadera.
- **Hipótesis inductiva:** $1 + 2 + \ldots + n = \frac{n(n+1)}{2}$.

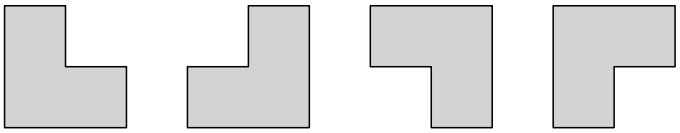


Figura 2.1: Una pieza de tromino es una figura formada por tres cuadrados adyacentes en forma de L.

- **Paso inductivo:** Veamos que la afirmación es verdadera para $n+1$. En efecto,

$$1+2+\cdots+n+(n+1) = \underbrace{(1+2+\cdots+n)}_{=\frac{n(n+1)}{2}} + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+2)}{2}.$$

Por el principio de inducción, concluimos que la afirmación es verdadera para todo $n \in \mathbb{N}$. □

En lo ejemplo anterior podemos ver un esquema clásico de las demostraciones por inducción: en el paso inductivo, buscamos como hacer aparecer la hipótesis inductiva, y luego usamos esa hipótesis para deducir la afirmación para $n+1$.

Veamos un segundo ejemplo, esta vez con una afirmación más compleja.

Proposición. Todo tablero de ajedrez de tamaño $2^n \times 2^n$ con una casilla faltante puede ser cubierto completamente con piezas de tromino.

Demostración.

- **Caso base:** Para $n = 0$, el tablero es de tamaño 1×1 y con una casilla faltante, por lo que no hay nada que cubrir, y la afirmación es verdadera.
- **Hipótesis inductiva:** Todo tablero de ajedrez de tamaño $2^n \times 2^n$ con una casilla faltante puede ser cubierto completamente con piezas de tromino.
- **Paso inductivo:** Veamos que la afirmación es verdadera para $n+1$. Particionemos el tablero de ajedrez de tamaño $2^{n+1} \times 2^{n+1}$ con una casilla faltante en cuatro subtableros de tamaño $2^n \times 2^n$. Uno de estos subtableros tendrá la casilla faltante, mientras que los otros tres estarán completos. Colocamos una pieza de tromino en el centro del tablero, cubriendo una casilla de cada uno de los tres subtableros completos. Esto deja cada uno de los cuatro subtableros con exactamente una casilla faltante. Por la hipótesis inductiva, cada uno de estos subtableros puede ser cubierto completamente con piezas de tromino. Por lo tanto, el tablero completo también puede ser cubierto completamente con piezas de tromino.

Por el principio de inducción, concluimos que la afirmación es verdadera para todo $n \in \mathbb{N}$. □

A veces el paso inductivo requiere usar una hipótesis más fuerte de lo que queremos demostrar. Usualmente uno logra darse cuenta de esto cuando intenta hacer el paso inductivo y se da cuenta de que no puede usar la hipótesis inductiva para deducir la afirmación para $n+1$. Esto podría parecer contraintuitivo, pero es simplemente el hecho de que una conclusión más poderosa, nos da una premisa más fuerte para usar en el paso inductivo, lo que nos permite deducir la afirmación para $n+1$.

Veamos un ejemplo de esto con el juego de las Torres de Hanoi. En el juego de las Torres de Hanoi, tenemos tres varillas A, B y C y n discos de diferentes tamaños que pueden deslizarse sobre

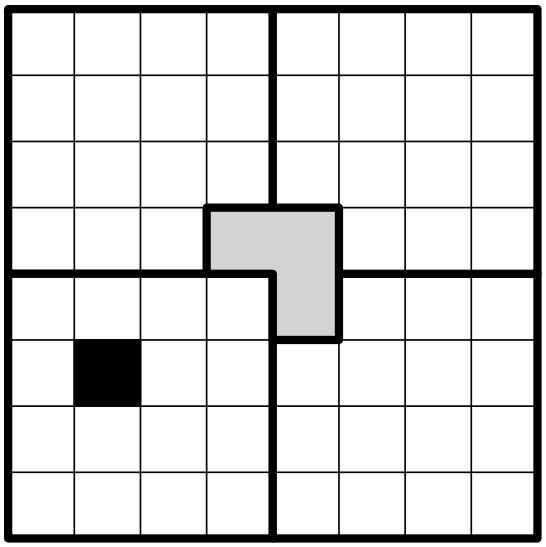


Figura 2.2: Esquema del paso inductivo para el problema del tromino.

cualquiera de las varillas. El juego comienza con los discos apilados en A en orden decreciente de tamaño, es decir, el disco más pequeño está en la parte superior y el más grande en la parte inferior. El objetivo es mover toda la pila de discos de A a C, siguiendo estas reglas:

- Solo se puede mover un disco a la vez.
- Cada movimiento consiste en tomar el disco superior de una de las pilas y deslizarlo sobre otra pila, encima de los discos que ya están allí.
- No se puede colocar un disco más grande sobre uno más pequeño.

Proposición. El juego de las Torres de Hanoi siempre es posible de resolver.

Demostración. Demostraremos una afirmación más fuerte que nos permitirá hacer el paso inductivo. Sea $P(n)$ la afirmación de que es posible mover una pila de n discos de una varilla X a otra varilla Y distinta de X.

- **Caso base:** Para $n = 0$, no hay discos que mover, por lo que la afirmación es verdadera.
- **Hipótesis inductiva:** $P(n)$ es verdadera, es decir, es posible mover una pila de n discos de una varilla X a otra varilla Y.
- **Paso inductivo:** Veamos que la afirmación es verdadera para $n + 1$. Para mover una pila de $n + 1$ discos de una varilla X a otra varilla Y, primero movemos los primeros n discos de la varilla X a la varilla $Z \notin \{X, Y\}$, usando la hipótesis inductiva. Luego, movemos el disco más grande (el disco número $n + 1$) de la varilla X a la varilla Y. Finalmente, movemos los n discos que habíamos movido a la varilla Z a la varilla Y, usando nuevamente la hipótesis inductiva.

Por el principio de inducción, concluimos que la afirmación es verdadera para todo $n \in \mathbb{N}$. En particular, es posible mover una pila de n discos de la varilla A a la varilla C, por lo que el juego siempre es posible de resolver. □

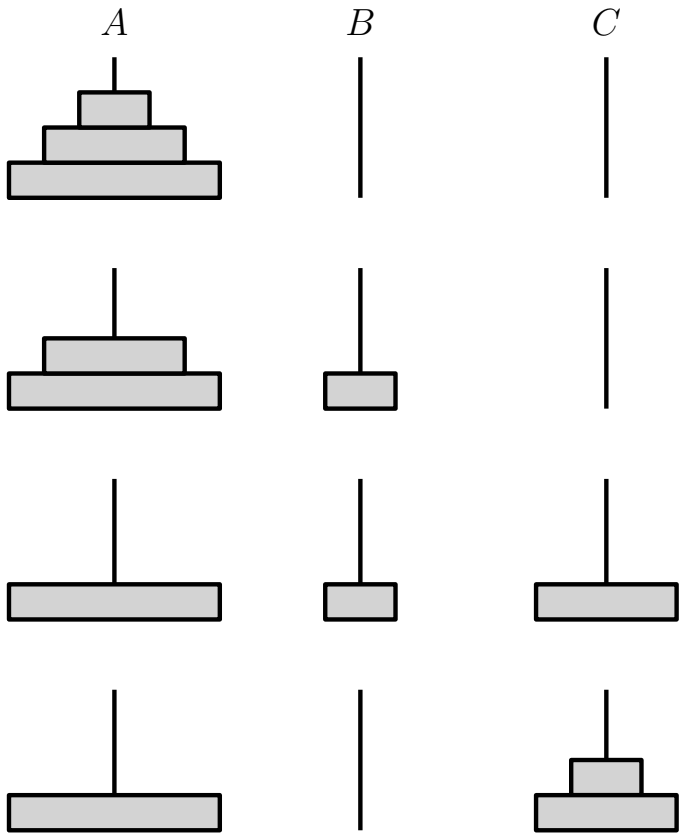


Figura 2.3: Una secuencia de 3 movimientos válidos en el juego de las Torres de Hanoi para $n = 3$.

2.2.1. Principio de inducción desde un caso base arbitrario

Una primera generalización a considerar es cuando el caso base no es $n = 0$, sino algún otro número natural n_0.

Definición. El *principio de inducción* establece que si $P(n)$ es una propiedad que depende de un número natural n, y se cumplen las siguientes condiciones:

- $P(n_0)$ es verdadera para algún $n_0 \in \mathbb{N}$ (caso base).
- Para todo $n \geq n_0$, si $P(n)$ es verdadera, entonces $P(n+1)$ también es verdadera (paso inductivo).

Entonces, $P(n)$ es verdadera para todo $n \in \mathbb{N}$ con $n \geq n_0$.

Veamos un ejemplo de cómo aplicar el principio de inducción general para demostrar una afirmación. Suponga que tenemos un parque con $2n + 1$ personas distribuidas de tal forma que cada persona tiene a otra persona a una distancia distinta de la distancia a cualquier otra persona. A la señal, cada persona dispara a la persona más cercana a ella.

Proposición. En el escenario descrito, al menos una persona no sufre un disparo.

Demostración. Sea $P(n)$ la afirmación de que si hay $2n+1$ personas distribuidas en un parque con

las condiciones dadas, entonces al menos una persona no sufre un disparo.

- **Caso base:** Para $n = 1$ tenemos tres personas, cada una a una distancia distinta de las otras dos. Sean A y B las dos personas más cercanas entre sí, y C la persona restante. Entonces, A dispara a B, B dispara a A, y C dispara a la persona más cercana, que es A o B. En cualquier caso, C no sufre un disparo, por lo que la afirmación es verdadera.
- **Hipótesis inductiva:** $P(n)$ es verdadera, es decir, si hay $2n + 1$ personas distribuidas en un parque con las condiciones dadas, entonces al menos una persona no sufre un disparo.
- **Paso inductivo:** Veamos que la afirmación es verdadera para $n+1$, es decir, si hay $2(n+1)+1$ personas distribuidas en un parque con las condiciones dadas, entonces al menos una persona no sufre un disparo. Sean A y B las dos personas más cercanas entre sí. Luego, A dispara a B y B dispara a A. Tenemos dos casos.
  - **Caso 1:** Nadie más dispara a A o B. Entonces, los restantes se disparan entre sí, y por la hipótesis inductiva, al menos una persona no sufre un disparo.
  - **Caso 2:** Alguien más dispara a A o B. Entonces, en el conjunto hay $2n+1$ personas y sólo ocurren a lo más $2n$ disparos, por lo que al menos una persona no sufre un disparo. □

2.2.2. Inducción fuerte

Nuevamente haremos el principio de inducción más fuerte, permitiendo que el paso inductivo use la hipótesis inductiva para todos los números naturales menores que n, en vez de sólo para n.

Definición. El *principio de inducción fuerte* establece que si $P(n)$ es una propiedad que depende de un número natural n, y se cumplen las siguientes condiciones:

- $P(n_0)$ es verdadera para algún $n_0 \in \mathbb{N}$ (caso base).
- Para todo $n \geq n_0$, si $P(k)$ es verdadera para todo k tal que $n_0 \leq k < n + 1$, entonces $P(n + 1)$ también es verdadera (paso inductivo).

Entonces, $P(n)$ es verdadera para todo $n \in \mathbb{N}$ con $n \geq n_0$.

Consideraremos un ejemplo de cómo aplicar el principio de inducción fuerte para demostrar una afirmación. Consideremos el siguiente juego de fichas de un jugador:

- Comenzamos con una pila de n fichas.
- En cada turno, podemos dividir una pila de tamaño k en dos pilas de tamaño a, b tales que $a + b = k$. Hacer esta jugada nos da un puntaje de ab.
- El juego termina cuando no podemos hacer más jugadas, es decir, cuando todas las pilas tienen tamaño 1.

Proposición. Para todo $n \in \mathbb{N}$, el puntaje total que se obtiene al jugar el juego de fichas de un jugador con una pila inicial de n fichas es $\frac{n(n-1)}{2}$ independientemente de las jugadas que se hagan.

Demostración.

- **Caso base:** Para $n = 1$ tenemos una pila de tamaño 1, por lo que no podemos hacer ninguna jugada, y el puntaje total es 0, que es igual a $\frac{1(1-1)}{2}$, por lo que la afirmación es verdadera.

- **Hipótesis inductiva:** $P(k)$ es verdadera para todo k tal que $1 \leq k < n$, es decir, para todo k tal que $1 \leq k < n$, el puntaje total que se obtiene al jugar el juego de fichas de un jugador con una pila inicial de k fichas es $\frac{k(k-1)}{2}$.

- **Paso inductivo:** Veamos que la afirmación es verdadera para $n+1$. Consideremos cualquier jugada que se haga en el primer turno, es decir, consideremos cualquier división de la pila de tamaño n en dos pilas de tamaño a, b tales que $a + b = n$. Por esta jugada, obtenemos un puntaje de ab. Luego, jugamos el juego con la pila de tamaño $a < n+1$ y el juego con la pila de tamaño $b < n+1$. Por la hipótesis inductiva, el puntaje total que se obtiene al jugar el juego con la pila de tamaño a es $\frac{a(a-1)}{2}$, y el puntaje total que se obtiene al jugar el juego con la pila de tamaño b es $\frac{b(b-1)}{2}$. Por lo tanto, el puntaje total que se obtiene al jugar el juego de fichas de un jugador con una pila inicial de n fichas es

$$ab + \frac{a(a-1)}{2} + \frac{b(b-1)}{2} = \frac{2ab + a^2 - a + b^2 - b}{2} = \frac{(a+b)^2 - (a+b)}{2} = \frac{n^2 - n}{2} = \frac{n(n-1)}{2}.$$

□

Consideraremos por último una variante del principio de inducción fuerte, en el que probaremos varios casos bases $\{n_0, \ldots, n_1\}$ con $n_0 < n_1$, y luego haremos el paso inductivo para $n > n_1$.

Definición. El *principio de inducción fuerte* establece que si $P(n)$ es una propiedad que depende de un número natural n, y se cumplen las siguientes condiciones:

- $P(n)$ es verdadera para todo n tal que $n_0 \leq n \leq n_1$ para algunos $n_0, n_1 \in \mathbb{N}$ con $n_0 < n_1$ (casos base).
- Para todo $n > n_1$, si $P(k)$ es verdadera para todo k tal que $n_0 \leq k < n+1$, entonces $P(n+1)$ también es verdadera (paso inductivo).

Entonces, $P(n)$ es verdadera para todo $n \in \mathbb{N}$ con $n \geq n_0$.

La gracia acá, es que sólo necesitamos probar el paso inductivo para $n > n_1$, pero podemos usar la hipótesis inductiva para todo k tal que $n_0 \leq k < n$. Veamos un ejemplo de cómo aplicar esta variante del principio de inducción fuerte para demostrar una afirmación.

Consideremos el siguiente problema de compra de nuggets: Una cadena de comida rápida vende nuggets en bolsas de 3 unidades o cajitas de 5 unidades.

Proposición. Para todo $n \geq 8$, es posible comprar exactamente n nuggets usando bolsas de 3 o cajitas de 5 unidades.

Demostración. Sea $P(n)$ la afirmación de que es posible comprar exactamente n nuggets usando bolsas de 3 o cajitas de 5 unidades.

- **Casos base:** Para $n = 8$, podemos comprar exactamente 8 nuggets usando una bolsa de 3 unidades y una cajita de 5 unidades. Para $n = 9$, podemos comprar exactamente 9 nuggets

usando tres bolsas de 3 unidades. Para $n = 10$, podemos comprar exactamente 10 nuggets usando dos cajitas de 5 unidades. Por lo tanto, $P(n)$ es verdadera para todo n tal que $8 \leq n \leq 10$.

- **Hipótesis inductiva:** $P(k)$ es verdadera para todo k tal que $8 \leq k < n$, es decir, para todo k tal que $8 \leq k < n$, es posible comprar exactamente k nuggets usando bolsas de 3 o cajitas de 5 unidades.
- **Paso inductivo:** Veamos que la afirmación es verdadera para $n > 10$. En efecto, por la hipótesis inductiva, es posible comprar exactamente $n-3$ nuggets usando bolsas de 3 o cajitas de 5 unidades, ya que $n - 3 \geq 8$. Entonces, podemos comprar exactamente n nuggets usando una bolsa de 3 unidades adicional a lo que ya compramos para obtener $n - 3$ nuggets. □

2.2.3. Principio del buen orden

Discutiremos un poco ahora sobre la validez del principio de inducción. Para esto consideraremos el principio del buen ordern, que puede ser considerado como un axioma de los números naturales.[1]

> **Definición.** El *principio del buen orden* establece que todo subconjunto no vacío de los números naturales tiene un elemento mínimo.

Antes de ver como se relaciona el principio del buen orden con el principio de inducción, veamos que el principio del buen orden es tambien una herramienta de demostración útil. Nuevamente, el objetivo es demostrar que una propiedad $P(n)$ es verdadera para todo $n \geq n_0$, pero esta vez el razonamiento será por usualmente por contradicción, definiendo el conjunto de contraejemplos a $P(n)$ como

$$S = \{n \in \mathbb{N} : n \geq n_0 \text{ y } P(n) \text{ es falsa}\}.$$

Luego, usamos el principio del buen orden para obtener un número m que es el menor número para el cual $P(m)$ es falsa, y luego construir un $m' < m$ para el cual $P(m')$ es falsa, lo que es una contradicción, ya que m es el menor número para el cual $P(m)$ es falsa. Por esto mismo, también se conocen como *demostraciones por contraejemplo minimal*.

Podemos redemostrar el teorema de los nuggets usando el principio del buen orden.

> **Proposición.** Para todo $n \geq 8$, es posible comprar exactamente n nuggets usando bolsas de 3 o cajitas de 5 unidades.

Demostración. Supongamos, en busca de una contradicción, que existe un número $n \geq 8$ tal que no es posible comprar exactamente n nuggets usando bolsas de 3 o cajitas de 5 unidades. Por el principio del buen orden, existe un número natural $m \geq 8$ tal que m es el menor número para el cual no es posible comprar exactamente m nuggets usando bolsas de 3 o cajitas de 5 unidades.

Más aún, como $8 = 3+5$, $9 = 3+3+3$, y $10 = 5+5$, tenemos que $m \geq 11$. Notemos que si fuera imposible comprar exactamente m nuggets, entonces también sería imposible comprar exactamente $m - 3$ nuggets, ya que si pudiéramos comprar exactamente $m - 3$ nuggets, podríamos comprar

[1] Uno podría demostrar este principio si fijamos una construcción de los números naturales (por ejemplo, la construcción de Von Neumann) y ocupando simplemente axiomas de la teoría de conjuntos de Zermelo-Fraenkel, pero escapa de los objetivos de este curso.

exactamente m nuggets comprando una bolsa de 3 unidades adicional a lo que ya compramos para obtener $m-3$ nuggets. Por lo tanto, $m-3 \geq 8$ es un número menor que m para el cual no es posible comprar exactamente $m-3$ nuggets usando bolsas de 3 o cajitas de 5 unidades, lo cual es una contradicción, ya que m es el menor número para el cual no es posible comprar exactamente m nuggets usando bolsas de 3 o cajitas de 5 unidades. □

2.3. Equivalencias principios de inducción

Cuando miramos la demostración anterior del teorema de los nuggets usando el principio del buen orden, podemos ver que la idea de la demostración es muy similar a la idea del paso inductivo en una demostración por inducción fuerte. Esto es pues ambos principios son equivalentes. Por tanto, si aceptamos el principio del buen orden como un axioma de los números naturales, entonces el principio de inducción es un teorema que se puede demostrar usando el principio del buen orden. La demostración sigue justamente esta idea de *contraejemplo minimal* provisto por el principio del buen orden.

Teorema. El principio de inducción es equivalente al principio del buen orden.

Lo demostraremos con respecto al pricipio de inducción débil, pero luego veremos que no hay diferencias esencialmente.

Demostración. Probemos ambas implicancias por separado.

($\Rightarrow$) Supongamos que el principio de inducción es verdadero. Supongamos, en busca de una contradicción, que existe $S \subseteq \mathbb{N}$ no vacío que sin elemento mínimo. Sea $P(n)$ la afirmación de que $S \cap \{0, 1, \ldots, n\} = \emptyset$.

- **Caso base:** $P(0)$ es verdadera, ya que S no tiene elemento mínimo, por lo que $0 \notin S$, y por lo tanto $S \cap \{0\} = \emptyset$.
- **Hipótesis inductiva:** $P(n)$ es verdadera, es decir, $S \cap \{0, 1, \ldots, n\} = \emptyset$.
- **Paso inductivo:** Veamos que $P(n+1)$ es verdadera, es decir, veamos que $S \cap \{0, 1, \ldots, n+1\} = \emptyset$. En efecto, por la hipótesis inductiva, $S \cap \{0, 1, \ldots, n\} = \emptyset$, por lo que $S \cap \{0, 1, \ldots, n+1\} = S \cap \{n+1\}$. Si $(n+1) \in S$, lo que no es posible, ya que $n+1$ sería el elemento mínimo de S. Concluimos entonces que $(n+1) \notin S$, por lo que $S \cap \{0, 1, \ldots, n+1\} = \emptyset$.

Por el principio de inducción, concluimos que $P(n)$ es verdadera para todo $n \in \mathbb{N}$, es decir, $S \cap \{0, 1, \ldots, n\} = \emptyset$ para todo $n \in \mathbb{N}$. Como S es no-vacío, existe $s \in S$, pero $S \cap \{0, 1, \ldots, s\} = \emptyset$, lo que es una contradicción.

($\Leftarrow$) Supongamos que el principio del buen orden es verdadero. Sea $P(n)$ una propiedad que depende de un número natural n, y supongamos que se cumplen las siguientes condiciones:

- $P(n_0)$ es verdadera para algún $n_0 \in \mathbb{N}$ (caso base).
- Para todo $n \in \mathbb{N}$ si $P(n)$ es verdadera, entonces $P(n+1)$ también es verdadera (paso inductivo).

Supongamos, en busca de una contradicción, que $P(n)$ no es verdadera para todo $n \geq n_0$. Por el principio del buen orden, existe un número natural $m \geq n_0$ tal que $P(m)$ es falsa y $P(k)$ es verdadera para todo k tal que $n_0 \leq k < m$. Notemos que $m \neq n_0$, ya que $P(n_0)$ es verdadera, por lo que $m - 1 \geq n_0$. Por lo tanto, $P(m-1)$ es verdadera, y por el paso inductivo, $P(m)$ también es verdadera, lo que es una contradicción. □

Por último, veremos que los principios de inducción que hemos visto, son todos equivalentes entre sí.

Teorema. El principio de inducción débil, el principio de inducción desde un caso base arbitrario, el principio de inducción fuerte, y el principio de inducción fuerte con varios casos base, son todos equivalentes entre sí.

Demostración. Notemos que:

- Inducción débil es un caso particular de inducción desde un caso base arbitrario.
- Inducción desde un caso base arbitrario se sigue de inducción fuerte.
- Inducción fuerte es un caso particular de inducción fuerte con varios casos base.

Por lo tanto, sólo tenemos que mostrar que inducción fuerte con varios casos base se sigue de inducción débil. Sea $P(n)$ una propiedad que depende de un número natural n, tal que se cumple lo siguiente:

(H1) $P(n)$ es verdadera para todo n tal que $n_0 \leq n \leq n_1$ para algunos $n_0, n_1 \in \mathbb{N}$ con $n_0 < n_1$ (casos base).

(H2) Para todo $n > n_1$, si $P(k)$ es verdadera para todo k tal que $n_0 \leq k < n$, entonces $P(n)$ también es verdadera (paso inductivo).

Queremos probar que $P(n)$ es verdadera para todo $n \in \mathbb{N}$ con $n \geq n_0$. Para esto, definamos $Q(m)$ como la afirmación de que $P(n + n_1)$ es verdadera para todo $n \in \mathbb{N}$ tal que $n_0 \leq n + n_1 \leq m + n_1$. Probemos que $Q(m)$ es verdadera para todo $m \in \mathbb{N}$ por inducción débil.

- **Caso base:** $Q(0)$ es verdadera, ya que $P(n)$ es verdadera para todo $n \in \mathbb{N}$ tal que $n_0 \leq n + 0 \leq n_1 + 0$.
- **Hipótesis inductiva:** $Q(m)$ es verdadera.
- **Paso inductivo:** Veamos que $Q(m+1)$ es verdadera. Como $Q(m)$ es verdadera, $P(n + n_1)$ es verdadera para todo $n \in \mathbb{N}$ tal que $n_0 \leq n + n_1 \leq m + n_1$. En particular, $P(k)$ es verdadera para todo k tal que $n_0 \leq k < (m+1) + n_1$. Por lo tanto, por (H2), $P((m+1) + n_1)$ también es verdadera, por lo que $Q(m+1)$ es verdadera. □

Concluimos que $Q(m)$ es verdadera para todo $m \in \mathbb{N}$, es decir, $P(n + n_1)$ es verdadera para todo $n \in \mathbb{N}$ tal que $n_0 \leq n + n_1$, lo que implica que $P(n)$ es verdadera para todo $n \in \mathbb{N}$ tal que $n \geq n_0$.

2.4. Inducción estructural

Otra forma de la inducción es la *inducción estructural*, que es una generalización de la inducción a objetos que no son números naturales, pero que tienen una estructura recursiva, como por ejemplo, listas, árboles, etc.

2.4.1. Definiciones recursivas

Partamos viendo que signfica que un objeto tenga una estructura recursiva.

Definición. Sea $\mathcal{U}$ un dominio de objetos. Supongamos que tenemos

- un conjunto $B \subseteq \mathcal{U}$ de *objetos básicos* y
- *reglas de producción* $\mathcal{F}$ donde cada regla de producción es una función $f : \mathcal{U}^k \to \mathcal{U}$ para algún $k \in \mathbb{N}$.

Diremos que un objeto $u \in \mathcal{U}$ se *construye* a partir de B y $\mathcal{F}$ si:

- $u \in B$, o
- $u = f(u_1, u_2, \ldots, u_k)$ para alguna regla de producción $f : \mathcal{U}^k \to \mathcal{U}$ y algunos objetos $u_1, u_2, \ldots, u_k$ que se construyen a partir de B y $\mathcal{F}$.

Luego podemos definir el *conjunto generado* por B y $\mathcal{F}$ como el conjunto de objetos que se construyen a partir de B y $\mathcal{F}$.

Ejemplo. Veamos algunos ejemplos de objetos con estructura recursiva.

- El conjunto de los números naturales $\mathbb{N}$ se puede generar a partir de $B = \{0\}$ y $\mathcal{F} = \{n \mapsto n + 1\}$.
- El conjunto de las potencias de 2 se puede generar a partir de $B = \{1\}$ y $\mathcal{F} = \{n \mapsto 2n\}$.
- El conjunto de strings binarios se puede generar a partir de $B = \{\epsilon\}$, donde ϵ es el string vacío, y $\mathcal{F} = \{f_0, f_1\}$, donde $f_0(s) = s0$ y $f_1(s) = s1$ donde la operación es concatenación.
- El conjunto de paréntesis balanceados se puede generar a partir de $B = \{\epsilon\}$, donde ϵ es el string vacío, y $\mathcal{F} = \{f_1, f_2\}$, donde $f_1(s) = (s)$ y $f_2(s, t) = st$ donde la operación es concatenación.
- El conjunto de strings binarios palíndromos se puede generar a partir de $B = \{\epsilon\}$ y $\mathcal{F} = \{f_0, f_1\}$, donde $f_0(s) = 0s0$ y $f_1(s) = 1s1$ donde la operación es concatenación.
- El conjunto de triangulaciones de un polígono con n vértices se puede generar a partir de $B = \{\triangle\}$ y $\mathcal{F} = \{f\}$, donde $f(T_1, T_2)$ es la triangulación que se obtiene al pegar las triangulaciones T_1 y T_2 por un lado común.

Como los objetos con estructura recursiva se construyen a partir de objetos básicos y reglas de producción, podemos usar esta estructura para definir funciones sobre estos objetos, o para demostrar propiedades sobre estos objetos. En dichos casos basta con definir la función para los casos bases y luego definir como se comportan bajo las reglas de producción.

Ejemplo. Podemos definir la función *paridad* sobre los naturales como sigue:

- $p(0) = \text{par}$.
- $p(n+1) = \begin{cases} \text{par} & \text{si } p(n) = \text{impar} \\ \text{impar} & \text{si } p(n) = \text{par} \end{cases}$

Definamos la función *largo* sobre strings como sigue:

- $|\epsilon| = 0$.
- Para todo $\sigma \in \Sigma$, $|f_\sigma(s)| = |s| + 1$.

Definamos el *reverso* de un string como sigue:

- $\epsilon^R = \epsilon$.
- Para todo $\sigma \in \Sigma$, $f_\sigma(s)^R = \sigma s^R$.

Definamos la función *número de σ's* sobre strings como sigue:

- $|\epsilon|_\sigma = 0$.
- $|f_\tau(s)|_\sigma = \begin{cases} |s|_\sigma + 1 & \text{si } \tau = \sigma \\ |s|_\sigma & \text{si } \tau \neq \sigma \end{cases}$

Por último, definamos la función *número de triángulos* sobre triangulaciones de un polígono con n vértices como sigue:

- $T(\triangle) = 1$.
- Para todo par de triangulaciones T_1, T_2, tenemos que $T(f(T_1, T_2)) = T(T_1) + T(T_2)$.

2.4.2. Principio de inducción estructural

El principio de inducción estructural establece que para demostrar que una propiedad P sobre objetos recursivos, basta con probarlo para los objetos básicos, y luego probar que P se mantiene bajo las reglas de producción.

> **Definición** (Principio de inducción estructural)**.** Sea S un conjunto generado por objetos básicos B y un conjunto de reglas de producción $\mathcal{F}$. Supongamos $P(s)$ es una propiedad que depende de un objeto $s \in S$ tal que se cumplen las siguientes condiciones:
>
> - $P(b)$ es verdadera para todo $b \in B$ (casos base).
> - Para toda regla de producción $f : \mathcal{U}^k \to \mathcal{U}$ y para todos objetos $s_1, s_2, \ldots, s_k \in S$, si $P(s_i)$ es verdadera para todo $i \in \{1, 2, \ldots, k\}$, entonces $P(f(s_1, s_2, \ldots, s_k))$ también es verdadera (paso inductivo).
>
> Entonces, $P(s)$ es verdadera para todo $s \in S$.

Nuevamente no nos preocuparemos por ahora de la validez del principio de inducción estructural, pero veremos como aplicarlo. En particular, veremos ejemplos en *strings* y en *arboles binarios*.

Strings Sea Σ un alfabeto. Consideraremos el conjunto Σ^* de *strings* sobre Σ, definido por:

- $\epsilon \in \Sigma^*$, donde ϵ es el string vacío.
- Si $\sigma \in \Sigma$ y $s \in \Sigma^*$, entonces $\sigma s \in \Sigma^*$.[2]

Además, consideraremos la *concatenación de strings* $x \cdot y$ definida por $\epsilon \cdot s = s \cdot \epsilon = s$ y $\sigma s \cdot t = \sigma(s \cdot t)$. Notaremos, sin demostrar, que esta operación es asociativa y que ϵ es su elemento neutro. El *reverso* de un string s se denota por s^R y se define por $\epsilon^R = \epsilon$ y $(\sigma s)^R = s^R \sigma$.

Proposición. Para todo $x, y \in \Sigma^*$, $(x \cdot y)^R = y^R \cdot x^R$.

Demostración. Probemos la afirmación por inducción estructural sobre x.

- **Caso base:** $x = \epsilon$. En este caso, $(\epsilon \cdot y)^R = y^R = y^R \cdot \epsilon = y^R \cdot x^R$.
- **Hipótesis inductiva:** Para $s \in \Sigma^*$, $(s \cdot y)^R = y^R \cdot s^R$.
- **Paso inductivo:**

$$\begin{aligned}((\sigma s) \cdot y)^R &= (\sigma(s \cdot y))^R \\ &= (s \cdot y)^R \sigma \\ &= (y^R \cdot s^R)\sigma \\ &= y^R \cdot (s^R \sigma) \\ &= y^R \cdot (\sigma s)^R.\end{aligned}$$

□

Veamos una segunda propiedad sobre strings y su reverso.

Proposición. Para todo $s \in \Sigma^*$, $(s^R)^R = s$.

Demostración. Probemos la afirmación por inducción estructural sobre s.

- **Caso base:** $s = \epsilon$. En este caso, $(\epsilon^R)^R = \epsilon^R = \epsilon$.
- **Hipótesis inductiva:** Para $t \in \Sigma^*$, $(t^R)^R = t$.
- **Paso inductivo:**

$$\begin{aligned}((\sigma t)^R)^R &= (t^R \sigma)^R \\ &= \sigma^R (t^R)^R \\ &= \sigma t.\end{aligned}$$

□

[2]Note que esto simplemente denota la regla de producción.

Árbol binario Definimos el conjunto $\mathcal{T}$ de *árboles binarios* como:

- $\square \in \mathcal{T}$, donde $\square$ es un símbolo que denota un árbol vacío.
- Si $L, R \in \mathcal{T}$, entonces $\bullet(L, R) \in \mathcal{T}$, donde $\bullet(L, R)$ es un árbol con un nodo llamado raíz del que sale un subárbol izquierdo L y subárbol derecho R.

El número de nodos de un árbol T se denota por $N(T)$ y se define por $N(\square) = 0$ y $N(\bullet(L, R)) = 1 + N(L) + N(R)$. El número de arboles vacíos que hay en un árbol T se denota por $E(T)$ y se define por $E(\square) = 1$ y $E(\bullet(L, R)) = E(L) + E(R)$.

Proposición. Para todo árbol $T \in \mathcal{T}$, $E(T) = N(T) + 1$.

Demostración. Probemos la afirmación por inducción estructural sobre T.

- **Caso base:** $T = \square$. En este caso, $E(\square) = 1$ y $N(\square) + 1 = 0 + 1 = 1$, por lo que $E(T) = N(T) + 1$.
- **Hipótesis inductiva:** $L, R \in \mathcal{T}$, nos dice que $E(L) = N(L) + 1$ y $E(R) = N(R) + 1$.
- **Paso inductivo:**

$$\begin{aligned} E(\bullet(L, R)) &= E(L) + E(R) \\ &= (N(L) + 1) + (N(R) + 1) \\ &= 1 + N(L) + N(R) + 1 \\ &= N(\bullet(L, R)) + 1. \end{aligned}$$

□

Consideremos ahora la *altura* de un árbol binario, que es el nodo a mayor distancia de la raíz. Formalmente, la altura de un árbol T se denota por $H(T)$ y se define por $H(\square) = 0$ y $H(\bullet(L, R)) = 1 + \max\{H(L), H(R)\}$.

Proposición. Para todo árbol $T \in \mathcal{T}$, $N(T) \leq 2^{H(T)} - 1$.

Demostración. Probemos la afirmación por inducción estructural sobre T.

- **Caso base:** $T = \square$. En este caso, $N(\square) = 0$ y $2^{H(\square)} - 1 = 2^0 - 1 = 0$, por lo que $N(T) \leq 2^{H(T)} - 1$.
- **Hipótesis inductiva:** $L, R \in \mathcal{T}$, nos dice que $N(L) \leq 2^{H(L)} - 1$ y $N(R) \leq 2^{H(R)} - 1$.
- **Paso inductivo:**

$$\begin{aligned} N(\bullet(L, R)) &= 1 + N(L) + N(R) \\ &\leq 1 + (2^{H(L)} - 1) + (2^{H(R)} - 1) \\ &= 2^{H(L)} + 2^{H(R)} - 1 \\ &\leq 2^{\max\{H(L),H(R)\}+1} - 1 \\ &= 2^{H(\bullet(L,R))} - 1. \end{aligned}$$

□

2.4.3. Equivalencia con inducción sobre los números naturales

Notemos que el principio de inducción estructural es una generalización del principio de inducción, pues al tomar $B = \{0\}$ y $\mathcal{F} = \{n \mapsto n + 1\}$ recuperamos el principio de inducción débil. De manera interesante, la conversa también es cierta, es decir, el principio de inducción estructural se sigue del principio de inducción débil, por lo que ambos principios son equivalentes entre sí.

Teorema. El principio de inducción estructural es equivalente al principio de inducción.

Demostración. Ya discutimos que el principio de inducción estructural es una generalización del principio de inducción, por lo que sólo tenemos que mostrar que el principio de inducción estructural se sigue del principio de inducción.

Sea S un conjunto generado por objetos básicos B y un conjunto de reglas de producción $\mathcal{F}$. Supongamos $P(s)$ es una propiedad que depende de un objeto $s \in S$ tal que se cumplen las siguientes condiciones:

(H1) $P(b)$ es verdadera para todo $b \in B$ (casos base).

(H2) Para toda regla de producción $f : \mathcal{U}^k \to \mathcal{U}$ y para todos objetos $s_1, s_2, \ldots, s_k \in S$, si $P(s_i)$ es verdadera para todo $i \in \{1, 2, \ldots, k\}$, entonces $P(f(s_1, s_2, \ldots, s_k))$ también es verdadera (paso inductivo).

Definamos $Q(n)$ como la afirmación de que $P(s)$ es verdadera para todo $s \in S$ que se construye a partir de B y $\mathcal{F}$ usando n reglas de producción. Probemos que $Q(n)$ es verdadera para todo $n \in \mathbb{N}$ por inducción fuerte sobre n.

- **Caso base:** $Q(0)$ es verdadera, ya que $P(b)$ es verdadera para todo $b \in B$ por (H1).
- **Hipótesis inductiva:** $Q(k)$ es verdadera para todo $k \leq n$ con $n \in \mathbb{N}$.
- **Paso inductivo:** Probemos que $Q(n + 1)$ es verdadera. Sea $s \in S$ que se construye a partir de B y $\mathcal{F}$ usando $n+1$ reglas de producción. Entonces, $s = f(s_1, s_2, \ldots, s_k)$ para alguna regla de producción $f : \mathcal{U}^k \to \mathcal{U}$ y algunos objetos $s_1, s_2, \ldots, s_k \in S$ que se construyen a partir de B y $\mathcal{F}$ usando a lo más n reglas de producción. Por la hipótesis inductiva, $P(s_i)$ es verdadera para todo $i \in \{1, 2, \ldots, k\}$ y por (H2), $P(s) = P(f(s_1, s_2, \ldots, s_k))$ también es verdadera, por lo que $Q(n + 1)$ es verdadera.

Como todo $s \in S$ se construye a partir de B y $\mathcal{F}$ usando n reglas de producción para algún $n \in \mathbb{N}$, concluimos que $P(s)$ es verdadera para todo $s \in S$. □

Capítulo 3

Relaciones

3.1. Relaciones binarias

Tal como en lógica de primer orden estudiamos predicados para describir propiedades y relaciones entre objetos, en matemáticas discretas muchas veces nos interesará simplemente estudiar el conjunto de pares que están relacionados. Esta idea aparece de manera natural en bases de datos, grafos, sistemas de recomendación, especificación formal de programas y la teoría de la computación, entre otras áreas. La noción matemática que captura esto es la de *relación binaria*.

Definición. Una *relación (binaria)* R entre dos conjuntos A y B es un subconjunto de $A \times B$. Es decir, cada elemento de R es un par ordenado (a, b) con $a \in A$ y $b \in B$.

Si $(a, b) \in R$, escribiremos simplemente $a \; R \; b$. En otras palabras, $a \; R \; b$ es una manera abreviada de decir que a y b están relacionados por R.

Ejemplo. Sean $A = \{1, 2, 3\}$ y $B = \{a, b\}$. Entonces

$$R = \{(1, a), (1, b), (3, b)\}$$

es una relación binaria entre A y B. En este caso, se cumple que $1 \; R \; a$, $1 \; R \; b$ y $3 \; R \; b$, mientras que $2 \; R \; a$ y $2 \; R \; b$ no se cumplen. El hecho de que *no* se cumpla una relación también es información relevante, y a menudo se denota escribiendo, por ejemplo, $a \not R \; b$.

3.1.1. Relaciones sobre un conjunto

El caso más importante para nosotros será cuando una relación conecta elementos de un conjunto consigo mismo. En ese escenario, podemos preguntarnos si la relación satisface ciertas propiedades estructurales.

Definición. Una relación binaria R sobre un conjunto A se dirá:

- *reflexiva* si $a \; R \; a$ para todo $a \in A$.
- *simétrica* si $a \; R \; b$ implica $b \; R \; a$ para todo $a, b \in A$.
- *antisimétrica* si $a \; R \; b$ y $b \; R \; a$ implican $a = b$ para todo $a, b \in A$.

- *transitiva* si $a\ R\ b$ y $b\ R\ c$ implican $a\ R\ c$ para todo $a, b, c \in A$.

En el dígrafo asociado a una relación sobre A, estas propiedades tienen una interpretación muy concreta. La reflexividad significa que cada vértice tiene un lazo hacia sí mismo. La simetría significa que cada vez que hay una flecha de a a b, también hay una flecha de b a a. La antisimetría significa que no puede haber flechas en ambos sentidos entre dos vértices distintos. Finalmente, la transitividad significa que siempre que haya un camino dirigido de largo 2 desde a hasta c, pasando por algún vértice intermedio b, entonces también debe existir la flecha directa de a hacia c.

Veamos algunos ejemplos de relaciones con estas propiedades.

Ejemplo. Consideremos las siguientes relaciones sobre $\mathbb{Z}$.

- Sea R_1 dada por $a\ R_1\ b$ si y sólo si $a \leq b$. Esta relación es reflexiva, antisimétrica y transitiva, pero no es simétrica.
- Sea R_2 dada por $a\ R_2\ b$ si y sólo si $a-b$ es par. Esta relación es reflexiva, simétrica y transitiva, pero no es antisimétrica.
- Sea R_3 dada por $a\ R_3\ b$ si y sólo si $a+b \leq 3$. Esta relación es simétrica, pero no es reflexiva ni transitiva.

3.1.2. Representación de relaciones

Cuando los conjuntos involucrados son finitos, una relación puede representarse simplemente listando sus pares, pero también mediante una tabla. La idea es poner los elementos de A en las filas, los de B en las columnas, y marcar aquellas posiciones correspondientes a pares que pertenecen a la relación.

Por ejemplo, la relación del ejemplo anterior puede escribirse como:

R	a	b
1	✓	✓
2		
3		✓

Esta representación es particularmente útil cuando pensamos en relaciones como estructuras de datos finitas, por ejemplo matrices de adyacencia o tablas de una base de datos.

También existen dos representaciones gráficas muy naturales de una relación.

Definición. Sea $R \subseteq A \times B$ una relación binaria entre dos conjuntos. El *grafo bipartito* asociado a R es el grafo cuyos vértices se dividen en dos partes, una correspondiente a los elementos de A y la otra a los elementos de B, y en el que hay una arista entre $a \in A$ y $b \in B$ si y sólo si $a\ R\ b$.

Definición. Sea $R \subseteq A \times A$ una relación binaria sobre un conjunto A. El *dígrafo* asociado a R es el grafo dirigido cuyos vértices son los elementos de A y en el que hay un arco dirigido desde a hacia b si y sólo si $a\ R\ b$.

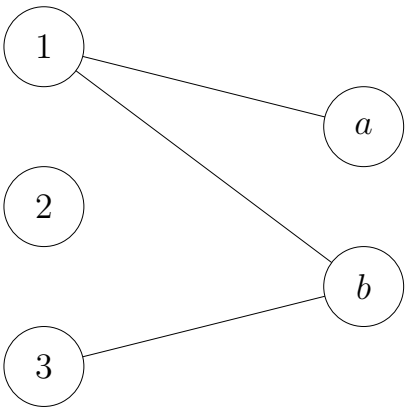


Figura 3.1: Representación de la relación del inicio $R \subseteq A \times A$ como grafo bipartito.

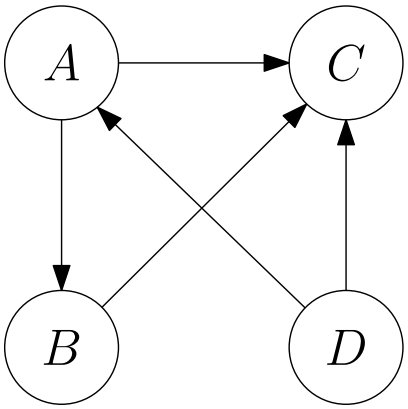


Figura 3.2: Representación de la relación x sigue a y en el conjunto $\{\text{Alicia}, \text{Bob}, \text{Carla}, \text{Darío}\}$.

(Ver Figuras 3.1 y 3.2.)

Estas representaciones son especialmente útiles en ciencias de la computación. Por ejemplo, un grafo bipartito puede modelar usuarios y recursos, o estudiantes y cursos, mientras que un dígrafo puede modelar transiciones de estado, dependencias entre tareas o enlaces entre páginas web.

3.1.3. Relaciones de orden y equivalencia

Las relaciones de orden nos permiten formalizar la idea de comparar elementos. Esto aparece, por ejemplo, cuando queremos hablar de precedencia entre tareas, inclusión entre conjuntos, divisibilidad o jerarquías entre objetos.

> **Definición.** Una relación binaria R sobre un conjunto A es un *orden parcial* si es reflexiva, antisimétrica y transitiva. Si además para todo $a, b \in A$ se cumple que $a \ R \ b$ o $b \ R \ a$, entonces diremos que R es un *orden total*.

Ejemplo. Algunos ejemplos importantes de relaciones de orden son los siguientes.

- La relación $\leq$ es un orden total sobre $\mathbb{N}$.
- La relación $\subseteq$ es un orden parcial sobre 2^X para cualquier conjunto X. Si X tiene al menos dos elementos, entonces no es un orden total.

- La relación de divisibilidad sobre $\mathbb{N} \setminus \{0\}$, definida por $a \; R \; b$ si y sólo si $a \mid b$, es un orden parcial. No es total, pues por ejemplo ni $2 \mid 3$ ni $3 \mid 2$.
- Si fijamos un alfabeto Σ con un orden total sobre sus símbolos, entonces el orden lexicográfico sobre Σ^* es un orden total. Este es el orden que se usa, por ejemplo, al ordenar palabras en un diccionario o strings en muchas estructuras de datos.
- La relación de prefijo sobre Σ^*, definida por $s \; R \; t$ si y sólo si s es prefijo de t, es un orden parcial. Es reflexiva, antisimétrica y transitiva, pero no es total, ya que por ejemplo 01 y 10 no son prefijo uno del otro.

En muchas situaciones no queremos comparar elementos, sino agruparlos en clases de objetos que consideraremos "iguales" bajo cierto criterio. La noción que formaliza esta idea es la de relación de equivalencia.

Definición. Una relación binaria $R \subseteq A \times A$ es una *relación de equivalencia* si es reflexiva, simétrica y transitiva.

Ejemplo. Fijado un entero $p \geq 2$, podemos definir una relación sobre $\mathbb{Z}$ por

$$a \equiv b \pmod{p} \quad \text{si y sólo si} \quad p \mid (a - b).$$

Esta es una relación de equivalencia. En particular, cuando $p = 3$ escribiremos $a \equiv_3 b$ para indicar que a y b tienen el mismo residuo módulo 3.

Ejemplo. También aparecen relaciones de equivalencia de manera natural al trabajar con strings.

- Sobre Σ^*, la relación definida por $s \; R \; t$ si y sólo si $|s| = |t|$ es una relación de equivalencia. Sus clases de equivalencia agrupan strings según su largo.
- Sobre el conjunto de strings decimales que representan enteros no negativos, podemos definir $s \; R \; t$ si y sólo si s y t representan el mismo número al ignorar ceros a la izquierda. Por ejemplo, 0012, 012 y 12 quedan en la misma clase de equivalencia.

3.1.4. Particiones

Una relación de equivalencia nos permite descomponer un conjunto en bloques de elementos equivalentes entre sí. Cada uno de estos bloques corresponde a una clase de equivalencia.

Definición. Dada una relación de equivalencia R sobre un conjunto A y un elemento $a \in A$, la *clase de equivalencia* de a es el conjunto

$$[a]_R = \{b \in A : a \; R \; b\}.$$

Ejemplo. Consideremos la relación de congruencia módulo 3 sobre $\mathbb{Z}$. Sus clases de equivalencia son:

$$[0]_{\equiv_3} = \{\ldots, -6, -3, 0, 3, 6, \ldots\},$$
$$[1]_{\equiv_3} = \{\ldots, -5, -2, 1, 4, 7, \ldots\},$$
$$[2]_{\equiv_3} = \{\ldots, -4, -1, 2, 5, 8, \ldots\}.$$

Proposición. Sea R una relación de equivalencia sobre un conjunto A. Para cualesquiera $a, b \in A$, son equivalentes las siguientes afirmaciones:

- $a \; R \; b$.
- $[a]_R = [b]_R$.
- $[a]_R \cap [b]_R \neq \emptyset$.

Demostración. Demostraremos las implicancias en ciclo.

($1 \Rightarrow 2$) Supongamos que $a \; R \; b$. Veamos que $[a]_R \subseteq [b]_R$. Si $x \in [a]_R$, entonces $a \; R \; x$. Como $a \; R \; b$ y R es simétrica, tenemos $b \; R \; a$. Luego, por transitividad, de $b \; R \; a$ y $a \; R \; x$ se sigue que $b \; R \; x$. Por lo tanto, $x \in [b]_R$.

El mismo argumento, intercambiando los papeles de a y b, muestra que $[b]_R \subseteq [a]_R$. Concluimos que $[a]_R = [b]_R$.

($2 \Rightarrow 3$) Si $[a]_R = [b]_R$, entonces en particular $a \in [a]_R$ por reflexividad. Como $[a]_R = [b]_R$, se tiene también $a \in [b]_R$. Por lo tanto, $a \in [a]_R \cap [b]_R$, y la intersección es no vacía.

($3 \Rightarrow 1$) Supongamos que $[a]_R \cap [b]_R \neq \emptyset$. Entonces existe $x \in A$ tal que $x \in [a]_R$ y $x \in [b]_R$. Esto significa que $a \; R \; x$ y $b \; R \; x$. Como R es simétrica, de $b \; R \; x$ obtenemos $x \; R \; b$. Luego, por transitividad, de $a \; R \; x$ y $x \; R \; b$ concluimos que $a \; R \; b$. □

3.1.5. Particiones

Una partición de un conjunto corresponde a dividir sus elementos en bloques disjuntos que cubren todo el conjunto. La relación entre particiones y relaciones de equivalencia es una de las ideas centrales de esta clase.

Definición. Una familia $\mathcal{P}$ de subconjuntos de un conjunto A es una *partición* de A si se cumplen las siguientes condiciones:

- Para todo $P \in \mathcal{P}$, se tiene $P \neq \emptyset$.
- Para todo $P, Q \in \mathcal{P}$, si $P \neq Q$, entonces $P \cap Q = \emptyset$.
- Para todo $a \in A$, existe $P \in \mathcal{P}$ tal que $a \in P$.

Las particiones aparecen naturalmente en las equivalencias: cada relación de equivalencia induce una partición cuyas partes son las clases de equivalencia.

Teorema. Dada una relación de equivalencia R sobre un conjunto A, el conjunto de clases de equivalencia

$$A/R = \{[a]_R : a \in A\}$$

es una partición de A. Dicho conjunto se llama el *conjunto cociente* de A por R.

Demostración. Debemos verificar las tres propiedades de una partición.

- Cada clase es no vacía. En efecto, si $a \in A$, entonces $a \; R \; a$ por reflexividad, y por lo tanto $a \in [a]_R$.
- Dos clases cualesquiera son iguales o disjuntas. En efecto, si $[a]_R \cap [b]_R \neq \emptyset$, entonces por la proposición anterior se tiene $a \; R \; b$, y nuevamente por la misma proposición se concluye que $[a]_R = [b]_R$.
- La unión de todas las clases es A. En efecto, si $a \in A$, entonces $a \in [a]_R$, por lo que a pertenece a alguna clase de equivalencia.

Concluimos que $\{[a]_R : a \in A\}$ es una partición de A. □

Por ejemplo, para la congruencia módulo 3 obtenemos:

$$\mathbb{Z}/\equiv_3 = \{[0]_{\equiv_3}, [1]_{\equiv_3}, [2]_{\equiv_3}\}.$$

De manera interesante, el proceso inverso también es cierto: cada partición de un conjunto induce una relación de equivalencia cuyas clases de equivalencia son exactamente los bloques de la partición.

Teorema. Dada una partición $\mathcal{P}$ de un conjunto A, la relación $R_\mathcal{P}$ definida por

$$a \; R_\mathcal{P} \; b \quad \text{si y sólo si} \quad \exists P \in \mathcal{P} \text{ tal que } a, b \in P$$

es una relación de equivalencia. Más aún, sus clases de equivalencia son exactamente los elementos de $\mathcal{P}$.

Demostración. Primero veremos que $R_\mathcal{P}$ es una relación de equivalencia.

- **Reflexividad.** Sea $a \in A$. Como $\mathcal{P}$ es una partición, existe $P \in \mathcal{P}$ tal que $a \in P$. Luego, $a \; R_\mathcal{P} \; a$.
- **Simetría.** Si $a \; R_\mathcal{P} \; b$, entonces existen $P \in \mathcal{P}$ tal que $a, b \in P$. Por lo tanto, también $b, a \in P$, y así $b \; R_\mathcal{P} \; a$.
- **Transitividad.** Supongamos que $a \; R_\mathcal{P} \; b$ y $b \; R_\mathcal{P} \; c$. Entonces existen $P, Q \in \mathcal{P}$ tales que $a, b \in P$ y $b, c \in Q$. Como $b \in P \cap Q$ y los elementos de una partición son disjuntos dos a dos, debe ser $P = Q$. En consecuencia, $a, c \in P$, y por lo tanto $a \; R_\mathcal{P} \; c$.

Veamos ahora que las clases de equivalencia de $R_\mathcal{P}$ son exactamente los elementos de $\mathcal{P}$. Sea $P \in \mathcal{P}$ y sea $a \in P$. Afirmamos que $[a]_{R_\mathcal{P}} = P$.

Si $x \in [a]_{R_\mathcal{P}}$, entonces $x \; R_\mathcal{P} \; a$, así que existe $Q \in \mathcal{P}$ tal que $x, a \in Q$. Como también $a \in P$, tenemos $a \in P \cap Q$, luego $P = Q$. Por lo tanto, $x \in P$, y así $[a]_{R_\mathcal{P}} \subseteq P$.

Recíprocamente, si $x \in P$, entonces x y a pertenecen al mismo elemento de la partición, por lo que $x \; R_\mathcal{P} \; a$. En consecuencia, $x \in [a]_{R_\mathcal{P}}$, y obtenemos $P \subseteq [a]_{R_\mathcal{P}}$.

Concluimos que $[a]_{R_\mathcal{P}} = P$. □

Ejemplo. Consideremos el conjunto $\{0,1\}^*$ de todos los strings binarios. Para cada $n \in \mathbb{N}$, definamos

$$L_n = \{w \in \{0,1\}^* : |w| = n\}.$$

Entonces la familia

$$\mathcal{P} = \{L_n : n \in \mathbb{N}\}$$

es una partición de $\{0,1\}^*$: cada string tiene un largo bien definido, y dos strings con largos distintos no pueden pertenecer al mismo bloque.

La relación de equivalencia inducida por esta partición es, de manera natural, la relación

$$s \; R_{\mathcal{P}} \; t \quad \text{si y sólo si} \quad |s| = |t|.$$

En efecto, dos strings están relacionados exactamente cuando pertenecen al mismo bloque L_n, es decir, cuando tienen el mismo largo.

Por ejemplo,

$$\varepsilon \in L_0, \qquad 0, 1 \in L_1, \qquad 00, 01, 10, 11 \in L_2.$$

Así, $01 \; R_{\mathcal{P}} \; 10$ porque ambos tienen largo 2, mientras que $1 \not R_{\mathcal{P}} \; 01$ porque sus largos son distintos.

Las clases de equivalencia de $R_{\mathcal{P}}$ son precisamente los conjuntos L_n.

3.2. Relaciones de otras aridades

Hasta ahora hemos trabajado únicamente con relaciones binarias, es decir, relaciones entre pares de elementos. Sin embargo, muchas veces nos interesa relacionar tres o más objetos a la vez. Esto nos lleva a la noción general de relación k-aria.

Definición. Una *relación k-aria* entre conjuntos $A_1, A_2, \ldots, A_k$ es un subconjunto de

$$A_1 \times A_2 \times \cdots \times A_k.$$

En el caso particular en que $A_1 = A_2 = \cdots = A_k = A$, diremos que es una relación k-aria sobre A.

Ejemplo. Algunos ejemplos de relaciones de otras aridades son los siguientes.

- La propiedad de ser primo puede verse como una relación unaria sobre $\mathbb{N}$, dada por el conjunto de todos los números primos.
- La suma puede verse como una relación ternaria sobre $\mathbb{N}$, definida por

$$S = \{(a, b, c) \in \mathbb{N}^3 : a + b = c\}.$$

- En una base de datos, una tabla que registra triples de la forma (estudiante, curso, semestre) puede verse como una relación ternaria entre esos tres dominios.
- En procesamiento de strings, una relación ternaria puede modelar la concatenación:

$$C = \{(x, y, z) \in (\Sigma^*)^3 : xy = z\}.$$

  Es decir, $(x, y, z) \in C$ cuando concatenar x con y produce exactamente el string z.

Ejercicios

1. En una plataforma de desarrollo de software consideremos los siguientes conjuntos:

   - D: el conjunto de desarrolladores,
   - M: el conjunto de módulos del sistema,
   - T: el conjunto de tareas de compilación,
   - S: el conjunto de archivos fuente.

   Definimos las siguientes relaciones:

   - Para $d \in D$ y $m \in M$, escribimos $d \; A \; m$ si el desarrollador d tiene permiso de escritura sobre el módulo m.
   - Para $t_1, t_2 \in T$, escribimos $t_1 \preceq t_2$ si completar t_1 es prerrequisito para completar t_2, o bien $t_1 = t_2$.
   - Para $s_1, s_2 \in S$, escribimos $s_1 \sim s_2$ si al eliminar espacios en blanco y comentarios de ambos archivos, los resultados coinciden exactamente.

   a) Indique, para cada una de las tres relaciones, si se trata de una relación entre dos conjuntos distintos o de una relación sobre un solo conjunto.

   b) Para cada relación, indique cuál de las representaciones vistas en clases es la más natural: tabla, grafo bipartito o dígrafo. Justifique brevemente.

   c) Suponiendo que el sistema de compilación es consistente, ¿qué propiedades de relaciones esperaría que satisfaga $\preceq$? Concluya qué tipo de relación es.

   d) Explique por qué $\sim$ modela una noción de equivalencia entre archivos fuente y describa qué representan sus clases de equivalencia. Mencione además una aplicación computacional donde estas clases puedan ser útiles.

2. Definimos la relación $\approx$ sobre $\mathbb{N} \times \mathbb{N}$ definida por

$$(a, b) \approx (c, d) \quad \text{si y sólo si} \quad a + d = b + c.$$

   a) Demuestre que $\approx$ es una relación de equivalencia.

   b) Demuestre que toda clase de equivalencia contiene al menos un elemento de la forma $(k, 0)$ o $(0, k)$ con $k \in \mathbb{N}$.

   c) Demuestre que ninguna clase de equivalencia puede contener dos elementos distintos de esa forma.

   d) Concluya que el conjunto cociente $(\mathbb{N} \times \mathbb{N})/\approx$ puede identificarse naturalmente con $\mathbb{Z}$.

3. Sea $\mathcal{F}$ el conjunto formulas proposicionales válidas. Definimos la relación $\sim$ sobre $\mathcal{F}$ por

$$\phi \sim \psi \quad \text{si y sólo si} \quad \phi \leftrightarrow \psi \text{ es una tautología.}$$

   a) Demuestre que $\sim$ es una relación de equivalencia.

b) Definimos ahora una relación $\preceq$ sobre $\mathcal{F}/\sim$ por

$$[\phi]_\sim \preceq [\psi]_\sim \quad \text{si y sólo si} \quad (\phi \wedge \psi) \leftrightarrow \psi.$$

Demuestre que $\preceq$ está bien definida, es decir, que no depende de los representantes elegidos de cada clase de equivalencia.

c) Demuestre que $\preceq$ es un orden sobre $\mathcal{F}/\sim$.

d) ¿Es $\preceq$ un orden total? Justifique su respuesta.

4. Sea A un conjunto finito y $\preceq$ un orden en A. Decimos que x es un *predecesor inmediato* de y si $x \preceq y$, $x \neq y$, y no existe $z \in A$ tal que $x \preceq z \preceq y$ con $z \neq x, y$. El *diagrama de Hasse* de $(A, \preceq)$ es el grafo dirigido cuyos vértices son los elementos de A y dónde hay un arco dirigido de x hacia y si y sólo si x es un predecesor inmediato de y.

a) Demuestre que en cualquier conjunto finito parcialmente ordenado, la relación $\preceq$ puede recuperarse a partir del diagrama de Hasse: más precisamente, pruebe que $x \preceq y$ si y sólo si existe un camino dirigido de x a y en el diagrama de Hasse, o bien $x = y$.

Indicación: Un camino dirigido de x a y es una secuencia en A: $x = a_1, a_2, \ldots, a_k = y$ con $a_1 \to a_2 \to \cdots \to a_k$ en el diagrama de Hasse.

b) ¿Puede un diagrama de Hasse contener ciclos dirigidos de largo $\ell \geq 2$? Justifique su respuesta.

Indicación: Un ciclo dirigido de largo ℓ es una secuencia en A: $a_1, \ldots, a_\ell$ con $a_1 \to a_2 \to \cdots \to a_\ell \to a_1$ en el diagrama de Hasse.

Ahora sea S_n el conjunto de permutaciones de $\{1, \ldots, n\}$; es decir, S_n son todos los strings en $\{1, \ldots, n\}^*$ que contienen exactamente una vez cada símbolo $1, 2, \ldots, n$. Para $\pi = \pi_1 \pi_2 \cdots \pi_n \in S_n$, definimos su *conjunto de inversiones* por

$$\mathrm{Inv}(\pi) = \{(i, j) : 1 \leq i < j \leq n \text{ y } \pi_i > \pi_j\}.$$

Definimos una relación sobre S_n por

$$\pi \preceq \sigma \quad \text{si y sólo si} \quad \mathrm{Inv}(\pi) \subseteq \mathrm{Inv}(\sigma).$$

A esta relación se le llama el *orden débil* sobre S_n.

c) Demuestre que el orden débil es efectivamente un orden parcial sobre S_n. ¿Es el orden débil un orden total?

d) Dibuje el diagrama de Hasse de $(S_3, \preceq)$.

e) Un *mínimo* de un orden parcial es un elemento x tal que no existe y con $y \prec x$. Un *máximo* es un elemento x tal que no existe y con $x \prec y$. Determine cuántos mínimos y máximos tiene el orden débil sobre S_n.

f) Sean x e y dos permutaciones de S_n tal que x es un prédecesor inmediato de y. ¿Qué relación existe entre x e y?

3.3. Operaciones sobre relaciones

3.3.1. Operaciones clásicas

Una de las ventajas de pensar una relación como un subconjunto de un producto cartesiano es que podemos combinar relaciones y construir nuevas a partir de otras ya conocidas. Esto es útil, por ejemplo, cuando modelamos dependencias entre tareas, permisos entre usuarios y recursos, o llamadas entre servicios de un sistema.

Definición (Operaciones clásicas)**.** Sean $R, S \subseteq A \times B$ relaciones entre los conjuntos A y B. Definimos:

- la *unión* $R \cup S$ como el conjunto de pares que pertenecen a R, a S o a ambas;
- la *intersección* $R \cap S$ como el conjunto de pares que pertenecen simultáneamente a R y a S;
- la *diferencia* $R \setminus S$ como el conjunto de pares que pertenecen a R pero no a S.

Ejemplo. Sean R y S las siguientes relaciones entre $A = \{1, 2, 3\}$ y $B = \{a, b\}$:

$$R = \{(1, a), (2, a), (3, b)\}, \qquad S = \{(1, a), (2, b)\}.$$

Entonces

$$\begin{aligned} R \cup S &= \{(1, a), (2, a), (2, b), (3, b)\}, \\ R \cap S &= \{(1, a)\}, \\ R \setminus S &= \{(2, a), (3, b)\}, \\ S \setminus R &= \{(2, b)\}. \end{aligned}$$

3.3.2. Inversión y composición

Definición. Sea $R \subseteq A \times B$ una relación. La *relación inversa* de R es

$$R^{-1} = \{(b, a) \in B \times A : (a, b) \in R\}.$$

Si además $S \subseteq B \times C$ es otra relación, entonces la *composición* de R y S es la relación

$$S \circ R = \{(a, c) \in A \times C : \exists b \in B \text{ tal que } (a, b) \in R \text{ y } (b, c) \in S\}.$$

Ejemplo. Consideremos los conjuntos

$$A = \{\text{Web}, \text{Móvil}\}, \quad B = \{\text{Auth}, \text{Pagos}\}, \quad C = \{\text{Usuarios}, \text{Transacciones}\}.$$

Sea $R \subseteq A \times B$ la relación “cliente usa servicio” dada por

$$R = \{(\text{Web}, \text{Auth}), (\text{Web}, \text{Pagos}), (\text{Móvil}, \text{Auth})\}$$

y sea $S \subseteq B \times C$ la relación "servicio usa base de datos" dada por

$$S = \{(\text{Auth}, \text{Usuarios}), (\text{Pagos}, \text{Transacciones})\}.$$

Entonces

$$S \circ R = \{(\text{Web}, \text{Usuarios}), (\text{Web}, \text{Transacciones}), (\text{Móvil}, \text{Usuarios})\}.$$

Es decir, la composición modela una relación indirecta: qué base de datos termina usando cada cliente a través de un servicio intermedio.

Cuando una relación se representa mediante un dígrafo, la composición puede interpretarse como seguir dos flechas consecutivas. Si existe una flecha de a a b en R y una flecha de b a c en S, entonces aparece una conexión indirecta de a a c en $S \circ R$.

Similarmente, cuando una relación se representa mediante un grafo bipartito, la composición puede interpretarse como seguir una flecha del primer conjunto al segundo, y luego otra flecha del segundo conjunto a un tercer conjunto.

3.3.3. Potencias de una relación

Definición. Sea R una relación sobre un conjunto A. Definimos sus *potencias* inductivamente por

$$R^1 = R, \qquad R^{n+1} = R^n \circ R \quad \text{para todo } n \geq 1.$$

En el dígrafo asociado a una relación sobre A, la relación R^n describe los pares (a, b) para los cuales existe un camino dirigido de largo exactamente n desde a hasta b.

Ejemplo. Sea D la relación sobre

$$A = \{\text{Gateway}, \text{Auth}, \text{Billing}, \text{DB}\}$$

dada por

$$D = \{(\text{Gateway}, \text{Auth}), (\text{Gateway}, \text{Billing}), (\text{Auth}, \text{DB}), (\text{Billing}, \text{DB})\}.$$

Si pensamos que $x\ D\ y$ significa "x llama directamente a y", entonces

$$D^2 = \{(\text{Gateway}, \text{DB})\}$$

porque desde `Gateway` se puede llegar a `DB` en exactamente dos pasos. En cambio,

$$D^3 = \emptyset.$$

3.4. Clausuras

En muchas situaciones una relación inicial no satisface alguna propiedad que nos interesa, pero sí queremos extenderla de la manera más económica posible para que la satisfaga. Esta idea da origen a las clausuras.

Definición. Sea R una relación sobre un conjunto A. La *clausura reflexiva* (resp. *simétrica*, *transitiva*, *reflexiva-transitiva*) de R es una relación T tal que:

- T es reflexiva (resp. simétrica, transitiva, reflexiva-transitiva);
- $R \subseteq T$;
- si T' es otra relación reflexiva (resp. simétrica, transitiva, reflexiva-transitiva) con $R \subseteq T'$, entonces $T \subseteq T'$.

En otras palabras, la clausura reflexiva (resp. simétrica, transitiva, reflexiva-transitiva) de R es la relación reflexiva (resp. simétrica, transitiva, reflexiva-transitiva) más pequeña que contiene a R.

Denotaremos las clausuras reflexiva, simétrica, transitiva y reflexiva-transitiva de R por R^{ref}, R^{sym}, R^+ y R^*, respectivamente.

Teorema. Sea R una relación sobre un conjunto A y sea

$$\Delta_A = \{(a, a) : a \in A\}.$$

Entonces:

- la clausura reflexiva de R es $R^{\text{ref}} = R \cup \Delta_A$;
- la clausura simétrica de R es $R^{\text{sym}} = R \cup R^{-1}$;
- la clausura transitiva de R está dada por
$$R^+ = \bigcup_{n \geq 1} R^n;$$
- la clausura reflexiva-transitiva de R se denota por R^* y está dada por
$$R^* = \Delta_A \cup R^+.$$

Demostración. Verifiquemos cada afirmación.

- Debemos verificar que se cumplen las tres condiciones de la definición de clausura reflexiva.
  - $R \cup \Delta_A$ es reflexiva pues para todo $a \in A$ se tiene $(a, a) \in \Delta_A \subseteq R \cup \Delta_A$.
  - $R \subseteq R \cup \Delta_A$ se cumple por definición de unión.
  - Sea T una relación reflexiva con $R \subseteq T$. Como es reflexiva, $\Delta_A \subseteq T$, combinando esto con $R \subseteq T$ se obtiene $R \cup \Delta_A \subseteq T$.
- Para la clausura simétrica, procedemos de la misma manera.
  - $R \cup R^{-1}$ es simétrica pues para todo $(a, b) \in R$, se tiene $(b, a) \in R^{-1}$, y por lo tanto $(a, b) \in R \cup R^{-1}$.

  - $R \subseteq R \cup R^{-1}$ se cumple por definición de unión.
  - Sea T una relación simétrica con $R \subseteq T$. Como es simétrica, $R^{-1} \subseteq T$, combinando esto con $R \subseteq T$ se obtiene $R \cup R^{-1} \subseteq T$.

- Nuevamente, verificamos las tres condiciones de la definición de clausura transitiva.
  - $\bigcup_{n\geq 1} R^n$ es transitiva pues si $(a,b) \in R^m$ y $(b,c) \in R^n$ para algunos $m, n \geq 1$, entonces $(a,c) \in R^{m+n}$, y por lo tanto $(a,c) \in \bigcup_{n\geq 1} R^n$.
  - $R \subseteq \bigcup_{n\geq 1} R^n$ se cumple porque $R = R^1$.
  - Sea T una relación transitiva con $R \subseteq T$. Mostremos por inducción que $R^n \subseteq T$ para todo $n \geq 1$. Para $n = 1$ se cumple $R^1 = R \subseteq T$ por hipótesis. Supongamos que $R^n \subseteq T$ para algún $n \geq 1$. Si $(a,b) \in R^{n+1} = R^n \circ R$, entonces existe $c \in A$ tal que $(a,c) \in R^n$ y $(c,b) \in R$. Por la hipótesis de inducción, $(a,c) \in T$, y por la hipótesis de que $R \subseteq T$, se tiene $(c,b) \in T$. Como T es transitiva, se concluye que $(a,b) \in T$. Por lo tanto, $R^{n+1} \subseteq T$. Por inducción, $R^n \subseteq T$ para todo $n \geq 1$, y así $\bigcup_{n\geq 1} R^n \subseteq T$.

- Finalmente, para la clausura reflexiva-transitiva, basta verificar que $\Delta_A \cup R^+$ cumple las tres condiciones de la definición de clausura reflexiva-transitiva.
  - $\Delta_A \cup R^+$ es reflexiva-transitiva pues es la unión de una relación reflexiva y una relación transitiva.
  - $R \subseteq \Delta_A \cup R^+$ se cumple porque $R = R^1 \subseteq R^+$.
  - Sea T una relación reflexiva-transitiva con $R \subseteq T$. Como es reflexiva, $\Delta_A \subseteq T$, y como es transitiva, $R^+ \subseteq T$, por lo que $\Delta_A \cup R^+ \subseteq T$. □

Ejemplo. Consideremos el conjunto de tareas

$$T = \{\text{Lexer}, \text{Parser}, \text{Checker}, \text{CodeGen}\}$$

y la relación P sobre T definida por

$$P = \{(\text{Lexer}, \text{Parser}), (\text{Parser}, \text{Checker}), (\text{Checker}, \text{CodeGen})\}.$$

Si $x \ P \ y$ significa "x debe completarse antes que y", entonces P contiene sólo las dependencias directas.

La clausura transitiva P^+ agrega además las dependencias indirectas:

$$(\text{Lexer}, \text{Checker}), \quad (\text{Parser}, \text{CodeGen}), \quad (\text{Lexer}, \text{CodeGen}).$$

Por otra parte, la clausura reflexiva-transitiva P^* agrega también los pares (x,x), que representan el hecho de que toda tarea es alcanzable desde sí misma mediante un camino de largo 0.

Proposición. Sea R una relación sobre un conjunto finito A con $|A| = n$. Entonces

$$R^+ = \bigcup_{1\leq i\leq n} R^i.$$

Demostración. La inclusión

$$\bigcup_{1\leq i\leq n} R^i \subseteq R^+$$

es inmediata por definición de R^+.

Para la otra inclusión, supongamos que $(a,b) \in R^+$. Entonces existe un camino dirigido desde a hasta b siguiendo flechas de R, de modo que $(a,b) \in R^m$ para algún $m \geq 1$.

Si este camino repite un vértice, podemos eliminar el ciclo comprendido entre las dos apariciones de ese vértice y obtener un camino más corto entre los mismos extremos. Repitiendo este proceso, obtenemos un camino sin repeticiones. Un camino sin repeticiones en un conjunto con n vértices tiene largo a lo más n. Por lo tanto, $(a,b) \in R^i$ para algún i con $1 \leq i \leq n$, y así

$$(a,b) \in \bigcup_{1\leq i\leq n} R^i.$$

Concluimos que

$$R^+ \subseteq \bigcup_{1\leq i\leq n} R^i.$$ □

3.5. Funciones

Las funciones forman una clase especialmente importante de relaciones. Intuitivamente, una función es una relación en la que cada entrada tiene exactamente una salida.

Definición. Una *función* $f : A \to B$ es una relación $f \subseteq A \times B$ tal que para todo $a \in A$ existe un único $b \in B$ con $(a,b) \in f$.

Definición. Sea $f : A \to B$ una función y sea $a \in A$. Si $(a,b) \in f$, escribimos $f(a) = b$ y decimos que b es la *imagen* de a. En este caso, también diremos que a es una *preimagen* de b. El conjunto A se llama el *dominio* de f, y el conjunto

$$\{b \in B : \exists a \in A \text{ tal que } f(a) = b\}$$

se llama el *rango* de f.

3.5.1. Clasificación de funciones

Definición. Sea $f : A \to B$ una función. Diremos que f es:

- *inyectiva* si para todos $a_1, a_2 \in A$, de $a_1 \neq a_2$ se sigue que $f(a_1) \neq f(a_2)$;
- *sobreyectiva* si para todo $b \in B$ existe $a \in A$ tal que $f(a) = b$;
- *biyectiva* si es inyectiva y sobreyectiva.

Ejemplo. Algunos ejemplos típicos son los siguientes.

- La función $f : \mathbb{N} \to \mathbb{N}$ dada por $f(n) = 2n$ es inyectiva, pero no es sobreyectiva.

- La función $g : \mathbb{N} \to \mathbb{N}$ dada por
$$g(0) = 0, \qquad g(n) = n - 1 \text{ para } n > 0$$
es sobreyectiva, pero no es inyectiva, pues $g(0) = g(1) = 0$.

3.5.2. Operaciones con funciones

Definición. Sean $f : A \to B$ y $g : B \to C$ funciones. La *composición* de f y g es la función
$$g \circ f : A \to C$$
definida por
$$(g \circ f)(a) = g(f(a)).$$
Si $f : A \to B$ es biyectiva, entonces existe una función
$$f^{-1} : B \to A$$
llamada la *inversa* de f, tal que
$$f^{-1}(f(a)) = a \quad \text{para todo } a \in A, \qquad f(f^{-1}(b)) = b \quad \text{para todo } b \in B.$$

Ejemplo. Sea $h : \mathbb{Z} \to \mathbb{Z}$ dada por $h(n) = n + 1$. Entonces h es biyectiva y su inversa es la función $h^{-1} : \mathbb{Z} \to \mathbb{Z}$ dada por
$$h^{-1}(n) = n - 1.$$

Proposición. Sean $f : A \to B$ y $g : B \to C$ funciones.

- Si f y g son inyectivas, entonces $g \circ f$ es inyectiva.
- Si f y g son sobreyectivas, entonces $g \circ f$ es sobreyectiva.

Demostración. Demostremos ambas afirmaciones por separado.

- Supongamos que f y g son inyectivas. Si
$$(g \circ f)(a_1) = (g \circ f)(a_2),$$
entonces
$$g(f(a_1)) = g(f(a_2)).$$
Como g es inyectiva, se obtiene $f(a_1) = f(a_2)$. Como f también es inyectiva, concluimos que $a_1 = a_2$.
- Supongamos ahora que f y g son sobreyectivas. Sea $c \in C$. Como g es sobreyectiva, existe $b \in B$ tal que $g(b) = c$. Como f es sobreyectiva, existe $a \in A$ tal que $f(a) = b$. Entonces
$$(g \circ f)(a) = g(f(a)) = g(b) = c.$$
Por lo tanto, $g \circ f$ es sobreyectiva. □

Otro ejemplo interesante de relación entre funciones aparece en la notación O de análisis de algoritmos. La estudiaremos en los ejercicios.

Ejercicios

1. En una arquitectura de microservicios consideremos el conjunto

$$S = \{\text{Gateway}, \text{Auth}, \text{Orders}, \text{Billing}, \text{DB}\}.$$

Definimos una relación D sobre S por

$$x \; D \; y \quad \text{si y sólo si} \quad x \text{ llama directamente a } y.$$

Suponga que

$$D = \{(\text{Gateway}, \text{Auth}), (\text{Gateway}, \text{Orders}), (\text{Orders}, \text{Billing}), (\text{Auth}, \text{DB}), (\text{Billing}, \text{DB})\}.$$

a) ¿Cuál es la representación vista en clases más natural para esta relación: tabla, grafo bipartito o dígrafo? Justifique brevemente.

b) Calcule D^2 e interprete el significado de cada par que aparece en esa relación.

c) Describa en palabras qué modela la clausura transitiva D^+.

d) Determine qué pares deben agregarse a D para obtener D^+.

e) Explique qué información adicional agrega la clausura reflexiva-transitiva D^* con respecto a D^+.

2. Sea $\mathcal{F} = \{f : \mathbb{N} \to \mathbb{R} : f(n) \geq 0 \text{ para todo } n \in \mathbb{N}\}$. Definimos una relación R_O sobre $\mathcal{F}$ por

$$f \; R_O \; g \quad \text{si y sólo si} \quad \exists c, k > 0 \text{ tales que } f(n) \leq c \cdot g(n) \text{ para todo } n > k.$$

En la notación usual, en vez de escribir $f \; R_O \; g$, escribimos simplemente

$$f = O(g).$$

a) Verifique que la función $f(n) = 3n^2 + 2n + 1$ satisface $f = O(n^2)$.

b) Demuestre que R_O es reflexiva y transitiva. ¿Cúal es la clausura reflexiva-transitiva de R_O?

c) Demuestre que R_O no es simétrica.

3. Sea R una relación sobre un conjunto A. Diremos que la *clausura de equivalencia* de R es la menor relación de equivalencia sobre A que contiene a R.

a) Sea S una relación simétrica sobre A. Demuestre que S^* es una relación de equivalencia.

b) Concluya que $(R^{\text{sym}})^*$ es una relación de equivalencia que contiene a R.

c) Sea E una relación de equivalencia sobre A tal que $R \subseteq E$. Demuestre que

$$(R^{\text{sym}})^* \subseteq E.$$

d) Concluya que la clausura de equivalencia de R es

$$R^{\text{eq}} = (R^{\text{sym}})^*.$$

e) Demuestre que

$$a \; R^{\text{eq}} \; b$$

si y sólo si existen elementos

$$a = a_0, a_1, \ldots, a_n = b$$

para algún $n \geq 0$, tales que para todo $i \in \{0, 1, \ldots, n-1\}$ se cumple

$$a_i \; R \; a_{i+1} \quad \text{o bien} \quad a_{i+1} \; R \; a_i.$$

f) Interprete esta caracterización de la clausura de equivalencia en términos de dígrafos y describa las clases de equivalencia de R^{eq} en términos de caminos no dirigidos en el dígrafo asociado a R.

4. Sea $A = \{1, 2, \ldots, n\}$ y sea R una relación sobre A. La *matriz de adyacencia* de R es la matriz binaria $M_R = (m_{ij})_{1 \leq i,j \leq n}$ definida por

$$m_{ij} = \begin{cases} 1 & \text{si } i \; R \; j, \\ 0 & \text{en otro caso.} \end{cases}$$

Si $M = (m_{ij})$ y $N = (n_{ij})$ son matrices binarias de tamaño $n \times n$, definimos su *producto booleano* $M \odot N = P = (p_{ij})$ por

$$p_{ij} = \bigvee_{k=1}^{n} (m_{ik} \wedge n_{kj}).$$

a) Demuestre que si S es otra relación sobre A, entonces $M_{S \circ R} = M_R \odot M_S$.

b) Defina recursivamente las potencias booleanas de una matriz por

$$M_R^{\odot 1} = M_R, \qquad M_R^{\odot (t+1)} = M_R^{\odot t} \odot M_R.$$

Demuestre que

$$M_{R^t} = M_R^{\odot t}$$

para todo $t \geq 1$.

c) Explique qué significa que la entrada (i, j) de $M_R^{\odot t}$ sea igual a 1.

d) Suponga que $|A| = n$. Describa, en términos de matrices booleanas, una matriz que represente la clausura transitiva R^+.

Capítulo 4

Combinatoria

4.1. Principios de conteo

Frecuentemente nos enfrentamos a preguntas de conteo en ciencias de la computación. ¿Cuántas claves posibles tiene un sistema? ¿Cuántos identificadores distintos podemos generar? ¿Cuántas configuraciones admite una estructura bajo ciertas restricciones? Más aún, muchas veces al analizar el rendimiento de un algoritmo, necesitamos contar la cantidad de pasos que se presentan en el peor escenario.

Este tipo de preguntas se ataca mediantes técnicas de conteo, que nos permiten determinar el número de objetos que cumplen ciertas propiedades sin necesidad de enumerarlos uno por uno. En esta clase veremos tres herramientas básicas: la regla del producto, la regla de la suma y el principio de inclusión-exclusión.

4.1.1. Regla del producto

Definición (Regla del producto)**.** Supongamos que un procedimiento consta de k tareas sucesivas $T_1, \ldots, T_k$. Si:

- la tarea T_1 puede realizarse de n_1 maneras;
- para cada $i \in \{2, \ldots, k\}$, una vez fijadas las tareas $T_1, \ldots, T_{i-1}$, la tarea T_i puede realizarse de n_i maneras;

entonces el procedimiento completo puede realizarse de $n_1 n_2 \cdots n_k$ maneras.

La idea es natural: si una decisión se toma en varias etapas, el número total de resultados se obtiene multiplicando la cantidad de opciones disponibles en cada etapa.

Ejemplo (Claves numéricas de 4 dígitos)**.** Consideremos una clave formada por cuatro dígitos decimales. Para cada posición hay 10 posibilidades. Por la regla del producto, la cantidad total de claves es

$$10 \cdot 10 \cdot 10 \cdot 10 = 10^4 = 10000.$$

Ejemplo (Números pares de 4 dígitos)**.** Queremos contar cuántos números de 4 dígitos son pares. El primer dígito tiene 9 opciones, pues no puede ser 0. El segundo y el tercer dígito tienen 10

opciones cada uno. El último dígito debe ser par, así que tiene 5 opciones: $0, 2, 4, 6, 8$. Por lo tanto, el número total es

$$9 \cdot 10 \cdot 10 \cdot 5 = 4500.$$

Proposición (Producto cartesiano)**.** Sean $A_1, \ldots, A_k$ conjuntos finitos. Entonces

$$|A_1 \times A_2 \times \cdots \times A_k| = |A_1| \cdot |A_2| \cdots |A_k|.$$

Demostración. Formar un elemento de $A_1 \times \cdots \times A_k$ equivale a elegir una k-upla

$$(a_1, \ldots, a_k)$$

con $a_i \in A_i$ para todo i. La primera coordenada puede elegirse de $|A_1|$ maneras, la segunda de $|A_2|$ maneras, y así sucesivamente. Aplicando la regla del producto, obtenemos

$$|A_1 \times \cdots \times A_k| = |A_1| \cdot |A_2| \cdots |A_k|.$$ □

Esta reformulación es útil porque muchas veces un objeto combinatorio puede modelarse directamente como una tupla. Contar las tuplas posibles es exactamente contar el producto cartesiano asociado.

Ejemplo (Patentes)**.** Ignorando restricciones regulatorias adicionales, consideremos los siguientes formatos para las patentes de automóviles:

- Si una patente tiene la forma
$$\alpha_1\alpha_2\alpha_3\alpha_4 d_1 d_2,$$
con $\alpha_i \in \{A, \ldots, Z\}$ y $d_j \in \{0, \ldots, 9\}$, entonces hay
$$26^4 \cdot 10^2 = 45697600$$
patentes posibles.
- Si ahora usamos el formato $\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5 d_1$, entonces hay
$$26^5 \cdot 10 = 118813760$$
posibilidades.
- En el formato de cuatro letras y dos dígitos, si además exigimos que no haya letras repetidas ni dígitos repetidos, entonces el número de posibilidades es
$$26 \cdot 25 \cdot 24 \cdot 23 \cdot 10 \cdot 9 = 32292000.$$

4.1.2. Regla de la suma

Definición (Regla de la suma)**.** Supongamos que una tarea puede realizarse de una entre k maneras alternativas, correspondientes a tareas $T_1, \ldots, T_k$. Si estas alternativas son incompatibles par a par, y T_i puede realizarse de n_i maneras para cada i, entonces la tarea total

puede realizarse de

$$n_1 + n_2 + \cdots + n_k$$

maneras.

La regla de la suma se usa cuando el procedimiento consiste en elegir entre casos que no se superponen. En ese escenario, contar el total equivale a sumar los tamaños de los casos.

Ejemplo (Elegir una entrada)**.** Supongamos que en un menú podemos elegir *o* una sopa *o* una ensalada como entrada. Si hay 2 opciones de sopa y 3 opciones de ensalada, entonces hay

$$2 + 3 = 5$$

maneras de elegir la entrada.

La clave es que no podemos elegir simultáneamente sopa y ensalada. Los casos son disjuntos.

Proposición (Unión disjunta)**.** Sean $A_1, \ldots, A_k$ conjuntos finitos y disjuntos par a par. Entonces

$$|A_1 \cup A_2 \cup \cdots \cup A_k| = |A_1| + |A_2| + \cdots + |A_k|.$$

Demostración. Elegir un elemento de la unión

$$A_1 \cup \cdots \cup A_k$$

equivale a elegir un elemento de exactamente uno de los conjuntos $A_1, \ldots, A_k$, porque son disjuntos par a par. Por la regla de la suma, el número total de posibilidades es

$$|A_1| + \cdots + |A_k|.$$ □

Ejemplo (Números de 4 dígitos con exactamente tres doses)**.** Queremos contar cuántos números de 4 dígitos tienen exactamente tres dígitos iguales a 2.

Los casos posibles son

$$222X, \qquad 22X2, \qquad 2X22, \qquad X222.$$

En los tres primeros casos, X puede ser cualquier dígito distinto de 2, por lo que hay 9 opciones. En el último caso, además debemos evitar $X = 0$, pues el número debe tener 4 dígitos. Por lo tanto, allí hay 8 opciones. Como los cuatro casos son incompatibles entre sí, por la regla de la suma obtenemos

$$9 + 9 + 9 + 8 = 35.$$

4.1.3. Principio de inclusión-exclusión

La regla de la suma deja de ser suficiente cuando los casos se superponen. Si dos propiedades pueden cumplirse simultáneamente, entonces al sumar los tamaños de los conjuntos correspondientes terminamos contando dos veces la intersección. El principio de inclusión-exclusión corrige exactamente ese doble conteo.

Teorema (Inclusión-exclusión para dos conjuntos)**.** Sean A y B conjuntos finitos. Entonces

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Demostración. Podemos escribir B como unión disjunta:

$$B = (B \setminus A) \cup (A \cap B).$$

Por lo tanto,

$$|B| = |B \setminus A| + |A \cap B|.$$

Además, la unión $A \cup B$ puede escribirse de manera disjunta como $A \cup (B \setminus A)$. Usando estos dos hechos, tenemos que

$$|A \cup B| = |A| + \underbrace{|B \setminus A|}_{=|B|-|A\cap B|} = |A| + |B| - |A \cap B|. \qquad \square$$

Ejemplo (Strings binarios de largo 8)**.** Queremos contar cuántos strings binarios de largo 8 empiezan con 1 o terminan con 00.

Sea A el conjunto de strings que empiezan con 1, y sea B el conjunto de strings que terminan con 00. Entonces:

- $|A| = 2^7$, porque fijamos el primer bit y quedan 7 bits libres.
- $|B| = 2^6$, porque fijamos los dos últimos bits y quedan 6 bits libres.
- $|A \cap B| = 2^5$, porque fijamos el primer bit y los dos últimos, y quedan 5 bits libres.

Por inclusión-exclusión,

$$|A \cup B| = 2^7 + 2^6 - 2^5 = 128 + 64 - 32 = 160.$$

4.1.4. Combinando reglas de conteo

Es común tener que usar varias reglas a la vez. Un mismo objeto puede clasificarse por casos y, dentro de cada caso, contarse mediante una aplicación de la regla del producto.

Ejemplo (Direcciones IPv4 de tipo A, B y C)**.** Consideremos direcciones IPv4 de 32 bits en el esquema clásico de tipos A, B y C. En cada caso distinguimos tres partes: un prefijo fijo, un *network id* y un *host id*.

Usaremos la siguiente simplificación:

- en tipo A, el *network id* tiene 7 bits y el *host id* tiene 24 bits;
- en tipo B, el *network id* tiene 14 bits y el *host id* tiene 16 bits;
- en tipo C, el *network id* tiene 21 bits y el *host id* tiene 8 bits.

Además, supondremos que el *network id* no puede ser todo unos, y que el *host id* no puede ser ni todo ceros ni todo unos.

Por lo tanto:

$$IP_A = (2^7 - 1)(2^{24} - 2),$$
$$IP_B = (2^{14} - 1)(2^{16} - 2),$$
$$IP_C = (2^{21} - 1)(2^8 - 2).$$

Como los tres tipos son incompatibles entre sí, por la regla de la suma obtenemos

$$IP_{\text{total}} = IP_A + IP_B + IP_C.$$

Numéricamente,

$$IP_A = 2130706178,$$
$$IP_B = 1073643522,$$
$$IP_C = 532676354.$$

En consecuencia,

$$IP_{\text{total}} = 3737026054.$$

En particular, el número total de direcciones es del orden de 10^9.

Ejercicios

1. Un sistema de autenticación ofrece los siguientes mecanismos:

   - un PIN de 4 dígitos;
   - un identificador alfanumérico de la forma

     $$\alpha_1\alpha_2\alpha_3\alpha_4 d_1 d_2,$$

     donde cada α_i es una letra de $\{A, \ldots, Z\}$ y cada d_j es un dígito de $\{0, \ldots, 9\}$;
   - un token binario de recuperación de 8 bits.

   a) ¿Cuántos PINes distintos pueden generarse?

   b) ¿Cuántos identificadores alfanuméricos distintos pueden generarse?

   c) ¿Cuántos identificadores alfanuméricos no tienen letras repetidas ni dígitos repetidos?

   d) Si un usuario puede autenticarse usando o bien un PIN o bien un identificador alfanumérico, ¿cuántas credenciales distintas admite el sistema?

   e) ¿Cuántos tokens binarios de recuperación empiezan con 1 o terminan con 00?

2. Sea A un conjunto finito con $|A| = n$. Para cada $k \in \{0, 1, \ldots, n\}$, definamos

   $$\mathcal{S}_k = \{X \subseteq A : |X| = k\}.$$

a) Demuestre que los conjuntos $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n$ son disjuntos par a par.

b) Demuestre que
$$\mathcal{P}(A) = \mathcal{S}_0 \cup \mathcal{S}_1 \cup \cdots \cup \mathcal{S}_n.$$

c) Concluya que
$$2^n = |\mathcal{S}_0| + |\mathcal{S}_1| + \cdots + |\mathcal{S}_n|.$$

d) Si denotamos por $\binom{n}{k}$ el número de subconjuntos de A con exactamente k elementos, deduzca que
$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

3. Sean A y B conjuntos finitos con $|A| = n$ y $|B| = m$.

a) Demuestre que el número total de funciones $f : A \to B$ es m^n.

b) Suponga ahora que $m \geq n$. Demuestre que el número de funciones inyectivas $f : A \to B$ es
$$m(m-1)\cdots(m-n+1).$$

c) Tomando ahora $B = \{0, 1\}$, cuente cuántas funciones
$$f : A \to \{0, 1\}$$
usan ambos valores 0 y 1.

d) Cuente para las funciones
$$f : A \to \{0, 1\},$$
según el valor de n:

- funciones arbitrarias;
- funciones inyectivas;
- funciones sobreyectivas;
- funciones a la vez inyectivas y sobreyectivas.

4.2. Objetos combinatoriales básicos

En la clase anterior vimos principios generales de conteo. Ahora estudiaremos algunas familias de objetos que aparecen una y otra vez: listas ordenadas, subconjuntos, selecciones con repetición e identidades entre cantidades combinatoriales.

4.2.1. Permutaciones

Comenzamos con estructuras donde el orden sí importa. Estas ideas aparecen, por ejemplo, al ordenar tareas, armar rankings o construir claves sin repetición.

Definición (Permutación y factorial)**.** Sea X un conjunto finito. Una *permutación* de X es una lista ordenada que contiene todos los elementos de X, cada uno exactamente una vez. Para $n \in \mathbb{N}$, definimos el *factorial* $n!$ como el número de permutaciones de un conjunto de n elementos.

Ejemplo. Si $X = \{a, b, c\}$, entonces las permutaciones de X son

$$abc, acb, bac, bca, cab, cba,$$

y hay 6 de ellas, así que $3! = 6$.

Proposición (Conteo de permutaciones)**.** Para todo $n \in \mathbb{N}$, se cumple

$$n! = n(n-1)\cdots 2 \cdot 1.$$

Demostración. Para formar una permutación de un conjunto de n elementos, el primer lugar puede elegirse de n maneras, el segundo de $n-1$ maneras, y así sucesivamente, hasta llegar a 1 manera para el último lugar. Por la regla del producto,

$$n! = n(n-1)\cdots 2 \cdot 1.$$ □

La noción de permutación usa todos los elementos del conjunto. En muchos problemas, en cambio, sólo queremos una lista ordenada de largo r con elementos distintos. Eso nos lleva a las r-permutaciones.

Definición (r-permutaciones)**.** Sean $n, r \in \mathbb{N}$ con $r \leq n$. Una *r-permutación* de un conjunto de n elementos es una lista ordenada de r elementos distintos.
Definimos $\Pi(n, r)$ como la *cantidad de r-permutaciones* de un conjunto de n elementos.

Ejemplo. Si $X = \{a, b, c\}$, entonces las 2-permutaciones de X son

$$ab, ac, ba, bc, ca, cb,$$

y hay 6 de ellas, así que $\Pi(3, 2) = 6$.

Proposición (Conteo de r-permutaciones)**.** Para todo $n, r \in \mathbb{N}$ con $r \leq n$, se cumple

$$\Pi(n, r) = n(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!}.$$

Demostración. Para formar una lista ordenada de largo r, el primer lugar puede elegirse de n maneras. Una vez elegido, el segundo puede elegirse de $n-1$ maneras. Continuando así, el lugar número r puede elegirse de $n-r+1$ maneras. Por la regla del producto,

$$\Pi(n, r) = n(n-1)\cdots(n-r+1).$$

La identidad con factoriales se obtiene reescribiendo este producto como $\frac{n!}{(n-r)!}$. □

Ejemplo (Playlist sorpresa)**.** Un reproductor tiene 12 canciones y queremos armar una lista de reproducción de largo 4 sin repetir canciones. Como el orden importa, estamos contando 4-permutaciones de un conjunto de 12 elementos. La cantidad posible es $\Pi(12,4) = 12 \cdot 11 \cdot 10 \cdot 9 = 11880$.

Ejemplo (Números de tres dígitos impares distintos)**.** Consideremos los dígitos impares $\{1,3,5,7,9\}$. Queremos formar números de tres dígitos sin repetición, así que estamos contando permutaciones ordenadas de largo 3 tomadas de un conjunto de 5 elementos. Entonces la cantidad buscada es $\Pi(5,3) = 5 \cdot 4 \cdot 3 = 60$.

4.2.2. Combinaciones

Hasta aquí el orden era esencial: cambiar la posición de los elementos cambiaba el resultado. Ahora pasamos al caso en que sólo importa qué elementos se eligen.

Definición (Combinaciones)**.** Sean $n, r \in \mathbb{N}$ con $r \leq n$. Una *r-combinación* de un conjunto de n elementos es un subconjunto de tamaño r.
Definimos el *coeficiente binomial* $\binom{n}{r}$ como la cantidad de r-combinaciones de un conjunto de n elementos.

Ejemplo. Si $X = \{a,b,c\}$, entonces las 2-combinaciones de X son

$$\{a,b\}, \{a,c\}, \{b,c\},$$

y hay 3 de ellas, así que $\binom{3}{2} = 3$.

Proposición (Conteo de combinaciones)**.** Para todo $n, r \in \mathbb{N}$ con $r \leq n$, se cumple

$$\binom{n}{r} = \frac{n!}{r!(n-r)!}.$$

Demostración. Toda r-combinación puede ordenarse de $r!$ maneras distintas, y toda r-permutación proviene de una única r-combinación junto con uno de esos órdenes. Por lo tanto, $\Pi(n,r) = \binom{n}{r} r!$. Usando el conteo de r-permutaciones,

$$\binom{n}{r} = \frac{\Pi(n,r)}{r!} = \frac{\frac{n!}{(n-r)!}}{r!} = \frac{n!}{r!(n-r)!}.$$ □

Ejemplo (Grupos, manos y strings binarios)**.** Las combinaciones aparecen de manera natural en muchos contextos.

- Si queremos formar un grupo de 3 personas entre 10, el número de posibilidades es $\binom{10}{3} = 120$.
- El número de manos de póker de 5 cartas tomadas de un mazo de 52 es $\binom{52}{5}$.
- El número de strings binarios de largo n con exactamente r unos es $\binom{n}{r}$, porque basta elegir las r posiciones donde irán los unos.

Ejemplo (Un substring fijo)**.** ¿Cuántas permutaciones de las letras de la A a la H contienen el substring ABC?

Si tratamos a ABC como un bloque, entonces debemos ordenar los siguientes 6 objetos ABC, D, E, F, G y H. Por lo tanto, la cantidad buscada es $\Pi(6,6) = 6!$.

Ejemplo (Anagramas)**.** Contemos cuántos anagramas distintos tiene la palabra `BANANA`.

Si consideráramos todas las letras distintas, habría 6! ordenamientos. Pero hay tres letras A indistinguibles y dos letras N indistinguibles, así que cada anagrama se cuenta $3!\,2!$ veces en ese total. En consecuencia, el número de anagramas distintos es

$$\frac{6!}{3!\,2!} = 60.$$

4.3. Strings y multiconjuntos

Hasta ahora siempre hemos supuesto que los elementos elegidos son distintos. Sin embargo, en problemas reales a veces las repeticiones están permitidas, e incluso forman parte esencial del modelo.

4.3.1. Strings

Definición (Strings)**.** Fijado un alfabeto de n símbolos, un *string de largo r* es una lista ordenada de largo r en la que se permiten repeticiones. En otros textos, estos objetos también se llaman *permutaciones con repetición*.

Ejemplo. Si el alfabeto es $\{a, b, c\}$, entonces los strings de largo 2 son

$$aa, ab, ac, ba, bb, bc, ca, cb, cc,$$

y hay 8 de ellos.

Proposición (Conteo de strings)**.** El número de strings de largo r sobre un alfabeto de n símbolos es n^r.

Demostración. Cada una de las r posiciones puede llenarse con cualquiera de los n elementos disponibles. Por la regla del producto, el número total de listas es

$$\underbrace{n \cdot n \cdots n}_{r \text{ veces}} = n^r. \qquad \square$$

Ejemplo (Strings binarios de largo 5)**.** Cada posición puede ser 0 o 1. Por lo tanto, el número de strings binarios de largo 5 es $2^5 = 32$.

4.3.2. Multiconjuntos

Cuando además dejamos de distinguir el orden, lo que obtenemos ya no son strings, sino multiconjuntos.

Definición (Multiconjuntos)**.** Un *multiconjunto de tamaño r sobre n tipos* es una selección de tamaño r en la que el orden no importa y las repeticiones están permitidas. En otros textos, estos objetos también se llaman *combinaciones con repetición*.

Ejemplo. Si los tipos son $\{a, b, c\}$, entonces los multiconjuntos de tamaño 2 son

$$\{a, a\}, \{a, b\}, \{a, c\}, \{b, b\}, \{b, c\}, \{c, c\},$$

y hay 6 de ellos.

Proposición (Conteo de multiconjuntos)**.** El número de multiconjuntos de tamaño r sobre n tipos es $\binom{n+r-1}{r}$.

Demostración. Pensamos el problema como distribuir r objetos indistinguibles en n cajas distinguibles. Si una caja recibe a_i objetos, entonces

$$a_1 + a_2 + \cdots + a_n = r$$

con $a_i \geq 0$.

Para representar una solución de lo anterior usaremos el método de estrellas y barras. Note que esto es lo mismo que repartir r estrellas y $n-1$ barras en una secuencia. Las estrellas indican objetos, y las barras separan las cajas. Por ejemplo, la secuencia

$$\star\star \mid \star \mid\mid \star\star\star$$

representa la distribución

$$(2, 1, 0, 3).$$

Entonces el problema se reduce a contar secuencias de r estrellas y $n-1$ barras. En total hay $r+n-1$ símbolos, y basta elegir cuáles r de ellos serán estrellas. Por lo tanto, el número de secuencias es $\binom{n+r-1}{r}$. □

Ejemplo (Helados)**.** Supongamos que una heladería ofrece 8 sabores y queremos elegir 3 bolas, permitiendo repetir sabores. Como el orden no importa, estamos contando multiconjuntos. La cantidad total es $\binom{8+3-1}{3} = \binom{10}{3} = 120$.

4.4. Demostraciones combinatoriales

En combinatoria es común encontrar identidades que admiten dos tipos de demostración. Una puede ser algebraica, manipulando expresiones via resultados previos o usando fórmulas explícitas. La otra puede ser una demostración combinatorial, es decir, mostrar que ambos lados de la identidad cuentan la misma cantidad de objetos, aunque de maneras distintas.

Proposición (Simetría del binomio)**.** Para todo $n, r \in \mathbb{N}$ con $r \leq n$,

$$\binom{n}{r} = \binom{n}{n-r}.$$

Demostración algebraica. A partir de la definición,

$$\binom{n}{r} = \frac{n!}{r!(n-r)!} = \frac{n!}{(n-r)!(n-(n-r))!} = \binom{n}{n-r}. \qquad \square$$

Demostración combinatorial. Elegir r elementos de un conjunto de n equivale a elegir cuáles $n-r$ elementos quedan fuera. Ambas decisiones describen exactamente el mismo conjunto de subconjuntos, luego tienen la misma cantidad. $\square$

Proposición (Suma binomial)**.** Para todo $n \in \mathbb{N}$,

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

Demostración algebraica. Aplicamos el teorema del binomio a

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}.$$

Tomando $x = y = 1$, obtenemos

$$2^n = \sum_{k=0}^{n} \binom{n}{k}. \qquad \square$$

Demostración combinatorial. Cada término $\binom{n}{k}$ cuenta los subconjuntos de tamaño k de un conjunto de n elementos. Al sumar sobre todos los posibles valores de k, contamos todos los subconjuntos del conjunto. Pero el número total de subconjuntos es 2^n. $\square$

Proposición (Identidad de Pascal)**.** Para todo $n, k \in \mathbb{N}$ con $1 \leq k \leq n$,

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}.$$

Demostración algebraica. Usando la definición de coeficiente binomial,

$$\binom{n}{k-1} + \binom{n}{k} = \frac{n!}{(k-1)!(n-k+1)!} + \frac{n!}{k!(n-k)!}.$$

Llevando ambas fracciones a denominador común, obtenemos

$$\frac{n!k}{k!(n-k+1)!} + \frac{n!(n-k+1)}{k!(n-k+1)!} = \frac{n!(n+1)}{k!(n-k+1)!} = \frac{(n+1)!}{k!(n+1-k)!} = \binom{n+1}{k}. \qquad \square$$

Demostración combinatorial. Fijemos un conjunto de $n+1$ elementos y distingamos un elemento especial a. Queremos contar los subconjuntos de tamaño k.

Hay dos posibilidades:

- el subconjunto contiene a a; en ese caso, faltan $k-1$ elementos por elegir entre los otros n, y hay $\binom{n}{k-1}$ posibilidades;
- el subconjunto no contiene a a; en ese caso, hay que elegir los k elementos entre los otros n, y hay $\binom{n}{k}$ posibilidades.

Como ambos casos son disjuntos y exhaustivos, el total es $\binom{n}{k-1} + \binom{n}{k}$. $\square$

4.5. Principio del palomar

El *principio del palomar* aparece una y otra vez en ciencias de la computación. Cada vez que muchas entradas compiten por pocos recursos, como buckets de una tabla de hash, salidas posibles de una función o clases de equivalencia, aparecen colisiones inevitables.

Es, además, una herramienta fundamental para demostrar resultados de existencia: si queremos mostrar que existe un objeto con cierta propiedad, a veces basta con mostrar que el número de objetos posibles es mayor que el número de objetos que no tienen esa propiedad.

> **Teorema** (Principio del palomar)**.** Si $k+1$ objetos se distribuyen en k cajas, entonces al menos una caja contiene al menos dos objetos.

Omitiremos la demostración, pues veremos una versión más general del principio del palomar al final de esta sección.

Ejemplo (Mismo resto módulo 10)**.** Dados 11 números enteros, siempre existen dos que tienen el mismo resto al dividir por 10.

En efecto, sólo hay 10 restos posibles: $0, 1, \ldots, 9$. Si asignamos a cada entero el resto de su división por 10, estamos distribuyendo 11 objetos en 10 cajas. Por el principio del palomar, dos de ellos caen en la misma caja. En particular, su diferencia es múltiplo de 10.

Ejemplo (Puntos en el cuadrado)**.** Sean n^2+1 puntos en el cuadrado $[0,1]\times[0,1]$. Entonces existen dos puntos a distancia a lo más $\frac{\sqrt{2}}{n}$.

En efecto, podemos dividir el cuadrado en n^2 subcuadrados de lado $\frac{1}{n}$. Si asignamos a cada punto el subcuadrado donde se encuentra, estamos distribuyendo n^2+1 objetos en n^2 cajas. Por el principio del palomar, dos de ellos caen en la misma caja. La distancia entre esos dos puntos es a lo más la diagonal del subcuadrado, que es $\frac{\sqrt{2}}{n}$.

Ejemplo (Divisibilidad de Erdős)**.** En cualquier subconjunto de $n+1$ números enteros distintos tomados de $\{1, 2, \ldots, 2n\}$, existe uno que divide a otro.

Todo entero positivo puede escribirse de la forma $2^k b$, con $k \geq 0$ y b impar entre 1 y $2n$. Entre 1 y $2n$ hay sólo n números impares posibles para b. Si elegimos $n+1$ enteros, dos de ellos deben compartir el mismo valor impar b. Entonces tienen la forma $2^{k_1} b$ y $2^{k_2} b$. Si $k_1 < k_2$, el primero divide al segundo.

> **Teorema** (Principio del palomar generalizado)**.** Si n objetos se distribuyen en k cajas, entonces al menos una caja contiene al menos $\left\lceil \frac{n}{k} \right\rceil$ objetos.

Demostración. Supongamos, buscando una contradicción, que toda caja contiene a lo más $\left\lceil \frac{n}{k} \right\rceil - 1$ objetos. Entonces el número total de objetos sería a lo más $k\left(\left\lceil \frac{n}{k} \right\rceil - 1\right)$. Pero por definición del techo,

$$\left\lceil \frac{n}{k} \right\rceil - 1 < \frac{n}{k},$$

así que

$$k\left(\left\lceil \frac{n}{k} \right\rceil - 1\right) < n,$$

contradicción. □

Ejemplo (Colisiones en una tabla de hash)**.** Si una tabla de hash tiene 128 buckets y almacenamos 1000 claves, entonces al menos un bucket contiene al menos $\lceil \frac{1000}{128} \rceil = 8$ claves. Este tipo de cota no nos dice dónde ocurre la colisión, pero sí garantiza que ocurre.

Ejemplo (La fiesta de Ramsey)**.** Ramsey organiza una fiesta a la que asisten 6 personas. En dicha fiesta, siempre existe un trío de personas que se conocen mutuamente, o un trío de personas que no se conocen mutuamente.

Elija una persona p. Entre las otras 5, por el principio del palomar aplicado a las dos cajas

$$\text{“conoce a } p\text{”} \quad \text{y} \quad \text{“no conoce a } p\text{”},$$

al menos 3 caen en la misma caja.

Supongamos primero que p conoce a a, b, c. Si alguno de los pares entre a, b, c se conoce, por ejemplo a y b, entonces p, a, b forman un trío de conocidos mutuos. Si ningún par entre a, b, c se conoce, entonces a, b, c forman un trío de desconocidos mutuos.

El caso en que p no conoce a a, b, c es análogo: si algún par entre ellos no se conoce, tenemos un trío de desconocidos; y si todos se conocen entre sí, tenemos un trío de conocidos. En cualquier caso, uno de los dos tipos de trío aparece inevitablemente.

Ejercicios

1. Una plataforma organiza una competencia de proyectos de software. En ella participan 10 equipos finalistas, hay 12 mentores disponibles para formar un jurado, el sistema ofrece 5 tipos de créditos de cómputo idénticos entre sí, y las propuestas se almacenan en una tabla de hash con 20 buckets.

   a) ¿De cuántas maneras se puede asignar un primer, segundo y tercer lugar entre los 10 equipos?

   b) ¿De cuántas maneras se puede formar un jurado de 4 mentores?

   c) ¿De cuántas maneras se puede elegir un paquete de 6 créditos de cómputo si sólo importa cuántos créditos de cada tipo se piden?

   d) Si se reciben 101 propuestas y cada una cae en uno de los 20 buckets, demuestre que al menos un bucket contiene al menos 6 propuestas.

   e) Indique en cada inciso si el orden importa y si se permiten repeticiones.

2. Sean $n, k \in \mathbb{N}$ con $1 \leq k \leq n$.

   a) Demuestre algebraicamente que

   $$k\binom{n}{k} = n\binom{n-1}{k-1}.$$

   b) Demuestre la misma identidad con una demostración combinatorial.

   c) Use la identidad de la parte *a*) para demostrar algebraicamente que

   $$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}.$$

d) Demuestre la misma identidad con una demostración combinatorial.

3. Los siguientes problemas muestran aplicaciones del principio del palomar.

   a) Ramsey organiza una segunda fiesta para n invitados, quienes se sientan en una mesa redonda. Como Ramsey es muy organizado, asigna a cada uno de sus invitados un asiento respectivo en la mesa redonda. Sin embargo, los invitados de Ramsey son bastante desconsiderados y cada uno se sienta en un puesto distinto al que le corresponde. Muestre que es posible girar la mesa de manera que al menos dos invitados queden en el puesto correcto.

   b) Cada punto en una grilla de 3×7 es coloreado de rojo o azul. Muestre que siempre existe un rectángulo cuyos vértices son puntos del mismo color.

4. Sean X e Y conjuntos finitos. En la clase anterior vimos que

$$|X \cup Y| = |X| + |Y| - |X \cap Y|.$$

   En este ejercicio, demostraremos el principio de inclusión-exclusión para k conjuntos finitos $A_1, A_2, \ldots, A_k$, que establece que

$$\left|\bigcup_{i=1}^{k} A_i\right| = \sum_{\emptyset \neq I \subseteq \{1,\ldots,k\}} (-1)^{|I|+1} \left|\bigcap_{i \in I} A_i\right|.$$

   Para esto, considere un elemento x que pertenece a $\bigcup_{i=1}^{k} A_i$. La idea es contar cuántas veces se cuenta a x en el lado derecho de la fórmula.

   a) Suponga que x pertenece a exactamente r de los conjuntos. Más aún, sin pérdida de generalidad, suponga que $x \in A_1, \ldots, A_r$ y $x \notin A_{r+1}, \ldots, A_k$. Muestre combinatorialmente que:

$$\sum_{\emptyset \neq I \subseteq \{1,\ldots,k\}} (-1)^{|I|+1} \left|\{x\} \cap \bigcap_{i \in I} A_i\right|$$
$$= |\{A_i \mid 1 \leq i \leq r\}| - |\{A_i \cap A_j \mid 1 \leq i < j \leq r\}| + \cdots + (-1)^{r+1} |\{A_1 \cap A_2 \cap \cdots \cap A_r\}|.$$

   b) Pruebe, de manera combinatorial, que la expresión anterior es igual a

$$\binom{r}{1} - \binom{r}{2} + \cdots + (-1)^{r+1} \binom{r}{r}.$$

   c) Muestre que la expresión anterior es igual a 1.

   d) Concluya el principio de inclusión-exclusión para k conjuntos finitos.

4.6. Relaciones de recurrencia

En las clases anteriores contamos objetos usando reglas generales e interpretando objetos combinatoriales. Ahora veremos una técnica complementaria: en vez de contar directamente los objetos de tamaño n, los relacionaremos con objetos de tamaños menores. Esta idea aparece de manera natural cuando un problema tiene una estructura recursiva.

Definición (Relación de recurrencia)**.** Una *relación de recurrencia* para una sucesión $(a_n)_{n\geq 0}$ es una ecuación que expresa a_n en función de uno o más términos anteriores de la sucesión, es decir, de $a_{n-1}, a_{n-2}, \ldots, a_0$.

Muchas veces buscamos *fórmulas cerradas* para las sucesiones, es decir, fórmulas que nos permitan calcular a_n sin necesidad de conocer los términos anteriores. Una recurrencia no nos proporciona esto, pero puede ser más fácil de encontrar que la fórmula cerrada, y a menudo es posible usar la recurrencia para deducir la fórmula cerrada. (Técnicas para encontrar fórmulas cerradas a partir de recurrencias las veremos en las siguientes clases.)

Ejemplo (Sueldo con aumento fijo)**.** Alicia recibe un sueldo inicial de 1 millón de pesos y cada mes obtiene un aumento de 10 %. Si a_n es su sueldo en el mes n, entonces podemos describir la sucesión por

$$a_0 = 1, \qquad a_n = 1.1a_{n-1} \quad \text{para todo } n \geq 1.$$

En este caso también es fácil reconocer la forma cerrada $a_n = 1.1^n$.

Ejemplo (Fibonacci)**.** La sucesión de Fibonacci se define por

$$f_0 = 0, \qquad f_1 = 1, \qquad f_n = f_{n-1} + f_{n-2} \quad \text{para todo } n \geq 2.$$

Aquí necesitamos dos condiciones iniciales, porque cada término nuevo depende de los dos términos anteriores.

4.6.1. Recurrencias por análisis de casos

La manera más común de obtener una recurrencia en problemas de conteo es hacer una partición por casos. La clave es que los casos sean disjuntos y que cada caso se pueda contar usando cantidades que ya conocemos.

Ejemplo (Strings binarios sin `00`)**.** Sea a_n el número de strings binarios de largo n que no contienen dos ceros consecutivos. Usaremos $a_0 = 1$, contando el string vacío.

Para $n \geq 2$, analizamos el final del string:

- si termina en `1`, entonces el prefijo de largo $n-1$ puede ser cualquier string válido, y hay a_{n-1} posibilidades;
- si termina en `0`, entonces el símbolo anterior debe ser `1`; por lo tanto, el string termina en `10`, y el prefijo de largo $n-2$ puede ser cualquier string válido.

Entonces

$$a_0 = 1, \qquad a_1 = 2, \qquad a_n = a_{n-1} + a_{n-2} \quad \text{para todo } n \geq 2.$$

Esta recurrencia es nuevamente de tipo Fibonacci, pero con condiciones iniciales distintas.

Ejemplo (Apretones de manos)**.** Supongamos que las personas llegan una por una a una reunión, y que cada nueva persona saluda a todas las que ya llegaron. Sea a_n el número total de apretones de manos después de que han llegado n personas.

Cuando llega la primera persona no ocurre ningún saludo, así que $a_1 = 0$. Para $n \geq 2$, la persona número n saluda a las $n-1$ personas anteriores. Por lo tanto,

$$a_1 = 0, \qquad a_n = a_{n-1} + (n-1) \quad \text{para todo } n \geq 2.$$

Ejemplo (Torres de Hanoi)**.** Sea h_n la mínima cantidad de movimientos necesarios para resolver Torres de Hanoi con n discos.

Para mover una torre de n discos desde una varilla origen a una varilla destino, necesariamente debemos:

- mover primero los $n-1$ discos superiores a la varilla auxiliar;
- mover el disco más grande a la varilla destino;
- mover los $n-1$ discos desde la varilla auxiliar a la varilla destino.

El mismo procedimiento muestra que esa cantidad de movimientos también se puede alcanzar. Por lo tanto,

$$h_1 = 1, \qquad h_n = 2h_{n-1} + 1 \quad \text{para todo } n \geq 2.$$

En este caso se puede verificar por inducción que la forma cerrada es $h_n = 2^n - 1$.

Ejemplo (Paréntesis balanceados y números de Catalán)**.** Sea c_n el número de palabras de paréntesis balanceados con n pares de paréntesis. Por ejemplo, para $n = 3$ algunas palabras válidas son

$$()()(), \qquad (())(), \qquad (()()).$$

Tenemos $c_0 = 1$, contando la palabra vacía. Para $n \geq 1$, tomemos una palabra balanceada con n pares. El primer paréntesis izquierdo tiene un paréntesis derecho que lo cierra. Supongamos que dentro de ese par hay i pares de paréntesis. Entonces después de ese par quedan $n-1-i$ pares.

Así, toda palabra balanceada se descompone de manera única como:

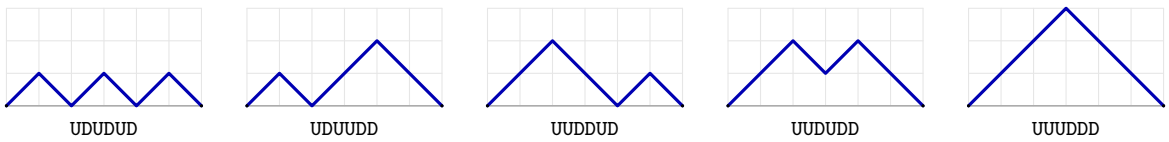
$$(\,A\,)\,B,$$

donde A es una palabra balanceada con i pares y B es una palabra balanceada con $n-1-i$ pares. Para cada valor de i, hay $c_i c_{n-1-i}$ posibilidades. Sumando sobre todos los valores posibles de i, obtenemos

$$c_0 = 1, \qquad c_n = \sum_{i=0}^{n-1} c_i c_{n-1-i} \quad \text{para todo } n \geq 1.$$

Esta sucesión es la de los *números de Catalán*.

Ejemplo (Caminos de Dyck y números de Catalán)**.** Un *camino de Dyck* de largo $2n$ es un camino que parte en altura 0, usa pasos $U = (1, 1)$ y $D = (1, -1)$, nunca baja de altura 0, y termina nuevamente en altura 0. Sea d_n el número de caminos de Dyck de largo $2n$. Por ejemplo, para $n = 3$ los cinco caminos válidos son:



Tenemos $d_0 = 1$, contando el camino vacío. Para $n \geq 1$, todo camino no vacío empieza con un paso U. Consideremos el paso D que lo devuelve por primera vez a altura 0. Si entre ese U inicial y ese D hay $2i$ pasos, entonces esa parte intermedia, vista una unidad más abajo, es un camino

de Dyck de largo $2i$. Después del primer retorno a altura 0 queda un camino de Dyck de largo $2(n-1-i)$.

Por lo tanto, todo camino de Dyck se descompone de manera única como

$$U\,A\,D\,B,$$

donde A es un camino de Dyck con i pares de pasos y B es un camino de Dyck con $n-1-i$ pares de pasos. Así,

$$d_0 = 1, \qquad d_n = \sum_{i=0}^{n-1} d_i d_{n-1-i} \quad \text{para todo } n \geq 1.$$

Es la misma recurrencia de los números de Catalán, así que $d_n = c_n$.

4.6.2. Sistemas de recurrencias

A veces una sola sucesión no conserva toda la información necesaria para hacer el paso recursivo. En esos casos introducimos estados auxiliares y definimos varias sucesiones simultáneamente.

> **Definición** (Sistema de recurrencias)**.** Un *sistema de recurrencias* es una colección de sucesiones definidas recursivamente de manera simultánea. Las ecuaciones de una sucesión pueden depender de los términos anteriores de las otras sucesiones del sistema.

Ejemplo (Paridad del número de unos)**.** Sea p_n el número de strings ternarios de largo n con un número par de unos, y sea q_n el número de strings ternarios de largo n con un número impar de unos.

Para $n = 0$, el string vacío tiene cero unos, así que

$$p_0 = 1, \qquad q_0 = 0.$$

Para construir un string de largo $n+1$, agregamos un símbolo al final:

- agregar `0` o `2` no cambia la paridad del número de unos;
- agregar `1` cambia la paridad.

Por lo tanto,

$$p_{n+1} = 2p_n + q_n, \qquad q_{n+1} = 2q_n + p_n \quad \text{para todo } n \geq 0.$$

El punto importante es que este sistema muestra cómo guardar el estado necesario para que la descripción recursiva sea local.

Ejercicios

1. Un sistema de caché registra cada solicitud como `H` si fue un *hit* y como `M` si fue un *miss*. Para proteger la base de datos, consideraremos válidas sólo las trazas en las que nunca aparecen tres `M` consecutivas. Sea a_n el número de trazas válidas de largo n.

   a) Modele el problema usando tres sucesiones auxiliares, según si la traza termina en `H`, en una única `M`, o en dos `M` consecutivas.

b) Escriba un sistema de recurrencias para esas tres sucesiones, incluyendo condiciones iniciales.

c) Deduzca una recurrencia sólo para a_n.

2. Sea f_n la sucesión de Fibonacci. Es decir, $f_0 = 0$, $f_1 = 1$, y $f_n = f_{n-1} + f_{n-2}$ para todo $n \geq 2$.

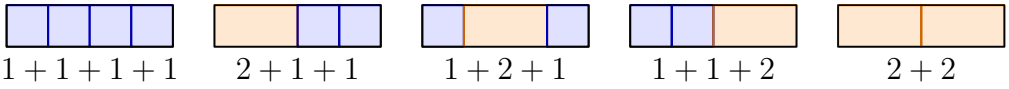
   a) Muestre que el número de formas de teselar un tablero de $1 \times n$ con fichas de tamaño 1×1 y 1×2 es f_{n+1}.

   b) Muestre que el número de subconjuntos de $\{1, 2, \ldots, n-1\}$ sin elementos consecutivos es también f_{n+1}.

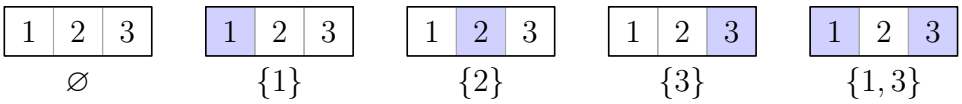
   c) Muestre que

$$\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k} = f_{n+1}.$$

   Para $n = 4$, tenemos $f_5 = 5$. Las cinco teselaciones de un tablero de 1×4 son:



   Para el mismo valor de f_5, los subconjuntos de $\{1, 2, 3\}$ sin elementos consecutivos son:



3. En el problema de Josephus hay n personas sentadas en círculo, etiquetadas $1, 2, \ldots, n$. Recorremos el círculo eliminando una de cada dos personas, empezando por eliminar a la persona 2. El proceso continúa hasta que queda una única persona. Sea j_n la etiqueta de la persona sobreviviente.

   a) Calcule $j_1, j_2, j_3, j_4, j_5, j_6$ simulando el proceso.

   b) Explique por qué, después de la primera vuelta, si $n = 2m$ entonces las personas que sobreviven tienen etiquetas impares.

   c) Deduzca que

$$j_1 = 1, \qquad j_{2m} = 2j_m - 1, \qquad j_{2m+1} = 2j_m + 1.$$

   d) Compute varios valores de j_n y conjeture una fórmula cerrada para j_n.

4. Sea u_n el número de formas de cubrir un tablero de $3 \times n$ con dominós de tamaño 2×1. Para modelar el problema, sea v_n el número de formas de cubrir un tablero de $3 \times n$ al que se le ha quitado la casilla superior derecha, para $n \geq 1$. Fijamos además $v_0 = 0$ por convención.

   a) Determine u_0, u_1, v_0, v_1.

   b) Analice cómo puede cubrirse la última columna de un tablero completo y derive una ecuación para u_n en términos de valores anteriores de u y v.

c) Analice cómo puede completarse el tablero con la casilla superior derecha removida y derive una ecuación para v_n.

d) Concluya que, para $n \geq 2$, se obtiene el sistema

$$u_n = u_{n-2} + 2v_{n-1}, \qquad v_n = u_{n-1} + v_{n-2}.$$

5. Sea $(c_n)_{n\geq 0}$ la sucesión definida por

$$c_0 = 1, \qquad c_n = \sum_{i=0}^{n-1} c_i c_{n-1-i} \quad \text{para todo } n \geq 1.$$

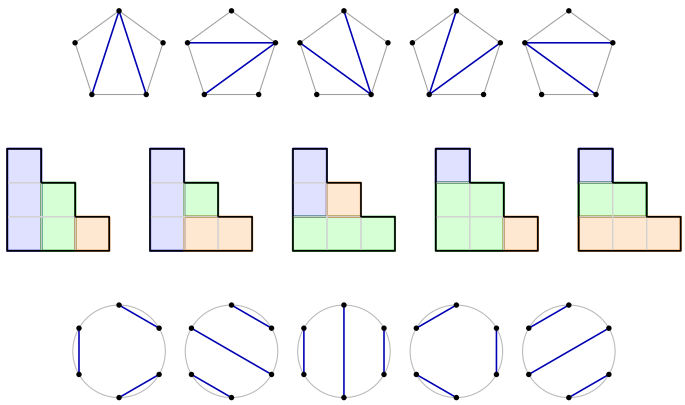
Muestre que las siguientes familias de objetos están contadas por c_n:

a) las triangulaciones de un polígono regular con $n+2$ lados.

b) las formas de teselar una escalera de n peldaños con exactamente n rectángulos, donde la escalera está formada por filas de longitudes $n, n-1, \ldots, 1$ y los rectángulos siguen las líneas de la grilla;

c) las formas de unir $2n$ puntos en una circunferencia mediante n cuerdas que no se intersectan entre sí.

Para $n = 3$, las cinco posibilidades en cada familia son:



4.7. Métodos básicos

En la sección anterior vimos cómo modelar problemas de conteo usando relaciones de recurrencia. En esta sección buscamos *resolverlas*, es decir, encontrar fórmulas cerradas para las sucesiones que aparecen. Notemos que esto vuelve a las recurrencias en poderosas herramientas para resolver problemas de conteo, primero modelando con recurrencias, y luego resolviendo esas recurrencias. Partiremos con tres métodos básicos: inspección, expansión y cambio de variable.

4.7.1. Inspección

El método más directo es calcular algunos términos, reconocer un patrón y luego demostrar por inducción que el patrón funciona. No siempre podemos reconocer a partir de pocos términos, pero cuando la sucesión es conocida o los primeros términos son sugerentes, suele ser el camino más corto.

Ejemplo (Torres de Hanoi por inspección)**.** Sea h_n la mínima cantidad de movimientos necesarios para resolver Torres de Hanoi con n discos. Recordemos que teníamos la recurrencia

$$h_n = 2h_{n-1} + 1,$$

con $h_0 = 0$. Los primeros términos son

n	0	1	2	3	4	5	6	7
h_n	0	1	3	7	15	31	63	127

Esto sugiere la fórmula $h_n = 2^n - 1$, que verificaremos por inducción. En efecto, para $n = 0$, se tiene $h_0 = 0 = 2^0 - 1$. La hipótesis inductiva es que $h_{n-1} = 2^{n-1} - 1$, entonces

$$h_n = 2h_{n-1} + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1.$$

Por lo tanto, $h_n = 2^n - 1$ para todo $n \geq 0$.

4.7.2. Expansión

En expansión la idea es aplicar la recurrencia repetidamente para escribir a_n en términos de a_{n-2}, luego a_{n-3}, y así sucesivamente, hasta llegar al caso base. Muchas veces, el resultado de la expansión es una suma que se puede resolver usando técnicas para calcular sumas o de conteo.

Ejemplo (Recurrencia suma)**.** La recurrencia prototipo de este método es de la forma

$$s_n = s_{n-1} + f(n),$$

con s_0 dado. Por ejemplo, consideremos la recurrencia

$$s_n = s_{n-1} + n,$$

con $s_0 = 0$. Al expandir,

$$\begin{aligned} s_n &= s_{n-1} + n \\ &= s_{n-2} + (n-1) + n \\ &= s_{n-3} + (n-2) + (n-1) + n \\ &\vdots \\ &= s_0 + \sum_{i=1}^{n} i. \end{aligned}$$

Como $s_0 = 0$, obtenemos $s_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Ejemplo (Recurrencia producto)**.** Otra recurrencia prototipo para este método es de la forma

$$p_n = c_n p_{n-1}.$$

Que nos lleva a una productoria al expandir. Por ejemplo, consideremos la recurrencia

$$p_n = 3p_{n-1}, \quad \text{con} p_0 = 1.$$

Al expandir obtenemos:

$$p_n = 3p_{n-1} = 3(3p_{n-2}) = 3^2 p_{n-2} = 3^3 p_{n-3} = \cdots = 3^n p_0 = 3^n.$$

Ejemplo (Recurrencia de MergeSort)**.** El algoritmo de MergeSort es un algoritmo de ordenamiento que divide el arreglo a ordenar en dos mitades, ordena cada mitad recursivamente, y luego mezcla las mitades ordenadas. Como la cantidad de operaciones para mezclar dos arreglos de tamaño $n/2$ es $\gamma \cdot n$ para alguna constante γ, la cantidad total de operaciones satisface

$$m_n = 2m_{n/2} + \gamma \cdot n,$$

con $m_1 = \gamma$. Supondremos que n es potencia de 2 para simplificar la discusión. Al expandir una vez,

$$m_n = 2\left(2m_{n/4} + \gamma \cdot \frac{n}{2}\right) + \gamma \cdot n = 4m_{n/4} + 2\gamma \cdot n.$$

Después de i expansiones,

$$m_n = 2^i m_{n/2^i} + i \cdot \gamma n.$$

Tomando $i = b = \log_2 n$, llegamos a $n/2^b = 1$, y por lo tanto

$$m_n = nm_1 + \gamma n \log_2 n = \gamma n + \gamma n \log_2 n.$$

4.7.3. Cambio de variable

La técnica del cambio de variable busca transformar la recurrencia original en otra más sencilla, a través de una función auxiliar. La idea es sumar o multiplicar por una función que cancele parte de la recurrencia, obteniendo una nueva recurrencia que se pueda resolver de manera más directa.

Ejemplo (Torres de Hanoi por cambio de variable I)**.** Consideremos nuevamente la recurrencia de Torres de Hanoi

$$h_n = 2h_{n-1} + 1,$$

con $h_0 = 0$. Sumemos 1 a ambos lados de la recurrencia, obteniendo

$$h_n + 1 = 2h_{n-1} + 2 = 2(h_{n-1} + 1).$$

Esto sugiere definir una nueva sucesión $u_n = h_n + 1$, que satisface $u_n = 2u_{n-1}$ y $u_0 = 1$. Por expansión, $u_n = 2^n u_0 = 2^n$, así que $h_n = u_n - 1 = 2^n - 1$.

Podemos generalizar esta técnica a recurrencias de la forma $a_n = ca_{n-1} + f(n)$, con c constante. Si $a_n = ca_{n-1} + f(n)$, entonces $b_n = a_n + g(n)$ satisface

$$b_n = cb_{n-1} + f(n) + g(n) - cg(n-1).$$

Así, si elegimos $g(n)$ tal que $f(n) + g(n) - cg(n-1) = 0$, entonces $b_n = cb_{n-1}$, que se puede resolver por expansión.

Ejemplo (Torres de Hanoi por cambio de variable II)**.** Veremos un segundo método para resolver la misma recurrencia, esta vez escalando la recurrencia. Dividimos la recurrencia por 2^n:

$$\frac{h_n}{2^n} = \frac{2h_{n-1}}{2^n} + \frac{1}{2^n} = \frac{h_{n-1}}{2^{n-1}} + \frac{1}{2^n}.$$

Ahora definimos $v_n = \frac{h_n}{2^n}$. La recurrencia se transforma en

$$v_n = v_{n-1} + \frac{1}{2^n},$$

con $v_0 = 0$. Esta es una recurrencia tipo suma, así que expandiendo obtenemos

$$v_n = \underbrace{\sum_{i=1}^{n} \frac{1}{2^i}}_{\text{geométrica}} = 1 - \frac{1}{2^n}.$$

Deshaciendo el cambio de variable nos queda $h_n = 2^n v_n = 2^n - 1$.

Podemos generalizar un poco esta técnica a recurrencias de la forma $a_n = ca_{n-1} + f(n)$, con c constante. Si $a_n = ca_{n-1} + f(n)$, entonces $b_n = c^{-n}a_n$ satisface

$$b_n = b_{n-1} + c^{-n}f(n),$$

que se puede resolver por expansión.

4.8. Polinomio característico

Los métodos anteriores funcionan bien en muchos casos particulares. Por ejemplo vimos en los ejemplos anteriores que las recurrencias de la forma $a_n = ca_{n-1} + f(n)$ se pueden resolver por cambio de variable.

Sin embargo, estos métodos no son tan automáticos: muchas veces no es fácil reconocer el patrón o encontrar un cambio de variable adecuado. Para esto, tenemos una gran clase de recurrencias que se pueden resolver sistemáticamente usando el polinomio característico.

Definición. Una relación de recurrencia es *lineal homogénea con coeficientes constantes* si tiene la forma

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} \quad \text{para todo } n \geq k,$$

donde $c_1, \ldots, c_k \in \mathbb{R}$ y $c_k \neq 0$. En este caso diremos que la recurrencia tiene *grado* k.

Ejemplo. Algunos ejemplos y contraejemplos son los siguientes.

- $a_n = 3a_{n-1} + 2a_{n-3}$ es una recurrencia lineal homogénea con coeficientes constantes de grado 3.
- $a_n = a_{n-1}^2$ no es lineal.
- $a_n = a_{n-1} + 1$ no es homogénea.

- $a_n = na_{n-1}$ no tiene coeficientes constantes.

La principal idea para resolver este tipo de recurrencias es buscar soluciones de la forma $a_n = r^n$. Notemos que este tipo de soluciones aparece naturalmente como parte de las soluciones de varias recurrencias.

Notemos que si probamos una solución de la forma $a_n = r^n$, para algún $r \in \mathbb{R}$, entonces r debe satisfacer

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k}.$$

(Por esto, también se conoce a este método como el método del *ansatz exponencial*.) Esto da una ecuación polinomial en r. Formalizamos esto con la noción de polinomio característico, que nos dará los candidatos exponenciales para la solución de la recurrencia.

Definición. Dada la recurrencia

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k},$$

su *polinomio característico* es

$$\chi(r) = r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k.$$

Las raíces de $\chi(r)$ son los candidatos exponenciales para la solución.

Ejemplo. Para la recurrencia $a_n = 3a_{n-1} + 2a_{n-3}$, el polinomio característico es $\chi(r) = r^3 - 3r^2 - 2$. Las raíces de este polinomio son -1, 1 y 2, por lo que los candidatos exponenciales para la solución de la recurrencia son $(-1)^n$, 1^n y 2^n.

¿Cómo podemos convertir los candidatos de solución en la solución general de la recurrencia? Para esto, notamos que cada raíz del polinomio representa uno de los grados de libertad que tenemos para elegir la solución. Por lo tanto, si el polinomio característico tiene k raíces distintas, entonces las potencias de esas raíces nos dan k candidatos independientes, que podemos combinar de manera ponderada para obtener la solución general.

Teorema (Raíces distintas)**.** Sea

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

una recurrencia lineal homogénea con coeficientes constantes. Si su polinomio característico tiene k raíces distintas $r_1, \ldots, r_k$, entonces la solución general tiene la forma

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \cdots + \alpha_k r_k^n,$$

donde las constantes $\alpha_1, \ldots, \alpha_k$ se determinan usando las condiciones iniciales.

Podremos probar este resultado con la maquinaria que desarrollaremos en las siguientes secciones.[1] Por ahora, veamos cómo usarlo para resolver recurrencias.

[1] Uno también puede demostrar esto vía técnicas de álgebra lineal.

Ejemplo. La sucesión de Torres de Hanoi también satisface

$$h_n = 3h_{n-1} - 2h_{n-2},$$

con $h_0 = 0$ y $h_1 = 1$. El polinomio característico es $\chi(r) = r^2 - 3r + 2 = (r-1)(r-2)$. Las raíces son 1 y 2, por lo que

$$h_n = \alpha_1 \cdot 1^n + \alpha_2 \cdot 2^n = \alpha_1 + \alpha_2 2^n.$$

Usando las condiciones iniciales,

$$0 = \alpha_1 + \alpha_2, \qquad 1 = \alpha_1 + 2\alpha_2.$$

Luego $\alpha_1 = -1$ y $\alpha_2 = 1$, así que

$$h_n = 2^n - 1.$$

4.8.1. Raíces repetidas

Si una raíz aparece repetida, perdemos candidatos exponenciales distintos. La manera de recuperar los grados de libertad es multiplicar por potencias de n. Así, una raíz r de multiplicidad m aporta

$$r^n, \quad nr^n, \quad n^2r^n, \quad \ldots, \quad n^{m-1}r^n.$$

Teorema (Solución general)**.** Sea

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

una recurrencia lineal homogénea con coeficientes constantes. Supongamos que su polinomio característico tiene raíces distintas $r_1, \ldots, r_t$, con multiplicidades $m_1, \ldots, m_t$. Entonces la solución general tiene la forma

$$a_n = \sum_{i=1}^{t} \left(\alpha_{i,0} + \alpha_{i,1} n + \cdots + \alpha_{i,m_i-1} n^{m_i-1}\right) r_i^n.$$

Donde las constantes $\alpha_{i,j}$ se determinan usando las condiciones iniciales.

Nuevamente, podremos probar este resultado con la maquinaria que desarrollaremos en las siguientes secciones.[2]

Ejemplo. Consideremos $a_n = 4a_{n-1} - 4a_{n-2}$, con $a_0 = 1$ y $a_1 = 4$. El polinomio característico es $\chi(r) = r^2 - 4r + 4 = (r-2)^2$. Entonces 2 es una raíz doble, y la solución general tiene la forma

$$a_n = \alpha_1 2^n + \alpha_2 n 2^n.$$

Usando las condiciones iniciales,

$$1 = \alpha_1, \qquad 4 = 2\alpha_1 + 2\alpha_2.$$

Por lo tanto $\alpha_1 = 1$ y $\alpha_2 = 1$, así que

$$a_n = (n+1)2^n.$$

[2]También se puede demostrar vía técnicas de álgebra lineal.

4.9. Recurrencias lineales no homogéneas

Muchas recurrencias naturales tienen además un término externo. Por ejemplo, en Torres de Hanoi, además de resolver dos subproblemas de tamaño menor, hay que mover el disco más grande una vez. Ese término adicional rompe la homogeneidad, pero todavía podemos usar la parte homogénea como punto de partida.

Definición. Una relación de recurrencia es *lineal no homogénea con coeficientes constantes* si tiene la forma

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n),$$

donde $c_1, \ldots, c_k \in \mathbb{R}$, $c_k \neq 0$, y $f(n)$ es una función dada. La *recurrencia homogénea asociada* se obtiene quitando el término $f(n)$.

Para resolver este tipo de recurrencias, la idea es encontrar una solución particular $a_n^{(p)}$ que satisfaga la recurrencia no homogénea, y luego sumar esa solución particular a la solución general de la homogénea asociada $a_n^{(g)}$. De esta manera, obtenemos la solución general de la recurrencia no homogénea.

Teorema. La solución general de una recurrencia lineal no homogénea con coeficientes constantes es

$$a_n = a_n^{(g)} + a_n^{(p)},$$

donde $a_n^{(g)}$ es la solución general de la homogénea asociada, y $a_n^{(p)}$ es una solución particular de la recurrencia no homogénea.

Demostración. Sea $a_n^{(p)}$ una solución particular de la recurrencia no homogénea, y sea b_n cualquier otra solución de la misma recurrencia. Entonces

$$a_n^{(p)} = c_1 a_{n-1}^{(p)} + \cdots + c_k a_{n-k}^{(p)} + f(n)$$

y

$$b_n = c_1 b_{n-1} + \cdots + c_k b_{n-k} + f(n).$$

Restando ambas igualdades, obtenemos que $b_n - a_n^{(p)}$ satisface la recurrencia homogénea asociada. Por lo tanto, $b_n - a_n^{(p)} = a_n^{(g)}$ para alguna solución homogénea $a_n^{(g)}$. Así, $b_n = a_n^{(g)} + a_n^{(p)}$. □

Ejemplo (Torres de Hanoi)**.** La cantidad mínima de movimientos para resolver Torres de Hanoi con n discos satisface

$$h_n = 2h_{n-1} + 1$$

para todo $n \geq 1$, con $h_0 = 0$. La homogénea asociada es $h_n = 2h_{n-1}$, cuya solución general es $h_n^{(g)} = \alpha 2^n$. Buscamos una solución particular constante $h_n^{(p)} = \gamma$. Sustituyendo,

$$\gamma = 2\gamma + 1,$$

de donde $\gamma = -1$. Luego $h_n = \alpha 2^n - 1$. Usando $h_0 = 0$ obtenemos $\alpha = 1$, y por lo tanto

$$h_n = 2^n - 1.$$

4.9.1. Encontrando soluciones particulares

Surge una pregunta natural: ¿cómo encontramos una solución particular? Para ciertos términos no homogéneos, podemos proponer una forma con coeficientes desconocidos y luego determinarlos sustituyendo en la recurrencia.

Teorema. Consideremos una recurrencia lineal no homogénea

$$a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k} + f(n),$$

donde $f(n) = p(n)s^n$ para algún polinomio p de grado t y alguna constante s. Entonces, para encontrar una solución particular, tenemos dos casos:

- Si s no es raíz del polinomio característico de la homogénea asociada, entonces existe una solución particular de la forma $q(n)s^n$, donde q es un polinomio de grado t.
- Si s es raíz de multiplicidad m, entonces existe una solución particular de la forma $n^m q(n)s^n$, donde q es un polinomio de grado t.

Ejemplo. Encontremos la solución de $a_n = 3a_{n-1} - 4n$, con $a_0 = 4$. La homogénea asociada es $a_n = 3a_{n-1}$, así que $a_n^{(g)} = \alpha 3^n$. Como el término no homogéneo es

$$f(n) = -4n = (-4n + 0)1^n,$$

buscamos una solución particular de la forma $a_n^{(p)} = cn + d$. Sustituyendo en la recurrencia,

$$cn + d = 3(c(n-1) + d) - 4n.$$

Al expandir e igualar coeficientes,

$$cn + d = (3c - 4)n + (3d - 3c),$$

por lo que

$$c = 3c - 4, \qquad d = 3d - 3c.$$

Concluimos que $c = 2$ y $d = 3$. Por lo tanto una solución particular es $2n + 3$, y la solución general es

$$a_n = \alpha 3^n + 2n + 3.$$

Usando la condición inicial, $4 = \alpha + 3$, así que $\alpha = 1$. Por lo tanto,

$$a_n = 3^n + 2n + 3.$$

4.9.2. Una recurrencia, tres métodos

Cerremos la sección comparando los métodos en una misma recurrencia. Consideremos

$$x_n = 2x_{n-1} + n,$$

con $x_0 = -1$. Con esta condición inicial, la recurrencia determina una única sucesión.

Ejemplo (Cambio de variable)**.** La parte no homogénea es lineal en n, así que buscamos un cambio que absorba ese término. Definimos

$$y_n = x_n + n + 2.$$

Entonces

$$\begin{aligned} y_n &= x_n + n + 2 \\ &= 2x_{n-1} + n + n + 2 \\ &= 2x_{n-1} + 2n + 2 \\ &= 2(x_{n-1} + n + 1) \\ &= 2y_{n-1}. \end{aligned}$$

Además $y_0 = x_0 + 2 = 1$. Por lo tanto, $y_n = 2^n$. Volviendo a x_n, obtenemos

$$x_n = y_n - n - 2 = 2^n - n - 2.$$

Ejemplo (Expansión)**.** Expandimos la recurrencia:

$$\begin{aligned} x_n &= 2x_{n-1} + n \\ &= 2(2x_{n-2} + (n-1)) + n \\ &= 2^2 x_{n-2} + 2(n-1) + n \\ &= 2^3 x_{n-3} + 2^2(n-2) + 2(n-1) + n \\ &\vdots \\ &= -2^n + \sum_{i=1}^{n} 2^{n-i} i. \end{aligned}$$

Factorizando 2^n, tenemos

$$x_n = 2^n \left(-1 + \sum_{i=1}^{n} \frac{i}{2^i} \right).$$

Ahora resolvemos la suma que apareció. Sea

$$r_n = \sum_{i=1}^{n} \frac{i}{2^i} = \frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \cdots + \frac{n}{2^n}.$$

Multiplicando por $\frac{1}{2}$,

$$\frac{1}{2} r_n = \frac{1}{2^2} + \frac{2}{2^3} + \frac{3}{2^4} + \cdots + \frac{n}{2^{n+1}}.$$

Restando ambas igualdades,

$$\begin{aligned} \frac{1}{2} r_n &= \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^n} - \frac{n}{2^{n+1}} \\ &= 1 - \frac{1}{2^n} - \frac{n}{2^{n+1}}. \end{aligned}$$

Por lo tanto,

$$r_n = 2 - \frac{n+2}{2^n}.$$

Luego

$$x_n = 2^n \left(1 - \frac{n+2}{2^n}\right) = 2^n - n - 2.$$

Ejemplo (Polinomio característico)**.** La homogénea asociada es $x_n = 2x_{n-1}$. Su polinomio característico es $\chi(r) = r - 2$, así que la solución general de la homogénea es $x_n^{(g)} = \alpha 2^n$. El término no homogéneo es n, que tiene la forma $p(n)1^n$ con p de grado 1. Como 1 no es raíz de $\chi(r)$, buscamos una solución particular de la forma

$$x_n^{(p)} = cn + d.$$

Sustituyendo, obtenemos que

$$cn + d = 2(c(n-1) + d) + n = (2c+1)n + (2d - 2c).$$

Igualando coeficientes, $c = 2c + 1$ y $d = 2d - 2c$. De aquí $c = -1$ y $d = -2$, por lo que

$$x_n = \alpha 2^n - n - 2.$$

Usando $x_0 = -1$, tenemos $-1 = \alpha - 2$, luego $\alpha = 1$. Finalmente,

$$x_n = 2^n - n - 2.$$

Ejercicios

1. Un algoritmo procesa un arreglo de largo n de la siguiente manera: primero procesa recursivamente los primeros $n-1$ elementos y luego hace una pasada final sobre el arreglo. En esa pasada realiza dos operaciones por cada elemento y una operación adicional para guardar el resultado. Sea t_n el número total de operaciones que realiza el algoritmo sobre un arreglo de largo n.

   a) Justifique que $t_0 = 0$ y que, para $n \geq 1$, se tiene

   $$t_n = t_{n-1} + 2n + 1.$$

   b) Resuelva esta recurrencia por expansión.

2. *a*) Resuelva vía dos cambios de variables distintos la recurrencia

   $$a_n = 3a_{n-1} + 2,$$

   con $a_0 = 0$.

   b) Resuelva de tres maneras distintas la recurrencia

   $$b_n = b_{n-1} + b_{n-2} - b_{n-3},$$

   con $b_0 = 0$, $b_1 = 1$ y $b_2 = 4$.

   c) Resuelva

   $$c_n = 2c_{n-1} + 2^n,$$

   con $c_0 = 0$, vía polinomio característico.

3. En este ejercicio veremos una demostración guíada del método del polinomio característico para recurrencias homogéneas de grado 2. Consideremos la recurrencia

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

para $n \geq 2$, con $c_2 \neq 0$, y su polinomio característico

$$\chi(r) = r^2 - c_1 r - c_2.$$

a) Suponga que r es raíz de χ. Demuestre que la sucesión $b_n = r^n$ satisface la recurrencia homogénea.

b) Suponga que χ tiene dos raíces distintas r_1 y r_2. Demuestre que, para cualquier par de constantes α, β, la sucesión

$$b_n = \alpha r_1^n + \beta r_2^n$$

satisface la recurrencia.

c) Dados valores iniciales a_0 y a_1, escriba el sistema que deben satisfacer α y β para que

$$a_n = \alpha r_1^n + \beta r_2^n$$

tenga esos valores iniciales. Explique por qué el sistema tiene solución única cuando $r_1 \neq r_2$.

d) Concluya que, si las raíces son distintas, las soluciones de la recurrencia tienen la forma

$$a_n = \alpha r_1^n + \beta r_2^n.$$

e) Suponga ahora que χ tiene una raíz repetida r_*. Muestre que $c_1 = 2r_*$ y $c_2 = -r_*^2$. Luego verifique directamente que la sucesión

$$b_n = n r_*^n$$

también satisface la recurrencia.

f) Concluya que, en el caso de raíz repetida, la forma de la solución es

$$a_n = \alpha r_*^n + \beta n r_*^n.$$

4.10. Funciones generatrices

Hasta ahora hemos descrito sucesiones de varias maneras: listando sus primeros términos, dando una recurrencia, o encontrando una fórmula cerrada. Por ejemplo, la sucesión de las torres de Hanoi se puede describir de las siguientes maneras:

- Listando sus primeros términos: $0, 1, 3, 7, 15, 31, \ldots$.
- Dando una recurrencia: $h_0 = 0$ y $h_{n+1} = 2h_n + 1$ para todo $n \geq 0$.
- Encontrando una fórmula cerrada: $h_n = 2^n - 1$ para todo $n \geq 0$.

Las funciones generatrices ofrecen una cuarta descripción. La idea es empaquetar toda la información de una sucesión en un objeto algebraico.

Definición. Sea $(a_n)_{n\geq 0}$ una sucesión. La *función generatriz (ordinaria)* de $(a_n)_{n\geq 0}$ es la serie formal

$$A(x) = \sum_{n\geq 0} a_n x^n.$$

Por ejemplo, para algunas sucesiones que ya conocemos, sus funciones generatrices comienzan así:

$$\begin{aligned} H(x) &= x + 3x^2 + 7x^3 + 15x^4 + 31x^5 + \cdots && \text{(Torres de Hanoi)},\\ F(x) &= x + x^2 + 2x^3 + 3x^4 + 5x^5 + 8x^6 + \cdots && \text{(Fibonacci)},\\ C(x) &= 1 + x + 2x^2 + 5x^3 + 14x^4 + 42x^5 + \cdots && \text{(Catalán)}. \end{aligned}$$

Recalcamos que en esta definición, x es una variable formal y $A(x)$ es una serie formal. Esto significa que no nos haremos preguntas sobre la convergencia de la serie, ni sobre su valor para algún x específico. Podemos pensar en estas series como polinomios con infinitos términos: lo importante no es evaluarlas, sino operar algebraicamente con sus coeficientes.

¿Cuál es la gracia de esta definición? Al tener esta interpretación algebraica, podemos usar herramientas de álgebra para manipular funciones generatrices y obtener información sobre las sucesiones que codifican. Intuitivamente, estas operaciones obedecen las mismas reglas que las operaciones sobre polinomios. En particular, consideraremos las siguientes operaciones sobre funciones generatrices.

Definición. Sean $A(x)$ y $B(x)$ funciones generatrices de las sucesiones $(a_n)_{n\geq 0}$ y $(b_n)_{n\geq 0}$ respectivamente. Definimos la *suma*, *multiplicación por escalar* $\lambda \in \mathbb{R}$ y *multiplicación por monomio* x^k (con $k \in \mathbb{N}$) de funciones generatrices como sigue:

$$\begin{aligned} A(x) + B(x) &= \sum_{n\geq 0} (a_n + b_n) x^n,\\ \lambda A(x) &= \sum_{n\geq 0} \lambda a_n x^n,\\ x^k A(x) &= \sum_{n\geq k} a_{n-k} x^n. \end{aligned}$$

Estas operaciones corresponden a operaciones naturales sobre sucesiones: la suma de funciones generatrices corresponde a la suma término a término de las sucesiones, la multiplicación por escalar corresponde a multiplicar cada término de la sucesión por el escalar, y la multiplicación por un monomio x^k corresponde a desplazar la sucesión k lugares hacia la derecha (rellenando con ceros los primeros k términos).

Operaremos con estas operaciones tal como si fueran las operaciones sobre polinomios (i.e., respetan asociatividad, conmutatividad, distributividad, etc.). Dichas operaciones nos permitirán obtener conclusiones sobre las sucesiones codificadas por las funciones generatrices.

Ejemplo. Consideremos la sucesión constante $a_n = 1$ para todo $n \geq 0$. Su función generatriz es

$$A(x) = \sum_{n\geq 0} x^n.$$

Podemos multiplicar por x y obtener otra función generatriz:

$$xA(x) = \sum_{n\geq 0} x^{n+1} = \sum_{n\geq 1} x^n = A(x) - 1.$$

De manera interesante, hemos obtenido una ecuación algebraica para $A(x)$:

$$xA(x) = A(x) - 1.$$

Despejando, obtenemos que $A(x) = \frac{1}{1-x}$.

Ejemplo (Sucesión constante)**.** Podemos usar la serie anterior para encontrar la función generatriz de la sucesión $a_n = c$ para todo $n \geq 0$:

$$A(x) = \sum_{n\geq 0} cx^n = c\sum_{n\geq 0} x^n = \frac{c}{1-x}.$$

Ejemplo (Sucesión exponencial)**.** De manera similar, la función generatriz de la sucesión $a_n = c^n$ para todo $n \geq 0$ es

$$A(x) = \sum_{n\geq 0} c^n x^n = \sum_{n\geq 0} (cx)^n = \frac{1}{1-cx}.$$

Ejemplo (Sucesión binomial)**.** Consideremos ahora la sucesión $a_n = \binom{m}{n}$ para $n \geq 0$. Notemos que $\binom{m}{n} = 0$ para $n > m$, así que la función generatriz es un polinomio de grado m:

$$A(x) = \sum_{n\geq 0} \binom{m}{n} x^n = \underbrace{\sum_{n=0}^{m} \binom{m}{n} x^n}_{\text{Teo. del binomio}} = (1+x)^m.$$

4.10.1. Resolviendo recurrencias vía funciones generatrices

Una aplicación central de las funciones generatrices es transformar una recurrencia en una ecuación algebraica. Luego despejamos la función generatriz y recuperamos los coeficientes. Veamos el procedimiento completo con una recurrencia con la cual ya estamos familiarizados: la de las torres de Hanoi.

Ejemplo. Sea $(h_n)_{n\geq 0}$ la sucesión de las torres de Hanoi, definida por

$$h_0 = 0, \qquad h_{n+1} = 2h_n + 1 \quad \text{para todo } n \geq 0.$$

Sea

$$H(x) = \sum_{n\geq 0} h_n x^n.$$

Multiplicamos la recurrencia por x^{n+1} y sumamos para $n \geq 0$:

$$\sum_{n\geq 0} h_{n+1} x^{n+1} = 2\sum_{n\geq 0} h_n x^{n+1} + \sum_{n\geq 0} x^{n+1}$$

El lado izquierdo es

$$\sum_{n\geq 0} h_{n+1}x^{n+1} = \sum_{n\geq 1} h_n x^n = H(x) - h_0 = H(x).$$

El lado derecho es

$$2\sum_{n\geq 0} h_n x^{n+1} + \sum_{n\geq 0} x^{n+1} = 2xH(x) + \frac{x}{1-x}.$$

Por lo tanto,

$$H(x) = 2xH(x) + \frac{x}{1-x}.$$

Despejando,

$$H(x) = \frac{x}{(1-x)(1-2x)}.$$

Hemos obtenido una fórmula para $H(x)$, la pregunta es: ¿cómo recuperamos h_n a partir de esta fórmula? Tratemos de reescribir $H(x)$ como funciones generatrices conocidas. Para esto, hacemos fracciones parciales.

$$\frac{x}{(1-x)(1-2x)} = x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right).$$

Como

$$\frac{1}{1-2x} = \sum_{n\geq 0} 2^n x^n \quad \text{y} \quad \frac{1}{1-x} = \sum_{n\geq 0} x^n,$$

obtenemos

$$H(x) = \sum_{n\geq 1}(2^n - 1)x^n = \sum_{n\geq 0}(2^n - 1)x^n.$$

Por lo tanto $h_n = 2^n - 1$ para todo $n \geq 0$.

Notemos que el procedimiento anterior se puede resumir en los siguientes pasos:

- definir $A(x) = \sum_{n\geq 0} a_n x^n$;
- multiplicar la recurrencia por una potencia conveniente de x (usualmente, si el mayor índice de la recurrencia es $n + k$, multiplicar por x^{n+k});
- sumar sobre los índices donde la recurrencia vale;
- reescribir todo en términos de $A(x)$ y despejar;
- expandir $A(x)$ como serie y leer sus coeficientes.

Notemos que el paso de expandir $A(x)$ como serie suele ser el más difícil. Muchas veces nos aparecerán funciones que buscamos reescribir como funciones generatrices conocidas. Técnicas como fracciones parciales (que es lo que hicimos en el ejemplo) nos ayudarán a esto.

Ejemplo (Fibonacci)**.** Sea $(f_n)_{n\geq 0}$ la sucesión de Fibonacci:

$$f_0 = 0, \qquad f_1 = 1, \qquad f_{n+1} = f_n + f_{n-1} \quad \text{para todo } n \geq 1.$$

Definamos

$$F(x) = \sum_{n\geq 0} f_n x^n.$$

Multiplicando por x^{n+1} y sumando desde $n = 1$, tenemos

$$\sum_{n\geq 1} f_{n+1}x^{n+1} = \sum_{n\geq 1} f_n x^{n+1} + \sum_{n\geq 1} f_{n-1}x^{n+1}.$$

El lado izquierdo es

$$\sum_{n\geq 1} f_{n+1}x^{n+1} = \sum_{n\geq 2} f_n x^n = F(x) - f_0 - f_1 x = F(x) - x.$$

El lado derecho es

$$\sum_{n\geq 1} f_n x^{n+1} + \sum_{n\geq 1} f_{n-1}x^{n+1} = x\sum_{n\geq 1} f_n x^n + x^2 \sum_{n\geq 1} f_{n-1}x^{n-1} = x(F(x)-f_0)+x^2F(x) = xF(x)+x^2F(x).$$

Por lo tanto,

$$F(x) - x = xF(x) + x^2F(x).$$

Despejando,

$$F(x) = \frac{x}{1-x-x^2}.$$

Notemos que las soluciones de $1 - x - x^2 = 0$ son $\phi = \frac{1+\sqrt{5}}{2}$ y $\psi = \frac{1-\sqrt{5}}{2}$, así que vía fracciones parciales obtenemos

$$\frac{x}{1-x-x^2} = \frac{1}{\sqrt{5}}\left(\frac{1}{1-\phi x} - \frac{1}{1-\psi x}\right).$$

Como

$$\frac{1}{1-\phi x} = \sum_{n\geq 0} \phi^n x^n \quad \text{y} \quad \frac{1}{1-\psi x} = \sum_{n\geq 0} \psi^n x^n,$$

obtenemos

$$F(x) = \sum_{n\geq 0} \frac{\phi^n - \psi^n}{\sqrt{5}} x^n.$$

Por lo tanto,

$$f_n = \frac{\phi^n - \psi^n}{\sqrt{5}}.$$

4.10.2. Convolución y producto

Para ampliar nuestro repertorio de funciones generatrices, necesitamos una operación que nos permita combinar funciones generatrices conocidas para obtener funciones generatrices de sucesiones más complicadas. Esta operación es el producto de funciones generatrices. Notemos que este producto se puede deducir de las operaciones básicas que definimos antes, obteniendo la siguiente fórmula para el producto de funciones generatrices.

Proposición. Sean

$$A(x) = \sum_{n\geq 0} a_n x^n, \qquad B(x) = \sum_{n\geq 0} b_n x^n.$$

Entonces,

$$A(x)B(x) = \sum_{n\geq 0}\left(\sum_{k=0}^{n} a_k b_{n-k}\right)x^n.$$

Antes de demostrar esta fórmula, introduciremos la notación del corchete de Iverson, que nos ayudará a hacer manipulaciones de manera más compacta.

Definición. Para una proposición ϕ, definimos el *corchete de Iverson* $[\phi]$ como

$$[\phi] = \begin{cases} 1 & \text{si } \phi \text{ es verdadera,} \\ 0 & \text{si } \phi \text{ es falsa.} \end{cases}$$

Podemos continuar ahora con la demostración de la fórmula del producto de funciones generatrices.

Demostración. Notemos que $a_k x^k B(x) = \sum_{n\geq k} a_k b_{n-k} x^n$, así que

$$A(x)B(x) = \sum_{k\geq 0} a_k x^k B(x) = \sum_{k\geq 0}\sum_{n\geq k} a_k b_{n-k} x^n.$$

Intercambiando el orden de las sumas, obtenemos

$$A(x)B(x) = \sum_{k\geq 0}\sum_{n\geq 0}[n\geq k]a_k b_{n-k}x^n = \sum_{n\geq 0}\sum_{k=0}^{n} a_k b_{n-k}x^n. \qquad \square$$

Ejemplo. Tratemos de calcular la función generatriz $A(x)$ de la sucesión $a_n = n+1$ para todo $n \geq 0$. Notemos que $n+1 = \sum_{k=0}^{n} 1\cdot 1$, así que a_n es la convolución de la sucesión constante 1 con ella misma. Por tanto,

$$A(x) = \sum_{n\geq 0}(n+1)x^n = \sum_{n\geq 0}\left(\sum_{k=0}^{n} 1\cdot 1\right)x^n = \left(\sum_{n\geq 0} x^n\right)^2 = \frac{1}{(1-x)^2}.$$

Ejemplo. Sea $(a_n)_{n\geq 0}$ la sucesión definida por $a_0 = 1$ y, para $n \geq 1$,

$$2a_n + \sum_{k=1}^{n-1} a_k a_{n-k} = (n+1)2^n.$$

Esta recurrencia es no lineal, porque aparecen productos de términos de la sucesión. Como $a_0 = 1$, podemos reescribirla de manera más simétrica como

$$\sum_{k=0}^{n} a_k a_{n-k} = (n+1)2^n \quad \text{para todo } n \geq 0.$$

Definamos

$$A(x) = \sum_{n\geq 0} a_n x^n.$$

Multiplicando por x^n y sumando sobre $n \geq 0$, obtenemos

$$\sum_{n\geq 0}\left(\sum_{k=0}^{n} a_k a_{n-k}\right)x^n = \sum_{n\geq 0}(n+1)2^n x^n.$$

Por la fórmula de convolución, el lado izquierdo es $A(x)^2$. Por otra parte,

$$\sum_{n\geq 0}(n+1)2^n x^n = \sum_{n\geq 0}\sum_{k=0}^{n} 2^k 2^{n-k} x^n = \left(\sum_{n\geq 0} 2^n x^n\right)^2 = \frac{1}{(1-2x)^2}.$$

Por lo tanto,

$$A(x)^2 = \frac{1}{(1-2x)^2}.$$

Tenemos entonces dos posibles valores para $A(x)$:

$$A(x) = \frac{1}{1-2x} \quad \text{o} \quad A(x) = \frac{-1}{1-2x}.$$

Las sucesiones codificadas por estas funciones generatrices son $a_n = 2^n$ y $a_n = -2^n$ respectivamente. Como $a_0 = 1$, concluimos que $A(x) = \frac{1}{1-2x}$ y $a_n = 2^n$ para todo $n \geq 0$.

Concluimos esta sección con un catálogo de funciones generatrices ordinarias básicas, que nos servirán como bloques de construcción para obtener funciones generatrices de sucesiones más complicadas.

Proposición (Catálogo de FGO)**.**

- $a_n = c \leftrightarrow A(x) = \frac{c}{1-x}$.
- $a_n = (-1)^n \leftrightarrow A(x) = \frac{1}{1+x}$.
- $a_n = r^n \leftrightarrow A(x) = \frac{1}{1-rx}$.
- $a_n = \binom{m}{n} \leftrightarrow A(x) = (1+x)^m$.
- $a_n = n+1 \leftrightarrow A(x) = \frac{1}{(1-x)^2}$.
- $a_n = [n=0] \leftrightarrow A(x) = 1$.
- $a_n = [n=m] \leftrightarrow A(x) = x^m$.
- $a_n = [d \,|\, n] \leftrightarrow A(x) = \frac{1}{1-x^d}$.
- $a_n = \binom{n+m-1}{n} \leftrightarrow A(x) = \frac{1}{(1-x)^m}$.

Demostración. Notemos que sólo falta mostrar las del lado derecho, ya que las del lado izquierdo las hemos mostrado a lo largo de la sección.

- Si $a_n = [n=0]$, entonces el único término no nulo de $A(x)$ es el término constante, es decir $A(x) = 1$.
- Si $a_n = [n=m]$, entonces el único término no nulo de $A(x)$ es el término x^m, es decir $A(x) = x^m$.
- Si $a_n = [d \,|\, n]$, entonces

$$A(x) = \sum_{n\geq 0}[d \,|\, n]x^n = \sum_{k\geq 0} x^{dk} = \sum_{k\geq 0}(x^d)^k = \frac{1}{1-x^d}$$

- Probaremos esta por inducción sobre m.
  - **Caso base:** Si $m = 1$, entonces $a_n = \binom{n}{n} = 1$ y $A(x) = \frac{1}{1-x}$.
  - **Hipótesis inductiva:** Supongamos que $a_n = \binom{n+m-1}{n}$ y $A(x) = \frac{1}{(1-x)^m}$.
  - **Paso inductivo:** Notemos que

$$\frac{1}{(1-x)^{m+1}} = \frac{1}{1-x} \cdot \underbrace{\frac{1}{(1-x)^m}}_{\text{H.I.}} = \left(\sum_{n\geq 0} x^n\right)\left(\sum_{n\geq 0}\binom{n+m-1}{n}x^n\right).$$

Por la fórmula de convolución, el coeficiente de x^n en esta función generatriz es

$$\sum_{k=0}^{n}\binom{k+m-1}{k} = \binom{n+m}{n}.$$

Para justificar la igualdad, contemos las formas de distribuir n objetos idénticos en $m+1$ cajas. El lado derecho cuenta esto por la fórmula de multiconjuntos. Para el lado izquierdo, fijamos k como el número de objetos que van a las primeras m cajas; esos k objetos se distribuyen de $\binom{k+m-1}{k}$ formas (por la fórmula de multiconjuntos), y la última caja recibe los $n-k$ objetos restantes. Sumando sobre $k = 0, \ldots, n$, obtenemos el lado izquierdo. Por lo tanto, $\frac{1}{(1-x)^{m+1}}$ es la FGO de la sucesión $a_n = \binom{n+m}{n}$, que es la fórmula deseada para $m+1$. □

Ejercicios

1. El objetivo de este ejercicio es encontrar la función generatriz de varias sucesiones.

   a) La sucesión $a_n = \alpha n^2 + \beta n + \gamma$ para $n \geq 0$, donde $\alpha, \beta, \gamma \in \mathbb{R}$ son constantes. Encuentre $A(x)$ la FGO de $(a_n)_{n\geq 0}$.

   b) Una alarma se activa en los tiempos múltiplos de 3 y además se activa dos veces en el tiempo 5. Sea b_n el número de veces que se activa la alarma en el tiempo n. Encuentre $B(x)$ la FGO de $(b_n)_{n\geq 0}$.

   c) Sea c_n una sucesión con función generatriz $C(x)$. Encuentre la función generatriz de la sucesión:

$$d_n = (0, 0, 1, c_3, c_4, c_5, \ldots)$$

   y de la sucesión

$$e_n = (a_0, 0, a_2, 0, a_4, 0, a_6, \ldots)$$

2. Un compilador muy pequeño genera programas de tamaño total n bytes. Tiene dos instrucciones de 1 byte y una macro-instrucción de 2 bytes. Sea p_n el número de programas posibles de tamaño total n bytes, donde importa el orden de las instrucciones.

   a) Modele p_n mediante una recurrencia, separando según la última instrucción.

   b) Defina $P(x) = \sum_{n\geq 0} p_n x^n$ y despeje $P(x)$.

c) Use fracciones parciales para obtener una fórmula cerrada para p_n.

3. Use funciones generatrices para encontrar una fórmula cerrada para cada sucesión.

   a) $a_0 = 0$ y $a_{n+1} = 2a_n + 2^n$ para $n \geq 0$.

   b) $b_0 = 1$ y, para $n \geq 1$,
   $$b_n = 1 + 3\sum_{k=0}^{n-1} b_k.$$

   c) $c_0 = 1$ y, para $n \geq 1$,
   $$c_n = 1 + \sum_{k=0}^{n-1} 2^k c_{n-1-k}.$$

4. Otro uso de las funciones generatrices es demostrar identidades combinatoriales.

   a) Probaremos que
   $$\sum_{k=0}^{n} \binom{n}{k} 2^k = 3^n.$$
   via funciones generatrices.

   1) Defina $s_n = \sum_{k=0}^{n} \binom{n}{k} 2^k$ y $S(x) = \sum_{n\geq 0} s_n x^n$. Intercambiando el orden de las sumas, obtenga
   $$S(x) = \frac{1}{1-x} \sum_{k\geq 0} 2^k \frac{x^k}{(1-x)^k}.$$

   2) Pruebe que $S(x) = \frac{1}{1-3x}$ y concluya la identidad deseada.

   El método anterior es conocido como el método de *Snake Oil* (aceite de serpiente) y es un método bastante vérsatil para demostrar identidades. Veamos dos ejemplos más.

   b) Defina
   $$t_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-k}{k}.$$
   Encuentre la FGO de $(t_n)_{n\geq 0}$ y concluya que $t_n = f_{n+1}$, donde $(f_n)_{n\geq 0}$ es la sucesión de Fibonacci.

   c) Para m fijo, pruebe que
   $$\sum_{k=0}^{n} \binom{k}{m} = \binom{n+1}{m+1}.$$

4.11. El teorema del binomio generalizado

A pesar de que ya tenemos un buen catálogo de funciones generatrices, hay varias recurrencias y funciones que no podemos resolver con las técnicas que hemos visto hasta ahora. Un caso emblemático es la función generatriz de los números de Catalán, que estaban dados por la recurrencia

$$c_0 = 1, \qquad c_{n+1} = \sum_{k=0}^{n} c_k c_{n-k} \quad \text{para todo } n \geq 0.$$

Una herramienta que nos ayudará a expandir funciones generatrices es el teorema del binomio generalizado con exponentes reales. Para esto partimos definiendo los coeficientes binomiales generalizados.

Definición. Para $r \in \mathbb{R}$ y $k \in \mathbb{N}$, definimos el *coeficiente binomial generalizado* $\binom{r}{k}$ como

$$\binom{r}{k} = \frac{r(r-1)(r-2)\cdots(r-k+1)}{k!}.$$

Notemos que con esta definición perdemos la interpretación combinatorial de los coeficientes binomiales usuales, pero ganamos en generalidad.

Ejemplo. Esta definición nos permite usar fracciones y números negativos como índices de los coeficientes binomiales. Para $r = 1/2$ y $k = 3$, tenemos

$$\binom{1/2}{3} = \frac{\frac{1}{2}\cdot(-\frac{1}{2})\cdot(-\frac{3}{2})}{3!} = \frac{1}{16}.$$

Un ejemplo con r negativo es $r = -2$ y $k = 3$:

$$\binom{-2}{3} = \frac{-2\cdot(-3)\cdot(-4)}{3!} = -4.$$

El nuevo coeficiente binomial generalizado extiende propiedades antiguas de los coeficientes binomiales usuales, e introduce algunas nuevas.

Proposición. Para $r \in \mathbb{R}$ y $k \in \mathbb{N}$, se cumplen las siguientes propiedades:

1. $k\binom{r}{k} = r\binom{r-1}{k-1}$ cuando $k \geq 1$.
2. $\binom{r}{k} = \binom{r-1}{k} + \binom{r-1}{k-1}$ cuando $k \geq 1$.
3. $\binom{1/2}{k} = (-1)^{k-1}\frac{1}{k2^{2k-1}}\binom{2k-2}{k-1}$ cuando $k \geq 1$.
4. $\binom{-r}{k} = (-1)^k\binom{r+k-1}{k}$.

Demostración. Como hemos perdido la interpretación combinatorial, los argumentos deberán ser de manipulación algebraica.

1. En efecto,

$$\begin{aligned} k\binom{r}{k} &= \frac{r(r-1)(r-2)\cdots(r-k+1)}{(k-1)!} \\ &= r\frac{(r-1)(r-2)\cdots(r-k+1)}{(k-1)!} \\ &= r\binom{r-1}{k-1}. \end{aligned}$$

2. Notemos que

$$\begin{aligned} \binom{r-1}{k} + \binom{r-1}{k-1} &= \frac{(r-1)(r-2)\cdots(r-k)}{k!} + \frac{(r-1)(r-2)\cdots(r-k+1)}{(k-1)!} \\ &= \frac{(r-1)(r-2)\cdots(r-k+1)}{k!}((r-k)+k) \\ &= \frac{r(r-1)(r-2)\cdots(r-k+1)}{k!} \\ &= \binom{r}{k}. \end{aligned}$$

3. Esta vez obtenemos

$$\binom{1/2}{k} = \frac{(\frac{1}{2})(-\frac{1}{2})(-\frac{3}{2})\cdots(\frac{1}{2}-k+1)}{k!} = (-1)^{k-1}\frac{1\cdot 3\cdot 5\cdots(2k-3)}{2^k k!}.$$

Multiplicando arriba y abajo por $2\cdot 4\cdot 6\cdots(2k-2) = 2^{k-1}(k-1)!$, obtenemos

$$\binom{1/2}{k} = (-1)^{k-1}\frac{(2k-2)!}{2^{2k-1}k!(k-1)!} = (-1)^{k-1}\frac{1}{k2^{2k-1}}\binom{2k-2}{k-1}.$$

4. Por último,

$$\begin{aligned} \binom{-r}{k} &= \frac{-r(-r-1)(-r-2)\cdots(-r-k+1)}{k!} \\ &= (-1)^k\frac{r(r+1)(r+2)\cdots(r+k-1)}{k!} \\ &= (-1)^k\binom{r+k-1}{k}. \end{aligned}$$

□

Ejemplo. Podemos usar estas propiedades para obtener fórmulas para otros coeficientes binomiales

generalizados. Para $k \geq 2$,

$$\begin{aligned}
\binom{k-\frac{5}{2}}{k} &= (-1)^k \binom{3/2}{k} && \text{(ítem 4)} \\
&= (-1)^k \left(\binom{1/2}{k} + \binom{1/2}{k-1} \right) && \text{(ítem 2)} \\
&= -\frac{1}{k2^{2k-1}} \binom{2k-2}{k-1} + \frac{1}{(k-1)2^{2k-3}} \binom{2k-4}{k-2} && \text{(ítem 3)} \\
&= -\frac{2k-3}{k(k-1)2^{2k-2}} \binom{2k-4}{k-2} + \frac{2k}{k(k-1)2^{2k-2}} \binom{2k-4}{k-2} \\
&= \frac{3}{k(k-1)2^{2k-2}} \binom{2k-4}{k-2}.
\end{aligned}$$

Lo más interesante de los coeficientes binomiales generalizados es que nos permitirán obtener sucesiones de funciones generatrices de la forma $(1+x)^r$ con $r \in \mathbb{R}$.

Teorema. Para $r \in \mathbb{R}$, se cumple la identidad de series formales

$$(1+x)^r = \sum_{k \geq 0} \binom{r}{k} x^k.$$

Omitiremos la demostración, que es bastante técnica. Es interesante notar que esto generaliza el teorema del binomio usual, que se obtiene al tomar $r \in \mathbb{N}$ y usar $\frac{x}{y}$ como variable.

Ejemplo. Consideremos la función generatriz

$$A(x) = \sqrt{1-4x} = (1-4x)^{1/2}.$$

Usando el teorema del binomio generalizado, obtenemos

$$A(x) = \sum_{k \geq 0} \binom{1/2}{k} (-4x)^k.$$

Usando la fórmula que obtuvimos para $\binom{1/2}{k}$, nos queda

$$A(x) = 1 + \sum_{k \geq 1} (-1)^{k-1} \frac{1}{k2^{2k-1}} \binom{2k-2}{k-1} (-4x)^k = 1 - \sum_{k \geq 1} \frac{2}{k} \binom{2k-2}{k-1} x^k.$$

Ejemplo. El binomio generalizado nos ayudará también a resolver recurrencias más complejas. Recordemos la recurrencia de los números de Catalán:

$$c_0 = 1, \qquad c_{n+1} = \sum_{k=0}^{n} c_k c_{n-k} \quad \text{para todo } n \geq 0.$$

Notemos que dicha recurrencia es no lineal, así que nuestras técnicas anteriores no aplican directamente.

Definamos $C(x) = \sum_{n\geq 0} c_n x^n$. Multiplicando por x^{n+1} y sumando sobre $n \geq 0$, obtenemos

$$\sum_{n\geq 0} c_{n+1}x^{n+1} = \sum_{n\geq 0}\left(\sum_{k=0}^{n} c_k c_{n-k}\right)x^{n+1}.$$

El lado izquierdo es

$$\sum_{n\geq 0} c_{n+1}x^{n+1} = \sum_{n\geq 1} c_n x^n = C(x) - c_0 = C(x) - 1.$$

El lado derecho es

$$x\sum_{n\geq 0}\left(\sum_{k=0}^{n} c_k c_{n-k}\right)x^n = xC(x)^2.$$

Por lo tanto,

$$C(x) - 1 = xC(x)^2 \implies xC(x)^2 - C(x) + 1 = 0.$$

Resolviendo esta ecuación cuadrática para $C(x)$, obtenemos

$$C(x) = \frac{1 \pm \sqrt{1-4x}}{2x}.$$

¿Cuál de las dos soluciones es la función generatriz de los números de Catalán? Notemos que como $2xC(x) = 1 \pm \sqrt{1-4x}$, entonces el lado derecho debe tener término constante 0. Notemos que $1 + \sqrt{1-4x}$ tiene término constante 2, mientras que $1 - \sqrt{1-4x}$ tiene término constante 0. Por lo tanto, la función generatriz de los números de Catalán es $C(x) = \frac{1-\sqrt{1-4x}}{2x}$.

Usando la expansión que obtuvimos para $\sqrt{1-4x}$, obtenemos

$$C(x) = \frac{1-\sqrt{1-4x}}{2x} = \sum_{n\geq 1}\frac{1}{n}\binom{2n-2}{n-1}x^{n-1} = \sum_{n\geq 0}\frac{1}{n+1}\binom{2n}{n}x^n.$$

De esta manera, obtenemos la fórmula cerrada para los números de Catalán $c_n = \frac{1}{n+1}\binom{2n}{n}$.

4.12. Conteo con funciones generatrices

Hasta ahora hemos usado las funciones generatrices para resolver indirectamente problemas de conteo: primero obtenemos una recurrencia y vía funciones generatrices obtenemos una fórmula cerrada para el problema de conteo original. Una segunda forma de usar funciones generatrices es modelar directamente el proceso de conteo con una función generatriz, y luego leer el coeficiente que nos interesa. Dicha técnica se suele conocer como el *método simbólico* y es una herramienta poderosa para resolver problemas de conteo.

Primero, introduciremos una notación para referirnos a los coeficientes de una función generatriz, y luego veremos tres principios fundamentales para modelar problemas de conteo con funciones generatrices: el principio de la suma, el principio convolutivo y el principio de las listas.

Definición. Sea $\mathcal{A}$ una clase de objetos, y sea a_n el número de objetos de $\mathcal{A}$ de peso n. La

función generatriz ordinaria de $\mathcal{A}$ es la serie formal

$$A(x) = \sum_{n \geq 0} a_n x^n.$$

Para una función generatriz $A(x)$, denotamos por $[x^n]A(x)$ al coeficiente de x^n en la expansión de $A(x)$, es decir, $[x^n]A(x) = a_n$.

Ejemplo. Algunos ejemplos de clases combinatoriales son:

1. **Bitstrings.** Los objetos son las palabras formadas con 0 y 1, y el peso de una palabra es su largo. Su función generatriz es

$$\frac{1}{1-2x} = 1 + 2x + 4x^2 + 8x^3 + \cdots,$$

y $[x^n]\frac{1}{1-2x} = 2^n$ es el número de palabras binarias de largo n.

2. **Caminos de Dyck.** Recordemos que un camino de Dyck es una secuencia de pasos $\nearrow$ y $\searrow$ que comienza y termina en el mismo nivel, y nunca pasa por debajo del nivel inicial. Usamos como peso el número de pasos $\nearrow$. La función generatriz de los caminos de Dyck es

$$C(x) = \frac{1 - \sqrt{1-4x}}{2x} = 1 + x + 2x^2 + 5x^3 + 14x^4 + \cdots,$$

y $[x^n]C(x) = \frac{1}{n+1}\binom{2n}{n}$ es el número de caminos de Dyck de peso n.

3. **Subconjuntos con k elementos.** Fijemos k. Para cada n, los objetos de peso n son los subconjuntos de $[n]$ con k elementos. Como hay $\binom{n}{k}$ de ellos, su función generatriz es

$$\sum_{n \geq 0} \binom{n}{k} x^n = \frac{x^k}{(1-x)^{k+1}}.$$

4.12.1. Principio aditivo

Teorema (Principio aditivo)**.** Sean $\mathcal{A}$ y $\mathcal{B}$ dos clases de objetos, con FGO $A(x)$ y $B(x)$ respectivamente. Si construimos un nuevo objeto eligiendo un objeto de $\mathcal{A}$ o un objeto de $\mathcal{B}$ de manera disjunta, entonces la cantidad de objetos de peso n que podemos construir es

$$[x^n](A(x) + B(x)).$$

Demostración. Para obtener peso n, podemos elegir un objeto de $\mathcal{A}$ de peso n o un objeto de $\mathcal{B}$ de peso n. Como las alternativas son disjuntas, hay $a_n + b_n$ formas de hacer esto. Por lo tanto,

$$[x^n](A(x) + B(x)) = a_n + b_n. \qquad \square$$

Este principio dice que una suma modela un o: usamos $A(x) + B(x)$ cuando el objeto pertenece a una de varias alternativas disjuntas.

Ejemplo (Elegir una ficha)**.** Tenemos dos fichas rojas, una marcada con un 1 y la otra con un 3. También tenemos dos fichas azules, una marcada con un 2 y la otra con un 3. Queremos contar cuántas formas hay de elegir una ficha según su marca.

La FGO de las fichas rojas es

$$R(x) = x + x^3,$$

y la FGO de las fichas azules es

$$Z(x) = x^2 + x^3.$$

Como elegimos una ficha roja o una ficha azul, y las alternativas son disjuntas, la FGO total es

$$F(x) = R(x) + Z(x) = x + x^2 + 2x^3.$$

Por ejemplo, $[x^3]F(x) = 2$, porque hay dos fichas marcadas con 3: una roja y una azul.

4.12.2. Principio convolutivo

Teorema (Principio convolutivo)**.** Sean $\mathcal{A}$ y $\mathcal{B}$ dos clases de objetos y $A(x)$ y $B(x)$ sus respectivas FGO. Si construimos un nuevo objeto eligiendo independientemente un objeto de $\mathcal{A}$ y uno de $\mathcal{B}$, y el peso total es la suma de ambos pesos, entonces la cantidad de objetos de peso n que podemos construir es $[x^n]A(x)B(x)$.

Demostración. Para obtener peso total n, podemos elegir un objeto de $\mathcal{A}$ de peso k y un objeto de $\mathcal{B}$ de peso $n-k$. Hay $a_k b_{n-k}$ formas de hacer esto. Por lo tanto,

$$[x^n]A(x)B(x) = \sum_{k=0}^{n} a_k b_{n-k}. \qquad \square$$

En problemas más elementales, el mismo principio aparece así: cada variable independiente aporta un factor, y el exponente total registra la suma.

Ejemplo (Suma con dos variables)**.** ¿Cuántas soluciones existen de $e_1 + e_2 = 8$ sujetas a

$$2 \le e_1 \le 5, \qquad 3 \le e_2 \le 6?$$

Para e_1 usamos $F_1(x) = x^2 + x^3 + x^4 + x^5$, y para e_2 usamos $F_2(x) = x^3 + x^4 + x^5 + x^6$. Entonces

$$F_1(x)F_2(x) = x^5 + 2x^6 + 3x^7 + 4x^8 + 3x^9 + 2x^{10} + x^{11}.$$

Como $[x^8]F_1(x)F_2(x) = 4$, hay 4 soluciones:

$$(2,6), \quad (3,5), \quad (4,4), \quad (5,3).$$

Ejemplo. ¿De cuántas maneras se pueden distribuir 8 galletas idénticas entre 3 perritos distintos, si cada uno recibe entre 2 y 4 galletas?

Si p_i es la cantidad de galletas del perrito i, entonces cada p_i se modela con $x^2 + x^3 + x^4$. Por el principio convolutivo, la FGO es

$$G(x) = (x^2 + x^3 + x^4)^3.$$

Expandiendo,

$$G(x) = x^6 + 3x^7 + 6x^8 + 7x^9 + 6x^{10} + 3x^{11} + x^{12}.$$

Como $[x^8]G(x) = 6$, hay 6 distribuciones posibles.

Ejemplo. Supongamos que tenemos billetes de 1, 2 y 5 mil pesos, y queremos contar las formas de pagar p mil pesos usando cualquier cantidad de billetes, sin importar el orden. Si el orden no importa, sólo registramos cuántos billetes usamos de cada tipo. Para billetes de valor d, usar j billetes aporta peso jd, así que el factor correspondiente es

$$1 + x^d + x^{2d} + x^{3d} + \cdots = \frac{1}{1-x^d}.$$

Por el principio convolutivo,

$$M(x) = (1 + x + x^2 + x^3 + \cdots)(1 + x^2 + x^4 + x^6 + \cdots)(1 + x^5 + x^{10} + x^{15} + \cdots).$$

Es decir,

$$M(x) = \frac{1}{(1-x)(1-x^2)(1-x^5)}.$$

La expansión comienza como

$$M(x) = 1 + x + 2x^2 + 2x^3 + 3x^4 + 4x^5 + 5x^6 + 6x^7 + \cdots .$$

Como $[x^5]M(x) = 4$, hay 4 formas de pagar 5 mil pesos, que son:

$$1+1+1+1+1, \quad 1+1+1+2, \quad 1+2+2, \quad 5.$$

Ejemplo (Planificar un curso)**.** Un curso tiene n días de clases. Queremos particionar el curso en dos bloques: los primeros k días serán teoría y los últimos $n-k$ serán laboratorio, con $1 \le k \le n-2$. Hay que planificar un control en el bloque de teoría y dos controles en el bloque de laboratorio.

Si el bloque de teoría tiene k días, hay k formas de planificar su control. La FGO para este bloque es

$$A(x) = \sum_{k\ge1} kx^k = x\sum_{k\ge0}(k+1)x^k = \frac{x}{(1-x)^2}.$$

Si el bloque de laboratorio tiene m días, hay $\binom{m}{2}$ formas de planificar dos controles. La FGO para este bloque es

$$B(x) = \sum_{m\ge2}\binom{m}{2}x^m = \sum_{m\ge2}\binom{m}{m-2}x^m = x^2\sum_{m\ge0}\binom{m+2}{m}x^m = \frac{x^2}{(1-x)^3}.$$

Por el principio convolutivo, si f_n es el número de planificaciones de un curso de n días,

$$F(x) = \sum_{n\ge0} f_n x^n = A(x)B(x) = \frac{x^3}{(1-x)^5}.$$

Usando el binomio generalizado,

$$F(x) = x^3\sum_{n\ge0}\binom{-5}{n}(-x)^n = x^3\sum_{n\ge0}\binom{n+4}{n}x^n = \sum_{n\ge3}\binom{n+1}{n-3}x^n = \sum_{n\ge3}\binom{n+1}{4}x^n.$$

Por lo tanto, $f_n = \binom{n+1}{4}$ para todo $n \ge 3$.

Ejemplo. Otra forma de usar los principios convolutivo y aditivo es para obtener una función generatriz para un objeto que se construye recursivamente. Por ejemplo, un árbol binario es un objeto que es vacío o es un nodo raíz con un árbol binario a la izquierda y otro a la derecha. Si $T(x)$ es la FGO de los árboles binarios, entonces por el principio convolutivo,

$$T(x) = 1 + xT(x)^2.$$

Veamos el porqué de cada término: el 1 corresponde al árbol vacío, y el $xT(x)^2$ corresponde a un nodo raíz (que aporta x) con un árbol binario a la izquierda (que aporta $T(x)$) y otro a la derecha (que aporta otro $T(x)$). Resolviendo esta ecuación cuadrática para $T(x)$, obtenemos

$$T(x) = \frac{1 \pm \sqrt{1-4x}}{2x}.$$

Como $T(x)$ tiene término constante 1, la función generatriz de los árboles binarios es $T(x) = \frac{1-\sqrt{1-4x}}{2x}$, que es la misma función generatriz de los números de Catalán.

4.12.3. Principio de las listas

El principio convolutivo sirve cuando el número de elecciones independientes está fijo. Cuando llenamos una cantidad arbitraria de posiciones, el principio convolutivo no es suficiente para modelar el proceso de conteo. Este es el caso de las listas, que son objetos con una cantidad arbitraria de posiciones, cada una de las cuales se llena con un objeto de una clase dada.

Teorema (Principio de las listas)**.** Sea $\mathcal{A}$ una clase de objetos contada por una FGO $A(x)$, con $a_0 = 0$. Consideremos la clase de listas finitas de objetos de $\mathcal{A}$, incluyendo la lista vacía, donde el peso de una lista es la suma de los pesos de sus elementos. Entonces la FGO de esta clase es

$$\frac{1}{1 - A(x)}.$$

Demostración. Para $m \geq 0$, consideremos primero las listas con exactamente m elementos. Por el principio convolutivo aplicado m veces, la FGO de estas listas es $A(x)^m$. En particular, cuando $m = 0$ obtenemos la lista vacía, que aporta 1.

Como las listas de largos distintos son disjuntas, por el principio aditivo la FGO de todas las listas es

$$\sum_{m \geq 0} A(x)^m = \frac{1}{1 - A(x)}.$$

El supuesto $a_0 = 0$ garantiza que no hay objetos de peso 0 en $\mathcal{A}$, así que cada coeficiente cuenta una cantidad finita de listas. □

Ejemplo (Billetes: el orden importa)**.** Si el orden sí importa y queremos usar exactamente m billetes de valores 1, 2 y 5 mil pesos, cada posición aporta un billete. Esto todavía es una aplicación del principio convolutivo:

$$H_m(x) = (x + x^2 + x^5)^m.$$

Por ejemplo,

$$H_2(x) = (x + x^2 + x^5)^2 = x^2 + 2x^3 + x^4 + 2x^6 + 2x^7 + x^{10}.$$

Como $[x^3]H_2(x) = 2$, obtenemos las secuencias $(1, 2)$ y $(2, 1)$.

Si ahora permitimos cualquier número de billetes, usamos el principio de las listas. Tomando $A(x) = x + x^2 + x^5$ para la clase de billetes, obtenemos

$$H(x) = \frac{1}{1 - A(x)} = \frac{1}{1 - x - x^2 - x^5}.$$

La expansión comienza como

$$H(x) = 1 + x + 2x^2 + 3x^3 + 5x^4 + 9x^5 + 15x^6 + 26x^7 + \cdots .$$

Como $[x^5]H(x) = 9$, hay 9 formas ordenadas de pagar 5 mil pesos.

Ejemplo (Unidades con comandante)**.** Tenemos n personas en una fila. Queremos dividir la fila en unidades consecutivas no vacías y elegir un comandante en cada unidad.

Si una unidad tiene tamaño k, hay k formas de elegir su comandante. La FGO para una unidad es

$$A(x) = \sum_{k \geq 1} kx^k = \frac{x}{(1 - x)^2}.$$

Como el número de unidades es arbitrario, usamos el principio de las listas:

$$H(x) = \frac{1}{1 - A(x)} = \frac{1}{1 - \frac{x}{(1-x)^2}} = 1 + \frac{x}{1 - 3x + x^2}.$$

Para obtener una fórmula cerrada, usamos fracciones parciales. Sean

$$\alpha = \frac{3 + \sqrt{5}}{2}, \qquad \beta = \frac{3 - \sqrt{5}}{2}.$$

Entonces $1 - 3x + x^2 = (1 - \alpha x)(1 - \beta x)$, y

$$\frac{x}{1 - 3x + x^2} = \frac{1}{\sqrt{5}} \left(\frac{1}{1 - \alpha x} - \frac{1}{1 - \beta x} \right).$$

Por lo tanto,

$$H(x) = 1 + \sum_{n \geq 1} \frac{\alpha^n - \beta^n}{\sqrt{5}} x^n.$$

Si h_n es el número de formas para una fila de n personas, entonces

$$h_0 = 1, \qquad h_n = \frac{1}{\sqrt{5}} \left(\left(\frac{3 + \sqrt{5}}{2} \right)^n - \left(\frac{3 - \sqrt{5}}{2} \right)^n \right) \quad \text{para } n \geq 1.$$

La expansión comienza como

$$H(x) = 1 + x + 3x^2 + 8x^3 + 21x^4 + \cdots .$$

Por ejemplo, hay 8 formas de dividir una fila de 3 personas en unidades consecutivas y elegir un comandante en cada unidad.

Ejercicios

1. Un servidor procesa trabajos en orden FIFO. Cada trabajo corresponde a una operación del sistema, y cada operación tiene un tiempo de ejecución fijo:

   - `parse-request`, que consume 1 unidad de tiempo.
   - `cache-lookup`, que consume 1 unidad de tiempo.
   - `db-query`, que consume 2 unidades de tiempo.
   - `serialize-response`, que consume 2 unidades de tiempo.

   Una cola se registra como una traza de ejecución, por ejemplo

   `parse-request`, `db-query`, `cache-lookup`.

   Dos colas son distintas si tienen distintos largos o si alguna posición contiene una operación distinta.

   a) Si a_n es el número de trazas de exactamente m trabajos que consumen n unidades de tiempo, encuentre su FGO $A_m(x)$.

   b) Si b_n es el número de trazas con cualquier número de trabajos que consumen n unidades de tiempo, encuentre su FGO $B(x)$.

   c) Encuentre una fórmula cerrada para b_n.

2. Un árbol binario coloreado es un árbol binario donde cada nodo es azul o rojo. Sea a_n el número de árboles binarios coloreados con n nodos.

   a) Calcule a_1, a_2 y a_3.

   b) Sea $A(x)$ la FGO de a_n. Muestre que

   $$A(x) = 2xA(x)^2 + 1.$$

   Indicación: puede diseñar una recurrencia o usar los principios de conteo asociados a funciones generatrices.

   c) Resuelva la ecuación cuadrática anterior y explique por qué

   $$A(x) = \frac{1 - \sqrt{1 - 8x}}{4x}.$$

   d) Use el teorema del binomio generalizado para demostrar que

   $$a_n = \frac{2^n}{n+1}\binom{2n}{n} \quad \text{para todo } n \geq 0.$$

   e) Interprete la fórmula anterior en términos de los números de Catalán.

3. *a*) Si a_n es el número de formas de entregar n peluches idénticos a 5 niños, de modo que cada uno reciba a lo más 3 peluches, encuentre su FGO $A(x)$.

b) Si b_n es el número de formas de pagar n mil pesos con billetes de 1, 2, 3, y 4 mil pesos cuando el orden no importa, encuentre su FGO $B(x)$.

c) Si c_n es el número de formas de pagar n mil pesos con billetes de 1, 2, 3, y 4 mil pesos cuando el orden sí importa, encuentre su FGO $C(x)$.

4. Sea $\mathcal{A}$ una clase de objetos con FGO

$$A(y) = \sum_{m \geq 0} a_m y^m,$$

donde un objeto de peso m tiene m posiciones distinguibles. Sea $\mathcal{B}$ una clase de objetos con FGO $B(x)$, con $b_0 = 0$. Construimos una nueva clase reemplazando cada posición de un objeto de $\mathcal{A}$ por un objeto de $\mathcal{B}$, y definiendo el peso total como la suma de los pesos de los objetos insertados.

a) Pruebe que la FGO de la nueva clase es $A(B(x))$.

b) Un camino de Dyck decorado con bitstrings es un camino de Dyck donde cada paso $\nearrow$ se reemplaza por un bitstring. El largo de un camino de Dyck decorado es la suma de los largos de los bitstrings que lo decoran. Si d_n es el número de caminos de Dyck decorados de largo n, encuentre su FGO $D(x)$.

Capítulo 5

Teoría de grafos

5.1. Introducción a grafos

Muchas estructuras que aparecen en computación tienen la misma forma básica: objetos conectados por relaciones. Por ejemplo:

- computadores conectados en una red,
- ciudades conectadas por rutas,
- personas conectadas por amistades o por relaciones de seguimiento,
- módulos de un programa conectados por dependencias,
- clases de un sistema conectadas por herencia,
- palabras conectadas por relaciones de sinonimia.

La teoría de grafos nos entrega un lenguaje común para estudiar todas estas situaciones. La idea es abstraer qué son los objetos y quedarnos solamente con la información de cuáles objetos están conectados con cuáles otros.

Definición. Un *grafo simple* es un par $G = (V, E)$, donde:

- V es un conjunto de *vértices* o *nodos*;
- E es un conjunto de *aristas*, donde cada arista es un conjunto de dos vértices distintos.

En esta clase pensaremos siempre en grafos finitos. Es decir, V y E tienen un número finito de elementos. Usualmente escribiremos

$$|V| = n \quad \text{y} \quad |E| = m.$$

Cuando el grafo no es dirigido, una arista entre u y v no tiene orientación: conecta u con v y también v con u. En ese caso escribiremos indistintamente uv, vu o $\{u, v\}$ para referirnos a la misma arista. Cuando el grafo es un digrafo, en cambio, el arco (u, v) va desde u hacia v y puede ser distinto del arco (v, u). Más adelante permitiremos aristas paralelas y loops; cuando no digamos nada adicional, asumiremos que el grafo es simple.

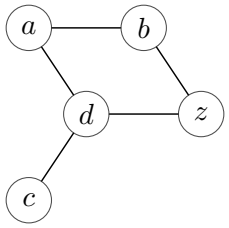
Ejemplo. Consideremos

$$V = \{a, b, c, d, z\}$$

y

$$E = \{ab, ad, bz, cd, dz\}.$$

Entonces $G = (V, E)$ es un grafo con 5 vértices y 5 aristas.



5.1.1. Adyacencia y vecindad

Definición. Sea $G = (V, E)$ un grafo no dirigido y sean $u, v \in V$. Decimos que u y v son *adyacentes* si $uv \in E$. La *vecindad* de un vértice v es el conjunto

$$N(v) = \{u \in V : uv \in E\}.$$

La vecindad de un vértice contiene exactamente los vértices que están conectados a él por una arista. Por eso, pensar en $N(v)$ es pensar en los contactos directos de v: sus amigos en una red social, sus vecinos en un mapa de rutas, o los servidores a los que está conectado directamente en una red de computadores.

Ejemplo. En el grafo del ejemplo anterior tenemos

$$N(a) = \{b, d\}, \qquad N(b) = \{a, z\}, \qquad N(c) = \{d\}, \qquad N(d) = \{a, c, z\}.$$

En particular, a y b son adyacentes, pero a y z no lo son.

5.1.2. Tipos de aristas y tipos de grafos

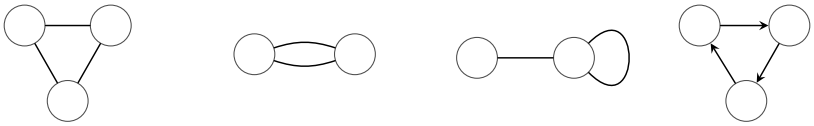
Definición. Sea $G = (V, E)$ un grafo.

- Una *arista paralela* es una segunda copia de una arista que ya existe entre los mismos dos vértices.
- Un *loop* o *bucle* es una arista de un vértice hacia sí mismo.
- Un *arco dirigido* es una arista con orientación, es decir, un par ordenado (u, v) que va desde u hacia v.

Estas tres posibilidades permiten distinguir varias clases de grafos. La siguiente tabla resume la nomenclatura que usaremos.

Tipo	Dirigido	Paralelas	Loops
grafo simple	no	no	no
multigrafo	no	sí	no
grafo	no	sí	sí
digrafo simple	sí	no	no
multidigrafo	sí	sí	no
digrafo	sí	sí	sí

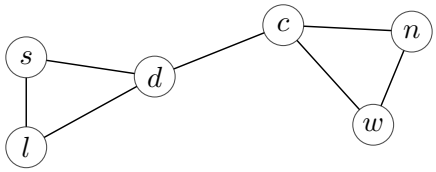
Visualmente, las diferencias son las siguientes.



simple paralelas loop digrafo

Si no especificamos que un grafo es un digrafo, asumiremos que no lo es. Además, en muchos contextos diremos simplemente "grafo" para referirnos a un grafo simple no dirigido, siempre que no haya riesgo de confusión.

Ejemplo. Una red de computadores donde cada enlace físico conecta dos computadores distintos, y donde no distinguimos rutas redundantes entre el mismo par de computadores, se modela naturalmente por un grafo simple.



Ejemplo. Si la misma red tiene rutas redundantes entre algunos pares de data centers, entonces conviene usar un multigrafo. En cambio, si además hay enlaces de diagnóstico local desde un computador hacia sí mismo, aparecen loops y debemos permitir grafos con loops.

Si los enlaces son unidireccionales, entonces el modelo correcto es un digrafo. Por ejemplo, en una red social de seguidores, que u siga a v no implica necesariamente que v siga a u.

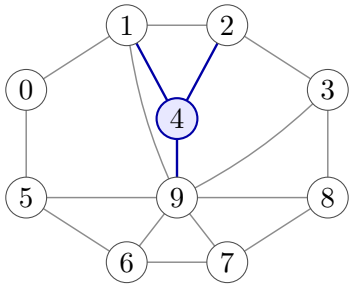
multigrafo con loop digrafo

5.1.3. Grado

Definición. Sea $G = (V, E)$ un grafo no dirigido y sea $v \in V$. El *grado* de v, denotado por $d(v)$, es el número de aristas incidentes en v. Si $d(v) = 0$, decimos que v es un *vértice aislado*.

Una arista uv es incidente en sus dos extremos u y v. Por convención, un loop en v aporta 2 al grado de v: una vez por cada extremo de la arista.

Ejemplo. Consideremos el siguiente grafo.

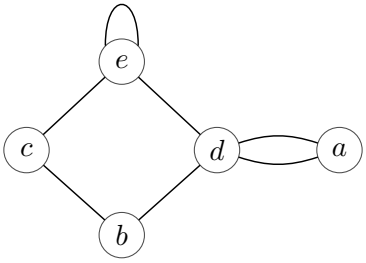


En este grafo,

$$N(4) = \{1, 2, 9\} \quad \text{y} \quad d(4) = 3.$$

El vértice 0 tiene grado 2.

Ejemplo. En un multigrafo, las aristas paralelas se cuentan por separado y un loop aporta 2 al grado. En el siguiente grafo, el vértice e tiene un loop y el par d, a está unido por dos aristas paralelas.



Por lo tanto,

$$d(e) = 4, \qquad d(d) = 4 \qquad \text{y} \qquad d(a) = 2.$$

Teorema (Apretón de manos)**.** Sea $G = (V, E)$ un grafo no dirigido. Entonces

$$\sum_{v \in V} d(v) = 2m.$$

Demostración. Contemos incidencias entre vértices y aristas de dos maneras. Por un lado, si sumamos $d(v)$ sobre todos los vértices $v \in V$, contamos una incidencia por cada extremo de cada arista. Por otro lado, cada arista tiene exactamente dos extremos. Por lo tanto, cada arista aporta exactamente 2 a la suma total de grados. Como hay m aristas, obtenemos

$$\sum_{v \in V} d(v) = 2m. \qquad \square$$

Proposición. Sea $G = (V, E)$ un grafo no dirigido. Entonces G tiene un número par de vértices de grado impar.

Demostración. Dividamos los vértices en dos conjuntos:

$$V_0 = \{v \in V : d(v) \text{ es par}\} \quad \text{y} \quad V_1 = \{v \in V : d(v) \text{ es impar}\}.$$

Por el teorema del apretón de manos,

$$\sum_{v \in V} d(v) = 2m,$$

así que la suma total de los grados es par. Además,

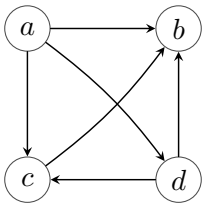
$$\sum_{v \in V} d(v) = \sum_{v \in V_0} d(v) + \sum_{v \in V_1} d(v).$$

La primera suma del lado derecho es par, porque es suma de números pares. Por lo tanto, la segunda suma también debe ser par. Pero la segunda suma es una suma de $|V_1|$ números impares. Una suma de números impares es par si y sólo si la cantidad de sumandos es par. Concluimos que $|V_1|$ es par. □

5.1.4. Grado en digrafos

Definición. Sea $G = (V, E)$ un digrafo y sea $v \in V$. El *grado de salida* de v, denotado por $d^+(v)$, es el número de arcos que salen de v. El *grado de entrada* de v, denotado por $d^-(v)$, es el número de arcos que entran a v.

Ejemplo. Consideremos el siguiente digrafo.



Sus grados de entrada y salida son:

vértice	$d^+(v)$	$d^-(v)$
a	3	0
b	0	3
c	1	2
d	2	1

Teorema (Apretón de manos en digrafos)**.** Sea $G = (V, E)$ un digrafo. Entonces

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = m.$$

Demostración. Cada arco tiene exactamente un vértice inicial y exactamente un vértice final. Si sumamos los grados de salida sobre todos los vértices, cada arco se cuenta exactamente una vez: en su vértice inicial. Por lo tanto,

$$\sum_{v \in V} d^+(v) = m.$$

Análogamente, si sumamos los grados de entrada sobre todos los vértices, cada arco se cuenta exactamente una vez: en su vértice final. Así,

$$\sum_{v \in V} d^-(v) = m.$$

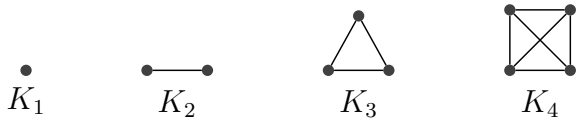
Esto prueba el resultado. □

5.2. Familias básicas de grafos

Algunos grafos aparecen con tanta frecuencia que tienen nombre propio. Estas familias serán nuestros primeros ejemplos de estructuras con propiedades reconocibles.

Definición. El *grafo completo* con n vértices, denotado por K_n, es el grafo simple en que todo par de vértices distintos es adyacente.

Ejemplo. Los primeros grafos completos son:



Ejemplo. El grafo completo K_n tiene

$$\binom{n}{2} = \frac{n(n-1)}{2}$$

aristas. En efecto, en K_n hay una arista por cada par de vértices distintos. Por lo tanto, el número de aristas es el número de subconjuntos de tamaño 2 de un conjunto de tamaño n, que es $\binom{n}{2}$.

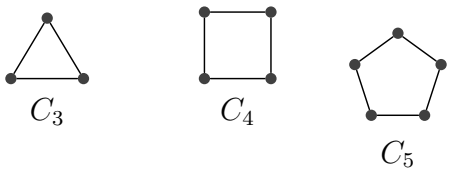
Definición. Para $n \geq 3$, el *ciclo* con n vértices, denotado por C_n, es el grafo con vértices

$$v_1, v_2, \ldots, v_n$$

y aristas

$$v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1.$$

Ejemplo. El grafo C_3 es un triángulo, C_4 es un cuadrado, y en general C_n corresponde a una cadena cerrada de n vértices. En particular, C_n tiene n vértices y n aristas.



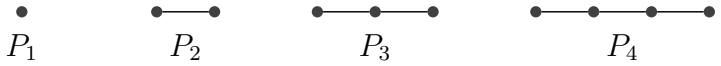
Definición. Para $n \geq 1$, el *camino* con n vértices, denotado por P_n, es el grafo con vértices

$$v_1, v_2, \ldots, v_n$$

y aristas

$$v_1v_2, v_2v_3, \ldots, v_{n-1}v_n.$$

Ejemplo. Los primeros caminos son:



Ejemplo. Las tres familias básicas tienen cantidades de aristas muy regulares:

$$|E(P_n)| = n - 1, \qquad |E(C_n)| = n, \qquad |E(K_n)| = \binom{n}{2}.$$

En P_n sólo aparecen las aristas entre vértices consecutivos, así que hay una menos que el número de vértices. En C_n aparece además la arista que cierra el ciclo, por lo que hay exactamente n aristas. En K_n, como vimos antes, hay una arista por cada par de vértices distintos.

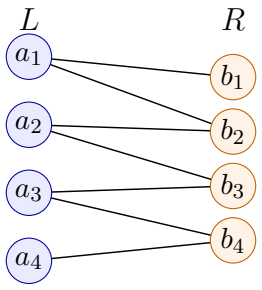
Definición. Un grafo $G = (V, E)$ es *bipartito* si existen conjuntos $L, R \subseteq V$ tales que:

- $V = L \cup R$;
- $L \cap R = \emptyset$;
- toda arista de G tiene un extremo en L y el otro extremo en R.

En ese caso, decimos que (L, R) es una *bipartición* de G.

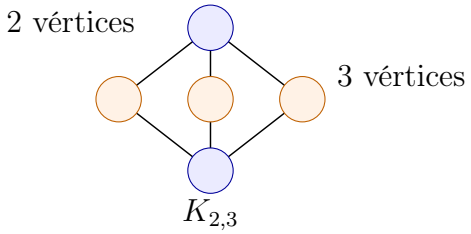
La condición importante es que no puede haber aristas dentro de L ni dentro de R. Todas las aristas deben cruzar de un lado al otro.

Ejemplo. El siguiente grafo es bipartito.



Definición. El *grafo bipartito completo* $K_{\ell,r}$ es el grafo bipartito con partes L y R, donde $|L| = \ell$ y $|R| = r$, y donde cada vértice de L es adyacente a cada vértice de R.

Ejemplo. El grafo $K_{2,3}$ tiene dos vértices a un lado, tres vértices al otro lado, y todas las aristas posibles entre ambos lados. En general, $K_{\ell,r}$ tiene $\ell + r$ vértices y ℓr aristas.



Definición. Un *2-coloreo* de un grafo $G = (V, E)$ es una asignación de uno de dos colores a cada vértice, de manera que los extremos de toda arista tienen colores distintos.

Proposición. Un grafo es bipartito si y sólo si es 2-coloreable.

Demostración. Primero supongamos que G es bipartito, con bipartición (L, R). Coloreamos todos los vértices de L con el primer color y todos los vértices de R con el segundo color. Como toda arista tiene un extremo en L y el otro en R, toda arista queda con extremos de colores distintos. Por lo tanto, esto es un 2-coloreo.

Recíprocamente, supongamos que G admite un 2-coloreo. Sea L el conjunto de vértices con el primer color y sea R el conjunto de vértices con el segundo color. Claramente $V = L \cup R$ y $L \cap R = \emptyset$. Además, como el coloreo es válido, ninguna arista tiene sus dos extremos con el mismo color. Por lo tanto, toda arista tiene un extremo en L y el otro en R. Así, G es bipartito. □

Ejemplo. Todo camino P_n es bipartito. Si sus vértices son $v_1, v_2, \ldots, v_n$ en orden, tomamos

$$L = \{v_i : i \text{ es impar}\} \quad \text{y} \quad R = \{v_i : i \text{ es par}\}.$$

Cada arista de P_n une dos vértices consecutivos, por lo que siempre conecta un vértice de L con uno de R. Equivalentemente, se obtiene un 2-coloreo alternando los colores a lo largo del camino.

Ejemplo. El ciclo C_n es bipartito si y sólo si n es par. Si n es par, usamos la misma partición por paridad de índices:

$$L = \{v_i : i \text{ es impar}\} \quad \text{y} \quad R = \{v_i : i \text{ es par}\}.$$

Todas las aristas del ciclo cruzan de L a R, incluyendo la arista $v_n v_1$, porque n es par.

Si n es impar, cualquier 2-coloreo tendría que alternar colores alrededor del ciclo. Al volver a la arista $v_n v_1$, los vértices v_n y v_1 quedarían con el mismo color. Por lo tanto, los ciclos impares no son bipartitos.

Ejemplo. El grafo completo K_n es bipartito sólo para $n \leq 2$. Los grafos K_1 y K_2 son bipartitos: en K_1 no hay aristas y en K_2 basta poner un vértice en cada parte. En cambio, si $n \geq 3$, el grafo K_n contiene tres vértices mutuamente adyacentes, es decir, una copia de C_3. Como C_3 no es bipartito, ningún K_n con $n \geq 3$ es bipartito.

Ejercicios

1. Para cada una de las siguientes situaciones, proponga un modelo usando grafos. En cada caso indique qué representan los vértices, qué representan las aristas o arcos, y qué tipo de grafo corresponde usar.

a) Una red social donde dos personas están conectadas si son amigas.

b) Una red social donde una persona puede seguir a otra sin que la relación sea recíproca.

c) Una red de computadores donde puede haber varias rutas físicas entre el mismo par de data centers.

d) Un sistema de dependencias entre tareas, donde una tarea puede ser prerrequisito de otra.

e) Una aplicación de mapas donde dos ciudades están conectadas si existe una ruta directa entre ellas.

2. *a*) Pruebe que todo grafo con al menos dos vértices tiene dos vértices con el mismo grado.

b) La *secuencia de grados* de un grafo $G = (V, E)$ es la secuencia de los grados de sus vértices, ordenada de mayor a menor. Decida si cada una de las siguientes secuencias puede ser la secuencia de grados de un grafo simple. Justifique su respuesta.

1) $(5, 4, 3, 2, 1)$
2) $(2, 2, 2, 2, 2)$.
3) $(3, 3, 2, 2, 1, 1)$.
4) $(4, 4, 4, 1, 1)$.
5) $(1, 1, 1, 1, 1)$.

c) El resultado de la parte (a) dice que no puede haber n grados distintos en un grafo con n vértices. ¿Puede haber $n - 1$ grados distintos? Construya ejemplos para $n = 2, 3, 4, 5, 6$. ¿Como sería una construcción general para cualquier $n \geq 2$?

3. Sea $G = (V, E)$ un grafo simple. El *complemento* de G es el grafo $\overline{G} = (V, \overline{E})$ donde, para vértices distintos $u, v \in V$,

$$uv \in \overline{E} \quad \text{si y sólo si} \quad uv \notin E.$$

a) Calcule cuántas aristas tiene $\overline{G}$ en función de $n = |V|$ y $m = |E|$.

b) Demuestre que para todo vértice $v \in V$ se cumple

$$d_G(v) + d_{\overline{G}}(v) = n - 1.$$

c) Determine el complemento de K_n, de C_4 y de $K_{1,n-1}$.

d) Diremos que un grafo G es *autocomplementario* si G y $\overline{G}$ son iguales salvo por renombrar sus vértices. Pruebe que si existe un grafo autocomplementario con n vértices, entonces

$$n \equiv 0 \pmod 4 \quad \text{o} \quad n \equiv 1 \pmod 4.$$

e) De ejemplos de grafos autocomplementario con 4 y 5 vértices.

5.3. Representaciones de grafos

Partiremos esta clase con formas distintas de representar un grafo en un computador.
La primera forma es la matriz de adyacencia.

Definición. Sea $G = (V, E)$ un grafo simple con

$$V = \{v_1, v_2, \ldots, v_n\}.$$

La *matriz de adyacencia* de G es la matriz $A = (a_{ij}) \in \{0,1\}^{n\times n}$ definida por

$$a_{ij} = \begin{cases} 1 & \text{si } v_i v_j \in E, \\ 0 & \text{si } v_i v_j \notin E. \end{cases}$$

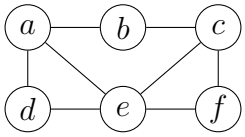
Ejemplo. Consideremos el grafo G con vértices

$$V = \{a, b, c, d, e, f\}$$

y aristas

$$E = \{ab, ad, ae, bc, ce, cf, de, ef\}.$$

Una forma de dibujarlo es la siguiente.



Si ordenamos los vértices como a, b, c, d, e, f, entonces su matriz de adyacencia es

	a	b	c	d	e	f
a	0	1	0	1	1	0
b	1	0	1	0	0	0
c	0	1	0	0	1	1
d	1	0	0	0	1	0
e	1	0	1	1	0	1
f	0	0	1	0	1	0

Proposición. Sea G un grafo simple no dirigido y sea A su matriz de adyacencia. Entonces A es simétrica y su diagonal está formada sólo por ceros.

Demostración. Como G no es dirigido, para todo par de vértices v_i, v_j tenemos

$$v_i v_j \in E \quad \text{si y sólo si} \quad v_j v_i \in E.$$

Por la definición de matriz de adyacencia, esto implica que $a_{ij} = a_{ji}$ para todo i, j. Por lo tanto, A es simétrica.

Además, como G es simple, no tiene bucles. Luego $v_i v_i \notin E$ para todo i, y por lo tanto $a_{ii} = 0$ para todo i. □

Definición. Sea $G = (V, E)$ un grafo. Una *lista de adyacencia* de G es una representación que, para cada vértice $v \in V$, almacena la lista de vértices adyacentes a v.

Ejemplo. Para el grafo anterior, la lista de adyacencia es

vértice	vértices adyacentes
a	b, d, e
b	a, c
c	b, e, f
d	a, e
e	a, c, d, f
f	c, e

La matriz de adyacencia es muy cómoda para preguntar rápidamente si dos vértices son adyacentes. La lista de adyacencia, en cambio, es muy natural cuando queremos recorrer los vecinos de un vértice. Estas son dos maneras distintas de guardar la misma información.

Definición. Sea $G = (V, E)$ un grafo no dirigido con

$$V = \{v_1, \ldots, v_n\} \quad \text{y} \quad E = \{e_1, \ldots, e_m\}.$$

La *matriz de incidencia* de G es la matriz $B = (b_{ij}) \in \{0, 1\}^{n \times m}$ definida por

$$b_{ij} = \begin{cases} 1 & \text{si } e_j \text{ es incidente en } v_i, \\ 0 & \text{en otro caso.} \end{cases}$$

Ejemplo. Para el grafo anterior, nombremos las aristas como

$$e_1 = ab, \quad e_2 = bc, \quad e_3 = ad, \quad e_4 = ae, \quad e_5 = de, \quad e_6 = cf, \quad e_7 = ce, \quad e_8 = ef.$$

Entonces la matriz de incidencia es

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
a	1	0	1	1	0	0	0	0
b	1	1	0	0	0	0	0	0
c	0	1	0	0	0	1	1	0
d	0	0	1	0	1	0	0	0
e	0	0	0	1	1	0	1	1
f	0	0	0	0	0	1	0	1

Proposición. Sea G un grafo simple no dirigido y sea B su matriz de incidencia. Entonces cada columna de B tiene exactamente dos entradas iguales a 1.

Demostración. Cada columna de B corresponde a una arista e_j. Como G es simple y no dirigido, la arista e_j tiene dos extremos distintos. Por definición de matriz de incidencia, la columna j tiene un 1 exactamente en las filas correspondientes a esos dos extremos, y tiene 0 en todas las demás filas. Por lo tanto, cada columna tiene exactamente dos entradas iguales a 1. □

Hay versiones dirigidas para estas representaciones. En la matriz de adyacencia de un digrafo, se usa $a_{ij} = 1$ si existe el arco $v_i \to v_j$. En una lista de adyacencia se pueden guardar sólo los vecinos de salida. En una matriz de incidencia dirigida se registra cómo incide cada arco, por ejemplo distinguiendo su extremo inicial y su extremo final.

5.4. Isomorfismos e invariantes

Ahora pensemos en una pregunta distinta. A veces dos dibujos se ven muy diferentes, pero representan la misma estructura. En ese caso no queremos decir que son grafos distintos de verdad, sino que son el mismo grafo con nombres o posiciones distintas. La noción formal que captura esta idea es la de isomorfismo.

Definición. Sean $G = (V, E)$ y $H = (W, F)$ dos grafos simples. Decimos que G y H son *isomorfos* si existe una biyección

$$f : V \to W$$

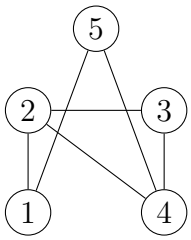
tal que, para todo par de vértices $u, v \in V$,

$$uv \in E \quad \text{si y sólo si} \quad f(u)f(v) \in F.$$

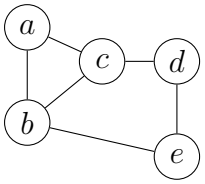
En ese caso, la función f se llama un *isomorfismo* entre G y H.

Un isomorfismo es un cambio de nombres de los vértices que preserva exactamente las adyacencias. Si dos vértices estaban conectados antes del cambio de nombres, sus imágenes deben quedar conectadas después. Si dos vértices no estaban conectados antes, sus imágenes tampoco deben quedar conectadas después.

Ejemplo. Consideremos los siguientes grafos.



G



H

Los grafos son isomorfos. Un isomorfismo está dado por

$$f(1) = e, \quad f(2) = b, \quad f(3) = d, \quad f(4) = c, \quad f(5) = a.$$

Para verificarlo, basta revisar que las aristas de G se transforman exactamente en las aristas de H:

arista de G	imagen en H
12	eb
23	bd
34	dc
45	ca
51	ae
24	bc

Definición. Una propiedad de grafos es un *invariante por isomorfismo* si cada vez que dos grafos son isomorfos, ambos tienen la misma propiedad.

Proposición. Si dos grafos simples G y H son isomorfos, entonces tienen:

1. el mismo número de vértices;
2. el mismo número de aristas;
3. la misma cantidad de vértices de cada grado.

Demostración. Sea $f : V(G) \to V(H)$ un isomorfismo. Como f es una biyección, G y H tienen el mismo número de vértices.

Además, por definición de isomorfismo, cada arista uv de G determina una arista $f(u)f(v)$ de H. Recíprocamente, como f es biyectiva y preserva no-adyacencias, cada arista de H viene de una única arista de G. Por lo tanto, hay una biyección entre las aristas de G y las aristas de H, y ambos grafos tienen el mismo número de aristas.

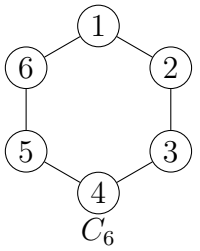
Finalmente, fijemos un vértice $u \in V(G)$. Los vecinos de u en G son exactamente los vértices que, bajo f, se transforman en vecinos de $f(u)$ en H. Así,

$$d_G(u) = d_H(f(u)).$$

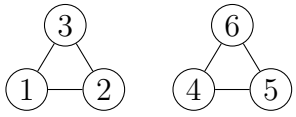
Como f es una biyección entre vértices, la cantidad de vértices de cada grado coincide en ambos grafos. □

Estos invariantes son muy útiles para demostrar que dos grafos no son isomorfos. Por ejemplo, si un grafo tiene 10 aristas y el otro tiene 11, entonces no pueden ser isomorfos. Sin embargo, que estos invariantes coincidan no alcanza para concluir que dos grafos sí son isomorfos.

Ejemplo. Los siguientes dos grafos tienen 6 vértices, 6 aristas y todos sus vértices tienen grado 2.



C_6



$C_3 \cup C_3$

No son isomorfos: el segundo contiene un ciclo de largo 3, mientras que el primero no.

5.5. Subgrafos

Definición. Sea $G = (V, E)$ un grafo. Un *subgrafo* de G es un grafo $G' = (V', E')$ tal que

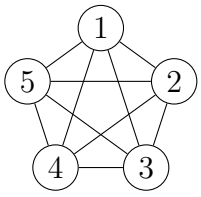
$$V' \subseteq V \quad \text{y} \quad E' \subseteq E,$$

donde cada arista de E' tiene sus extremos en V'. Si $G' \neq G$, decimos que G' es un *subgrafo propio* de G.

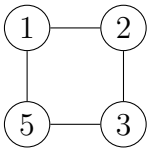
Ejemplo. Si $G = K_5$, podemos obtener un subgrafo eliminando algunos vértices y algunas aristas. Por ejemplo, el grafo con vértices $\{1, 2, 3, 5\}$ y aristas

$$\{12, 23, 35, 15\}$$

es un subgrafo propio de K_5.



$G = K_5$



G'

Contener ciertos subgrafos también es un invariante por isomorfismo. Por ejemplo, contener un K_ℓ, un P_ℓ o un C_ℓ no depende de los nombres de los vértices.

> **Proposición.** Si dos grafos simples G y H son isomorfos, entonces para todo $k \geq 1$, G tiene un ciclo de largo k si y sólo si H tiene un ciclo de largo k.

Demostración. Sea $f : V(G) \to V(H)$ un isomorfismo. Supongamos que G tiene un ciclo de largo k, digamos

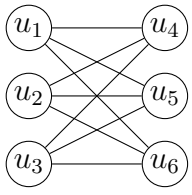
$$v_0, v_1, \ldots, v_{k-1}, v_0,$$

con aristas consecutivas en G. Como f preserva adyacencias,

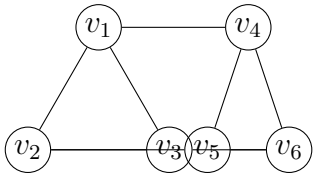
$$f(v_0), f(v_1), \ldots, f(v_{k-1}), f(v_0)$$

es un ciclo de largo k en H. Esto prueba una implicancia. La otra se obtiene aplicando el mismo argumento al isomorfismo inverso f^{-1}. □

Ejemplo. Los siguientes dos grafos tienen el mismo número de vértices, el mismo número de aristas y la misma secuencia de grados. De hecho, ambos tienen 6 vértices, 9 aristas, y todos sus vértices tienen grado 3. Sin embargo, no son isomorfos.



$G = K_{3,3}$



H

El grafo H contiene un ciclo de largo 3, por ejemplo v_1, v_2, v_3, v_1. El grafo $G = K_{3,3}$ no contiene ciclos de largo 3, porque es bipartito y en un grafo bipartito no puede haber un ciclo impar. Por la proposición anterior, G y H no son isomorfos.

5.6. Complemento

Definición. Sea $G = (V, E)$ un grafo simple. El *complemento* de G es el grafo

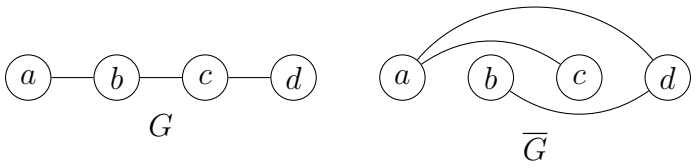
$$\overline{G} = (V, \overline{E}),$$

donde para todo par de vértices distintos $u, v \in V$,

$$uv \in \overline{E} \quad \text{si y sólo si} \quad uv \notin E.$$

El complemento tiene los mismos vértices que el grafo original. La diferencia está en las aristas: dos vértices están conectados en $\overline{G}$ exactamente cuando no estaban conectados en G.

Ejemplo. Consideremos el camino $a - b - c - d$. Su complemento tiene las aristas ac, ad y bd.



Sea $G = (V, E)$ un grafo simple con $|V| = n$ y $|E| = m$. Entonces

$$|E(\overline{G})| = \binom{n}{2} - m.$$

Demostración. En un grafo simple con conjunto de vértices V, cada arista posible corresponde a un par de vértices distintos. Hay $\binom{n}{2}$ pares de este tipo. Cada uno de esos pares aparece como arista en exactamente uno de los grafos G y $\overline{G}$: si está en G, no está en $\overline{G}$, y si no está en G, está en $\overline{G}$. Por lo tanto,

$$|E| + |E(\overline{G})| = \binom{n}{2}.$$

Como $|E| = m$, despejamos

$$|E(\overline{G})| = \binom{n}{2} - m.$$

□

Sea $G = (V, E)$ un grafo simple con $|V| = n$. Para todo vértice $v \in V$ se cumple

$$d_G(v) + d_{\overline{G}}(v) = n - 1.$$

Demostración. Fijemos un vértice $v \in V$. Hay exactamente $n - 1$ vértices distintos de v. Cada uno de ellos es adyacente a v en exactamente uno de los grafos G y $\overline{G}$. Por lo tanto, al sumar el número de vecinos de v en G y el número de vecinos de v en $\overline{G}$, contamos exactamente esos $n - 1$ vértices. Así,

$$d_G(v) + d_{\overline{G}}(v) = n - 1.$$

□

Definición. Un grafo simple G es *auto-complementario* si G es isomorfo a su complemento $\overline{G}$.

Ejemplo. El camino con 4 vértices, P_4, es auto-complementario. En efecto, si

$$E(P_4) = \{ab, bc, cd\},$$

entonces

$$E(\overline{P_4}) = \{ac, ad, bd\}.$$

Ambos grafos son caminos con 4 vértices, aunque dibujados con nombres distintos.

Si G es un grafo auto-complementario con n vértices, entonces

$$\binom{n}{2}$$

es par. Equivalentemente, $n \equiv 0$ o $n \equiv 1 \pmod 4$.

Demostración. Sea $m = |E(G)|$. Como G es auto-complementario, G y $\overline{G}$ son isomorfos. En particular, tienen el mismo número de aristas. Por la fórmula para el complemento,

$$|E(\overline{G})| = \binom{n}{2} - m.$$

Como $|E(\overline{G})| = m$, tenemos

$$2m = \binom{n}{2}.$$

Por lo tanto, $\binom{n}{2}$ es par.

Ahora,

$$\binom{n}{2} = \frac{n(n-1)}{2}.$$

Este número es par si y sólo si $n(n-1)$ es divisible por 4. Como n y $n-1$ son consecutivos, exactamente uno de ellos es par. Entonces $n(n-1)$ es divisible por 4 si y sólo si n es divisible por 4 o $n-1$ es divisible por 4. Esto equivale a

$$n \equiv 0 \quad \text{o} \quad n \equiv 1 \pmod 4.$$

□

5.7. Conectividad

Una vez que tenemos un grafo, una de las preguntas más naturales es si podemos movernos de un vértice a otro siguiendo aristas. Por ejemplo, en un mapa de rutas queremos saber si una ciudad es alcanzable desde otra. En una red de computadores queremos saber si dos servidores pueden comunicarse. En un grafo de flujo de control, un camino representa una posible secuencia de instrucciones durante una ejecución.

Definición. Sea $G = (V, E)$ un grafo no dirigido. Un *camino* de largo k desde u hasta v es una secuencia de vértices

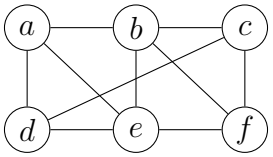
$$v_0, v_1, \ldots, v_k$$

tal que

$$v_0 = u, \quad v_k = v, \quad \text{y} \quad v_{i-1}v_i \in E \text{ para todo } 1 \leq i \leq k.$$

Equivalentemente, es una secuencia de k aristas consecutivas.

Ejemplo. Consideremos el siguiente grafo.



La secuencia

$$d, e, b, c$$

es un camino simple. La secuencia

$$d, e, b, c, d, e, f$$

es un camino no simple, porque repite la arista de. La secuencia

$$d, e, b, a, d$$

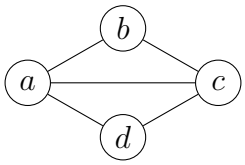
es un camino cerrado.

Lema. Si existe un camino desde u hasta v en un grafo no dirigido, entonces existe un camino desde u hasta v que no repite vértices.

Demostración. Tomemos un camino desde u hasta v de largo mínimo entre todos los caminos que van desde u hasta v. Supongamos que este camino repite algún vértice. Entonces existen índices $i < j$ tales que $v_i = v_j$. Si eliminamos la parte del camino entre las dos apariciones de ese vértice, obtenemos otro camino desde u hasta v estrictamente más corto. Esto contradice la minimalidad del camino elegido. Por lo tanto, el camino de largo mínimo no repite vértices. □

Definición. Un grafo no dirigido G es *conexo* si para todo par de vértices distintos $u, v \in V(G)$ existe un camino desde u hasta v.

Ejemplo. El grafo



es conexo. En cambio, el grafo

no es conexo, porque no hay camino desde a hasta c.

Definición. Sea G un grafo no dirigido. Una *componente conexa* de G es un subgrafo H de G tal que:

- H es conexo;
- H no es subgrafo propio de ningún subgrafo conexo de G.

Es decir, una componente conexa es un subgrafo conexo maximal.

Proposición. Todo grafo no dirigido es la unión disjunta de sus componentes conexas.

Demostración. Definamos una relación $\sim$ sobre los vértices de G por

$$u \sim v \quad \text{si y sólo si} \quad u = v \text{ o existe un camino desde } u \text{ hasta } v.$$

Esta relación es reflexiva, porque $u = u$. Es simétrica, porque en un grafo no dirigido todo camino desde u hasta v puede recorrerse al revés para obtener un camino desde v hasta u. Es transitiva, porque si existe un camino desde u hasta v y existe un camino desde v hasta w, podemos concatenarlos para obtener un camino desde u hasta w. Por lo tanto, $\sim$ es una relación de equivalencia.

Las clases de equivalencia particionan $V(G)$. Cada clase induce un subgrafo conexo, porque todos sus vértices son alcanzables entre sí. Además, ese subgrafo es maximal con respecto a ser conexo: si agregáramos un vértice de otra clase, no habría camino entre ese vértice y los vértices de la clase original. Por lo tanto, estas clases inducen exactamente las componentes conexas de G. □

Proposición. Si dos grafos simples son isomorfos, entonces tienen la misma cantidad de componentes conexas.

Demostración. Sea $f : V(G) \to V(H)$ un isomorfismo. Primero observemos que si $v_0, v_1, \ldots, v_k$ es un camino en G, entonces

$$f(v_0), f(v_1), \ldots, f(v_k)$$

es un camino en H, porque cada arista $v_{i-1}v_i$ se transforma en la arista $f(v_{i-1})f(v_i)$. Así, vértices conectados por caminos en G se transforman en vértices conectados por caminos en H. Aplicando el mismo argumento a f^{-1}, obtenemos también la implicancia recíproca. Por lo tanto, f induce una biyección entre las componentes conexas de G y las componentes conexas de H. □

Corolario. Si un grafo es conexo y otro no lo es, entonces no son isomorfos.

Demostración. Un grafo conexo tiene exactamente una componente conexa. Un grafo no conexo tiene al menos dos componentes conexas. Por la proposición anterior, dos grafos isomorfos deben tener la misma cantidad de componentes conexas. Por lo tanto, un grafo conexo y uno no conexo no pueden ser isomorfos. □

5.8. Conectividad en digrafos

En digrafos, la dirección de los arcos cambia la noción de alcanzabilidad. No es lo mismo poder llegar desde u hasta v que poder llegar desde v hasta u. Por eso aparecen dos nociones de conectividad: fuerte y débil.

Definición. Sea $G = (V, E)$ un digrafo. Un *dicamino* de largo k desde u hasta v es una secuencia de vértices

$$v_0, v_1, \ldots, v_k$$

tal que

$$v_0 = u, \quad v_k = v, \quad \text{y} \quad (v_{i-1}, v_i) \in E \text{ para todo } 1 \leq i \leq k.$$

Definición. Sea $G = (V, E)$ un digrafo. Un *diciclo* es un dicamino desde un vértice v hasta v.

Definición. Un digrafo G es *fuertemente conexo* si para todo par de vértices $u, v \in V(G)$ existe un dicamino desde u hasta v y existe un dicamino desde v hasta u.

Ejemplo. El diciclo

$$a \to b \to c \to d \to a$$

es fuertemente conexo: desde cualquier vértice podemos llegar a cualquier otro siguiendo el ciclo. En cambio, el digrafo

$$a \to b \to c$$

no es fuertemente conexo, porque existe un dicamino desde a hasta c, pero no desde c hasta a.

Definición. Sea $G = (V, E)$ un digrafo. El *grafo no dirigido subyacente* de G es el grafo no dirigido G^{und} con el mismo conjunto de vértices, donde u y v son adyacentes si y sólo si

$$(u, v) \in E \quad \text{o} \quad (v, u) \in E.$$

Definición. Un digrafo G es *débilmente conexo* si su grafo no dirigido subyacente G^{und} es conexo.

Ejemplo. El digrafo

$$a \to b, \qquad c \to b$$

no es fuertemente conexo, porque no podemos llegar desde b hasta a ni desde b hasta c. Sin embargo, sí es débilmente conexo: si olvidamos las direcciones, obtenemos el camino $a - b - c$.

Proposición. Todo digrafo fuertemente conexo es débilmente conexo.

Demostración. Sea G un digrafo fuertemente conexo. Tomemos dos vértices $u, v \in V(G)$. Como G es fuertemente conexo, existe un dicamino desde u hasta v. Si olvidamos las direcciones de sus arcos, obtenemos un camino desde u hasta v en el grafo no dirigido subyacente G^{und}. Como esto vale para todo par u, v, el grafo G^{und} es conexo. Por lo tanto, G es débilmente conexo. □

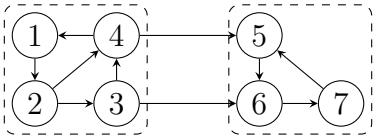
Ejemplo. La recíproca no es cierta. El digrafo $a \to b \to c$ es débilmente conexo, porque su grafo subyacente es el camino $a - b - c$. Pero no es fuertemente conexo, porque no existe un dicamino desde c hasta a.

Definición. Sea G un digrafo. Una *componente fuertemente conexa* de G es un subgrafo H de G tal que:

- H es fuertemente conexo;
- H no es subgrafo propio de otro subgrafo fuertemente conexo de G.

Es decir, una componente fuertemente conexa es un subgrafo fuertemente conexo maximal.

Ejemplo. En el siguiente digrafo hay dos componentes fuertemente conexas grandes.



Dentro de cada región punteada podemos ir de cualquier vértice a cualquier otro siguiendo direcciones. Además, hay arcos desde la primera región hacia la segunda, pero no hay arcos que permitan volver.

Ejercicios

1. Un sistema de compilación mantiene un *grafo de dependencias* entre módulos. Los vértices son

$$V = \{\texttt{core}, \texttt{parser}, \texttt{typecheck}, \texttt{optimizer}, \texttt{codegen}, \texttt{cli}, \texttt{tests}\}.$$

   Hay un arco dirigido desde un módulo x hacia un módulo y si un cambio en x puede afectar directamente a y. Suponga que

$$E = \{(\texttt{core}, \texttt{parser}), (\texttt{core}, \texttt{typecheck}), (\texttt{parser}, \texttt{typecheck}), (\texttt{typecheck}, \texttt{optimizer}),$$
$$(\texttt{optimizer}, \texttt{codegen}), (\texttt{codegen}, \texttt{cli}), (\texttt{typecheck}, \texttt{tests}), (\texttt{tests}, \texttt{typecheck}), (\texttt{parser}, \texttt{cli})\}.$$

   a) Explique por qué este modelo debe ser un grafo dirigido.

   b) Interprete, en términos del sistema de compilación, qué significa que exista un camino dirigido desde un módulo x hasta un módulo y.

   c) Si se modifica `core`, ¿qué módulos podrían verse afectados?

   d) Determine las componentes fuertemente conexas del grafo.

   e) Determine si el grafo es débilmente conexo y si es fuertemente conexo.

2. Diremos que dos ciclos de largo ℓ de un grafo son distintos si tienen distinto conjunto de aristas.

   a) Demuestre que si G y H son isomorfos, entonces para todo $\ell \geq 3$ tienen la misma cantidad de ciclos de largo ℓ.

   Indicación: si $v_0, v_1, \ldots, v_{\ell-1}, v_0$ es un ciclo en G, aplique el isomorfismo a todos sus vértices.

b) Sea P el grafo prisma triangular, con vértices

$$\{a_1, a_2, a_3, b_1, b_2, b_3\}$$

y aristas

$$E(P) = \{a_1a_2, a_2a_3, a_3a_1, b_1b_2, b_2b_3, b_3b_1, \\ a_1b_1, a_2b_2, a_3b_3\}.$$

Sea $K_{3,3}$ el grafo bipartito completo con partes

$$A = \{a_1, a_2, a_3\} \quad \text{y} \quad B = \{b_1, b_2, b_3\}.$$

Verifique que ambos grafos tienen 6 vértices, 9 aristas y todos sus vértices tienen grado 3. Luego use la parte anterior para demostrar que P y $K_{3,3}$ no son isomorfos.

3. Sea G un grafo simple no dirigido con vértices

$$V(G) = \{v_1, \ldots, v_n\}$$

y matriz de adyacencia $A = (a_{ij})_{1 \leq i,j \leq n}$. Recordemos que

$$(A^2)_{ij} = \sum_{k=1}^{n} a_{ik}a_{kj}.$$

a) Demuestre que, si $i \neq j$, entonces $(A^2)_{ij}$ es el número de vecinos comunes de v_i y v_j.

b) Demuestre que

$$(A^2)_{ii} = \deg(v_i)$$

para todo $i \in \{1, \ldots, n\}$.

c) Demuestre que el número de triángulos de G es

$$\frac{1}{6} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(A^2)_{ij}.$$

Indicación: piense cuántas veces cuenta la suma a cada triángulo.

4. Sea G un grafo simple no dirigido con al menos dos vértices.

a) Demuestre que si G no es conexo, entonces $\overline{G}$ es conexo.
Indicación: tome dos vértices u, v. Si están en componentes conexas distintas de G, entonces son adyacentes en $\overline{G}$. Si están en la misma componente conexa de G, use un vértice w de otra componente.

b) Concluya que no puede ocurrir que G y $\overline{G}$ sean ambos no conexos.

c) Demuestre que si G es auto-complementario, entonces G es conexo.

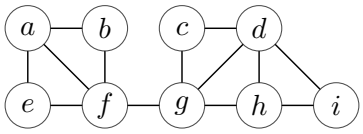
5.9. Recorrer aristas

Venimos de hablar de representación, isomorfismos y conectividad. Ahora la pregunta cambia un poco: no sólo queremos saber si podemos llegar de un lugar a otro, sino si podemos recorrer una red completa sin desperdiciar pasos.

El ejemplo clásico es tal vez el origen de la teoría de grafos: los puentes de Königsberg. La pregunta era si se podía caminar cruzando cada puente exactamente una vez y volver al punto de partida. Al olvidar la geometría del mapa, las zonas de tierra se vuelven vértices y los puentes se vuelven aristas. Así, un problema de paseo se transforma en una pregunta sobre recorridos en multigrafos.

Definición. Sea $G = (V, E)$ un multigrafo. Un *camino euleriano* de G es un recorrido que usa cada arista de G exactamente una vez.
Un *circuito euleriano* de G es un camino euleriano que empieza y termina en el mismo vértice.

Ejemplo. Consideremos el siguiente multigrafo y preguntémonos si admite un circuito euleriano.



El obstáculo aparece cuando intentamos cerrar un recorrido que use todas las aristas. Los vértices a y h tienen grado impar. En un recorrido cerrado, cada vez que entramos a un vértice también debemos salir de él por otra arista; las aristas incidentes quedan emparejadas. Un vértice de grado impar rompe exactamente esa idea. Por lo tanto, este grafo no puede tener un circuito euleriano.

5.10. Circuitos eulerianos

5.10.1. Argumentos de camino maximal

Antes de atacar directamente un circuito que use todas las aristas, busquemos una estructura más sencilla: ciclos. Esta es una maniobra que aparece mucho en grafos. Tomamos un camino que no se pueda extender más; si todavía hay suficiente grado en el último vértice, la única forma de no poder extenderlo es que volvamos a un vértice ya visitado. Eso fuerza un ciclo.

Lema. Sea $G = (V, E)$ un multigrafo finito no vacío. Si $d(v) \geq 2$ para todo $v \in V$, entonces G tiene un ciclo.

Demostración. Tomemos un camino simple maximal

$$v_0, v_1, \ldots, v_k.$$

Como $d(v_k) \geq 2$, hay una arista e incidente en v_k que no es la arista del camino por la que llegamos a v_k; si el camino tiene largo 0, tomamos cualquier arista incidente. Sea u el otro extremo de e. Por maximalidad del camino, u no puede ser un vértice nuevo: si no apareciera entre $v_0, \ldots, v_k$, podríamos agregarlo al final y obtener un camino simple más largo. Por lo tanto $u = v_i$ para algún $i \leq k$, y la parte del camino desde v_i hasta v_k, junto con la arista e, cierra un ciclo. □

Este lema es útil porque nos da el primer paso para construir circuitos eulerianos. Nos garantiza un primer ciclo. Después viene la idea clave: encontrar un ciclo, borrarlo, resolver lo que queda, e insertar las soluciones de vuelta en el ciclo original. No es sólo una demostración; también es un algoritmo.

Teorema (Caracterización de Euler)**.** Sea $G = (V, E)$ un multigrafo conexo. Entonces G tiene un circuito euleriano si y sólo si todos sus vértices tienen grado par.

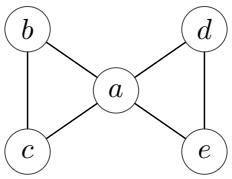
Demostración. Supongamos primero que G tiene un circuito euleriano. En un recorrido cerrado no hay vértices especiales de inicio o término: cada vez que pasamos por un vértice usamos una arista para entrar y otra arista distinta para salir. Como el circuito usa todas las aristas exactamente una vez, las aristas incidentes en cada vértice quedan emparejadas en pares entrada-salida. Por lo tanto, todos los grados son pares.

Para la recíproca, probaremos por inducción en $m = |E(G)|$ que todo multigrafo conexo con todos sus grados pares tiene un circuito euleriano. El caso $m = 0$ corresponde al recorrido trivial. Supongamos entonces que $m > 0$. Como G es conexo y $m > 0$, no hay vértices aislados. Además, todos los grados son pares, así que todo vértice tiene grado al menos 2. Por el lema anterior, existe un ciclo C en G.

Eliminemos las aristas de C, conservando todos los vértices, y llamemos H al multigrafo resultante. Al borrar el ciclo, cada vértice pierde 0 o 2 aristas incidentes, de modo que todos los grados de H siguen siendo pares. Cada componente conexa de H que aún tiene aristas tiene menos de m aristas, así que por hipótesis de inducción posee un circuito euleriano. Además, tal componente debe contener algún vértice de C; de lo contrario, ya habría estado separada del ciclo en el grafo original, contradiciendo la conexidad de G.

Finalmente recorremos C y vamos insertando los circuitos de las componentes restantes. Cuando llegamos a un vértice de C que pertenece a una componente de H con aristas, recorremos completo el circuito euleriano de esa componente y volvemos al mismo vértice; desde ahí continuamos por C. Después de hacer esto para todas las componentes, obtenemos un recorrido cerrado que usa cada arista de C y cada arista de H exactamente una vez, es decir, un circuito euleriano de G. □

Ejemplo. En el siguiente grafo todos los vértices tienen grado par. Por el teorema anterior, no tenemos que adivinar; sabemos que existe un circuito euleriano.



Un circuito euleriano es

$$a, b, c, a, d, e, a.$$

La gracia del teorema es que esta respuesta se podía predecir mirando sólo los grados.

5.11. Caminos eulerianos

La caracterización de caminos eulerianos se obtiene reduciendo el problema al caso de circuitos. Para pasar de caminos a circuitos, agregamos una arista entre los dos extremos. Si después encon-

tramos un circuito euleriano en el grafo aumentado, borrar esa arista artificial abre el circuito y deja un camino.

Teorema (Caminos eulerianos)**.** Sea $G = (V, E)$ un multigrafo conexo. Entonces G tiene un camino euleriano que no es circuito si y sólo si tiene exactamente dos vértices de grado impar.

Demostración. Supongamos primero que existe un camino euleriano desde s hasta t, con $s \neq t$. En todo vértice distinto de los extremos, cada visita usa una arista para entrar y otra para salir, por lo que sus aristas incidentes quedan emparejadas. En cambio, el vértice inicial s tiene una salida sin entrada correspondiente, y el vértice final t tiene una entrada sin salida correspondiente. Así, los únicos vértices de grado impar son precisamente s y t.

Recíprocamente, supongamos que los únicos vértices de grado impar son s y t. Agregamos una arista artificial entre ellos. En el nuevo multigrafo todos los grados son pares y la conexidad se conserva, por lo que el teorema de Euler nos entrega un circuito euleriano. Ese circuito usa la arista artificial exactamente una vez; al borrarla, el circuito se abre y queda un camino euleriano en el grafo original, con extremos s y t. □

Ejemplo. En el grafo del primer ejemplo, los únicos vértices de grado impar son a y h. Por lo tanto, existe un camino euleriano desde uno de ellos hasta el otro. Un ejemplo de tal camino es

$$a, b, f, e, a, f, g, c, d, i, h, g, d, h.$$

5.12. La versión dirigida

En grafos dirigidos debemos distinguir entre entrar y salir de un vértice. La condición de paridad se reemplaza por una condición de balance entre grados de entrada y salida. La conectividad también cambia: para recorrer todos los arcos basta que el grafo sea conexo al olvidar las orientaciones. No necesitamos que sea fuertemente conexo, porque el recorrido euleriano puede imponer una dirección local sin permitir viajes arbitrarios entre todos los pares de vértices.

Teorema (Circuitos eulerianos dirigidos)**.** Sea $G = (V, E)$ un multidigrafo sin vértices aislados. Entonces G tiene un circuito euleriano dirigido si y sólo si:

1. G es débilmente conexo, y
2. para todo vértice v se cumple
$$d^-(v) = d^+(v).$$

Otra vez buscamos primero la estructura más sencilla. En el caso dirigido esa estructura no es un ciclo cualquiera, sino un diciclo: un ciclo que respeta la orientación de cada arco.

Lema. Sea $G = (V, E)$ un multidigrafo finito no vacío. Si $d^+(v) \geq 1$ para todo $v \in V$, entonces G tiene un ciclo dirigido.

Demostración. Tomemos un camino dirigido simple maximal

$$v_0, v_1, \ldots, v_k.$$

Como $d^+(v_k) \geq 1$, existe un arco desde v_k hacia algún vértice u. Por maximalidad del camino, u no puede ser un vértice nuevo: si no apareciera entre $v_0, \ldots, v_k$, podríamos extender el camino dirigido agregando u al final. Escribamos entonces $u = v_i$. La parte final del camino, junto con el arco $v_k \to v_i$, da

$$v_i, v_{i+1}, \ldots, v_k, v_i$$

que es un ciclo dirigido. □

Demostración del teorema. Si existe un circuito euleriano dirigido, entonces cada vez que el recorrido visita un vértice entra por un arco y sale por otro. Como todos los arcos se usan exactamente una vez, esto empareja los arcos entrantes y salientes de cada vértice, y por lo tanto $d^-(v) = d^+(v)$ para todo v. Además, al olvidar las orientaciones, el mismo recorrido conecta todos los vértices no aislados; de ahí la conectividad débil.

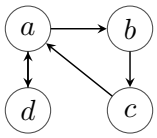
Para la recíproca, usamos inducción en $m = |E(G)|$. Si $m = 0$, no hay nada que recorrer. Supongamos entonces que $m > 0$. Como no hay vértices aislados y los grados están balanceados, todo vértice tiene al menos un arco saliente. Por el lema, G contiene un ciclo dirigido C.

Borramos los arcos de C. Cada vértice pierde 0 arcos, o bien pierde exactamente un arco de entrada y uno de salida, así que el balance $d^- = d^+$ se mantiene. Cada componente débilmente conexa que aún tiene arcos tiene menos de m arcos y, ignorando vértices aislados, satisface nuevamente las hipótesis del teorema; por inducción, posee un circuito euleriano dirigido. Como el grafo original era débilmente conexo, cada una de esas componentes contiene algún vértice de C.

Ahora recorremos el ciclo dirigido C. Cuando llegamos a un vértice de C que pertenece a una componente restante, recorremos el circuito euleriano dirigido de esa componente y regresamos al mismo vértice; luego seguimos por C. Al insertar todos esos circuitos, usamos cada arco borrado y cada arco restante exactamente una vez, y obtenemos un circuito euleriano dirigido de todo G. □

El lema dirigido cumple el mismo papel que en el caso no dirigido. Exactamente igual que antes, nos da un algoritmo. La única diferencia es que ahora las piezas que borramos e insertamos son diciclos, y las componentes se miran como débilmente conexas.

Ejemplo. En el siguiente multidigrafo podemos ver explícitamente un circuito euleriano dirigido.



El recorrido

$$a, b, c, a, d, a$$

usa todos los arcos exactamente una vez.

Teorema (Caminos eulerianos dirigidos)**.** Sea $G = (V, E)$ un multidigrafo sin vértices aislados. Entonces G tiene un camino euleriano dirigido que no es circuito si y sólo si G es débilmente conexo y existen dos vértices distintos s, t tales que

$$d^+(s) = d^-(s) + 1,$$

$$d^-(t) = d^+(t) + 1,$$

y para todo vértice $v \neq s, t$ se cumple

$$d^-(v) = d^+(v).$$

Demostración. Si existe un camino euleriano dirigido desde s hasta t, entonces todo vértice interno recibe tantas entradas como salidas durante el recorrido. El vértice inicial s tiene una salida extra, y el vértice final t tiene una entrada extra; esto entrega exactamente las igualdades del enunciado. Además, al olvidar las orientaciones, el camino recorre todos los arcos y por lo tanto conecta todos los vértices no aislados.

Para probar la recíproca, agregamos un arco artificial desde t hacia s. Con ese arco, todos los vértices quedan balanceados, y la conectividad débil no cambia. Por el teorema anterior, el nuevo multidigrafo tiene un circuito euleriano dirigido. Ese circuito usa el arco artificial $t \to s$ exactamente una vez; al borrarlo, el circuito se abre y deja un camino euleriano dirigido desde s hasta t en el grafo original. □

Ejemplo. Consideremos ahora un ejemplo donde no hay circuito euleriano dirigido, pero sí hay camino euleriano dirigido. El multidigrafo tiene arcos

$$(a, b), (b, c), (c, a), (a, d), (d, b).$$

Aquí a tiene una salida más que entradas, b tiene una entrada más que salidas, y los demás vértices están balanceados. Entonces existe un camino euleriano dirigido desde a hasta b:

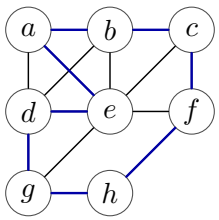
$$a, d, b, c, a, b.$$

5.13. Recorrer vértices

Ahora cambiamos la pregunta: en vez de recorrer aristas, queremos recorrer vértices. Los problemas eulerianos recorren aristas. La versión Hamiltoniana pide recorrer vértices. Este cambio parece pequeño, pero cambia completamente el tipo de problema. En el mundo euleriano la respuesta queda capturada por grados: paridad o balance. En el mundo Hamiltoniano no hay una caracterización comparable, y el problema está muy cerca del Travelling Salesperson Problem.

Definición. Sea $G = (V, E)$ un grafo. Un *camino Hamiltoniano* de G es un camino que usa cada vértice de G exactamente una vez.
Un *ciclo Hamiltoniano* de G es un ciclo que usa cada vértice de G exactamente una vez.

Ejemplo. En el siguiente grafo queremos decidir si existe un ciclo Hamiltoniano.



No es tan fácil argumentar por qué o por qué no. En este dibujo, las aristas destacadas muestran un ciclo Hamiltoniano. La dificultad está en que los grados altos ayudan, pero no dicen por sí solos qué vértices deben quedar consecutivos en el ciclo.

5.14. Condiciones hamiltonianas

Conviene separar dos tipos de condiciones. Primero, condiciones suficientes: hipótesis fuertes que garantizan que existe un ciclo Hamiltoniano. Después, condiciones necesarias: propiedades que todo ciclo Hamiltoniano tendría que cumplir, y que por lo tanto sirven para descartar ejemplos.

5.14.1. Condiciones suficientes

Una primera condición suficiente viene de la densidad del grafo. La intuición es que, si cada vértice tiene muchas opciones para entrar y salir, entonces hay suficiente libertad para cerrar un ciclo que pase por todos. El teorema de Dirac formaliza una versión muy limpia de esa idea.

> **Teorema** (Dirac)**.** Sea $G = (V, E)$ un grafo simple con $n \geq 3$ vértices. Si $d(v) \geq \frac{n}{2}$ para todo $v \in V$, entonces G tiene un ciclo Hamiltoniano.

Dirac es una condición suficiente, no una caracterización. Por ejemplo, el ciclo C_n es Hamiltoniano para todo $n \geq 3$, pero si $n \geq 5$ todos sus vértices tienen grado $2 < \frac{n}{2}$. Es decir, no satisfacer Dirac no significa no ser Hamiltoniano.

5.14.2. Condiciones necesarias

Para obtener condiciones necesarias, una buena forma de pensar es preguntar qué tendría que ocurrir dentro de cualquier ciclo Hamiltoniano. En un ciclo, cada vértice tiene exactamente dos vecinos del ciclo: uno por donde entramos y otro por donde salimos.

> **Proposición.** Si $G = (V, E)$ tiene un ciclo Hamiltoniano, entonces todo vértice de G tiene grado al menos 2.

Demostración. En un ciclo Hamiltoniano, cada vértice aparece exactamente una vez, pero dentro del ciclo queda unido a dos vecinos: el vértice anterior y el vértice siguiente. Esas dos aristas también pertenecen a G. Aunque G pueda tener más aristas además de las del ciclo, ya esas dos aristas obligan a que todo vértice de G tenga grado al menos 2. □

En particular, un grafo con ciclo Hamiltoniano tiene exactamente 0 vértices de grado 1. Para caminos Hamiltonianos la restricción es un poco más débil: todo vértice de grado 1 debe ser extremo del camino. Como un camino tiene sólo dos extremos, un grafo con camino Hamiltoniano puede tener a lo más dos vértices de grado 1. Estas condiciones son necesarias, no suficientes.

La frase "a lo más dos" es importante. Dos hojas no garantizan un camino Hamiltoniano; sólo dicen que el obstáculo obvio de grado 1 no aparece todavía.

> **Proposición.** Sea $G = (V, E)$ un grafo bipartito con bipartición $L \cup R = V$. Si $|L| \neq |R|$, entonces G no tiene ciclo Hamiltoniano.

Demostración. En un grafo bipartito, todo ciclo alterna entre vértices de L y vértices de R. Por eso, cada vez que el ciclo avanza desde una parte, el siguiente vértice cae en la otra parte, y al cerrar el ciclo no puede quedar una parte usada más veces que la otra. En particular, un ciclo Hamiltoniano,

que pasa por todos los vértices, tendría que usar la misma cantidad de vértices de L y de R. Si $|L| \neq |R|$, tal ciclo no puede existir. □

Ejemplo. Determinemos cuándo el grafo bipartito completo $K_{\ell,r}$ tiene ciclo Hamiltoniano. Por la proposición anterior, es necesario que $\ell = r$. Si $\ell = r \geq 2$, podemos alternar los vértices de las dos partes y cerrar el ciclo. Por lo tanto, $K_{\ell,r}$ tiene ciclo Hamiltoniano si y sólo si $\ell = r \geq 2$.

Ejercicios

1. Un equipo debe revisar los enlaces físicos entre servidores de un pequeño centro de datos. Cada enlace debe revisarse exactamente una vez. Los servidores son

$$A, B, C, D, E, F$$

y los enlaces son

$$AB, AB, AC, BC, BD, CD, DE, EF, FD, CF.$$

Observe que hay dos enlaces entre A y B.

   a) Modele la situación como un multigrafo.

   b) ¿Puede el equipo empezar en un servidor, revisar todos los enlaces exactamente una vez, y volver al servidor inicial?

   c) Si no se exige volver al servidor inicial, ¿puede revisar todos los enlaces exactamente una vez?

   d) Si la respuesta anterior es afirmativa, determine en qué servidores podría empezar y terminar la revisión.

2. Estudiaremos una familia que separa muy bien las nociones eulerianas y hamiltonianas. Definimos Q_n como el siguiente grafo: sus vértices son los strings binarios de largo n, y dos vértices son adyacentes si difieren en exactamente una coordenada. Dicho grafo se conoce como el *hipercubo* de dimensión n.

   a) ¿Cuántos vértices y cuántas aristas tiene Q_n?

   b) ¿Es Q_n conexo? ¿Es Q_n bipartito?

   c) Determine para qué valores de n el grafo Q_n tiene un camino euleriano.

   d) Pruebe que Q_n tiene un ciclo Hamiltoniano para todo $n \geq 2$.

3. *a*) Sea G un grafo simple y sea $\delta(G)$ su grado mínimo; es decir, $\delta(G) = \min_{v \in V(G)} d(v)$. Pruebe que si $\delta(G) \geq k$ para algún $k \geq 2$, entonces G contiene un ciclo de largo al menos $k + 1$.

   b) Sea G un multigrafo conexo con exactamente $2r$ vértices de grado impar, con $r \in \mathbb{N}$. Muestre que las aristas de G pueden cubrirse con r caminos que no repitan aristas.

4. Sea G un grafo simple. El *grafo de líneas* de G, denotado $L(G)$, se define así: los vértices de $L(G)$ son las aristas de G, y dos vértices de $L(G)$ son adyacentes si las aristas correspondientes de G comparten un extremo.

a) Dibuje $L(K_4)$.

b) ¿Cúal es el grafo de lineas de un ciclo C_n? ¿Y el grafo de líneas de un camino P_n?

c) Pruebe que si G tiene un circuito euleriano, entonces $L(G)$ tiene un ciclo Hamiltoniano.

d) Pruebe que si G tiene un camino euleriano, entonces $L(G)$ tiene un camino Hamiltoniano.

e) ¿Es cierta la recíproca de la primera parte? Es decir, si $L(G)$ tiene un ciclo Hamiltoniano, ¿debe G tener un circuito euleriano?

5.15. Caminos más cortos

Hasta ahora hemos usado grafos para modelar relaciones, conectividad y recorridos. En muchas aplicaciones, sin embargo, no basta con saber si existe un camino. También queremos saber qué tan caro es usarlo.

Por ejemplo, en una red de computadores, los vértices pueden representar servidores y las aristas pueden representar conexiones. Si cada conexión tiene un tiempo de respuesta, entonces no queremos simplemente llegar desde un servidor a otro: queremos llegar lo más rápido posible. Lo mismo ocurre con rutas entre ciudades, costos de transporte, latencias, o dependencias con distintos tiempos de ejecución.

Formalicemos esta idea agregando pesos a los arcos de un digrafo.

Definición. Un *digrafo con pesos positivos* es una terna $G = (V, E, w)$, donde (V, E) es un digrafo y $w : E \to \mathbb{R}_{>0}$ asigna un peso positivo a cada arco.
Si $P = v_0, v_1, \ldots, v_k$ es un dicamino, su *largo* o *costo* es $w(P) = \sum_{i=1}^{k} w(v_{i-1}v_i)$.

Con esta definición, podemos definir la distancia entre dos vértices como el costo mínimo de un dicamino entre ellos.

Definición. Sea $G = (V, E, w)$ un digrafo con pesos positivos y sean $u, v \in V$. La *distancia* de u a v, denotada $d(u, v)$, es

$$d(u,v) = \begin{cases} \infty & \text{si no existe un dicamino de } u \text{ a } v, \\ \min\{w(P) : P \text{ es un dicamino de } u \text{ a } v\} & \text{en otro caso.} \end{cases}$$

En esta clase nos interesa el siguiente problema. Fijamos un vértice de origen s y queremos calcular la distancia desde s hacia todos los demás vértices. Este problema se conoce como *camino más corto desde un origen*, o *SSSP* por su nombre en inglés: *single-source shortest paths*.

En esta clase trabajaremos sólo con pesos positivos. Notemos que esto garantiza que los dicaminos de costo mínimo no repiten vértices.

Proposición. Sea $G = (V, E, w)$ un digrafo con pesos positivos. Si existe un dicamino de costo mínimo desde s hasta v, entonces existe uno que no repite vértices.

Demostración. Consideremos un dicamino de costo mínimo desde s hasta v. Si repite algún vértice, entonces contiene un diciclo. Como todos los pesos son positivos, el costo de ese diciclo es positivo. Al borrar el diciclo obtenemos otro dicamino desde s hasta v de costo estrictamente menor, lo que

contradice la optimalidad del dicamino original. Por lo tanto, un dicamino de costo mínimo no necesita repetir vértices. □

5.15.1. La elección avara de Dijkstra

Una idea natural para resolver el problema de SSSP es construir de manera incremental un conjunto U de vértices para los cuales ya conocemos la distancia desde s. Para hacer esa elección miramos los arcos que salen desde U hacia vértices que todavía no están en U.

Definición. Sea $G = (V, E, w)$ un digrafo y sea $U \subseteq V$. Los *vecinos de salida* de U son

$$N^+(U) = \{v \in V \setminus U : \text{existe } u \in U \text{ tal que } (u, v) \in E\}.$$

Pensemos que U es el conjunto de vértices para los cuales ya conocemos la distancia exacta desde s. Si $v \notin U$, una forma natural de llegar a v es llegar primero a algún $u \in U$ y luego cruzar el arco (u, v). El costo de esa posibilidad es $d(s, u) + w(u, v)$. Entonces, para cada vértice de $N^+(U)$, nos interesa la mejor forma de entrar a él desde el conjunto ya resuelto.

Definición. Sea $U \subseteq V$ tal que $s \in U$. Para $v \in N^+(U)$ definimos su *costo candidato desde U* como

$$c_U(v) = \min\{d(s, u) + w(u, v) : u \in U \text{ y } (u, v) \in E\}.$$

Figura 5.1: La frontera de Dijkstra. A la izquierda, el arco destacado representa una entrada candidata desde U hacia un vértice aún no resuelto. A la derecha, todo dicamino desde s hacia un vértice exterior tiene un primer arco que sale de U.

Lema. Sea $G = (V, E, w)$ un digrafo con pesos positivos. Sea $U \subseteq V$ tal que $s \in U$. Si $v^\star \in N^+(U)$ minimiza $c_U(v)$ entre todos los vértices de $N^+(U)$, entonces $d(s, v^\star) = c_U(v^\star)$.

Demostración. Primero, por definición de $c_U(v^\star)$, existe un arco $(u, v^\star)$ con $u \in U$ tal que $c_U(v^\star) = d(s, u) + w(u, v^\star)$. Tomando un dicamino más corto de s a u y luego el arco $(u, v^\star)$, obtenemos un dicamino de s a $v^\star$ de costo $c_U(v^\star)$. Por lo tanto, $d(s, v^\star) \leq c_U(v^\star)$.

Para probar la otra desigualdad, tomemos un dicamino más corto $P : s = x_0, x_1, \ldots, x_k = v^\star$. Como $x_0 = s \in U$ y $x_k = v^\star \notin U$, el dicamino debe salir de U en algún momento. Sea (x_{i-1}, x_i) el primer arco de P que sale de U. Entonces $x_{i-1} \in U$ y $x_i \in N^+(U)$.

Como los pesos son positivos, el costo total de P es al menos el costo del prefijo hasta x_i. Además, como conocemos correctamente la distancia hasta x_{i-1},

$$d(s, v^\star) = w(P) \geq d(s, x_{i-1}) + w(x_{i-1}, x_i).$$

Por definición de $c_U(x_i)$, tenemos $d(s, x_{i-1}) + w(x_{i-1}, x_i) \geq c_U(x_i)$. Finalmente, como $v^\star$ minimiza el costo candidato en $N^+(U)$, tenemos que $c_U(x_i) \geq c_U(v^\star)$. Juntando las desigualdades obtenemos $d(s, v^\star) \geq c_U(v^\star)$. Concluimos que $d(s, v^\star) = c_U(v^\star)$. □

El lema anterior justifica el paso central del algoritmo de Dijkstra. En cada iteración miramos todas las formas de cruzar desde U hacia afuera y agregamos el vértice cuyo costo candidato es menor. El lema dice que, cuando agregamos ese vértice, su distancia ya quedó determinada correctamente.

En el pseudocódigo usaremos $\delta(v)$ para el valor que el algoritmo ha calculado para la distancia desde s hasta v.

Algoritmo 1 Dijkstra, versión de frontera

```
Entrada: un digrafo con pesos positivos G = (V, E, w) y un origen s ∈ V
Salida: distancias desde s hacia los vértices alcanzables
U ← {s}
δ(s) ← 0          ▷ Para reconstruir caminos, se puede guardar el arco que realiza cada mínimo.
while N⁺(U) ≠ ∅ do
    Elegir v* ∈ N⁺(U) que minimice c_U(v)
    δ(v*) ← c_U(v*)
    U ← U ∪ {v*}
return δ para los vértices de U
```

> **Teorema.** Sea $G = (V, E, w)$ un digrafo con pesos positivos y sea $s \in V$. El algoritmo de Dijkstra calcula correctamente $d(s, v)$ para todo vértice $v \in V$ alcanzable desde s.

Demostración. Probamos por inducción sobre el número de iteraciones que, cada vez que un vértice pertenece a U, su distancia desde s ya fue calculada correctamente.

Al comienzo $U = \{s\}$ y $d(s, s) = 0$, así que la afirmación es cierta. Supongamos ahora que la afirmación es cierta para el conjunto actual U. El algoritmo elige un vértice $v^\star \in N^+(U)$ de costo candidato mínimo. Por la propiedad segura de Dijkstra, $d(s, v^\star) = c_U(v^\star)$. Por lo tanto, al agregar $v^\star$ a U, su distancia queda correctamente calculada. Esto mantiene la invariante.

Como en cada iteración se agrega un vértice nuevo, el algoritmo termina. Si al terminar un vértice v alcanzable desde s no perteneciera a U, entonces un dicamino de s a v tendría un primer arco que sale de U, contradiciendo que $N^+(U)$ es vacío. Por lo tanto, todos los vértices alcanzables desde s quedan resueltos. □

5.16. Coloreo

Pasemos ahora a una pregunta distinta. Supongamos que queremos asignar recursos a objetos que tienen conflictos entre sí. Por ejemplo, queremos asignar horarios a evaluaciones, frecuencias a antenas, registros a variables, o hábitats a animales que no pueden convivir.

Una forma natural de modelar esto es construir un grafo: los vértices son los objetos, y ponemos una arista entre dos vértices si esos objetos no pueden recibir el mismo recurso. Entonces el problema se transforma en colorear los vértices de modo que vértices adyacentes tengan colores distintos.

Definición. Sea $G = (V, E)$ un grafo simple. Un *coloreo válido* de G es una asignación de colores a los vértices de G tal que, si $uv \in E$, entonces u y v reciben colores distintos. Decimos que G es *k-coloreable* si tiene un coloreo válido usando a lo más k colores.

Definición. El *número cromático* de un grafo simple G, denotado $\chi(G)$, es el menor número de colores necesarios para colorear propiamente G.

Ejemplo. Todo grafo con n vértices puede colorearse con n colores: basta asignar un color distinto a cada vértice. La pregunta interesante es cuántos colores podemos ahorrar.

Por ejemplo, un grafo sin aristas tiene número cromático 1. En cambio, en el grafo completo K_n, todos los pares de vértices son adyacentes, así que no se puede reutilizar ningún color.

Proposición. Para todo $n \geq 1$, $\chi(K_n) = n$.

Demostración. Como todos los pares de vértices de K_n son adyacentes, dos vértices distintos no pueden recibir el mismo color. Por lo tanto, todo coloreo válido usa al menos n colores. Por otra parte, usando un color distinto para cada vértice obtenemos un coloreo válido con n colores. Luego $\chi(K_n) = n$. □

Proposición. Para todo $n \geq 3$,

$$\chi(C_n) = \begin{cases} 2 & \text{si } n \text{ es par,} \\ 3 & \text{si } n \text{ es impar.} \end{cases}$$

Demostración. Si n es par, podemos colorear los vértices de C_n alternando dos colores alrededor del ciclo. Como el número de vértices es par, al volver al inicio no aparece conflicto. Entonces $\chi(C_n) \leq 2$. Como C_n tiene al menos una arista, no puede colorearse con un solo color, así que $\chi(C_n) = 2$.

Supongamos ahora que n es impar. Si intentamos colorear C_n con dos colores, una vez que fijamos el color de un vértice, los colores de todos los demás quedan forzados al alternar alrededor del ciclo. Como el ciclo tiene largo impar, el último vértice queda con el mismo color que el primero, pero ambos son adyacentes. Por lo tanto, no existe un coloreo válido con dos colores. Así, $\chi(C_n) \geq 3$. Por otra parte, podemos colorear un camino con $n - 1$ vértices alternando dos colores y usar un tercer color para el último vértice del ciclo. Entonces $\chi(C_n) = 3$. □

Ejemplo. El *grafo rueda* W_n se obtiene tomando un ciclo con $n-1$ vértices y agregando un vértice central adyacente a todos los vértices del ciclo. Entonces el vértice central necesita un color distinto de los colores usados en el ciclo. Por lo tanto,

$$\chi(W_n) = \begin{cases} 3 & \text{si } n \text{ es impar,} \\ 4 & \text{si } n \text{ es par,} \end{cases}$$

para $n \geq 4$. En efecto, si n es impar, el ciclo exterior tiene largo par y usa dos colores; si n es par, el ciclo exterior tiene largo impar y necesita tres colores.

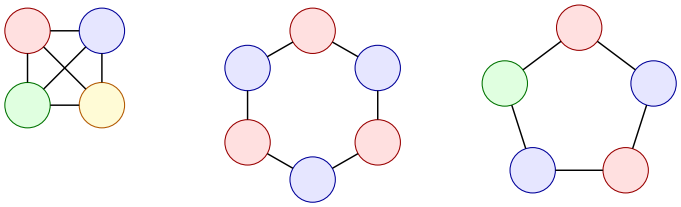


Figura 5.2: Tres comportamientos básicos del coloreo. De izquierda a derecha: en K_4 todos los vértices son adyacentes entre sí; el ciclo par C_6 admite un coloreo alternando dos colores; el ciclo impar C_5 necesita un tercer color.

Una aplicación típica es el problema de asignar recursos incompatibles. Si dos objetos no pueden compartir recurso, los conectamos por una arista. Entonces cada color representa un recurso. El número cromático es exactamente la mínima cantidad de recursos necesarios.

Por ejemplo, si los vértices son animales y una arista indica que dos animales no pueden estar en el mismo hábitat, entonces $\chi(G)$ es el número mínimo de hábitats necesarios. Si los vértices son evaluaciones y una arista indica que hay estudiantes en común, entonces $\chi(G)$ es el número mínimo de bloques horarios necesarios.

5.17. Planaridad

Los mapas entregan otra motivación natural para colorear grafos. Si queremos colorear regiones de un mapa de modo que regiones vecinas reciban colores distintos, podemos construir un grafo: cada región es un vértice, y unimos dos vértices si las regiones correspondientes comparten frontera.

Los grafos que aparecen así tienen una propiedad geométrica especial: se pueden dibujar en el plano sin que las aristas se crucen.

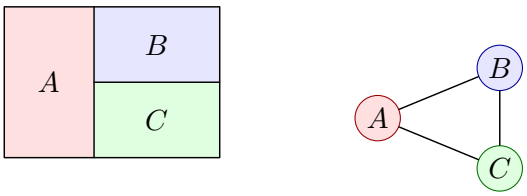


Figura 5.3: De un mapa a un grafo. La figura de la izquierda muestra tres regiones; la figura de la derecha representa las regiones como vértices, uniendo dos vértices cuando las regiones correspondientes comparten frontera.

Definición. Un grafo simple G es *planar* si se puede dibujar en el plano de modo que sus aristas no se crucen, salvo en vértices comunes. A un dibujo con esta propiedad lo llamamos una *representación planar* de G.

Definición. Una representación planar de un grafo conexo divide el plano en *regiones*, también llamadas caras. La región no acotada, que queda "afuera" del dibujo, también cuenta como región.

Para entender qué se puede decir sobre coloreos de mapas, primero conviene estudiar qué restricciones combinatoriales impone la planaridad. La herramienta básica es la fórmula de Euler.

Teorema (Fórmula de Euler)**.** Sea $G = (V, E)$ un grafo simple, conexo y planar. Si r es el número de regiones de una representación planar de G, entonces $|V| - |E| + r = 2$. Equivalentemente, $r = |E| - |V| + 2$.

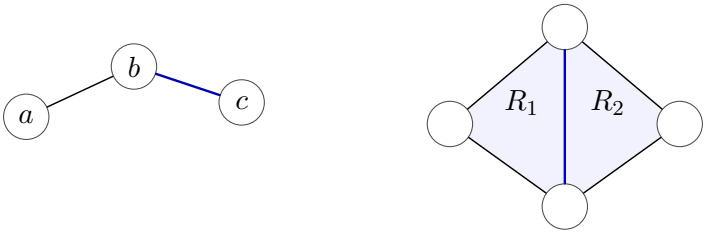


Figura 5.4: Los dos pasos del argumento de Euler. A la izquierda, al agregar una arista que trae un vértice nuevo, $|V|$ y $|E|$ aumentan en 1 y el número de regiones no cambia. A la derecha, al agregar una arista entre vértices ya presentes, $|E|$ y r aumentan en 1. En ambos casos $|V| - |E| + r$ se conserva.

Demostración. Fijemos una representación planar de G. Construiremos el grafo desde un vértice inicial, agregando aristas una por una, siempre manteniendo un subgrafo conexo de G. Esto se puede hacer porque G es conexo: mientras falten aristas o vértices, podemos agregar una arista incidente al subgrafo que ya tenemos.

Al comienzo tenemos un solo vértice, ninguna arista y una sola región. Entonces $|V| - |E| + r = 1 - 0 + 1 = 2$.

Veamos qué ocurre al agregar una arista. Hay dos casos.

Primero, la arista puede traer un vértice nuevo. En ese caso, el número de vértices aumenta en 1, el número de aristas aumenta en 1, y el número de regiones no cambia. Por lo tanto, el valor de $|V| - |E| + r$ permanece igual.

Segundo, la arista puede conectar dos vértices que ya estaban en el subgrafo. Como estamos usando la representación planar fijada, esta arista se dibuja dentro de una región y la divide en dos. En ese caso, el número de aristas aumenta en 1, el número de regiones aumenta en 1, y el número de vértices no cambia. Nuevamente, el valor de $|V| - |E| + r$ permanece igual.

Como al final obtenemos todo G, concluimos que para G también se cumple $|V| - |E| + r = 2$. □

La fórmula de Euler es muy útil porque convierte una propiedad geométrica en una igualdad combinatorial. A partir de ella podemos obtener restricciones sobre cuántas aristas puede tener un grafo planar. Para eso usamos una versión del apretón de manos, pero ahora contando incidencias entre aristas y regiones.

Lema (Apretón de manos para regiones)**.** Sea G un grafo simple, conexo y planar con una representación planar fija. Si $\ell(R)$ denota el largo de frontera de una región R, contando una arista dos veces si aparece dos veces en la frontera de la misma región, entonces $\sum_R \ell(R) = 2|E|$.

Demostración. Cada arista tiene dos lados. Al recorrer los bordes de todas las regiones, cada lado de cada arista aparece exactamente una vez. Por lo tanto, cada arista contribuye 2 a la suma total de largos de bordes. □

Corolario. Si G es un grafo simple, conexo y planar con $n \geq 3$ vértices y m aristas, entonces $m \leq 3n - 6$.

Demostración. En un grafo simple, toda región tiene borde de largo al menos 3. Si r es el número de regiones, entonces $3r \leq 2m$. Por la fórmula de Euler, $r = m - n + 2$. Reemplazando obtenemos $3(m - n + 2) \leq 2m$, y al reordenar queda $m \leq 3n - 6$. □

Ejemplo. El grafo K_5 no es planar. En efecto, tiene $n = 5$ vértices y $m = 10$ aristas. Si fuera planar, debería cumplir $10 \leq 3 \cdot 5 - 6 = 9$, lo que es imposible.

Corolario. Si G es un grafo simple, conexo, planar y bipartito con $n \geq 3$ vértices y m aristas, entonces $m \leq 2n - 4$.

Demostración. En un grafo bipartito no hay ciclos de largo impar, y en particular no hay triángulos. Por lo tanto, en una representación planar cada región tiene borde de largo al menos 4. Usando el apretón de manos para regiones, $4r \leq 2m$. Por Euler, $r = m - n + 2$. Luego $4(m - n + 2) \leq 2m$, y reordenando obtenemos $m \leq 2n - 4$. □

Ejemplo. El grafo $K_{3,3}$ no es planar. Es bipartito, tiene $n = 6$ vértices y $m = 9$ aristas. Si fuera planar, por la cota anterior tendría que cumplirse $9 \leq 2 \cdot 6 - 4 = 8$, contradicción.

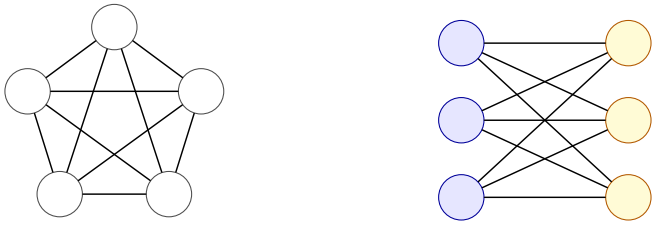


Figura 5.5: Los dos ejemplos clásicos de grafos no planares que detectan las cotas de aristas. A la izquierda, K_5 tiene $n = 5$ y $m = 10$, contradiciendo $m \leq 3n - 6$. A la derecha, $K_{3,3}$ es bipartito, tiene $n = 6$ y $m = 9$, contradiciendo $m \leq 2n - 4$.

Las cotas anteriores tienen una consecuencia muy importante para el coloreo: todo grafo planar tiene un vértice de grado pequeño. Esto sugiere una estrategia inductiva para colorear grafos planares con pocos colores: borrar un vértice de grado pequeño, colorear el resto, y luego volver a insertar el vértice borrado.

Lema. Todo grafo simple planar tiene un vértice de grado a lo más 5.

Demostración. Podemos asumir que el grafo tiene al menos tres vértices, pues si tiene uno o dos vértices el resultado es inmediato. También podemos asumir que es conexo: si no lo es, basta aplicar el argumento a una componente conexa.

Sea $n = |V|$ y $m = |E|$. Como el grafo es simple, conexo y planar, tenemos $m \leq 3n - 6$. Por el apretón de manos, $\sum_{v \in V} d(v) = 2m \leq 6n - 12$. Si todos los vértices tuvieran grado al menos 6, entonces $\sum_{v \in V} d(v) \geq 6n$, lo que contradice la desigualdad anterior. Por lo tanto, existe un vértice de grado a lo más 5. □

Teorema (Teorema de los seis colores)**.** Si G es un grafo simple planar, entonces $\chi(G) \leq 6$.

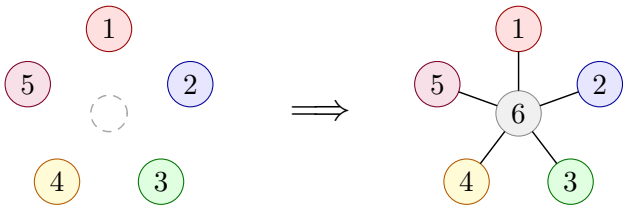


Figura 5.6: Paso inductivo del teorema de los seis colores. Después de colorear $G - v$, el vértice borrado tiene a lo más cinco vecinos, así que entre seis colores siempre queda un color disponible para reinsertarlo.

Demostración. Demostraremos el resultado por inducción en el número de vértices. Si G tiene un solo vértice, el resultado es claro.

Supongamos que todo grafo simple planar con menos de n vértices es 6-coloreable, y sea G un grafo simple planar con n vértices. Por el lema anterior, G tiene un vértice v de grado a lo más 5. Consideremos el grafo $G - v$ que se obtiene eliminando v y todas sus aristas incidentes. Este grafo sigue siendo simple y planar, y tiene menos vértices que G. Por la hipótesis inductiva, $G - v$ admite un coloreo válido con a lo más 6 colores.

Ahora volvamos a insertar v. El vértice v tiene a lo más 5 vecinos, así que entre los 6 colores disponibles hay al menos un color que no aparece en sus vecinos. Asignamos ese color a v. El coloreo sigue siendo válido, pues el único vértice nuevo es v y no comparte color con ninguno de sus vecinos. Por lo tanto, G es 6-coloreable. □

Ejercicios

1. Un *fullereno* es un grafo simple, conexo y planar en el que todo vértice tiene grado 3, y toda región tiene borde de largo 5 o 6.

   a) Sea n el número de vértices y m el número de aristas. Use el apretón de manos para demostrar que $2m = 3n$.

   b) Sea p el número de regiones de largo 5 y sea h el número de regiones de largo 6. Use el apretón de manos para regiones para demostrar que $5p + 6h = 2m$.

c) Use la fórmula de Euler para concluir que $p = 12$.

d) Interprete el resultado en el caso de un balón de fútbol idealizado, formado por pentágonos y hexágonos.

2. En los siguientes problemas, describa una reducción a una instancia de caminos más cortos. En cada caso, especifique el grafo auxiliar, los pesos de sus arcos, qué camino más corto hay que calcular, y cómo se recupera una solución del problema original.

   a) **Máxima probabilidad de supervivencia.** Una red está modelada por un digrafo $G = (V, E)$. Cada arco e tiene una probabilidad de supervivencia $p(e) \in (0, 1)$, y la probabilidad de supervivencia de un dicamino P es

   $$\prod_{e \in P} p(e).$$

   Dados dos vértices $s, t \in V$, queremos encontrar un dicamino de s a t que maximice esa probabilidad. Reduzca el problema a uno de camino más corto usando pesos de la forma $-\log p(e)$, y pruebe que la reducción preserva las soluciones óptimas.

   b) **Costos que dependen del tiempo.** Una persona parte en s en el instante 0 y quiere llegar a t a más tardar en el instante T. El tiempo es discreto. Al cruzar un arco $e = (u, v)$ en el instante τ, se llega a v en el instante $\tau + 1$ y se paga un costo positivo $c_e(\tau)$, que puede depender de τ. También está permitido esperar un instante en cualquier vértice pagando costo 0. Construya un grafo expandido en el tiempo, con vértices de la forma (v, τ), que reduzca el problema a encontrar un camino más corto desde $(s, 0)$ hasta algún vértice (t, τ) con $0 \leq \tau \leq T$. Si quiere tener un único destino, agregue un vértice terminal $t^\star$ y arcos de costo 0 desde cada (t, τ) hacia $t^\star$. Explique por qué los caminos en el grafo expandido corresponden exactamente a recorridos factibles en la red original.

   c) **Cupones de descuento.** En un digrafo con costos positivos $w(e)$, se quiere ir de s a t. Se tienen a lo más k cupones, y cada cupón puede usarse al cruzar un arco para pagar costo 0 por ese arco. Construya un grafo por capas, donde la capa registra cuántos cupones se han usado, que reduzca el problema a un camino más corto. Indique cuáles son los arcos que se quedan en la misma capa y cuáles avanzan a la capa siguiente. El camino buscado debe ir desde $(s, 0)$ hasta alguna copia (t, j) con $0 \leq j \leq k$.

3. En este ejercicio dirigido demostraremos el *teorema de los cinco colores*: todo grafo simple planar es 5-coloreable.

   a) Trataremos de seguir la estrategia inductiva del teorema de los seis colores, pero ahora con solo cinco colores disponibles. Use el lema de grado pequeño para elegir un vértice v de grado a lo más 5, y explique por qué el caso $d(v) \leq 4$ se resuelve inmediatamente después de colorear $G - v$ con cinco colores.

   b) Suponga entonces que $d(v) = 5$. Fije una representación planar de G y escriba los vecinos de v en orden cíclico alrededor de v como v_1, v_2, v_3, v_4, v_5. Explique por qué, si dos vecinos de v reciben el mismo color en un 5-coloreo de $G - v$, entonces se puede colorear v.

c) Queda el caso en que los vecinos de v usan cinco colores distintos. Renombre los colores para que v_i tenga color i. Para dos colores i, j, sea H_{ij} el subgrafo de $G - v$ inducido por los vértices cuyos colores son i o j. Demuestre que intercambiar los colores i y j dentro de una componente conexa de H_{ij} mantiene un coloreo válido.

d) Demuestre que si v_1 y v_3 están en componentes distintas de H_{13}, entonces se puede intercambiar los colores 1 y 3 en la componente de v_1 y después colorear v con el color 1. Formule el argumento análogo para v_2 y v_4 usando H_{24}.

e) Suponga, para obtener una contradicción, que v_1 y v_3 están en la misma componente de H_{13}, y que v_2 y v_4 están en la misma componente de H_{24}. Tome un camino P_{13} de v_1 a v_3 en H_{13}. Explique por qué P_{13} junto con las aristas vv_1 y vv_3 forma una curva cerrada que separa a v_2 de v_4 en la representación planar.

f) Use la separación anterior para justificar que no puede existir simultáneamente un camino de v_2 a v_4 en H_{24}. Concluya que al menos uno de los dos intercambios de colores es posible y complete la demostración inductiva.

5.18. Árboles

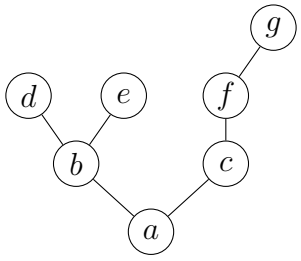
Ya hemos visto varias familias de grafos: ciclos, cliques, grafos bipartitos, grafos planares. En esta clase estudiaremos una de las familias más importantes: los árboles.

Los árboles aparecen todo el tiempo en computación. Por ejemplo, el código Morse se puede leer siguiendo decisiones sucesivas: punto o raya. Un sistema de archivos se organiza jerárquicamente desde un directorio raíz. Un árbol de búsqueda binaria permite ordenar datos para buscarlos eficientemente. Un torneo deportivo de eliminación directa también tiene una estructura de árbol.

La idea común es que hay una forma de conexión sin redundancia. Podemos llegar de un lugar a otro, pero no hay ciclos que nos permitan dar vueltas.

> **Definición.** Un grafo simple no dirigido T se llama *árbol* si es conexo y no tiene ciclos. Un grafo simple no dirigido F se llama *bosque* si cada una de sus componentes conexas es un árbol.

Ejemplo. El siguiente grafo es un árbol.



Es conexo, y no hay manera de volver a un vértice inicial sin repetir alguna arista.

En cambio, si agregamos la arista ef, aparece un ciclo. Si borramos la arista ac, el grafo deja de ser conexo. En ambos casos dejamos de tener un árbol.

La definición de árbol es corta, pero es sorprendentemente rígida. Conectividad y ausencia de ciclos se fuerzan mutuamente de varias maneras. El resto de esta sección consiste en hacer explícitas esas formas equivalentes de mirar la misma estructura.

Lema. Sea T un árbol. Entonces entre cada par de vértices distintos de T existe un único camino simple.

Demostración. Como T es conexo, existe al menos un camino simple entre cada par de vértices distintos. Debemos probar que no puede haber dos.

Supongamos, buscando una contradicción, que existen dos caminos simples distintos P y Q entre dos vértices u y v. Recorremos ambos caminos desde u hasta que se separan por primera vez. Luego, como ambos terminan en v, eventualmente se vuelven a encontrar. Si llamamos x al primer vértice donde se separan y y al primer vértice posterior donde se reencuentran, entonces el tramo de P entre x e y junto con el tramo de Q entre x e y forman un ciclo.

Esto contradice que T no tiene ciclos. Por lo tanto, el camino simple entre u y v es único. □

Lema. Todo árbol con al menos dos vértices tiene una hoja, es decir, un vértice de grado 1.

Demostración. Tomemos un camino simple de largo máximo en T, digamos

$$v_0, v_1, \ldots, v_k.$$

Como T tiene al menos dos vértices, tenemos $k \geq 1$.

Afirmamos que v_0 es una hoja. En efecto, si v_0 tuviera un vecino w distinto de v_1, entonces w no puede aparecer en el camino, pues si apareciera se formaría un ciclo. Pero entonces

$$w, v_0, v_1, \ldots, v_k$$

sería un camino simple más largo, contradiciendo la elección del camino original. Por lo tanto, v_0 tiene grado 1. □

Teorema. Sea $T = (V, E)$ un grafo simple no dirigido con $|V| = n$. Las siguientes propiedades son equivalentes.

1. T es un árbol.
2. Entre cada par de vértices distintos existe un único camino simple.
3. T es conexo y $|E| = n - 1$.
4. T es conexo, y al borrar cualquier arista de T el grafo se vuelve no conexo.
5. T es acíclico y $|E| = n - 1$.
6. T es acíclico, y al agregar cualquier arista entre dos vértices distintos de T se forma un ciclo.

Demostración. Probemos las implicancias principales.

Ya vimos que si T es un árbol, entonces existe un único camino simple entre cada par de vértices distintos.

Supongamos ahora que existe un único camino simple entre cada par de vértices distintos. Entonces T es conexo. Además, si T tuviera un ciclo, dos vértices consecutivos del ciclo estarían

conectados por dos caminos simples distintos: la arista entre ellos, y el resto del ciclo. Esto contradice la unicidad. Por lo tanto, T es un árbol.

Probemos ahora que todo árbol con n vértices tiene $n-1$ aristas. Lo hacemos por inducción en n. Para $n = 1$ no hay aristas, así que la fórmula vale. Para $n \geq 2$, por el lema anterior existe una hoja v. Al borrar v y su única arista incidente, el grafo que queda sigue siendo un árbol con $n-1$ vértices. Por hipótesis inductiva tiene $n-2$ aristas. Al volver a agregar v, agregamos exactamente una arista, así que T tiene $n-1$ aristas.

Recíprocamente, supongamos que T es conexo y tiene $n-1$ aristas. Si T tuviera un ciclo, podríamos borrar una arista de ese ciclo y el grafo seguiría siendo conexo. Repitiendo este procedimiento hasta eliminar todos los ciclos, obtendríamos un árbol generador sobre los mismos n vértices con estrictamente menos de $n-1$ aristas. Pero acabamos de probar que todo árbol con n vértices tiene $n-1$ aristas. Esto es imposible. Luego T no tiene ciclos y, como era conexo, es un árbol.

Finalmente, si T es un árbol y borramos una arista xy, el único camino entre x e y desaparece. Por lo tanto, el grafo se vuelve no conexo. Recíprocamente, si T es conexo y al borrar cualquier arista se desconecta, entonces T no puede tener ciclos: una arista de un ciclo puede borrarse sin perder conectividad. Así, T es conexo y sin ciclos, es decir, un árbol. □

Corolario. Si F es un bosque con n vértices y c componentes conexas, entonces

$$|E(F)| = n - c.$$

Demostración. Cada componente conexa de F es un árbol. Si sus tamaños son $n_1, \ldots, n_c$, entonces la componente i tiene $n_i - 1$ aristas. Por lo tanto,

$$|E(F)| = \sum_{i=1}^{c} (n_i - 1) = \sum_{i=1}^{c} n_i - c = n - c.$$ □

5.19. Árboles con raíz

Muchas aplicaciones de árboles no sólo usan conexiones, sino también jerarquías. En un sistema de archivos hay un directorio raíz. En un árbol de búsqueda binaria hay un primer nodo desde el cual comenzamos la búsqueda. En un árbol de decisiones hay una primera pregunta.

Para capturar esta información elegimos un vértice especial, la raíz, y miramos todo el árbol desde ahí. La unicidad de caminos nos permite orientar todas las aristas de manera natural: alejándonos de la raíz.

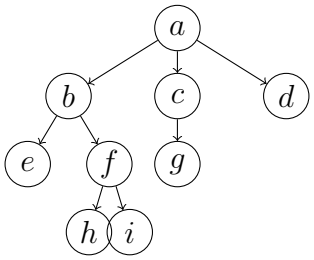
Definición. Un *árbol con raíz* es un árbol T junto con un vértice distinguido r, llamado la *raíz*. Si $v \neq r$, el *padre* de v es el vértice anterior a v en el único camino simple desde r hasta v. En ese caso, decimos que v es un *hijo* de su padre.
Dos vértices son *hermanos* si tienen el mismo padre.

Definición. Sea T un árbol con raíz r.

- Los *ancestros* de un vértice v son los vértices que aparecen en el camino desde r hasta v.

- Los *descendientes* de v son los vértices que tienen a v como ancestro.
- Una *hoja* es un vértice sin hijos.
- Un *vértice interno* es un vértice con al menos un hijo.
- La *altura* del árbol es el largo máximo de un camino desde la raíz hasta una hoja.
- La *aridad* del árbol es el número máximo de hijos de un vértice.

Ejemplo. Consideremos el árbol con raíz a.



El padre de f es b. Los vértices b, c, d son hermanos. Las hojas son d, e, g, h, i. La altura es 3 y la aridad es 3.

Definición. Sea $m \geq 1$. Un árbol con raíz es *m-ario* si cada vértice interno tiene a lo más m hijos. Es *m-ario completo* si cada vértice interno tiene exactamente m hijos.

Proposición. Si T es un árbol m-ario de altura h, entonces T tiene a lo más m^h hojas.

Demostración. Imaginemos que completamos el árbol hasta altura h agregando hijos ficticios cuando haga falta, de manera que todo vértice interno tenga exactamente m hijos y todas las hojas queden en el nivel h. Este procedimiento sólo puede aumentar el número de hojas.

En un árbol m-ario completo de altura h, el nivel 0 tiene 1 vértice, el nivel 1 tiene m vértices, el nivel 2 tiene m^2 vértices, y así sucesivamente. Por inducción, el nivel h tiene m^h vértices.

Por lo tanto, el árbol original tenía a lo más m^h hojas. □

Proposición. Si T es un árbol m-ario completo con i vértices internos, entonces T tiene

$$mi + 1$$

vértices en total.

Demostración. Como T es un árbol, si tiene n vértices entonces tiene $n - 1$ aristas. Por otro lado, cada arista conecta a un padre con un hijo. Como cada vértice interno tiene exactamente m hijos, el número de aristas también es mi. Así,

$$n - 1 = mi.$$

Por lo tanto, $n = mi + 1$. □

5.20. Árboles generadores

Volvamos ahora a grafos generales. Supongamos que tenemos varias ciudades y algunos caminos posibles entre ellas. Queremos pavimentar suficientes caminos para poder ir de cualquier ciudad a cualquier otra, pero no queremos pavimentar caminos innecesarios.

Un árbol es exactamente una red conexa sin redundancia. Por eso, si queremos conectar todos los vértices de un grafo usando la menor cantidad posible de aristas, buscamos un árbol dentro del grafo que use todos los vértices.

Definición. Sea G un grafo simple no dirigido. Un *árbol generador* de G es un subgrafo T de G tal que:

- T contiene todos los vértices de G, y
- T es un árbol.

Ejemplo. El ciclo C_n tiene muchos árboles generadores. En efecto, basta borrar una arista cualquiera del ciclo. El resultado es un camino con n vértices, que es un árbol generador de C_n.

Teorema. Un grafo simple no dirigido G tiene un árbol generador si y sólo si G es conexo.

Demostración. Si G tiene un árbol generador T, entonces entre cada par de vértices existe un camino en T. Como T es subgrafo de G, ese camino también existe en G. Luego G es conexo.

Recíprocamente, supongamos que G es conexo. Si G no tiene ciclos, entonces G mismo es un árbol y terminamos. Si G tiene un ciclo, borramos una arista de ese ciclo. El grafo sigue siendo conexo, porque los extremos de la arista borrada siguen conectados por el resto del ciclo.

Repetimos este procedimiento mientras existan ciclos. Como el grafo tiene un número finito de aristas, el proceso termina. Al final obtenemos un subgrafo conexo, sin ciclos, y con todos los vértices de G. Por lo tanto, es un árbol generador de G. □

La demostración anterior no sólo prueba existencia. También sugiere un método: borrar aristas que sobran hasta destruir todos los ciclos. Sin embargo, encontrar ciclos y decidir qué borrar no siempre es la forma más natural de construir un árbol generador.

Hay dos procedimientos básicos que aparecen en muchos algoritmos: búsqueda en profundidad y búsqueda en anchura. Ambos recorren el grafo desde un vértice inicial y guardan las aristas que descubren vértices nuevos.

Definición. Sea G un grafo conexo y sea r un vértice inicial. Un *árbol de búsqueda* desde r es un árbol generador con raíz r que se obtiene agregando, para cada vértice $v \neq r$, una arista que conecta a v con un vértice desde el cual fue descubierto.

En *búsqueda en profundidad* (DFS) intentamos avanzar lo más posible por un camino antes de retroceder. Cuando llegamos a un vértice sin vecinos nuevos, volvemos hacia atrás hasta encontrar una nueva opción.

En *búsqueda en anchura* (BFS) exploramos por niveles. Primero visitamos todos los vecinos de la raíz, luego todos los vértices a distancia dos, luego todos los vértices a distancia tres, y así sucesivamente.

Proposición. Si G es conexo, tanto DFS como BFS construyen un árbol generador de G.

Demostración. En ambos procedimientos, cada vez que se descubre un vértice nuevo $v \neq r$, se guarda exactamente una arista que conecta a v con un vértice ya descubierto. Por lo tanto, si al final se descubren todos los vértices, las aristas guardadas conectan todos los vértices con la raíz.

Además, no se puede crear un ciclo al agregar una arista hacia un vértice nuevo: antes de agregar esa arista, el vértice nuevo no estaba conectado al árbol construido. Así, las aristas guardadas forman un grafo conexo y sin ciclos.

Como G es conexo, desde la raíz se puede llegar a todos los vértices, así que el procedimiento eventualmente descubre todo vértice. Por lo tanto, el resultado es un árbol generador. □

Ejemplo. La diferencia entre DFS y BFS no está en si producen o no un árbol generador, sino en cuál árbol producen. En una rueda W_n, si comenzamos en el centro, BFS tiende a producir una estrella: descubre todos los vértices exteriores inmediatamente. En cambio, DFS puede avanzar por el ciclo exterior antes de retroceder. Ambos árboles usan todos los vértices y tienen $n - 1$ aristas, pero capturan estructuras distintas del mismo grafo.

5.21. Árboles generadores de peso mínimo

Muchas veces las aristas tienen costos. En una red de ciudades, una arista puede representar el costo de construir una carretera. En una red de computadores, puede representar el costo de instalar fibra. En diseño de circuitos, puede representar el largo de una conexión.

Entonces no basta con encontrar cualquier árbol generador. Queremos uno de costo mínimo.

Definición. Un *grafo con pesos* es un grafo $G = (V, E)$ junto con una función

$$w : E \to \mathbb{R}.$$

Si T es un subgrafo de G, su *peso* es

$$w(T) = \sum_{e \in E(T)} w(e).$$

Un *árbol generador de peso mínimo* de G es un árbol generador T de G con peso mínimo entre todos los árboles generadores de G.

La propiedad estructural que explica los algoritmos clásicos es la siguiente. Si partimos los vértices en dos lados, cualquier árbol generador debe usar alguna arista que cruce de un lado al otro. Entonces, una arista muy barata que cruza ese corte es una candidata segura. Esta es la versión para árboles generadores de peso mínimo de la idea que uno usa informalmente en Prim y Kruskal.

Definición. Sea $G = (V, E)$ un grafo y sea $S \subseteq V$ con $S \neq \varnothing$ y $S \neq V$. El *corte* definido por S es el conjunto de aristas con un extremo en S y el otro en $V \setminus S$.

Lema (Propiedad de corte)**.** Sea G un grafo conexo con pesos y sea $S \subseteq V(G)$ un conjunto no vacío y propio. Sea e una arista de peso mínimo entre las aristas que cruzan el corte definido por S. Entonces existe un árbol generador de peso mínimo que contiene a e.
Más aún, si e es la única arista de peso mínimo que cruza ese corte, entonces todo árbol generador de peso mínimo contiene a e.

Demostración. Sea T un árbol generador de peso mínimo. Si $e \in E(T)$, no hay nada que probar.

Supongamos entonces que $e \notin E(T)$. Al agregar e a T, aparece un único ciclo. Como e cruza el corte, para volver al lado inicial el ciclo debe usar alguna otra arista f que también cruza el mismo corte. Por la elección de e, tenemos $w(e) \leq w(f)$.

Consideremos

$$T' = T + e - f.$$

Al borrar una arista de ese ciclo, el grafo vuelve a ser un árbol generador. Además,

$$w(T') = w(T) + w(e) - w(f) \leq w(T).$$

Como T ya era de peso mínimo, T' también es de peso mínimo y contiene a e.

Si e es la única arista más barata del corte, entonces $w(e) < w(f)$. En ese caso, ningún árbol generador de peso mínimo puede omitir a e, pues el intercambio anterior produciría un árbol de peso estrictamente menor. □

Esta propiedad permite entender los dos algoritmos clásicos sin entrar en detalles de implementación. Prim hace crecer un solo árbol. Kruskal hace crecer un bosque. En ambos casos, cada arista que se agrega es segura porque es la más barata para conectar piezas que todavía están separadas.

Definición (Algoritmo de Prim)**.** Partimos desde un vértice cualquiera y mantenemos un conjunto U de vértices ya conectados. En cada paso agregamos una arista de menor peso que tenga un extremo en U y el otro en $V \setminus U$. Repetimos hasta que $U = V$.

Definición (Algoritmo de Kruskal)**.** Ordenamos las aristas de menor a mayor peso y partimos con un bosque vacío. Recorremos las aristas en ese orden. Agregamos una arista si conecta dos componentes distintas del bosque actual, y la descartamos si crearía un ciclo. Repetimos hasta tener un árbol generador.

Teorema. Sea G un grafo conexo con pesos. Los algoritmos de Prim y Kruskal construyen árboles generadores de peso mínimo de G.

Demostración. En ambos casos usaremos la misma invariante: después de cada paso, las aristas elegidas están contenidas en algún árbol generador de peso mínimo.

Consideremos primero Prim. Supongamos que las aristas ya elegidas forman un árbol sobre un conjunto de vértices U, y que están contenidas en algún árbol generador mínimo T. Sea e la arista elegida por Prim, es decir, una arista de peso mínimo que cruza el corte entre U y $V \setminus U$. Si $e \in E(T)$, la invariante sigue valiendo. Si $e \notin E(T)$, entonces $T + e$ contiene un único ciclo. Ese ciclo debe usar otra arista f que también cruza el corte entre U y $V \setminus U$. Como e era una arista de peso mínimo en ese corte, $w(e) \leq w(f)$. Luego

$$T' = T + e - f$$

es un árbol generador de peso a lo más $w(T)$, y por lo tanto también es mínimo. Además contiene todas las aristas ya elegidas y también e. Así, la invariante se mantiene. Al terminar, Prim ha construido un árbol generador contenido en un árbol generador mínimo; por lo tanto, es mínimo.

Consideremos ahora Kruskal. Supongamos que el bosque actual F está contenido en algún árbol generador mínimo T, y que Kruskal agrega una arista $e = uv$ que conecta dos componentes distintas de F. Si $e \in E(T)$, la invariante sigue valiendo. Si $e \notin E(T)$, entonces $T + e$ contiene un único ciclo. En el camino de T entre u y v debe existir una arista f que conecta dos componentes distintas del bosque F; de lo contrario, u y v ya estarían conectados en F.

Afirmamos que $w(e) \leq w(f)$. En efecto, si $w(f) < w(e)$, entonces Kruskal habría considerado f antes que e. En ese momento los extremos de f todavía estaban en componentes distintas, pues las componentes sólo se fusionan a medida que avanza el algoritmo. Por lo tanto, Kruskal habría agregado f, contradiciendo que f conecta dos componentes distintas de F justo antes de agregar e.

Entonces

$$T' = T + e - f$$

es un árbol generador de peso a lo más $w(T)$. Por minimalidad, también es mínimo, y contiene $F \cup \{e\}$. La invariante se mantiene. Al final, Kruskal obtiene un bosque conexo con $n - 1$ aristas; es decir, un árbol generador. Por la invariante, este árbol es de peso mínimo. □

Ejercicios

1. Considere la siguiente representación de directorios en Linux.

```
/
|-  bin
|-  etc
|   |-    nginx
|   ‘-    ssh
|-  home
|   |-    ana
|   |     |-     docs
|   |     ‘-     img
|   ‘-    beto
|         ‘-     src
‘-  usr
    |-    bin
    ‘-    lib
```

   Modelamos esto con un grafo que tiene un vértice por directorio y una arista entre cada directorio y su padre.

   a) Explique por qué este grafo debe ser un árbol con raíz `/`.

   b) Determine sus hojas, sus vértices internos, su altura y su aridad.

   c) Dibuje el árbol con raíz `/` y marque el padre de `/home/ana/img`, sus hermanos y sus ancestros.

2. Sea T un árbol con raíz r. Para cada vértice v, definimos su *nivel* como la distancia desde r hasta v.

   a) Demuestre que si uv es una arista de T, entonces los niveles de u y v difieren exactamente en 1.

   b) Concluya que si A es el conjunto de vértices de nivel par y B es el conjunto de vértices de nivel impar, entonces toda arista de T tiene un extremo en A y el otro en B.

   c) Demuestre que esta partición es única salvo intercambiar A y B.

3. Sea T un árbol no dirigido. Demuestre que cualesquiera dos caminos simples de largo máximo en T tienen al menos un vértice en común.

   Indicación: si existieran dos caminos máximos disjuntos, considere el único camino simple que une un vértice de uno con un vértice del otro.

4. Sea G un grafo conexo y sea T un árbol generador de G.

   a) Demuestre que si e es una arista de G que no pertenece a T, entonces $T + e$ contiene exactamente un ciclo.

   b) Demuestre que si e es una arista de T, entonces $T - e$ tiene exactamente dos componentes conexas.

   c) Use las partes anteriores para explicar por qué un árbol generador representa una forma de conectar todos los vértices sin redundancia.

5. Sea G un grafo simple conexo. Diremos que una arista e de G es un *puente* si $G - e$ no es conexo. Demuestre que una arista e pertenece a todo árbol generador de G si y sólo si e es un puente.

   Luego concluya que G tiene un único árbol generador si y sólo si G es un árbol.

6. Sea T un árbol con al menos dos vértices. Definimos una operación que borra simultáneamente todas las hojas de T. Si queda un grafo con al menos dos vértices, repetimos la operación.

   a) Demuestre que después de cada paso, si queda algún vértice, el grafo restante sigue siendo un árbol.

   b) Demuestre que el proceso termina con un único vértice o con una única arista.

   c) Interprete el resultado como una forma de encontrar el "centro" de un árbol.

5.22. Árboles

Ya hemos visto varias familias de grafos: cliques, ciclos, grafos bipartitos y grafos planares. En esta clase estudiaremos una de las familias más importantes: los *árboles*.

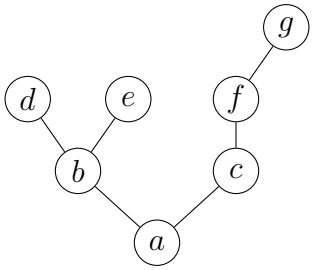
Los árboles aparecen en muchos lugares de computación. Un código Morse se puede leer como una secuencia de decisiones: punto o raya. Un sistema de archivos se organiza desde un directorio raíz. Un árbol de búsqueda binaria permite buscar elementos comparando sucesivamente con nodos intermedios. Incluso un torneo de eliminación directa tiene una estructura de árbol.

La idea común es que un árbol representa una forma de conexión *sin redundancia*. Podemos movernos entre sus vértices, pero no hay ciclos que nos permitan dar vueltas. Esta tensión entre conectividad y ausencia de ciclos es lo que hace que los árboles tengan tanta estructura.

Definición. Un grafo simple no dirigido T se llama *árbol* si es conexo y acíclico, es decir, si no contiene ciclos. Un grafo simple no dirigido F se llama *bosque* si es acíclico.

Como un bosque no tiene ciclos, cada una de sus componentes conexas es un árbol. Por eso el nombre es natural: un bosque es una colección disjunta de árboles.

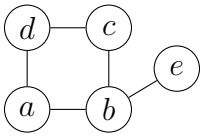
Ejemplo. El siguiente grafo es un árbol.



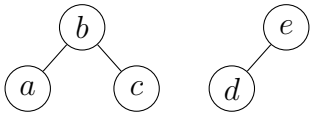
Es conexo, y no contiene ciclos.

Si agregamos la arista ef, aparece un ciclo. Si borramos la arista ac, el grafo deja de ser conexo. En ambos casos perdemos la propiedad de ser árbol.

Ejemplo. El siguiente grafo no es un árbol, porque contiene un ciclo.



En cambio, el siguiente grafo es un bosque: no contiene ciclos, pero tampoco es conexo.



5.22.1. Estructura de los árboles

La definición de árbol es muy corta, pero tiene consecuencias fuertes. La primera es que en un árbol no solamente existe un camino entre dos vértices: existe exactamente uno. Esto es una de las formas más útiles de reconocer árboles.

Lema. Sea T un árbol. Entonces entre cada par de vértices distintos de T existe un único camino simple.

Demostración. Como T es conexo, existe al menos un camino simple entre cada par de vértices distintos. Debemos probar que no pueden existir dos.

Supongamos, buscando una contradicción, que existen dos caminos simples distintos P y Q entre dos vértices u y v. Recorremos ambos caminos desde u. Como son distintos, en algún momento se separan. Sea x el primer vértice donde se separan. Después de separarse, como ambos caminos terminan en v, eventualmente se vuelven a encontrar. Sea y el primer vértice posterior donde esto ocurre.

El tramo de P entre x e y y el tramo de Q entre x e y son distintos y no comparten vértices internos. Por lo tanto, juntos forman un ciclo. Esto contradice que T es acíclico. Luego el camino simple entre u y v es único. □

Otra consecuencia básica es que los árboles siempre tienen extremos. Estos extremos son importantes porque permiten hacer pruebas por inducción: borramos una hoja, usamos la hipótesis inductiva, y luego la agregamos de vuelta.

Definición. En un árbol no enraizado, una *hoja* es un vértice de grado 1.

Lema. Todo árbol con al menos dos vértices tiene al menos dos hojas.

Demostración. Tomemos un camino simple de largo máximo en T,

$$v_0, v_1, \ldots, v_k.$$

Como T tiene al menos dos vértices, tenemos $k \geq 1$.

Afirmamos que v_0 es una hoja. En efecto, si v_0 tuviera un vecino w distinto de v_1, entonces w no puede aparecer entre $v_1, \ldots, v_k$, pues eso formaría un ciclo. Pero entonces

$$w, v_0, v_1, \ldots, v_k$$

sería un camino simple más largo, contradiciendo la elección del camino original. Por lo tanto, v_0 tiene grado 1.

El mismo argumento muestra que v_k también tiene grado 1. Luego T tiene al menos dos hojas. □

Proposición. Si $T = (V, E)$ es un árbol con $|V| = n$, entonces

$$|E| = n - 1.$$

Demostración. Lo probaremos por inducción en n.

Si $n = 1$, entonces no hay aristas, así que $|E| = 0 = n - 1$.

Supongamos que el resultado vale para todo árbol con $n - 1$ vértices, y sea T un árbol con $n \geq 2$ vértices. Por el lema anterior, T tiene una hoja v. Borramos v y su única arista incidente. El grafo resultante sigue siendo conexo y acíclico, luego es un árbol con $n - 1$ vértices. Por hipótesis inductiva, tiene $n - 2$ aristas.

Al volver a agregar v, agregamos exactamente una arista. Por lo tanto, T tiene

$$(n - 2) + 1 = n - 1$$

aristas. □

Teorema (Caracterizaciones de árboles)**.** Sea $T = (V, E)$ un grafo simple no dirigido con $|V| = n$ y $|E| = m$. Las siguientes propiedades son equivalentes.

1. T es un árbol.
2. Entre cada par de vértices distintos de T existe un único camino simple.
3. T es conexo minimal: es conexo, y al borrar cualquier arista deja de ser conexo.
4. T es conexo y $m = n - 1$.
5. T es acíclico maximal: es acíclico, y al agregar cualquier arista entre dos vértices no adyacentes aparece un ciclo.
6. T es acíclico y $m = n - 1$.

Demostración. Ya probamos que si T es un árbol, entonces entre cada par de vértices distintos existe un único camino simple. Recíprocamente, si existe un único camino simple entre cada par de vértices distintos, entonces T es conexo. Además, si T tuviera un ciclo, dos vértices consecutivos de ese ciclo estarían unidos por dos caminos simples distintos: la arista directa y el resto del ciclo. Esto contradice la unicidad. Por lo tanto, T es conexo y acíclico, es decir, un árbol.

Veamos ahora la caracterización minimal. Si T es un árbol y borramos una arista uv, entonces desaparece el único camino simple entre u y v. Por lo tanto, el grafo se vuelve no conexo. Recíprocamente, si T es conexo minimal, entonces no puede contener un ciclo: si una arista pertenece a un ciclo, podemos borrarla y los extremos siguen conectados por el resto del ciclo. Esto contradice la minimalidad. Así, T es conexo y acíclico.

Probemos la caracterización con $m = n-1$. Ya sabemos que todo árbol con n vértices tiene $n-1$ aristas. Recíprocamente, supongamos que T es conexo y tiene $n - 1$ aristas. Si T tuviera un ciclo, podríamos borrar una arista de ese ciclo y el grafo seguiría siendo conexo. Repitiendo este proceso mientras existan ciclos, obtendríamos un árbol sobre los mismos n vértices con estrictamente menos de $n-1$ aristas. Esto contradice la proposición anterior. Luego T no tiene ciclos, y como es conexo, es un árbol.

Finalmente, consideremos la caracterización acíclica maximal. Si T es un árbol y agregamos una arista uv que no estaba en T, entonces ya existía un único camino simple entre u y v. La nueva arista, junto con ese camino, forma un ciclo. Por lo tanto, T es acíclico maximal. Recíprocamente, si T es acíclico maximal pero no fuera conexo, podríamos agregar una arista entre dos componentes conexas distintas. Esa arista no crea ningún ciclo, contradiciendo la maximalidad. Así, T es conexo y acíclico.

La equivalencia con la condición "acíclico y $m = n-1$" se obtiene de forma análoga. Un árbol es acíclico y tiene $n-1$ aristas. Recíprocamente, si un grafo acíclico no fuera conexo, sus componentes conexas serían árboles. Si esas componentes tienen tamaños $n_1, \ldots, n_c$, con $c \geq 2$, entonces el número total de aristas sería

$$\sum_{i=1}^{c}(n_i - 1) = n - c < n - 1,$$

contradiciendo $m = n - 1$. Por lo tanto, el grafo debe ser conexo y acíclico. □

Corolario. Si F es un bosque con n vértices y c componentes conexas, entonces

$$|E(F)| = n - c.$$

Demostración. Cada componente conexa de F es un árbol. Si sus tamaños son $n_1, \ldots, n_c$, entonces la componente i tiene $n_i - 1$ aristas. Por lo tanto,

$$|E(F)| = \sum_{i=1}^{c}(n_i - 1) = \sum_{i=1}^{c} n_i - c = n - c.$$

□

5.22.2. Árboles con raíz

En muchas aplicaciones no nos basta con saber que un grafo es un árbol. También queremos mirar el árbol desde un vértice especial. Ese vértice funciona como punto de partida de una jerarquía.

Por ejemplo, en un sistema de archivos hay un directorio raíz. En un árbol de búsqueda binaria comenzamos desde una clave inicial. En el árbol que codifica el código Morse, partimos desde el estado inicial y en cada paso elegimos punto o raya.

Definición. Un *árbol con raíz* es un árbol T junto con un vértice distinguido r, llamado la *raíz*.

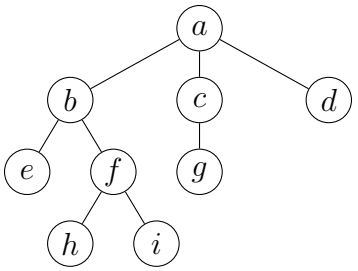
Una vez elegida la raíz, cada vértice queda ubicado a cierta distancia de ella. La unicidad de caminos en árboles permite definir relaciones familiares sin ambigüedad.

Definición. Sea T un árbol con raíz r.

- Si $v \neq r$, el *padre* de v es el vértice anterior a v en el único camino simple desde r hasta v.
- Si u es el padre de v, decimos que v es un *hijo* de u.
- Dos vértices son *hermanos* si tienen el mismo padre.
- Los *ancestros* de v son los vértices que aparecen en el camino desde r hasta v.
- Los *descendientes* de v son los vértices que tienen a v como ancestro.
- Una *hoja* es un vértice sin hijos.
- Un *vértice interno* es un vértice con al menos un hijo.

Definición. Sea T un árbol con raíz. La *altura* de T es el largo máximo de un camino desde la raíz hasta una hoja. La *aridad* de T es el máximo número de hijos que tiene un vértice de T.

Ejemplo. Consideremos el siguiente árbol con raíz a.



El padre de f es b. Los vértices b, c, d son hermanos. Los ancestros de i son a, b, f, i si incluimos al propio vértice, o a, b, f si hablamos de ancestros propios. Las hojas son d, e, g, h, i. La altura es 3, y la aridad es 3.

La raíz no cambia el grafo subyacente, pero cambia la jerarquía. El mismo árbol puede verse muy distinto si elegimos otra raíz. Por eso, cuando hablamos de padres, hijos, ancestros o altura, siempre debemos tener claro cuál es la raíz.

5.22.3. Árboles k-arios completos

En algunas aplicaciones queremos controlar cuántas decisiones pueden salir de cada nodo. Por ejemplo, en un árbol binario de búsqueda cada nodo tiene a lo más dos hijos. En un árbol de decisión con preguntas de respuesta múltiple, cada pregunta puede tener a lo más k respuestas posibles.

> **Definición.** Sea $k \geq 1$. Un árbol con raíz es *k-ario* si cada vértice interno tiene a lo más k hijos. Es *k-ario completo* si cada vértice interno tiene exactamente k hijos.

Ejemplo. Todo árbol binario de búsqueda es un árbol 2-ario. No necesariamente es 2-ario completo: un nodo puede tener sólo un hijo, o ninguno.

En cambio, un torneo de eliminación directa perfectamente balanceado puede modelarse como un árbol binario completo si cada partido combina exactamente dos participantes o equipos que vienen de rondas anteriores.

> **Proposición.** Sea T un árbol k-ario de altura h. Entonces T tiene a lo más
>
> $$k^h$$
>
> hojas.

Demostración. Lo probaremos por inducción en la altura h.

Si $h = 0$, el árbol sólo tiene la raíz, así que tiene una hoja y $1 = k^0$.

Supongamos ahora que $h \geq 1$. La raíz tiene a lo más k hijos. Cada hijo es la raíz de un subárbol k-ario de altura a lo más $h - 1$. Por hipótesis inductiva, cada uno de esos subárboles tiene a lo más k^{h-1} hojas. Por lo tanto, T tiene a lo más

$$k \cdot k^{h-1} = k^h$$

hojas. □

Proposición. Sea T un árbol k-ario completo con i vértices internos. Entonces T tiene

$$ki + 1$$

vértices en total.

Demostración. Sea n el número total de vértices de T. Como T es un árbol, tiene $n - 1$ aristas.

Por otro lado, cada arista conecta a un padre con un hijo. Como cada uno de los i vértices internos tiene exactamente k hijos, el número de aristas también es ki. Por lo tanto,

$$n - 1 = ki.$$

Concluimos que $n = ki + 1$. □

Corolario. Si T es un árbol k-ario completo con i vértices internos y ℓ hojas, entonces

$$\ell = (k - 1)i + 1.$$

Demostración. Por la proposición anterior, el número total de vértices es $ki + 1$. Estos vértices se dividen en internos y hojas, así que

$$i + \ell = ki + 1.$$

Despejando, obtenemos

$$\ell = (k - 1)i + 1.$$ □

5.22.4. Árboles generadores

Los árboles también aparecen dentro de grafos más grandes. Supongamos que tenemos un grafo que representa caminos posibles entre pueblos. Queremos elegir algunos caminos para que todos los pueblos queden conectados, pero sin pagar por caminos redundantes.

Como un árbol es exactamente una red conexa sin ciclos, lo que buscamos es un árbol que use todos los vértices del grafo original.

Definición. Sea G un grafo simple no dirigido. Un *árbol generador* de G es un subgrafo T de G tal que:

- T contiene todos los vértices de G;
- T es un árbol.

Ejemplo. El ciclo C_n tiene varios árboles generadores. Basta borrar una arista cualquiera del ciclo. El resultado es un camino con n vértices, y por lo tanto es un árbol generador de C_n.

Teorema. Un grafo simple no dirigido G tiene un árbol generador si y sólo si G es conexo.

Demostración. Supongamos primero que G tiene un árbol generador T. Como T es un árbol, entre cada par de vértices existe un camino en T. Pero T es subgrafo de G, así que ese camino también existe en G. Por lo tanto, G es conexo.

Recíprocamente, supongamos que G es conexo. Si G no tiene ciclos, entonces G mismo es un árbol y terminamos. Si G tiene un ciclo, borramos una arista de ese ciclo. El grafo sigue siendo conexo, porque los extremos de la arista borrada siguen conectados por el resto del ciclo.

Repetimos este procedimiento mientras existan ciclos. Como el grafo tiene un número finito de aristas, el proceso termina. Al final obtenemos un subgrafo conexo, acíclico, y con todos los vértices de G. Por lo tanto, obtenemos un árbol generador de G. □

Esta demostración también entrega una forma conceptual de construir un árbol generador: eliminar aristas que pertenecen a ciclos hasta que ya no quede redundancia. Más adelante veremos algoritmos de exploración, como BFS y DFS, que construyen árboles generadores de manera más directa.

Ejercicios

1. Considere la siguiente representación de directorios en Linux.

```
/
|-  bin
|-  etc
|   |-    nginx
|   ‘-    ssh
|-  home
|   |-    ana
|   |     |-     docs
|   |     ‘-     img
|   ‘-    beto
|         ‘-     src
‘-  usr
    |-    bin
    ‘-    lib
```

   Modelamos esta estructura con un grafo que tiene un vértice por directorio y una arista entre cada directorio y su directorio padre.

   a) Explique por qué este grafo es un árbol con raíz `/`.

   b) Determine las hojas, los vértices internos, la altura y la aridad del árbol.

   c) Dibuje el árbol con raíz `/` y marque el padre de `/home/ana/img`, sus hermanos y sus ancestros.

2. Sea T un árbol con raíz r. Para cada vértice v, definimos su *nivel* como la distancia desde r hasta v.

   a) Demuestre que si uv es una arista de T, entonces los niveles de u y v difieren exactamente en 1.

   b) Concluya que si A es el conjunto de vértices de nivel par y B es el conjunto de vértices de nivel impar, entonces toda arista de T tiene un extremo en A y el otro en B.

c) Demuestre que esta partición es única salvo intercambiar A y B.

d) Si a y b son hermanos, demuestre que a y b tienen el mismo nivel.

3. Sea T un árbol no dirigido. Demuestre que dos caminos simples de largo máximo en T tienen al menos un vértice en común.

4. Sea G un grafo conexo y sea T un árbol generador de G.

   a) Demuestre que si e es una arista de G que no pertenece a T, entonces $T + e$ contiene exactamente un ciclo.

   b) Demuestre que si e es una arista de T, entonces $T - e$ tiene exactamente dos componentes conexas.

   c) Use las partes anteriores para explicar por qué un árbol generador representa una forma de conectar todos los vértices sin redundancia.

5. Sea G un grafo simple conexo. Diremos que una arista e de G es un *puente* si $G - e$ no es conexo. Demuestre que una arista e pertenece a todo árbol generador de G si y sólo si e es un puente.

   Luego concluya que G tiene un único árbol generador si y sólo si G es un árbol.

5.23. Árboles generadores

En la clase anterior vimos que un árbol es, en cierto sentido, una forma mínima de conectividad: conecta sus vértices, pero no contiene ciclos. Ahora queremos buscar esa misma estructura dentro de grafos más grandes.

Pensemos en una red de ciudades, servidores o módulos de software. El grafo original puede tener muchos enlaces posibles, pero si el objetivo es garantizar que todo quede conectado, no necesitamos conservarlos todos. Podemos quedarnos con una subred que conecte todos los vértices sin redundancia.

Definición. Sea G un grafo simple. Un *árbol generador* de G es un subgrafo T de G tal que:

- T contiene todos los vértices de G;
- T es un árbol.

La palabra "generador" indica que no estamos descartando vértices: el subgrafo debe usar todo el conjunto de vértices de G. La palabra "árbol" indica que la conexión no tiene redundancia: no hay ciclos, y por tanto no hay aristas que podamos quitar sin cambiar la conectividad interna del subgrafo.

Si G tiene n vértices, entonces cualquier árbol generador de G tiene exactamente $n - 1$ aristas. Así, un árbol generador es una forma compacta de certificar que un grafo es conexo: en vez de exhibir caminos entre todos los pares de vértices, exhibimos una estructura que los conecta a todos.

Ejemplo. El ciclo C_n tiene n árboles generadores distintos. En efecto, al borrar una de sus n aristas obtenemos uno de ellos. Lo que queda es un camino con n vértices, que es un árbol generador de C_n.

En cambio, si un grafo no es conexo, no puede tener árbol generador. Ningún subgrafo puede crear una conexión entre vértices que ya estaban en componentes distintas del grafo original.

Teorema. Sea G un grafo simple. Entonces G tiene un árbol generador si y sólo si G es conexo.

Demostración. Supongamos primero que G tiene un árbol generador T. Como T es un árbol, entre cada par de vértices existe un camino en T. Como T es un subgrafo de G, ese mismo camino también existe en G. Por lo tanto, G es conexo.

Para la otra dirección, supongamos que G es conexo. Si G no contiene ciclos, entonces G mismo es un árbol y terminamos. Si G contiene un ciclo, podemos borrar una arista de ese ciclo sin desconectar el grafo: sus extremos siguen conectados usando el resto del ciclo.

Repetimos este procedimiento mientras queden ciclos. Como el grafo tiene finitas aristas, el proceso termina. El subgrafo final sigue siendo conexo, no tiene ciclos y conserva todos los vértices de G. Por lo tanto, es un árbol generador de G. □

La demostración anterior construye un árbol generador eliminando aristas redundantes. Para diseñar algoritmos, sin embargo, suele ser más útil mirar la construcción en la dirección opuesta: partir con un vértice y hacer crecer un árbol agregando un vértice nuevo en cada paso. Esa será la idea común detrás de los algoritmos de exploración.

5.23.1. Exploración incremental

La propiedad que permite hacer crecer un árbol es la siguiente. Si partimos con un árbol y agregamos un vértice nuevo usando una sola arista que lo conecta al árbol, el resultado sigue siendo un árbol. Esta observación explica por qué BFS, DFS y Prim tienen la misma forma básica.

Lema. Sea $G = (V, E)$ un grafo simple, $U \subseteq V$ y (U, T) subgrafo árbol de G. Si $e = uv$ conecta $u \in U$ y $v \notin U$, entonces $(U \cup \{v\}, T \cup \{uv\})$ es un árbol.

Demostración. Escribamos

$$T' = (U \cup \{v\}, T \cup \{uv\}).$$

La conectividad se conserva por la forma en que construimos T'. Los vértices de U ya estaban conectados entre sí en (U, T), y el nuevo vértice v queda conectado a ellos mediante la arista uv. Por lo tanto, todos los vértices de T' quedan en una misma componente conexa.

Falta ver que no aparece un ciclo. El grafo (U, T) no tenía ciclos. Cualquier ciclo nuevo tendría que usar la arista uv, porque es la única arista que agregamos. Para cerrar un ciclo con uv, tendría que existir además un camino desde v hasta u usando sólo aristas de (U, T). Eso es imposible, pues $v \notin U$ y (U, T) sólo contiene vértices de U.

Así, T' es conexo y acíclico. Por lo tanto, es un árbol. □

Algoritmo de exploración.

Partir con $U \leftarrow \{r\}$ y $T \leftarrow \emptyset$.
while $U \neq V$:
    Elegir $uv \in E$ con $u \in U$ y $v \notin U$.
    $U \leftarrow U \cup \{v\}$, $T \leftarrow T \cup \{uv\}$.
return (U, T).

La gracia del algoritmo es que siempre funciona, sin importar cómo se elija la arista entre U y U^c.

Teorema. Sea $G = (V, E)$ un grafo simple y conexo. El algoritmo de exploración genérico devuelve un árbol generador.

Demostración. La demostración es por invariante. Después de cada paso, el subgrafo (U, T) es un árbol.

Al comienzo, $U = \{r\}$ y $T = \varnothing$. Por lo tanto, (U, T) es el árbol con un único vértice. Si en algún paso agregamos una arista uv con $u \in U$ y $v \notin U$, entonces por el lema anterior el nuevo subgrafo sigue siendo un árbol.

Falta justificar que el algoritmo nunca se queda atascado antes de visitar todos los vértices. Mientras $U \neq V$, debe existir una arista que salga de U hacia $V \setminus U$. Como G es conexo, para cualquier vértice $x \in V \setminus U$ existe un camino desde r hasta x. Ese camino parte dentro de U y termina fuera de U, así que en algún momento cruza desde U hacia $V \setminus U$. Por lo tanto, existe una arista uv con $u \in U$ y $v \notin U$.

En cada paso agregamos un vértice nuevo a U. Como V es finito, el proceso termina con $U = V$. En ese momento (V, T) es un árbol que contiene todos los vértices de G, es decir, un árbol generador. □

El teorema separa dos ideas. La primera es estructural: agregar un vértice nuevo mediante una arista mantiene la propiedad de ser árbol. La segunda es algorítmica: si el grafo es conexo, siempre hay una arista disponible para seguir creciendo.

Las distintas formas de elegir esa arista producen distintos algoritmos de exploración.

5.23.2. Búsqueda en anchura y búsqueda en profundidad

Una forma natural de elegir la arista a agregar es mirar el orden en que los vértices fueron incorporados al árbol.

Definición (Tiempo de llegada)**.** En una ejecución del algoritmo de exploración, el *tiempo de llegada* de un vértice x es el instante en que x se agrega al conjunto U. Denotamos este tiempo por $\tau(x)$. La raíz tiene tiempo de llegada 0 y, cada vez que se agrega un vértice nuevo, se le asigna el siguiente tiempo de llegada.

Primero consideremos la regla que siempre continúa desde el vértice de menor tiempo de llegada que todavía permite agregar un vértice nuevo.

Definición (Búsqueda en anchura (BFS))**.** La *búsqueda en anchura*, o *BFS*, es el algoritmo de exploración con la siguiente regla de elección. En cada paso escogemos una arista $uv \in E$ con $u \in U$ y $v \notin U$ tal que u tenga el menor tiempo de llegada posible:

$$\tau(u) = \min\{\tau(x) : x \in U \text{ y existe } y \notin U \text{ tal que } xy \in E\}.$$

Luego agregamos la arista uv y asignamos a v el siguiente tiempo de llegada. Si hay varias aristas que cumplen la regla, cualquiera de ellas puede elegirse.

BFS explora completamente los vecinos de cada vértice antes de pasar al siguiente. Por eso su implementación natural usa una cola: los vértices que llegan primero son también los primeros que se expanden.

El efecto geométrico es que BFS construye capas alrededor de la raíz. Primero aparecen los vértices a distancia 1, luego los de distancia 2, y así sucesivamente. Por eso BFS computa distancias mínimas si todas las aristas tienen peso 1.

Proposición (BFS computa distancias mínimas)**.** Sea G un grafo simple conexo y sea $r \in V(G)$. Si todas las aristas tienen peso 1, BFS computa distancias mínimas desde r. Más precisamente, si BFS construye un árbol con raíz r, entonces cada vértice v queda en el nivel

$$d_G(r, v),$$

es decir, en la distancia mínima desde r hasta v en G.

Demostración. Sea $\ell(v)$ el nivel en que BFS incorpora a v. El árbol de BFS contiene un camino desde r hasta v de largo $\ell(v)$. Por lo tanto,

$$d_G(r, v) \leq \ell(v).$$

Probemos ahora la desigualdad contraria. Supongamos, buscando una contradicción, que existe un vértice v tal que

$$d_G(r, v) < \ell(v).$$

Tomemos un contraejemplo de distancia mínima, y sea

$$r = x_0, x_1, \ldots, x_k = v$$

un camino más corto desde r hasta v. Entonces $k = d_G(r, v)$.

El vértice x_{k-1} tiene distancia $k - 1$, así que por la minimalidad de v no es un contraejemplo. Luego BFS lo incorpora en el nivel $k - 1$. Cuando BFS explora los vértices de ese nivel, ve la arista $x_{k-1}v$. Por lo tanto, v debió ser incorporado a nivel a lo más k. Es decir,

$$\ell(v) \leq k = d_G(r, v),$$

contradiciendo la elección de v.

Concluimos que $\ell(v) = d_G(r, v)$ para todo vértice v. □

La segunda regla natural siempre continúa desde el vértice de mayor tiempo de llegada que todavía permite agregar un vértice nuevo.

Definición (Búsqueda en profundidad (DFS))**.** La *búsqueda en profundidad*, o *DFS*, es el algoritmo de exploración con la siguiente regla de elección. En cada paso escogemos una arista $uv \in E$ con $u \in U$ y $v \notin U$ tal que u tenga el mayor tiempo de llegada posible:

$$\tau(u) = \text{máx}\{\tau(x) : x \in U \text{ y existe } y \notin U \text{ tal que } xy \in E\}.$$

Luego agregamos la arista uv y asignamos a v el siguiente tiempo de llegada. Si hay varias aristas que cumplen la regla, cualquiera de ellas puede elegirse.

DFS explora lo más profundo posible antes de retroceder. Su implementación natural usa una pila: el último vértice que llega es el primero que se intenta expandir.

DFS es bueno para encontrar ciclos y otras estructuras. También suele usar menos memoria que BFS, porque no necesita mantener una frontera completa por niveles.

Ejemplo. En una rueda W_n, si comenzamos BFS desde el centro, el árbol de búsqueda incorpora inmediatamente todos los vértices del ciclo exterior. El árbol resultante se parece a una estrella.

En cambio, si hacemos DFS desde el centro y luego avanzamos por el ciclo exterior, el árbol puede verse como una rama larga que rodea la rueda antes de retroceder. Ambos son árboles generadores, pero registran información distinta sobre el grafo.

Una manera útil de recordar la diferencia es la siguiente. BFS responde bien preguntas de distancia: "¿qué tan lejos está este vértice de la raíz?". DFS responde bien preguntas de exploración: "¿hasta dónde puedo avanzar antes de quedarme sin opciones?".

5.23.3. Árboles generadores de costo mínimo

Ahora agreguemos costos. Si las aristas representan enlaces de red, caminos o cables, conectar todos los vértices ya no es el único objetivo. Queremos conectarlos pagando lo menos posible.

Esto nos lleva al problema de encontrar un árbol generador de costo mínimo.

Definición. Sea G un grafo simple con pesos w. Un *árbol generador de costo mínimo (MST)* de G con pesos w es un subgrafo (U, T) tal que

- (U, T) es un árbol generador,
- minimiza $\sum_{e \in T} w(e)$.

Nuevamente, pensemos en soluciones de manera incremental. Si buscamos agregar una arista entre un conjunto U de vértices ya conectados y el resto del grafo, lo más natural es elegir la arista de menor costo. Dicha elección es segura, en el sentido de que después de tomarla todavía existe una solución óptima compatible con lo que hemos construido.

Lema (Regla azul)**.** Sea $G = (V, E)$ un grafo simple, conexo y con pesos w, $U \subseteq V$. Si $e = uv$ es una arista de menor costo entre las aristas con $u \in U$ y $v \notin U$, entonces existe un MST que la contiene.

Demostración. Sea M un MST de G. Si $e \in E(M)$, entonces M ya es un MST que contiene a e.

Supongamos entonces que $e \notin E(M)$. Al agregar e a M, se forma un único ciclo. Como e tiene un extremo en U y el otro en $V \setminus U$, ese ciclo debe cruzar entre U y $V \setminus U$ otra vez para volver al lado donde empezó. Por lo tanto, existe una arista $f \in E(M)$, distinta de e, que también tiene un extremo en U y el otro en $V \setminus U$.

Como e era de menor costo entre las aristas que cruzan de U a $V \setminus U$,

$$w(e) \leq w(f).$$

Consideremos

$$M' = M + e - f.$$

Al borrar una arista de un ciclo, el grafo vuelve a ser un árbol generador. Además,

$$w(M') = w(M) + w(e) - w(f) \leq w(M).$$

Como M ya tenía costo mínimo, M' también es un MST. Además, M' contiene a e. □

La regla azul explica el algoritmo de Prim. Mantenemos un conjunto U de vértices ya conectados y elegimos la arista más barata que sale de U. La regla azul dice que esa elección es segura: después de tomarla, todavía existe una solución óptima compatible con lo que hemos construido.

Algoritmo de Prim.

Partir con $U \leftarrow \{r\}$ y $T \leftarrow \emptyset$.
while $U \neq V$:
    Elegir $uv \in E$ con $u \in U$ y $v \notin U$ de menor costo.
    $U \leftarrow U \cup \{v\}$, $T \leftarrow T \cup \{uv\}$.
return (U, T).

Formalizamos el hecho de que Prim funciona en el siguiente teorema.

Teorema. Sea $G = (V, E)$ un grafo simple y conexo. El algoritmo de Prim devuelve un árbol generador de costo mínimo.

Demostración. Primero veamos que Prim efectivamente construye un árbol generador. Por el lema del paso incremental, las aristas elegidas siempre forman un árbol sobre el conjunto U de vértices ya incorporados. Como G es conexo, mientras $U \neq V$ existe alguna arista con un extremo en U y el otro en $V \setminus U$. Por lo tanto, Prim termina con un árbol generador.

Falta probar que ese árbol tiene costo mínimo. Mantendremos la siguiente invariante: después de cada paso, existe un MST que contiene todas las aristas que Prim ha elegido hasta ese momento.

Al inicio, no hemos elegido aristas, así que cualquier MST cumple la invariante. Supongamos que la invariante vale antes de un paso, y sea A el conjunto de aristas ya elegidas. Sea M un MST que contiene a A. Prim elige una arista $e = uv$ de menor costo entre las aristas con $u \in U$ y $v \notin U$.

Si $e \in E(M)$, entonces M contiene $A \cup \{e\}$ y terminamos. Si $e \notin E(M)$, entonces $M + e$ contiene un único ciclo. Como e tiene un extremo en U y el otro en $V \setminus U$, ese ciclo contiene otra arista f que también tiene un extremo en U y el otro en $V \setminus U$. Además, ninguna arista de A cruza entre U y $V \setminus U$, pues A está completamente dentro de U. Por lo tanto, $f \notin A$.

Como e es de menor costo entre las aristas con un extremo en U y el otro fuera de U, tenemos $w(e) \leq w(f)$. Entonces

$$M' = M + e - f$$

es un árbol generador de costo a lo más $w(M)$. Por lo tanto, M' también es un MST. Además, M' contiene todas las aristas de A y también contiene e. Así, la invariante se mantiene después del paso.

Al terminar, Prim produjo un árbol generador T contenido en algún MST. Como ambos son árboles generadores sobre el mismo conjunto de vértices, ambos tienen $|V| - 1$ aristas. Luego T coincide con ese MST. Por lo tanto, T es un árbol generador de costo mínimo. □

La regla azul nos dice que las aristas más baratas de la conectividad se pueden incluir. La regla dual mira la aciclicidad. Si en un ciclo hay una arista demasiado cara, podemos evitarla: el resto del ciclo ya entrega una forma alternativa de conectar sus extremos.

Esta es la regla roja.

Lema (Regla roja)**.** Sea G un grafo simple, conexo y con pesos w, y C las aristas de un ciclo en G. Si $e = uv$ es la arista de C de mayor costo, entonces existe un MST que la evita.

Demostración. Sea T un MST de G. Si $e \notin E(T)$, entonces T ya es un MST que evita a e.

Supongamos que $e \in E(T)$. Al borrar e de T, el árbol se separa en dos componentes conexas. Como las aristas de C forman un ciclo, el resto de ese ciclo conecta los extremos de e. Al recorrerlo, necesariamente encontramos alguna arista $f \neq e$ que cruza entre las dos componentes que dejó $T - e$.

Entonces

$$T' = T - e + f$$

vuelve a ser un árbol generador. Como e tiene mayor costo en C, tenemos $w(f) \leq w(e)$. Por lo tanto,

$$w(T') = w(T) - w(e) + w(f) \leq w(T).$$

Así, T' también es un MST, pero ahora evita la arista e. □

Kruskal puede entenderse como una aplicación sistemática de estas reglas. Ordenamos las aristas de menor a mayor peso. Cuando una arista conecta dos componentes distintas del bosque actual, la agregamos. Cuando una arista cerraría un ciclo, la descartamos: en ese ciclo, todas las aristas que ya estaban fueron consideradas antes, así que la nueva arista es una de las más caras y puede evitarse.

Algoritmo de Kruskal.

Ordenar las aristas por peso $w(e_1) \leq w(e_2) \leq \cdots \leq w(e_m)$.
Partir con $T \leftarrow \emptyset$.
for $i = 1, \ldots, m$:
    if $(V, T \cup \{e_i\})$ es acíclico:
        $T \leftarrow T \cup \{e_i\}$.
return (V, T).

Nuevamente, formalizamos el hecho de que Kruskal funciona en el siguiente teorema.

Teorema. Sea $G = (V, E)$ un grafo simple y conexo. El algoritmo de Kruskal devuelve un árbol generador de costo mínimo.

Demostración. Durante todo el algoritmo, el conjunto A de aristas elegidas es acíclico, porque Kruskal sólo agrega una arista cuando no crea ciclos. Veamos que al final también conecta todos los vértices. Si quedaran dos componentes distintas en (V, A), como G es conexo habría alguna arista de G que cruza entre ellas. Cuando Kruskal procesó esa arista, sus extremos estaban en componentes distintas, así que agregarla no habría creado un ciclo. Por lo tanto, Kruskal la habría agregado, contradiciendo que esas componentes quedaron separadas al final. Luego (V, A) es conexo y acíclico, es decir, un árbol generador.

Probemos ahora que tiene costo mínimo. Usaremos la invariante: después de cada arista agregada por Kruskal, existe un MST que contiene todas las aristas elegidas hasta ese momento.

Al comienzo no hemos elegido aristas, así que cualquier MST cumple la invariante. Supongamos que A está contenido en algún MST M y que Kruskal agrega una arista $e = uv$. Esto significa que u y v están en componentes distintas del bosque (V, A).

Si $e \in E(M)$, entonces M ya contiene $A \cup \{e\}$. Supongamos entonces que $e \notin E(M)$. Al agregar e a M, aparece un único ciclo. En el camino de M entre u y v debe existir una arista f cuyos extremos están en componentes distintas de (V, A). Si no existiera, ese camino conectaría u con v usando sólo aristas internas a las componentes de A, contradiciendo que u y v estaban en componentes distintas. En particular, $f \notin A$.

Afirmamos que $w(e) \leq w(f)$. Si $w(f) < w(e)$, entonces Kruskal habría considerado f antes que e. En ese momento sus extremos todavía estaban en componentes distintas, porque las componentes del bosque sólo se fusionan con el tiempo. Por lo tanto, Kruskal habría agregado f, contradiciendo que $f \notin A$ justo antes de agregar e.

Definimos

$$M' = M + e - f.$$

Este es un árbol generador, y

$$w(M') = w(M) + w(e) - w(f) \leq w(M).$$

Por minimalidad de M, el árbol M' también es un MST. Además contiene $A \cup \{e\}$. Así, la invariante se mantiene después de agregar e.

Al terminar, el árbol construido por Kruskal está contenido en algún MST. Como ambos tienen $|V| - 1$ aristas, coinciden. Por lo tanto, Kruskal devuelve un MST. □

Ejercicios

1. Un equipo quiere instalar túneles privados entre servicios de una aplicación distribuida. Los servicios son

   $$A, B, C, D, E, F$$

   y los posibles túneles, junto con su costo mensual, son

túnel	AB	AC	BC	BD	CD	CE	DE	DF	EF	BF
costo	4	2	5	1	3	7	2	6	4	8

   Se quiere que todos los servicios puedan comunicarse entre sí, directamente o a través de otros servicios, minimizando el costo total.

a) Modele el problema como un problema de árboles generadores de costo mínimo.

b) Explique por qué una solución óptima no debería contener ciclos.

c) Use las reglas azul y roja para justificar una elección de túneles de costo mínimo.

d) Determine el costo mínimo total.

2. Sea G un grafo simple conexo y sea $r \in V(G)$. Para cada vértice v, sea $d(v) = d_G(r, v)$.

   a) Demuestre que si $uv \in E(G)$, entonces

   $$|d(u) - d(v)| \leq 1.$$

   b) Suponga que no existe ninguna arista uv con $d(u) = d(v)$. Demuestre que G es bipartito.

   c) Suponga que existe una arista uv con $d(u) = d(v)$. Demuestre que G contiene un ciclo de largo impar.

3. Hay n viviendas que se quieren conectar al suministro de agua. Entre las viviendas i y j se puede construir una cañería de costo c_{ij}. Además, la vivienda i puede conectarse directamente a la matriz principal de agua con costo C_i. Se quiere que todas las viviendas tengan acceso al agua. ¿Cómo puede resolver este problema usando algoritmos que ya conoce?

4. Sea G un grafo simple, conexo y con pesos. Suponga que todos los pesos de las aristas son distintos. Demuestre que G tiene un único árbol generador de costo mínimo.

5. Sea G un grafo simple, conexo y con pesos, y sea T un MST de G.

   a) Para dos vértices $u, v \in V(G)$, sea $P_T(u, v)$ el único camino de u a v en T. Demuestre que $P_T(u, v)$ minimiza el peso de la arista más pesada del camino.

   Más precisamente, demuestre que para todo camino P de u a v en G se cumple

   $$\max_{e \in P_T(u,v)} w(e) \leq \max_{e \in P} w(e).$$

   b) Alicia se desplaza por una ciudad que tiene lugares de interés etiquetados por $1, 2, \ldots, n$. Alicia sabe que en cada trayecto ij hay un nivel de ruido r_{ij}. Fijados dos lugares u y v, ¿cómo puede Alicia asegurarse de que el máximo nivel de ruido al que está expuesta se minimiza?