Apuntes de Matemáticas Discretas para la Computación

Andrés Abeliuk

2025

Este apunte han sido elaborado a partir de las diapositivas del curso CC3101, Matemáticas Discretas, impartido por el DCC de la Universidad de Chile por Alejandro Hevia, Federico Olmedo y Jocelyn Simmonds.

Índice general

Parte I

Fundamentos de La Lógica

Capítulo 1

Lógica Proposicional

1.1. ¿Qué es la lógica?

La lógica es una disciplina fundamental para el razonamiento y la estructura del conocimiento. Se centra en determinar cuándo un argumento es válido, es decir, en la relación entre premisas y conclusión, sin importar el contenido específico de las afirmaciones.

1.1.1. Breve Historia de la Lógica

La lógica ha experimentado una evolución a lo largo de la historia. Desde sus inicios en la antigua Grecia hasta su aplicación en la computación moderna, la lógica ha dejado una huella en diversas áreas del saber.

Todo comenzó con **Aristóteles** en el siglo IV a.C., quien estableció los cimientos de la lógica formal al introducir el silogismo. Este método de razonamiento deductivo, basado en premisas y conclusiones, sentó las bases para el desarrollo de sistemas lógicos más complejos.

Siglos después, la **lógica estoica** (siglos III a.C.-II d.C.) emergió con un enfoque innovador. Los estoicos desarrollaron un sistema lógico basado en operadores y reglas inferenciales, anticipando ideas esenciales que hoy son pilares de la lógica contemporánea.

Avanzando en el tiempo, **George Boole** (1815-1864) revolucionó el campo con su lógica booleana. Este sistema algebraico para el razonamiento lógico no solo transformó la lógica, sino que también sentó las bases para la computación moderna.

Gottlob Frege (1848-1925) llevó la lógica a otro nivel al introducir el cálculo de predicados y la cuantificación lógica. Estos conceptos se convirtieron en fundamentos esenciales para la computación, las matemáticas y la filosofía analítica.

Finalmente, **Claude Shannon** (1937) aplicó la lógica booleana a circuitos eléctricos en su tesis de maestría, *A Symbolic Analysis of Relay and Switching Circuits*. Este trabajo seminal marcó el inicio de la era de la computación digital, demostrando cómo la lógica puede ser utilizada para diseñar y optimizar sistemas electrónicos.

Estos hitos históricos ilustran cómo la lógica ha evolucionado y se ha adaptado, proporcio-

nando herramientas poderosas para el razonamiento y la innovación en diversas disciplinas.

1.1.2. Aplicaciones de la Lógica en Ciencias de la Computación

La lógica, en sus diversas formas, es una herramienta esencial en el campo de las ciencias de la computación. En el diseño de **lenguajes de programación**, la lógica formal proporciona las bases para definir sintaxis y semántica, asegurando que los programas sean coherentes y funcionales.

En la **verificación de software**, la lógica garantiza que los programas cumplan con sus especificaciones, detectando errores y mejorando la calidad del código. Las **bases de datos y los motores de búsqueda** también se benefician de la lógica, permitiendo la formulación de consultas eficientes y la extracción de información relevante de grandes volúmenes de datos.

La lógica se utiliza en **ciberseguridad y criptografía** para analizar protocolos y detectar vulnerabilidades, asegurando la integridad y confidencialidad de la información. Además, en el desarrollo de **algoritmos y estructuras de datos**, la lógica proporciona un marco riguroso para diseñar soluciones eficientes y optimizadas. Finalmente,en la utilizainteligencia artificial se utiliza para modelar el razonamiento y la toma de decisiones, permitiendo la creación de sistemas autónomos y adaptativos.

En todos estos ámbitos, la lógica proporciona un marco riguroso para estructurar el conocimiento y alcanzar conclusiones fundamentadas.

1.1.3. Tipos de Lógica

El razonamiento formal se apoya en diversos sistemas lógicos, cada uno con su propia expresividad y aplicaciones. A continuación, se presentan algunos de los más importantes.

La **lógica proposicional** es fundamental para el razonamiento lógico, utilizando proposiciones atómicas y conectivos lógicos para formar expresiones complejas. Se enfoca en las relaciones entre proposiciones y es esencial en matemáticas, computación y sistemas formales.

La **lógica de primer orden, o lógica de predicados**, amplía la lógica proposicional al introducir cuantificadores y predicados, permitiendo afirmaciones más detalladas sobre objetos en un dominio. Es más expresiva y se usa ampliamente en matemáticas, filosofía y ciencias de la computación.

La **lógica modal** introduce operadores para razonar sobre modalidades como la posibilidad y la necesidad, permitiendo considerar lo que es verdadero en diferentes contextos. Tiene aplicaciones en filosofía, inteligencia artificial y teoría de la información. La **lógica temporal** es una extension con operadores temporales para expresar conceptos como “siempre”, “eventualmente” y “hasta que”, lo que permite modelar procesos dinámicos.

En este curso, nos centraremos en la **lógica proposicional y de primer orden**, Estos sistemas son la base para construir modelos más avanzados de inferencia y representación del conocimiento.

1.2. Razonamiento Formal

La lógica proposicional es una de las formas más fundamentales de razonamiento formal, proporcionando una base estructurada para la inferencia y el análisis lógico de argumentos.

1.2.1. Definiciones Básicas

- Una **proposición** es cualquier enunciado que puede ser verdadero o falso, pero no ambos simultáneamente.
- Un **argumento** es una secuencia de proposiciones donde las primeras se llaman premisas y la última se llama conclusión.
- Un argumento es **válido** si la verdad de sus premisas garantiza la verdad de su conclusión.

1.2.2. Argumentos

Un **argumento** es una secuencia de proposiciones que se estructuran de manera tal que una o más premisas conducen a una conclusión. Aristóteles fue pionero en el estudio de los argumentos, desarrollando la teoría del silogismo para analizar patrones lógicos de razonamiento.

Un ejemplo clásico de silogismo es el siguiente:

Todas las personas son mortales.
Sócrates es una persona.

Por lo tanto, Sócrates es mortal.

Este es un ejemplo de *razonamiento deductivo*, donde la conclusión se deduce necesariamente de las premisas. La validez del silogismo reside en que, si las premisas son verdaderas, la conclusión debe serlo también. En este caso, la inferencia se basa en el hecho de que todas las personas comparten la propiedad de ser mortales, y Sócrates pertenece a este grupo.

1.2.3. Validez de un Argumento

La **validez** de un argumento se refiere a la relación lógica entre las premisas y la conclusión. Un argumento es válido si, bajo la suposición de que todas sus premisas son verdaderas, la conclusión también debe ser necesariamente verdadera. Es decir, un argumento válido garantiza que no es posible que las premisas sean verdaderas y la conclusión falsa al mismo tiempo.

Por ejemplo, consideremos el siguiente argumento:

Todas las especies de aves pueden volar.
Un pingüino es un ave.

Por lo tanto, un pingüino puede volar.

Aunque la primera premisa es falsa, la estructura lógica del argumento sigue siendo válida, ya que la conclusión se deriva formalmente de las premisas. Es importante resaltar que la validez no implica que las premisas sean realmente ciertas en el mundo real, sino que la inferencia es correcta desde un punto de vista lógico.

Un argumento **válido** es aquel en el que, si todas las premisas son verdaderas, la conclusión necesariamente también lo es. Esto garantiza que no puede haber un caso en el que las premisas sean verdaderas y la conclusión falsa.

Ejemplo de argumento válido:

Si llueve, entonces el suelo estará mojado.
Está lloviendo.

Por lo tanto, el suelo está mojado.

Un argumento tiene una **forma lógica**, es decir, una representación simbólica que abstrae su estructura subyacente mediante variables y conectores lógicos. En esta representación, las proposiciones se representan con letras (por ejemplo, p, q, r) y los conectores lógicos (como $\rightarrow$ para implicación, $\vee$ para disyunción, $\wedge$ para conjunción, y $\neg$ para negación) se utilizan para mostrar las relaciones entre las proposiciones.

El argumento anterior tiene la siguiente forma lógica:

$$\begin{array}{l} p \rightarrow q \\ \underline{p} \\ q \end{array}$$

Este razonamiento sigue la estructura del *modus ponens*, una regla de inferencia fundamental en lógica proposicional. Dado que la primera premisa establece una implicación ("si llueve, entonces el suelo estará mojado") y la segunda premisa afirma la verdad de su antecedente ("está lloviendo"), la conclusión ("el suelo está mojado") se sigue necesariamente.

En contraste, un **argumento inválido** es aquel en el que, aunque las premisas sean verdaderas, la conclusión no necesariamente lo es.

Ejemplo de argumento inválido:

Si llueve, entonces el suelo estará mojado.
El suelo está mojado.

Por lo tanto, está lloviendo.

Este argumento es inválido porque, aunque la premisa "el suelo está mojado" sea verdadera, existen otras posibles causas para que el suelo esté mojado, como el riego de un jardín. Por lo tanto, la conclusión "está lloviendo" no necesariamente se sigue de las premisas.

Comprender la distinción entre un argumento válido e inválido es fundamental para el análisis lógico. Un argumento válido garantiza que, si las premisas son verdaderas, la conclusión también será verdadera, pero no asegura la verdad de las premisas. La validez, por lo tanto, se refiere únicamente a la relación lógica entre las premisas y la conclusión, no a la veracidad de las afirmaciones contenidas en las premisas.

1.2.4. Ejemplos de Argumentos y su Forma Lógica

En esta sección analizaremos diferentes tipos de argumentos y su representación lógica. Los argumentos en los **ejemplos 1 y 2** pueden expresarse utilizando únicamente **proposiciones atómicas** y **conectivos lógicos**. El argumento en el **ejemplo 3** requiere una lógica más expresiva, incorporando **predicados** y **cuantificadores**.

Ejemplo 1: Argumento válido

Argumento:

Si Ana estudia en el DCC o en el DIM, entonces tomará CC3101.
Ana estudia en el DCC.

Por lo tanto, Ana tomará CC3101.

Forma lógica:

$$\begin{array}{l} (p \vee q) \rightarrow r \\ p \\ \hline r \end{array}$$

Análisis: Este argumento es válido porque la conclusión se sigue necesariamente de las premisas. La estructura lógica muestra que si Ana estudia en el DCC (p), entonces tomará CC3101 (r).

Ejemplo 2: Argumento no válido

Argumento:

Si llueve, entonces se moja el suelo.
No llovió.

Por lo tanto, no se mojó el suelo.

Forma lógica:

$$\begin{array}{l} p \rightarrow q \\ \neg p \\ \hline \neg q \end{array}$$

Análisis: Este argumento comete la **falacia de la negación del antecedente**, ya que de $p \to q$ y $\neg p$ no se sigue necesariamente $\neg q$. La implicación solo garantiza que si p ocurre, entonces q también; sin embargo, q podría darse por otras razones. En este caso, el suelo podría mojarse de otra forma (por ejemplo, con una manguera), por lo que la conclusión no es válida.

Ejemplo 3: Argumento válido

Argumento:

Todas las personas son mortales.
Sócrates es una persona.

Por lo tanto, Sócrates es mortal.

Forma lógica:

$$\frac{\forall x.\ \ persona(x) \to mortal(x) \qquad persona(\mathsf{Socrates})}{mortal(\mathsf{Socrates})}$$

Análisis: Este argumento es válido porque la conclusión se sigue necesariamente de las premisas. La estructura lógica muestra que si Sócrates es una persona ($persona(\mathsf{Socrates})$), entonces Sócrates es mortal ($mortal(\mathsf{Socrates})$).

1.3. Lógica Proposicional

La lógica proposicional es el punto de partida en el estudio del razonamiento lógico. Este enfoque se centra en las **proposiciones**, que son enunciados que pueden ser verdaderos o falsos, pero no ambos simultáneamente.

Una **proposición** es cualquier enunciado que puede ser evaluado como verdadero o falso. Por ejemplo, las siguientes son proposiciones:

- Ana estudia en el DCC.
- Santiago es la capital de Chile.
- $2 > 4$.

En contraste, las siguientes no son proposiciones porque no pueden ser evaluadas como verdaderas o falsas:

- ¿Qué hora es?
- $x == 2$.

La validez de un argumento en la lógica proposicional puede ser determinada mediante dos métodos principales: **tablas de verdad** y **sistemas de inferencia**. Las tablas de verdad permiten evaluar todas las combinaciones posibles de valores de verdad para las proposiciones involucradas, mientras que los sistemas de inferencia proporcionan reglas para derivar nuevas proposiciones a partir de las conocidas.

1.3.1. Sintaxis

La sintaxis en lógica proposicional define las reglas que determinan cuándo una secuencia de símbolos constituye una **fórmula bien formada**.

La sintaxis es fundamental porque establece las reglas que deben seguirse para construir expresiones lógicas válidas. Estas reglas aseguran que las proposiciones sean claras y sin ambigüedades, lo cual es crucial para el razonamiento lógico.

Fórmulas bien formadas: $p \land q$, $\neg p \to p$, q
Fórmulas no bien formadas: $p \land$, $q\ p \neg\ p$, $\lor$

Conjunto de Fórmulas Bien Formadas

Una fórmula bien formada es o bien una **fórmula atómica**, formada por una proposición, o bien una **fórmula compuesta**, formada combinando proposiciones a través de conectivos lógicos.

Definición (sintaxis de la lógica proposicional)

Dado un conjunto $P = \{p, q, r, \ldots\}$ de proposiciones, el conjunto de **fórmulas bien formadas** sobre P, notado $\mathcal{L}(P)$, se define inductivamente de la siguiente manera:

Caso base: Si $p \in P$, entonces $p \in \mathcal{L}(P)$.
Caso inductivo: Si $\alpha, \beta \in \mathcal{L}(P)$, entonces $(\neg\alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \to \beta) \in \mathcal{L}(P)$.

Para evitar el aglutinamiento de paréntesis en las fórmulas nos permitimos omitir el par de paréntesis más externos e introducimos un orden de precedencia sobre los conectivos lógicos: $\neg$, $\land$, $\lor$, $\to$.

Ejemplo:

Fórmula	Forma Abreviada
$((\neg p) \to q)$	$\neg p \to q$
$((p \land q) \lor r)$	$p \land q \lor r$

1.3.2. Semántica

La semántica de un lenguaje lógico define las reglas que permiten determinar cuándo una fórmula bien formada es verdadera o falsa. Es decir, establece el significado preciso de las expresiones del lenguaje y cómo su veracidad depende de los valores asignados a sus componentes.

Una herramienta clave en esta definición es la **función de verdad**, que asigna a cada combinación de valores de las proposiciones atómicas un valor de verdad para la fórmula completa:

Función de verdad: $\{0,1\}^n \to \{0,1\}$

El valor de verdad de una fórmula $\alpha \in \mathcal{L}(P)$ depende únicamente del valor de verdad de las proposiciones en $\mathcal{L}(P)$ y del significado de los conectivos lógicos, dado por su **tabla de verdad**.

Para ilustrar, consideremos la fórmula $\alpha = p \wedge q$. Su tabla de verdad es:

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

La función de verdad asigna un valor de verdad (0 para falso, 1 para verdadero) a cada combinación de valores de verdad de las proposiciones atómicas. Esto se realiza mediante la tabla de verdad correspondiente a la fórmula α.

Tabla de Verdad de la Negación ($\neg$)

α	$\neg\alpha$
1	0
0	1

El valor de verdad de $\neg\alpha$ es opuesto al de α.

Tabla de Verdad de la Conjunción ($\wedge$)

α	β	$\alpha \wedge \beta$
1	1	1
1	0	0
0	1	0
0	0	0

$\alpha \wedge \beta$ es verdadero si y sólo si α y β son verdaderos.

Tabla de Verdad de la Disyunción ($\vee$)

α	β	$\alpha \vee \beta$
1	1	1
1	0	1
0	1	1
0	0	0

$\alpha \vee \beta$ es verdadero si y sólo si α o β son verdaderos.

Tabla de Verdad de la Implicancia ($\rightarrow$)

α	β	$\alpha \rightarrow \beta$
1	1	1
1	0	0
0	1	1
0	0	1

$\alpha \rightarrow \beta$ es siempre verdadero, excepto cuando α (el antecedente) es verdadero y β (el consecuente) es falso.

Valor de Verdad de Fórmulas Compuestas

El valor de verdad de una fórmula lógica compuesta se determina a partir de los valores de verdad de sus proposiciones atómicas y de las reglas establecidas por los conectivos lógicos.

Ejemplo: Determinar el valor de verdad de la siguiente fórmula para cada valuación de las proposiciones p y q:

$$(p \vee q) \rightarrow (p \wedge q)$$

p	q	$p \vee q$	$p \wedge q$	$(p \vee q) \rightarrow (p \wedge q)$
1	1	1	1	1
1	0	1	0	0
0	1	1	0	0
0	0	0	0	1

En la tabla de verdad anterior, se observa cómo el valor de la fórmula se obtiene paso a paso aplicando las definiciones de los conectivos lógicos. Esto nos permite verificar su comportamiento en todas las posibles valuaciones de p y q.

Semántica de la Lógica Proposicional

Alternativamente, podemos formalizar lo anterior de la siguiente manera:

Definición (semántica de la lógica proposicional)

Sea

$$\sigma \colon P \to \{0, 1\}$$

una **valuación** de las proposiciones en $P = \{p, q, r, \ldots\}$, es decir una función que le asigna un valor de verdad a las proposiciones en P.
La función

$$\hat{\sigma} \colon \mathcal{L}(P) \to \{0, 1\}$$

le asigna un valor de verdad a cada fórmula en $\mathcal{L}(P)$ de la siguiente manera:

$$\begin{array}{lcl} \hat{\sigma}(p) & = & \sigma(p) \quad \text{para } p \in P \\ \hat{\sigma}(\neg\alpha) & = & 1 - \hat{\sigma}(\alpha) \\ \hat{\sigma}(\alpha \wedge \beta) & = & \text{mín}\,\{\hat{\sigma}(\alpha), \hat{\sigma}(\beta)\} \\ \hat{\sigma}(\alpha \vee \beta) & = & \text{máx}\,\{\hat{\sigma}(\alpha), \hat{\sigma}(\beta)\} \\ \hat{\sigma}(\alpha \to \beta) & = & \text{máx}\,\{1 - \hat{\sigma}(\alpha), \hat{\sigma}(\beta)\} \end{array}$$

1.3.3. Definición Formal de un Argumento Válido

Un **argumento** es válido si y solo si, dado un conjunto de premisas $P_1, P_2, \ldots, P_n$ y una conclusión C, siempre que todas las premisas sean verdaderas, la conclusión también debe ser verdadera. Formalmente, esto se expresa de la siguiente manera:

$$(P_1 \wedge P_2 \wedge \cdots \wedge P_n) \to C$$

Es decir, un argumento $\langle P_1, P_2, \ldots, P_n, C\rangle$ es válido si y solo si, en cualquier interpretación donde todas las premisas $P_1, P_2, \ldots, P_n$ sean verdaderas, la conclusión C también es verdadera. Si existe una interpretación en la que todas las premisas sean verdaderas y la conclusión falsa, entonces el argumento no es válido.

Ejemplos

Ejemplo 1: Modus Ponens

$$\begin{array}{l} p \to q \\ p \\ \hline q \end{array}$$

Para determinar la validez de este argumento, construimos su tabla de verdad:

p	q	$p \to q$	q
1	1	1	1
1	0	0	0
0	1	1	1
0	0	1	0

En este caso, observamos que siempre que p es verdadero y $p \to q$ también es verdadero, q debe ser verdadero. Por lo tanto, el argumento es válido.

Ejemplo 2: Silogismo Hipotético

Consideremos el siguiente argumento:

$$\begin{array}{l} p \to q \\ q \to r \\ \hline p \to r \end{array}$$

Para determinar si este argumento es válido, construimos su tabla de verdad:

p	q	r	$p \to q$	$q \to r$	$p \to r$
1	1	1	1	1	1
1	1	0	1	0	0
1	0	1	0	1	1
1	0	0	0	1	0
0	1	1	1	1	1
0	1	0	1	0	1
0	0	1	1	1	1
0	0	0	1	1	1

En la tabla de verdad, observamos que, en todas las filas en las que tanto $p \to q$ como $q \to r$ son verdaderos, $p \to r$ también es verdadero. Esto significa que siempre que las premisas son verdaderas, la conclusión también es verdadera. Por lo tanto, el argumento es válido.

Este tipo de razonamiento, conocido como **Silogismo Hipotético** se basa en la propiedad transitiva de las implicaciones en la lógica proposicional.

Ejemplo 3: Argumento No Válido

Considere el siguiente argumento:

$$\begin{array}{l} p \to r \\ \neg p \\ \hline \neg r \end{array}$$

Para determinar si este argumento es válido, construimos su tabla de verdad:

p	r	$p \to r$	$\neg p$	$\neg r$
1	1	1	0	0
1	0	0	0	1
0	1	1	1	0
0	0	1	1	1

En este caso, observamos que en la tercera fila, donde p es falso, $p \to r$ es verdadero, y $\neg p$ es verdadero, pero la conclusión $\neg r$ es falsa. Esto muestra que es posible que todas las premisas sean verdaderas mientras que la conclusión sea falsa. Por lo tanto, este argumento no es válido.

En resumen, un argumento es válido si y solo si, en todas las interpretaciones donde las premisas son verdaderas, la conclusión también lo es. Si existe una interpretación en la que las premisas sean verdaderas y la conclusión falsa, entonces el argumento no es válido.

1.4. Clasificación de Fórmulas en Lógica Proposicional

En esta sección, estudiaremos cómo se pueden clasificar distintas fórmulas de la lógica proposicional según sus propiedades lógicas. Esto nos permitirá entender mejor la estructura de los enunciados lógicos y su comportamiento bajo distintas valuaciones.

1.4.1. Satisfactibilidad

En lógica proposicional, una de las propiedades fundamentales de un conjunto de fórmulas es su **satisfactibilidad**. Esta propiedad nos indica si existe al menos una asignación de valores de verdad a las variables proposicionales que haga verdaderas todas las fórmulas de un conjunto dado.

Definición: Satisfactibilidad

Un conjunto de fórmulas $\Gamma = \{\alpha, \beta, \gamma, \ldots\} \subseteq \mathcal{L}(P)$ se dice **satisfactible** si y solo si existe al menos una valuación $\sigma\colon P \to \{0,1\}$ tal que:

$$\hat{\sigma}(\alpha) = \hat{\sigma}(\beta) = \hat{\sigma}(\gamma) = \cdots = 1.$$

En caso contrario, si no existe ninguna valuación que haga verdaderas todas las fórmulas del conjunto simultáneamente, se dice que Γ es **insatisfactible**.

Ejemplos de Satisfactibilidad e Insatisfactibilidad

Para comprender mejor esta definición, consideremos algunos ejemplos concretos:

- $\{p \vee q\}$ es un conjunto satisfactible, ya que, por ejemplo, la valuación $\sigma(p) = 1, \sigma(q) = 0$ hace que la fórmula sea verdadera.

- $\{p \wedge q, p \wedge \neg q\}$ es insatisfactible, porque no hay una valuación que pueda hacer ambas fórmulas verdaderas al mismo tiempo. En efecto, la primera exige que p y q sean verdaderos, mientras que la segunda requiere que p sea verdadero pero q sea falso, lo cual es contradictorio.

Aplicaciones de la Satisfactibilidad

El problema de satisfactibilidad proposicional (SAT) fue el primero en demostrarse **NP-completo** (Teorema de Cook, 1971). Ser NP-completo implica que la resolución de SAT en su forma general puede requerir un tiempo exponencial en función del número de variables, lo que hace que su solución sea computacionalmente difícil a gran escala. in embargo, en la práctica, existen algoritmos avanzados llamados **SAT-solvers** que pueden resolver muchas instancias grandes de manera eficiente mediante heurísticas y optimización.

Tiene múltiples aplicaciones en diversas áreas de la informática y la matemática:

- **Verificación de software y hardware:** Se utiliza en la verificación formal de circuitos digitales y programas para determinar si existen errores lógicos en su funcionamiento.
- **Optimización combinatoria:** La satisfactibilidad booleana es la base de problemas de optimización como el problema del corte mínimo, la asignación de tareas y la generación de horarios.
- **Criptografía y seguridad informática:** Problemas SAT se usan en el análisis de protocolos criptográficos para detectar vulnerabilidades. También se aplican en la recuperación de claves cifradas y en la verificación de la resistencia de sistemas de seguridad.

1.4.2. Tautologías y Contradicciones

Otra clasificación importante en lógica proposicional es la de **tautologías** y **contradicciones**. Estas nociones permiten identificar fórmulas que son siempre verdaderas o siempre falsas, independientemente de la valuación que se elija.

Definición: Tautología y Contradicción

Una fórmula $\alpha \in \mathcal{L}(P)$ es una **tautología** si y solo si es verdadera bajo *todas* las valuaciones posibles, es decir, si para toda valuación $\sigma\colon P \to \{0,1\}$ se cumple que:

$$\hat{\sigma}(\alpha) = 1.$$

Una fórmula $\alpha \in \mathcal{L}(P)$ es una **contradicción** si y solo si es falsa bajo *todas* las valuaciones posibles, es decir, si para toda valuación $\sigma\colon P \to \{0,1\}$ se cumple que:

$$\hat{\sigma}(\alpha) = 0.$$

Ejemplos de Tautologías y Contradicciones

A continuación, se presentan algunos ejemplos para ilustrar estas definiciones:

- $p \vee \neg p$ es una tautología, ya que siempre será verdadera sin importar la valuación de p.
- $p \to (p \vee q)$ es también una tautología, pues siempre que p sea verdadero, entonces $p \vee q$ también lo será.
- $p \wedge \neg p$ es una contradicción, ya que nunca puede ser verdadera, sin importar la valuación de p.

1.4.3. Relación entre Contradicción e Insatisfactibilidad

Una contradicción es una fórmula que es falsa bajo todas las valuaciones posibles. Por lo tanto, un conjunto que contenga al menos una contradicción es insatisfactible, ya que no existe ninguna valuación que lo haga verdadero.

De manera más general, un conjunto de fórmulas es insatisfactible cuando la conjunción de todas sus fórmulas forma una contradicción. Esto significa que no hay una asignación de valores de verdad que satisfaga simultáneamente todas las fórmulas del conjunto.

Ejemplo

El conjunto $\{p \vee q, \neg p, \neg q\}$ es insatisfactible, ya que su conjunción equivale a la contradicción $(p \vee q) \wedge \neg p \wedge \neg q$, la cual es falsa bajo cualquier valuación.

1.5. Equivalencia de fórmulas

Comprender la equivalencia de fórmulas es útil para la simplificación de expresiones lógicas y la verificación de argumentos. Nos permite transformar fórmulas en formas más manejables y demostrar propiedades de los sistemas lógicos.

Informalmente, dos fórmulas $\alpha, \beta \in \mathcal{L}(P)$ se dicen **equivalentes** si tienen la misma tabla de verdad. Formalmente:

Definición: Equivalencia

Dos fórmulas $\alpha, \beta \in \mathcal{L}(P)$ son (semánticamente) **equivalentes**, notado $\alpha \equiv \beta$, si y sólo si

$$\hat{\sigma}(\alpha) = \hat{\sigma}(\beta)$$

para toda valuación $\sigma \colon P \to \{0, 1\}$.

Ejemplo

Demostremos que $\neg(p \vee q) \equiv \neg p \wedge \neg q$ mediante su tabla de verdad.

p	q	$\neg(p \vee q)$	$\neg p \wedge \neg q$
1	1	0	0
1	0	0	0
0	1	0	0
0	0	1	1

Se observa que ambas columnas son iguales, por lo que las fórmulas son equivalentes.

1.5.1. Algunas equivalencias útiles

Las siguientes equivalencias son herramientas esenciales en la manipulación de expresiones lógicas:

Equivalencias útiles

Leyes de De Morgan	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
	$\neg(p \vee q) \equiv \neg p \wedge \neg q$
Asociatividad de $\wedge$ y $\vee$	$p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$
	$p \vee (q \vee r) \equiv (p \vee q) \vee r$
Caracterización alternativa de $\rightarrow$	$p \rightarrow q \equiv \neg p \vee q$

Ejercicio de Equivalencia

¿Es el conectivo de implicación ($\rightarrow$) asociativo?

Para responder esta pregunta, basta con analizar la equivalencia:

$$(p \rightarrow q) \rightarrow r \equiv p \rightarrow (q \rightarrow r)$$

Que podemos verificar si son iguales usando las tablas de verdad.

1.6. Consecuencia Lógica

La noción de **consecuencia lógica** es clave en lógica matemática, ya que permite formalizar cuándo una **conclusión** se **deriva** lógicamente de un **conjunto de premisas**. Esta noción subyace en el concepto de inferencia lógica y es clave para la validez de argumentos.

Sea Γ un conjunto de fórmulas (premisas) y α una fórmula (conclusión). Decimos que α es **consecuencia lógica** de Γ si cada vez que todas las fórmulas en Γ son verdaderas, entonces α también es verdadera. Escribimos esto como:

$$\Gamma \models \alpha.$$

Es decir, si asumimos que las premisas son verdaderas, entonces la conclusión necesariamente también debe serlo. Si encontramos al menos una interpretación en la que todas las premisas son verdaderas y la conclusión es falsa, entonces no hay consecuencia lógica.

En lógica, la **validez** es una propiedad que tienen los argumentos cuando las premisas implican la conclusión. Si la conclusión es una consecuencia lógica de las premisas, se dice que el argumento es **deductivamente válido**. En este contexto, consideraremos estas dos nociones como idénticas.

1.6.1. Ejemplos Clásicos de Consecuencia Lógica

Existen reglas clásicas de inferencia que ejemplifican el concepto de consecuencia lógica. Algunas de ellas son:

- **Modus ponens**: $\{p, p \to q\} \models q$
- **Modus tollens**: $\{\neg q, p \to q\} \models \neg p$
- **Transitividad de la implicación**: $\{p \to q, q \to r\} \models p \to r$

Estos esquemas de inferencia pueden verificarse mediante tablas de verdad como lo hicimos al verificar si un argumento es valido.

1.6.2. Definición Formal

Definición: Consecuencia lógica

Sea Γ un conjunto de fórmulas en un lenguaje lógico $\mathcal{L}(P)$ y α una fórmula en el mismo lenguaje. Decimos que α es **consecuencia lógica** de Γ, denotado $\Gamma \models \alpha$, si y sólo si para toda valuación $\sigma : P \to \{0, 1\}$,

$$\text{si } \hat{\sigma}(\Gamma) = 1, \text{ entonces } \hat{\sigma}(\alpha) = 1.$$

Esto significa que α debe ser verdadero en toda valuación en la que todas las fórmulas de Γ son verdaderas.

Notar que no nos importan aquellas valuaciones σ donde $\hat{\sigma}(\Gamma) \neq 1$. El valor de α en esas valuaciones es irrelevante.

1.6.3. Propiedades Importantes

La consecuencia lógica cumple con varias propiedades fundamentales que nos permiten derivar resultados de manera más estructurada y comprender mejor las relaciones entre distintas proposiciones. A continuación, presentamos algunas propiedades relevantes:

Para todo conjunto de fórmulas Γ y todas las fórmulas α, β se cumplen las siguientes propiedades:

- **Equivalencia lógica:** $\alpha \equiv \beta$ si y sólo si $\{\alpha\} \models \beta$ y $\{\beta\} \models \alpha$. Esto significa que ambas fórmulas tienen el mismo valor de verdad en todas las interpretaciones.
- **Relación con insatisfactibilidad:** $\Gamma \models \alpha$ si y sólo si $\Gamma \cup \{\neg\alpha\}$ es insatisfactible. Esta propiedad es útil para demostrar la validez de inferencias mediante el método de reducción al absurdo.
- **Regla de deducción:** $\Gamma \models \alpha \to \beta$ si y sólo si $\Gamma \cup \{\alpha\} \models \beta$. Es un recurso esencial para derivar conclusiones en sistemas formales.
- **Monotonía:** Si $\Gamma \models \alpha$ entonces $\Gamma \cup \{\beta\} \models \alpha$. Esto significa que al agregar más premisas a Γ, no se pierde la capacidad de inferir α.

Ejemplo de Monotonía con un Conjunto Insatisfactible

Pregunta: Sabemos que $\{p,\ p \to q\} \models q$. Por la propiedad de monotonía de la consecuencia lógica, también se cumple:

$$\{p,\ p \to q,\ \neg q\} \models q.$$

¿Cómo es esto posible?

La respuesta radica en que $\{p, p \to q, \neg q\}$ es un conjunto *insatisfactible*; no existe valuación que haga todas estas fórmulas verdaderas simultáneamente. En lógica proposicional, cuando un conjunto de premisas es insatisfactible, se sigue trivialmente que se puede derivar cualquier conclusión.

- Si un conjunto es insatisfactible, no hay asignaciones de verdad que satisfagan simultáneamente todas sus fórmulas.
- Por definición, en cada valuación que "satisface" el conjunto (no existe ninguna), la conclusión es verdadera de manera vacía o trivial.

Este hecho, aunque pueda parecer extraño, se desprende directamente de nuestras definiciones y pone de manifiesto por qué queremos evitar conjuntos insatisfactibles en el razonamiento lógico, ya que de ellos se podría derivar cualquier cosa.

1.6.4. Ejercicios

Ejercicio 1: Demuestre la relación con insatisfactibilidad: $\Gamma \models \alpha$ si y sólo si $\Gamma \cup \{\neg\alpha\}$ es insatisfactible.

Ejercicio 2: Demuestre la regla de deducción: $\Gamma \models \alpha \rightarrow \beta$ si y sólo si $\Gamma \cup \{\alpha\} \models \beta$.

Ejercicio 3: Explique por qué, a partir de un conjunto de premisas insatisfactible, se puede derivar cualquier conclusión.

1.7. Aplicaciones de Satisfactibilidad

1.7.1. Programación de exámenes universitarios

Se deben programar tres exámenes en dos franjas horarias disponibles: **mañana** ($t1$) y **tarde** ($t2$). Los exámenes son:

- **Matemáticas** ($\boldsymbol{M}$)
- **Física** ($\boldsymbol{P}$)
- **Ciencias de la Computación** ($\boldsymbol{C}$)

Las restricciones son:

1. Cada examen debe programarse en **exactamente una** franja horaria.
2. Los exámenes de Matemáticas y Física **no pueden coincidir** en la misma franja.
3. El examen de Ciencias de la Computación **debe programarse antes** que el de Matemáticas.

Definición de Variables Booleanas

Cada variable X_i indica si el examen X está programado en la franja ti:

- M_1, M_2: Matemáticas en la franja $t1$ o $t2$.
- P_1, P_2: Física en la franja $t1$ o $t2$.
- C_1, C_2: Ciencias de la Computación en la franja $t1$ o $t2$.

Codificación en lógica proposicional

1. Cada examen debe estar en una única franja

$$(M_1 \vee M_2) \wedge \neg(M_1 \wedge M_2)$$
$$(P_1 \vee P_2) \wedge \neg(P_1 \wedge P_2)$$
$$(C_1 \vee C_2) \wedge \neg(C_1 \wedge C_2)$$

2. Matemáticas y Física no pueden coincidir

$$\neg(M_1 \wedge P_1), \quad \neg(M_2 \wedge P_2)$$

3. Ciencias de la Computación antes que Matemáticas

$C_2 \rightarrow \neg M_1$.
(Si C está en $t2$, M no puede estar en $t1$)

Solución SAT

Una asignación válida que cumple todas las restricciones es:

$$C_1 = \text{true}, \quad C_2 = \text{false}$$
$$M_2 = \text{true}, \quad M_1 = \text{false}$$
$$P_2 = \text{true}, \quad P_1 = \text{false}$$

Interpretación de la Solución:

- **Ciencias de la Computación** $\rightarrow$ Mañana ($T1$)
- **Matemáticas** $\rightarrow$ Tarde ($T2$)
- **Física** $\rightarrow$ Tarde ($T2$)

La asignación respeta todas las restricciones, garantizando que **Ciencias de la Computación ocurre antes que Matemáticas** y que **Matemáticas y Física no coinciden**.

1.7.2. Juego del Gato

En este problema, modelamos el juego del Gato (*Tic-Tac-Toe*) como un problema de **satisfactibilidad (SAT)** en lógica proposicional. El objetivo es verificar si el jugador X tiene una **jugada ganadora** en su próximo turno.

Definición del Problema

Dado un tablero 3×3, donde algunas casillas ya están ocupadas por X o O, queremos determinar si existe una celda vacía donde X pueda jugar para ganar inmediatamente.

Representación del Tablero

Definimos el estado de cada celda (i, j) usando dos variables booleanas:

- $X_{i,j}$: La celda (i, j) está ocupada por X.
- $O_{i,j}$: La celda (i, j) está ocupada por O.

Las celdas vacías no se modelan explícitamente, sino que se deducen por la ausencia de $X_{i,j}$ y $O_{i,j}$.

Restricciones del Problema

1. Cada celda debe estar en exactamente un estado, lo que se expresa con la siguiente restricción:

$$(\neg X_{i,j} \wedge \neg O_{i,j}) \vee ((X_{i,j} \vee O_{i,j}) \wedge \neg(X_{i,j} \wedge O_{i,j})), \quad \forall i, j. \tag{1.1}$$

   Esta ecuación garantiza que cada celda la celda esté vacía o esté ocupada por X o O, pero no por ambos a la vez.

2. Condición 1: X debe jugar en una celda vacía

   Para que X pueda realizar una jugada, debe colocar su marca en una celda que aún no esté ocupada. Dado que las celdas vacías se identifican como aquellas donde $X_{i,j}$ y $O_{i,j}$ son falsas, la condición de jugada se expresa como:

$$\bigvee_{i,j} ((\neg X_{i,j} \wedge \neg O_{i,j}) \rightarrow X_{i,j}). \tag{1.2}$$

   Esta ecuación establece que si una celda no está ocupada por X ni O, entonces X puede colocar su marca en ella.

3. Condición 2: X debe completar una fila, columna o diagonal.

   - **Fila completa:**

$$(X_{1,1} \wedge X_{1,2} \wedge X_{1,3}) \vee (X_{2,1} \wedge X_{2,2} \wedge X_{2,3}) \vee (X_{3,1} \wedge X_{3,2} \wedge X_{3,3}). \tag{1.3}$$

   - **Columna completa:**

$$(X_{1,1} \wedge X_{2,1} \wedge X_{3,1}) \vee (X_{1,2} \wedge X_{2,2} \wedge X_{3,2}) \vee (X_{1,3} \wedge X_{2,3} \wedge X_{3,3}). \tag{1.4}$$

   - **Diagonal completa:**

$$(X_{1,1} \wedge X_{2,2} \wedge X_{3,3}) \vee (X_{1,3} \wedge X_{2,2} \wedge X_{3,1}). \tag{1.5}$$

Ejemplo de Tablero y Evaluación

Supongamos que el tablero actual es:

X	X	$\emptyset$
O	O	X
$\emptyset$	O	$\emptyset$

En este caso: - X puede jugar en la celda $(1, 3)$ y completar la primera fila. - Como la fórmula es satisfactible, se confirma que X tiene una jugada ganadora.

1.8. Deducción Natural

Métodos de Demostración en Lógica Proposicional

Cuando queremos demostrar que una afirmación es verdadera, podemos utilizar distintos enfoques. En lógica proposicional, los dos principales métodos son:

- **Deducción Natural:** Un sistema de reglas que nos permite derivar proposiciones paso a paso a partir de premisas iniciales.
- **Tablas de Verdad:** Un método exhaustivo que evalúa todas las combinaciones posibles de valores de verdad para determinar si una proposición es válida en todos los casos.

Ambos métodos tienen ventajas y limitaciones, y en esta sección nos centraremos en la **deducción natural**.

1.8.1. ¿Qué es la Deducción Natural?

La **deducción natural** es un sistema formal que imita la forma en que razonamos de manera estructurada: partiendo de **premisas**, aplicamos **reglas de inferencia** para derivar conclusiones que serán correctas siempre que las premisas lo sean. Este proceso exige un enfoque **creativo y metódico**, parecido a la resolución de problemas en programación o demostración de teoremas, pues no siempre es evidente qué regla usar ni en qué orden.

A diferencia de las **tablas de verdad**, que comprueban la validez de una proposición al evaluar todas las combinaciones posibles de valores de verdad, la deducción natural construye el razonamiento paso a paso, ofreciendo una comprensión detallada de cómo se encadenan las inferencias hasta llegar a la conclusión.

Ventajas y Desventajas de la Deducción Natural

- **Ventajas:**
  - Permite demostrar teoremas de manera sistemática y organizada.

  - Es aplicable a lógica de primer orden y sistemas más complejos.
  - Es proceso generativo que permite descubrir nuevas conclusiones.
  - Es más eficiente que las tablas de verdad para proposiciones con muchas variables.

- **Desventajas:**
  - Requiere práctica y habilidad para construir demostraciones.
  - No proporciona una forma directa de comprobar si una proposición es una tautología.

A continuación, exploraremos algunas reglas esenciales de la deducción natural, junto con ejemplos prácticos.

1.8.2. Reglas Básicas de la Deducción Natural

Regla de la Conjunción ($\wedge$)

Ejemplo: Supongamos que sabemos que:

- **Pedro tiene una bicicleta.**
- **Pedro tiene un casco.**

De esto, podemos concluir:

Pedro tiene una bicicleta y un casco.

Esto es un caso de la **introducción de la conjunción**.

Regla de Introducción de la Conjunción

Si tenemos dos afirmaciones ϕ y ψ, podemos combinarlas con $\wedge$.

$$\frac{\phi \quad \psi}{\phi \wedge \psi}(\wedge i)$$

Sobre las líneas están las premisas de la regla. Abajo de la línea se encuentra la conclusión. A la derecha de la regla tenemos el nombre de la regla abreviado.

Ejemplo: Si sabemos que:

Ana es ingeniera y Ana es programadora.

Podemos concluir:

Ana es programadora.

Regla de Eliminación de la Conjunción

Si sabemos que algo es cierto en conjunto $(\phi \wedge \psi)$, podemos extraer cualquiera de sus partes.

$$\frac{\phi \wedge \psi}{\phi}(\wedge e_1) \quad \frac{\phi \wedge \psi}{\psi}(\wedge e_2)$$

Ejercicio: Queremos probar:

$$p \wedge q, r \vdash q \wedge r$$

Representación gráfica de la demostración:

$$\dfrac{\dfrac{p \wedge q}{q}[\wedge e_2] \qquad r}{q \wedge r}[\wedge i]$$

Regla de la Doble Negación

Motivación: Si alguien dice:

"**No es falso que Pedro es honesto.**"

¿Significa esto que Pedro es honesto? Sí. Esto es un caso de eliminación de la doble negación.

Regla de la Doble Negación

Una afirmación es equivalente a su doble negación.

$$\frac{\neg\neg\phi}{\phi}(\neg\neg e) \quad \frac{\phi}{\neg\neg\phi}(\neg\neg i)$$

Regla de Eliminación de la Implicación (Modus Ponens)

Ejemplo: Sabemos que:

- Si estudio, aprobaré el examen.
- Estudié.

Podemos concluir:

Aprobaré el examen.

Esto es un caso del **Modus Ponens**.

Modus Ponens ($\to e$)

Si tenemos ϕ y $\phi \to \psi$, podemos deducir ψ.

$$\frac{\phi \quad \phi \to \psi}{\psi}(\to e)$$

Modus Tollens (MT)

Si tenemos $\phi \to \psi$ y $\neg\psi$, podemos deducir $\neg\phi$.

$$\frac{\phi \to \psi \quad \neg\psi}{\neg\phi}(MT)$$

Regla de Introducción de la Implicación

Ejemplo: Supongamos que queremos demostrar la siguiente afirmación:

Si Juan estudia, aprobará el examen.

¿Cómo lo haríamos? Podemos asumir temporalmente que Juan estudia, y si logramos demostrar que en ese caso aprueba el examen, entonces la afirmación original es válida.

Regla de Introducción de la Implicación

Para probar $\phi \to \psi$, asumimos temporalmente ϕ y demostramos ψ.

$$\frac{\begin{array}{|c|}\hline \phi \\ \vdots \\ \psi \\ \hline \end{array}}{\phi \to \psi}(\to i)$$

La regla expresa que para probar $\phi \to \psi$ debemos asumir temporalmente ϕ y entonces probar ψ. En la prueba de ψ se pueden usar cualquiera de las fórmulas obtenidas hasta el momento, como premisas y conclusiones provisionales.

Ejercicio: Queremos demostrar

$$p \to q \vdash \neg q \to \neg p$$

La mecánica de la regla es más complicada que las anteriores.

1	$p \to q$	premisa
2	$\neg q$	supuesto
3	$\neg p$	MT 1,2
4	$\neg q \to \neg p$	$\to i$ 2–3

El cuadro sirve para delimitar el alcance del supuesto temporal $\neg q$. No podemos usar suposiciones dentro del cuadro en reglas fuera del cuadro.

Reglas de la Disyunción ($\vee$)

Introducción de la Disyunción

Podemos introducir una disyunción a partir de cualquier proposición:

$$\frac{\phi}{\phi \vee \psi}(\vee i_1) \qquad \frac{\psi}{\phi \vee \psi}(\vee i_2)$$

Supongamos que queremos probar una proposición χ asumiendo que $\phi \vee \psi$. Puesto que no sabemos que argumento de la disyunción es verdadero, debemos obtener dos pruebas por separado que deberemos combinar en un solo argumento:

Eliminación de la Disyunción

Si queremos probar χ asumiendo $\phi \vee \psi$, necesitamos dos pruebas separadas para cada caso:

$$\dfrac{\phi \vee \psi \qquad \boxed{\begin{array}{c}\phi \\ \vdots \\ \chi\end{array}} \qquad \boxed{\begin{array}{c}\psi \\ \vdots \\ \chi\end{array}}}{\chi}(\vee e)$$

Regla de la Negación

Una **contradicción** en lógica proposicional se expresa como $\phi \wedge \neg\phi$, donde ϕ es cualquier fórmula. A esta contradicción se le denomina $\bot$. La regla de la negación describe cómo usar una contradicción y cómo introducir una negación al demostrar que un supuesto conduce a $\bot$.

Regla de la Negación

$$\dfrac{\phi \quad \neg\phi}{\bot}[\neg e] \qquad \dfrac{\bot}{\phi}[\bot e] \qquad \dfrac{\begin{array}{c}\phi \\ \vdots \\ \bot\end{array}}{\neg\phi}[\neg i]$$

- $\neg e$ (eliminación de la negación) nos dice que si ϕ y $\neg\phi$ se derivan simultáneamente, obtenemos $\bot$.

- $\bot e$ nos indica que, a partir de $\bot$, podemos concluir *cualquier* fórmula (principio de explosión).

- $\neg i$ (introducción de la negación) expresa que, si al suponer ϕ derivamos una contradicción ($\bot$), entonces concluimos $\neg\phi$.

Ejercicio Final: Demostración con Disyunciones y Conjunciones

Veamos un ejemplo que combina varias reglas de inferencia para demostrar:

$$p \wedge (q \vee r) \ \vdash \ (p \wedge q) \ \vee \ (p \wedge r).$$

La idea general es extraer p y la disyunción $q \vee r$ de la premisa, y luego considerar **dos casos** (uno donde q es verdadero, y otro donde r es verdadero) para demostrar la disyunción final. La prueba se detalla a continuación:

1	$p \wedge (q \vee r)$	*premisa*
2	p	$\wedge e_1$ (de 1)
3	$q \vee r$	$\wedge e_2$ (de 1)
4	q	*supuesto*
5	$p \wedge q$	$\wedge i$ (de 2,4)
6	$(p \wedge q) \vee (p \wedge r)$	$\vee i_1$ (de 5)
7	r	*supuesto*
8	$p \wedge r$	$\wedge i$ (de 2,7)
9	$(p \wedge q) \vee (p \wedge r)$	$\vee i_2$ (de 8)
10	$(p \wedge q) \vee (p \wedge r)$	$\vee e$ (3, 4–6, 7–9)

Conclusión

La deducción natural es más que un conjunto de reglas: se trata de un sistema de prueba que garantiza, por un lado, la **solidez** (no se puede demostrar algo falso) y, por otro, la **completitud** (todo lo que es válido puede probarse dentro del sistema), al menos en el contexto de la lógica proposicional. Estas dos propiedades son cruciales para confiar en cualquier sistema lógico: nos aseguran que no probaremos fórmulas falsas y que no dejaremos sin demostrar las verdaderas.

Preguntas para Reflexionar

- **Solidez (Soundness):** Si podemos derivar α a partir de Γ mediante deducción natural, entonces α es una consecuencia lógica de Γ en todas las interpretaciones. En términos formales:
$$\Gamma \vdash \alpha \quad \Longrightarrow \quad \Gamma \models \alpha.$$
- **Completitud (Completeness):** Si α es una consecuencia lógica de Γ (es decir, verdadera en todas las interpretaciones donde Γ lo es), entonces también podemos derivar α dentro del sistema de deducción natural. En términos formales:
$$\Gamma \models \alpha \quad \Longrightarrow \quad \Gamma \vdash \alpha.$$

Este **enfoque riguroso y ordenado** para el razonamiento lógico tiene implicaciones tanto en el ámbito académico—donde se emplea para demostrar teoremas y validar argumentos formales—como en la vida diaria, al entrenar nuestras habilidades de *pensamiento analítico*. Aun si no aplicamos conscientemente el lenguaje lógico en cada situación, los principios de consistencia y completitud ofrecen un horizonte de *certeza* para saber que los razonamientos formales se construyen de forma sólida y exhaustiva.

Capítulo 2

Lógica de Predicados

2.1. Introducción a Lógica de Predicados

2.1.1. Limitaciones de la Lógica Proposicional

La lógica proposicional es una herramienta poderosa para modelar razonamientos formales. Sin embargo, tiene ciertas limitaciones, ya que no puede expresar relaciones más complejas entre objetos ni generalizaciones. Consideremos el siguiente razonamiento:

> "Todas las personas son mortales. Sócrates es una persona. Por lo tanto, Sócrates es mortal."

Este argumento parece válido, pero en lógica proposicional no tenemos forma de expresar la idea de que "todas las personas" tienen la propiedad de ser mortales.

Para capturar este tipo de razonamiento necesitamos:

- Describir **propiedades de** o **relaciones entre** los elementos del dominio en cuestión. Por ejemplo:
$$persona(x), \quad mortal(x) \tag{2.1}$$
- **Cuantificar** sobre los elementos del dominio, permitiendo generalizaciones como:
$$\forall x \ (persona(x) \rightarrow mortal(x)) \tag{2.2}$$

Así, podemos representar el argumento formalmente en lógica de predicados:

$$\begin{array}{ll} \forall x.\ persona(x) \rightarrow mortal(x) & \text{(Premisa general)} \\ persona(\text{Sócrates}) & \text{(Premisa específica)} \\ mortal(\text{Sócrates}) & \text{(Conclusión)} \end{array}$$

Este tipo de lógica, llamada **lógica de predicados** o **lógica de primer orden**, permite expresar con mayor precisión una variedad de razonamientos que en lógica proposicional serían imposibles de representar.

Ejemplo con una Relación Binaria

No solo podemos representar propiedades individuales, sino también relaciones entre elementos. Consideremos el razonamiento:

> "Si un usuario sigue a otro en una red social, entonces puede ver sus publicaciones. Alice sigue a Bob. Por lo tanto, Alice puede ver las publicaciones de Bob."

Este argumento puede expresarse en lógica de predicados como:

$$\begin{array}{ll} \forall x, y.\ sigue(x, y) \rightarrow puedeVerPublicaciones(x, y) & \text{(Regla general)} \\ sigue(\text{Alice}, \text{Bob}) & \text{(Hecho específico)} \\ puedeVerPublicaciones(\text{Alice}, \text{Bob}) & \text{(Conclusión)} \end{array}$$

Diferencias Clave con la Lógica Proposicional

Para resumir, la lógica de predicados introduce dos elementos clave que la diferencian de la lógica proposicional:

1. **Predicados:** Nos permiten expresar propiedades y relaciones entre objetos.
2. **Cuantificadores:** Nos permiten hacer afirmaciones sobre todos los elementos de un dominio ($\forall$) o sobre la existencia de al menos un elemento que cumple una propiedad ($\exists$).

Estos elementos hacen que la lógica de predicados sea mucho más expresiva y útil para modelar razonamientos matemáticos y computacionales.

2.1.2. Sintaxis de la Lógica de Predicados

Las fórmulas de la lógica de predicados se construyen a partir de un vocabulario compuesto por:

- **Símbolos de relación** $P, Q, R, \ldots$, cada uno con una **aridad** determinada, es decir, el número de argumentos que toma. Si un símbolo de relación R tiene aridad n, se denota como $R(x_1, x_2, \ldots, x_n)$.
- **Cuantificadores**: universal ($\forall$) y existencial ($\exists$), que permiten expresar generalizaciones y la existencia de elementos en el dominio.
- **Variables**: $x, y, z, \ldots$, que representan elementos arbitrarios del dominio. Dos variables pueden referirse al mismo objeto si y solo si se cumple $x = y$.
- **Conectivos lógicos**: $\neg, \wedge, \vee, \rightarrow$

Ejemplos de Propiedades de Relaciones

Las relaciones pueden cumplir distintas propiedades fundamentales:

- **Reflexividad**: $\forall x\ R(x, x)$ (ejemplo: mayor o igual que)
- **Simetría**: $\forall x \forall y (R(x, y) \rightarrow R(y, x))$ (ejemplo: igualdad)
- **Antisimetría**: $\forall x \forall y ((R(x, y) \wedge R(y, x)) \rightarrow x = y)$ (ejemplo: mayor estricto que)
- **Transitividad**: $\forall x \forall y \forall z ((R(x, y) \wedge R(y, z)) \rightarrow R(x, z))$

Alcance de los Cuantificadores

Los cuantificadores determinan una "zona de influencia"de las variables, llamada **rango** o **alcance** del cuantificador:

$$R(z) \wedge \exists y\ \forall x (S(x) \implies T(x, y)) \tag{2.3}$$

- Las ocurrencias verdes de x están dentro del alcance del cuantificador universal y son çapturadas"por éste.
- La ocurrencia roja de y está dentro del alcance del cuantificador universal, pero no es capturada por éste.
- La ocurrencia roja de y está dentro del alcance del cuantificador existencial y sí es capturada por éste.
- La variable z es **libre** porque no está en el alcance de ningún cuantificador.

Para entender mejor la importancia del alcance de los cuantificadores, consideremos dos expresiones aparentemente similares, pero con significados distintos:

- $\forall x \exists y\, T(x, y)$ (Para todo x, existe al menos un y que cumple la relación $T(x, y)$).
- $\exists y \forall x\, T(x, y)$ (Existe un único y tal que para todo x, se cumple la relación $T(x, y)$).

2.2. Semántica de la Lógica de Predicados

En lógica, no solo nos interesa la *forma* de las fórmulas (su *estructura sintáctica*), sino también su *significado* (su *interpretación* en un dominio). Mientras que en la lógica proposicional basta con asignar valores de verdad a cada variable proposicional y evaluar la fórmula mediante conectivos lógicos, en la **lógica de primer orden** debemos dar significado a *términos*, *funciones* y *relaciones* sobre algún conjunto de referencia, llamado **dominio**.

2.2.1. Estructuras e Interpretaciones

Para capturar el significado de las fórmulas de primer orden, trabajamos con objetos llamados **estructuras**:

$$\mathcal{A} = \big(A, P^{\mathcal{A}}, Q^{\mathcal{A}}, R^{\mathcal{A}}, \dots\big)$$

donde:

- A es el **dominio** (o universo) de la estructura. Todas las variables de la fórmula tomarán valores en este conjunto.
- $P^{\mathcal{A}}, Q^{\mathcal{A}}, R^{\mathcal{A}}, \dots$ son las **interpretaciones** de los símbolos de relación (o predicados) del lenguaje. Por ejemplo, si nuestro lenguaje incluye un predicado binario P, entonces $P^{\mathcal{A}} \subseteq A \times A$ especifica para qué pares de elementos de A se considera verdadero $P(x, y)$.
- Si el lenguaje tiene símbolos de función f, g, etc., sus interpretaciones $f^{\mathcal{A}}, g^{\mathcal{A}}, \dots$ son funciones definidas sobre A. Por ejemplo, si f es una función de n variables, entonces $f^{\mathcal{A}} : A^n \to A$.
- Si el lenguaje incluye símbolos de constante como c, su interpretación $c^{\mathcal{A}}$ es un elemento concreto de A.

Para evaluar una fórmula con variables *libres*, necesitamos además una **valuación** σ, que es una función:

$$\sigma : \{\text{variables}\} \longrightarrow A.$$

La valuación asigna a cada variable un elemento concreto del dominio A. De esta forma, podemos "sustituir" cada variable por su valor en A y verificar si una fórmula resulta verdadera o falsa bajo tal asignación.

2.2.2. Ejemplos de Estructuras

Ejemplo 1: Grafos

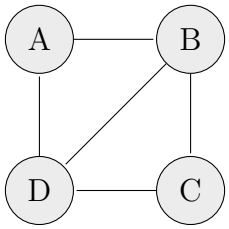
Sea $H = (V, E)$ un grafo en el sentido usual (conjunto de vértices V y conjunto de aristas E). Podemos representarlo como la estructura:

$$\mathcal{G} = (G, E^{\mathcal{G}})$$

donde:

- G es el conjunto de vértices. Este será el *dominio* de nuestra estructura.
- $E^{\mathcal{G}} \subseteq G \times G$ es la *interpretación* del símbolo de relación binaria E. Para $(u, v) \in E^{\mathcal{G}}$ decimos que hay arista entre u y v.

Ejemplo concreto. Considere el siguiente grafo no dirigido con vértices A, B, C, D y aristas $\{A, B\}$, $\{B, C\}$, $\{C, D\}$, $\{D, A\}$, $\{B, D\}$.



Interpretación como estructura:

- El *dominio* de la estructura es $G = \{A, B, C, D\}$.
- La *interpretación* de E se describe como:

  $$E^{\mathcal{G}} = \{(A, B), (B, A), (B, C), (C, B), (C, D), (D, C), (D, A), (A, D), (B, D), (D, B)\}.$$

  En otras palabras, si $(u, v) \in E^{\mathcal{G}}$, hay arista entre u y v.

Ejemplo 2: Números Naturales

Consideremos el lenguaje de la aritmética de Peano, con símbolos de constante 0, 1, la función sucesor $s(\cdot)$, las operaciones $+$ y $\cdot$, y la relación $<$, entre otros. Podemos representar a los números naturales mediante la estructura:

$$\mathcal{N} = \left(\mathbb{N},\, 0^{\mathbb{N}},\, 1^{\mathbb{N}},\, s^{\mathbb{N}},\, +^{\mathbb{N}},\, \cdot^{\mathbb{N}},\, <^{\mathbb{N}}\right)$$

donde:

- El dominio es $A = \mathbb{N}$, el conjunto de los números naturales.

- $0^{\mathbb{N}}$ y $1^{\mathbb{N}}$ son la interpretación de las constantes 0 y 1, respectivamente.
- $s^{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ es la función sucesor, que asigna a cada natural n el valor $n + 1$.
- $+^{\mathbb{N}} : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ y $\cdot^{\mathbb{N}} : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$ son las interpretaciones de la suma y la multiplicación, respectivamente.
- $<^{\mathbb{N}} \subseteq \mathbb{N} \times \mathbb{N}$ es la relación usual de orden sobre los naturales.

2.2.3. Definición Formal de Satisfacción

Dado un lenguaje de primer orden y una estructura $\mathcal{A}$ para dicho lenguaje, decimos que *una fórmula* ϕ es verdadera o se **satisface** en $\mathcal{A}$ bajo la valuación σ, y escribimos:

$$(\mathcal{A}, \sigma) \models \phi,$$

si al "interpretar" todos los símbolos de relación y función como indica $\mathcal{A}$, y al asignar las variables según σ, la fórmula ϕ resulta verdadera.

La definición de "$\models$" se da inductivamente según la forma de la fórmula ϕ. Algunos casos importantes:

$(\mathcal{A}, \sigma) \models P(x_1, \ldots, x_n)$	si y solo si $\big(\sigma(x_1), \ldots, \sigma(x_n)\big) \in P^{\mathcal{A}}$,
$(\mathcal{A}, \sigma) \models \phi \vee \psi$	si y solo si $(\mathcal{A}, \sigma) \models \phi$ o $(\mathcal{A}, \sigma) \models \psi$,
$(\mathcal{A}, \sigma) \models \neg\phi$	si y solo si $(\mathcal{A}, \sigma) \not\models \phi$,
$(\mathcal{A}, \sigma) \models \forall x\, \phi(x)$	si y solo si $(\mathcal{A}, \sigma[x/a]) \models \phi(x)$ para todo $a \in A$,

donde $\sigma[x/a]$ es una nueva valuación que coincide con σ en todas las variables excepto en x, a la que se le asigna el valor a.

- **Fórmulas cerradas (oraciones)**: Si ϕ no tiene variables libres, no es necesario especificar una valuación σ, pues no hay variables por asignar. En ese caso, escribimos simplemente $\mathcal{A} \models \phi$ para indicar que la estructura $\mathcal{A}$ satisface la oración ϕ.

La semántica de la lógica de predicados nos dice cómo interpretar y evaluar la veracidad de fórmulas de primer orden en dominios concretos. Una *estructura* fija tanto el conjunto de referencia (dominio) como la interpretación de los símbolos del lenguaje (relaciones, funciones, constantes). Para las variables libres, definimos una *valuación* que determina a qué elemento del dominio corresponde cada variable.

2.2.4. Ejemplos de Satisfacción

Ejemplo 1: Propiedad sobre una relación

Considera la fórmula:

$$\forall x\, \forall y\, \big(R(x, y) \to \exists z\, \big(R(x, z) \wedge R(z, y)\big)\big).$$

Esta fórmula afirma, en términos informales, que "si hay relación entre x e y, entonces existe un z que se relaciona con ambos".

En el dominio de los **números reales** con $R(x, y)$ interpretado como $x < y$, la afirmación puede leerse así: "si un número x es menor que un número y, entonces existe otro número z tal que $x < z$ y $z < y$". Esto es verdadero en $\mathbb{R}$, porque siempre podemos encontrar un número intermedio entre x e y.

Pregunta: ¿Sucede lo mismo si cambiamos el dominio de discurso a los **enteros** $\mathbb{Z}$ con la misma interpretación $(<)$?

Ejemplo 2: Asignaciones específicas

Sea la fórmula $R(x, y)$ en el dominio de los **reales**, con $R(x, y)$ interpretado como $x < y$. Bajo la valuación σ definida por $\sigma(x) = 1$ y $\sigma(y) = 2$, tenemos:

$$(\mathcal{A}, \sigma) \models R(x, y) \quad \iff \quad 1 < 2.$$

Dado que $1 < 2$ es verdadero, la fórmula se satisface.

Ejemplo 3: Propiedades en Grafos

Consideremos una estructura de grafo no dirigido $\mathcal{G} = (V, E)$, donde V es el conjunto de vértices y $(u, v) \in E$ representa que hay una arista (sin dirección) entre u y v.

- La fórmula
$$\forall x \, \forall y \, \big(E(x, y) \to E(y, x)\big)$$
expresa que el grafo es **no dirigido**. Si (x, y) es una arista, entonces (y, x) también lo es. Esta fórmula se satisface en todo grafo no dirigido, pero no en uno dirigido.

- Otra propiedad interesante es la **transitividad**:
$$\forall x \, \forall y \, \forall z \, \big(\big(E(x, y) \wedge E(y, z)\big) \to E(x, z)\big).$$
Si hay una arista de x a y y de y a z, entonces debe existir una arista de x a z. - Esto es cierto, por ejemplo, en *cliques completas* (todos los vértices están conectados con todos) y en cerraduras transitivas de un grafo. - En un grafo cualquiera, no suele cumplirse.

2.3. Equivalencia de Fórmulas

Definición:

Dos fórmulas cerradas ϕ y ψ se dicen **equivalentes**, notado como $\phi \equiv \psi$, si y solo si para cada estructura $\mathcal{A}$ se cumple:

$$\mathcal{A} \models \phi \quad \text{si y solo si} \quad \mathcal{A} \models \psi. \tag{2.4}$$

Es decir, ambas fórmulas tienen el mismo valor de verdad en todas las interpretaciones posibles.

2.3.1. Ejemplos de equivalencia

Algunas equivalencias fundamentales en lógica de predicados incluyen:

- $\neg\exists x.P(x) \equiv \forall x.\neg P(x)$ (Leyes de De Morgan para cuantificadores).
- $\neg\forall x.P(x) \equiv \exists x.\neg P(x)$.
- $\forall x.(P(x) \land Q(x)) \equiv (\forall x.P(x)) \land (\forall x.Q(x))$ (Distribución de cuantificadores).
- $\exists x.(P(x) \lor Q(x)) \equiv (\exists x.P(x)) \lor (\exists x.Q(x))$ (Distribución de cuantificadores).

2.3.2. Demostraciones

En la lógica de predicado no podemos comprobar equivalencias con una tabla de verdad-como hacíamos en la lógica proposicional-ya que no hay proposiciones a partir de las cuales construir una tabla. Entonces, razonemos.

Leyes de De Morgan para cuantificadores **Demostración de** $\neg\exists x.P(x) \equiv \forall x.\neg P(x)$:

Si $\neg\exists x.P(x)$ es verdadero, significa que no hay ningún x para el cual $P(x)$ sea verdadero, lo que equivale a decir que para todo x, $P(x)$ es falso, es decir, $\forall x.\neg P(x)$.

Recíprocamente, si $\forall x.\neg P(x)$ es verdadero, entonces no hay ningún x donde $P(x)$ sea verdadero, lo que implica $\neg\exists x.P(x)$.

Demostración de $\neg\forall x.P(x) \equiv \exists x.\neg P(x)$:

Si $\neg\forall x.P(x)$ es verdadero, significa que no es cierto que $P(x)$ sea verdadero para todos los valores de x, es decir, existe al menos un x para el cual $P(x)$ es falso, lo que equivale a $\exists x.\neg P(x)$.

La demostración inversa sigue la misma lógica.

Distribución de cuantificadores **Demostración de** $\forall x.(P(x) \land Q(x)) \equiv (\forall x.P(x)) \land (\forall x.Q(x))$:

Dirección ($\Rightarrow$): Suponemos que $\forall x.(P(x) \land Q(x))$ es verdadero. Esto significa que para cada x, tanto $P(x)$ como $Q(x)$ son verdaderos. Entonces, en particular, podemos afirmar que:

$$\forall x.P(x) \quad \text{y} \quad \forall x.Q(x)$$

por lo que $(\forall x.P(x)) \land (\forall x.Q(x))$ también es verdadero.

Dirección ($\Leftarrow$): Si $(\forall x.P(x)) \land (\forall x.Q(x))$ es verdadero, significa que para cada x, $P(x)$ es verdadero y $Q(x)$ es verdadero, lo que implica que $P(x) \land Q(x)$ es verdadero para cada x, es decir:

$$\forall x.(P(x) \land Q(x))$$

Demostración de $\exists x.(P(x) \vee Q(x)) \equiv (\exists x.P(x)) \vee (\exists x.Q(x))$:

Dirección ($\Rightarrow$): Si $\exists x.(P(x) \vee Q(x))$ es verdadero, significa que existe al menos un x_0 tal que $P(x_0) \vee Q(x_0)$ es verdadero. Esto implica que al menos uno de los dos ($P(x_0)$ o $Q(x_0)$) es verdadero. En cualquiera de los casos, se cumple $\exists x.P(x)$ o $\exists x.Q(x)$, lo que implica que $(\exists x.P(x)) \vee (\exists x.Q(x))$ es verdadero.

Dirección ($\Leftarrow$): Si $(\exists x.P(x)) \vee (\exists x.Q(x))$ es verdadero, significa que existe un x_1 tal que $P(x_1)$ es verdadero, o bien existe un x_2 tal que $Q(x_2)$ es verdadero. En ambos casos, podemos decir que existe un x tal que $P(x) \vee Q(x)$ es verdadero, es decir, $\exists x.(P(x) \vee Q(x))$.

2.3.3. Satisfacibilidad

Definición:

Una fórmula ϕ es **satisfacible** si existe una estructura $\mathcal{A}$ tal que $\mathcal{A} \models \phi$, es decir, si al menos hay una interpretación en la que la fórmula es verdadera.

Decidibilidad de la Satisfacibilidad

La satisfacibilidad es un concepto fundamental en lógica y computación. Su complejidad depende del sistema lógico en el que se enmarca:

- En **lógica proposicional**, decidir si una fórmula es satisfacible es un problema NP-completo. Esto significa que encontrar la solución puede requerir tiempo exponencial.
- En **lógica de primer orden**, el problema de satisfacibilidad es **indecidible**, según el Teorema de Church-Turing (1936). No existe un algoritmo general que pueda determinar, en todos los casos, si una fórmula de primer orden es satisfacible.

Este resultado tiene implicaciones profundas en la computabilidad y la teoría de modelos. Por ejemplo, no se puede construir un procedimiento automático que siempre determine si una afirmación matemática arbitraria es verdadera o falsa.

2.3.4. Consecuencia Lógica

Diremos que ϕ es una consecuencia lógica de Γ si todo modelo de Γ es un modelo de ϕ.

Consecuencia lógica

$\Gamma \models \phi$ quiere decir que para todo modelo $\mathcal{M}$, si $\mathcal{M} \models \Gamma$ entonces $\mathcal{M} \models \phi$

2.4. Inferencia en Lógica de Predicados

La inferencia en lógica de predicados permite derivar conclusiones a partir de un conjunto de fórmulas bien formadas (fbf) llamadas **premisas**. Aplicando reglas de inferencia, obtenemos una nueva fbf, la **conclusión**.

A diferencia de la lógica proposicional, la lógica de predicados introduce cuantificadores y variables, lo que la hace más expresiva. Como la lógica proposicional es un subconjunto de la lógica de predicados, sus reglas de inferencia también se aplican aquí. Además, existen reglas adicionales para manipular cuantificadores, permitiendo inferencias sobre conjuntos de elementos.

A continuación, exploramos las reglas fundamentales de inferencia en lógica de predicados con ejemplos ilustrativos.

2.4.1. Eliminación de cuantificador universal

Intuición: Si $\phi(x)$ se cumple para todo x, podemos concluir que $\phi(x)$ se cumple para cualquier t dado.

$$\frac{\forall x \phi(x)}{\phi(t)} [\forall x\ e]$$

Ejemplo

$F(t), \forall x(F(x) \to \neg M(x)) \vdash \neg M(t)$

1	$F(t)$	premisa
2	$\forall x(F(x) \to \neg M(x))$	premisa
3	$F(t) \to \neg M(t)$	$\forall x\ e$ 2
4	$\neg M(t)$	$\to e$ 3,1

2.4.2. Introducción de cuantificador universal

Intuición: Podemos concluir que ϕ se cumple para todo x si elegimos una variable x_0 arbitraria y nueva y demostramos $\phi(x)$.

$$\frac{\begin{array}{c} x_0 \\ \vdots \\ \phi(x_0) \end{array}}{\forall x \phi} [\forall x\ i]$$

Ejemplo

$$\forall x(P(x) \to Q(x)), \forall x P(x) \vdash \forall x Q(x)$$

1		$\forall x(P(x) \to Q(x))$	premisa
2		$\forall x P(x)$	premisa
3	x_0	$P(x_0) \to Q(x_0)$	$\forall x$ *e* 1
4		$P(x_0)$	$\forall x$ *e* 2
5		$Q(x_0)$	$\to$ *e* 3,4
6		$\forall x Q(x)$	$\forall x$ *i* 3–5

2.4.3. Reglas para la cuantificación existencial

Introducción del cuantificador existencial: Si podemos demostrar que un objeto específico t cumple una propiedad $\phi(t)$, entonces podemos concluir que **existe** algún x que también la cumple, es decir, $\exists x \phi(x)$.

$$\frac{\phi(t)}{\exists x \phi}[\exists x\ i]$$

Ejemplo: Si sabemos que "Sócrates es mortal" (M(Sócrates)), entonces podemos afirmar que "Existe alguien que es mortal" ($\exists x M(x)$).

Eliminación del cuantificador existencial: Si sabemos que **existe** un x que cumple $\phi(x)$, podemos razonar sobre un **elemento arbitrario** t que cumple $\phi(t)$. La clave es que este t debe ser tratado como un símbolo nuevo sin suposiciones previas.

$$\frac{\exists x \phi(x) \qquad \boxed{\begin{array}{ll} t & \phi(t) \\ & \vdots \\ & \chi \end{array}}}{\chi}[\exists e]$$

Para concluir algo que **no depende de** t, debemos demostrar que la afirmación resultante es válida sin importar qué elemento t represente.

Ejemplo: Si sabemos que "Existe una persona que es estudiante", podemos asumir temporalmente que hay una persona arbitraria t que es estudiante y usar esta suposición para demostrar otra propiedad.

Ejemplo

$$\forall x(P(x) \to Q(x)), \exists x P(x) \vdash \exists x Q(x)$$

1		$\forall x(P(x) \to Q(x))$	premisa
2		$\exists x P(x)$	premisa
3	x_0	$P(x_0)$	supuesto
4		$P(x_0) \to Q(x_0)$	$\forall x$ *e* 1
5		$Q(x_0)$	$\to$ *e* 4,3
6		$\exists x Q(x)$	$\exists x$ *i* 5
7		$\exists x Q(x)$	$\exists x$ *e* 2,3–6

2.4.4. Reglas para la igualdad

Recordar que la relación de igualdad, $x = y$, se incluye en todos los lenguajes de primer orden.

$$\frac{}{t = t}\,[\text{Reflexividad*}] \qquad \frac{t_1 = t_2}{t_2 = t_1}\,[\text{Simetría}]$$

$$\frac{t_1 = t_2 \qquad t_2 = t_3}{t_1 = t_3}\,[\text{Transitividad}] \qquad \frac{s = t \qquad \phi(s)}{\phi(t)}\,[\text{Sustitución*}]$$

Ejercicio

Demostrar que sólo las reglas con * son necesarias para derivar el resto.

2.5. Robustez y Completitud

2.5.1. Teorema de Completitud de Gödel (1930)

La **completitud** establece que si una fórmula es lógicamente válida, entonces existe una prueba formal de ella en un sistema deductivo adecuado. En otras palabras, si $\Gamma \models \phi$, entonces también se tiene que $\Gamma \vdash \phi$.

La **Solidez** que afirma la dirección opuesta: si una fórmula es demostrable en un sistema formal ($\Gamma \vdash \phi$), entonces es válida en todos los modelos de Γ ($\Gamma \models \phi$). Es decir, todo lo que se puede probar es necesariamente cierto en cualquier interpretación válida.

Teorema de Solidez y Completitud de Gödel (1929-1930)

Para cualquier conjunto de fórmulas Γ y cualquier fórmula ϕ, se cumple:

$$\Gamma \vdash \phi \quad \text{si y solo si} \quad \Gamma \models \phi.$$

Esto implica que la lógica de primer orden es tanto **completa** como **sólida**, estableciendo una correspondencia entre las pruebas sintácticas ($\vdash$) y la validez semántica ($\models$).

Sin embargo, ningún sistema lógico que sea más expresivo que la lógica de primer orden puede tener simultáneamente un cálculo de pruebas que sea **tanto sólido como completo**. Esto significa que en sistemas más poderosos, como la lógica de segundo orden, siempre se sacrifica o bien la solidez o bien la completitud.

2.5.2. Demostración completitud de lógica de predicados

Más tarde, Leon Henkin proporcionó otra prueba más simple, formulando el **Teorema de la existencia del modelo de Henkin (1949)**.

Teorema de la existencia Henkin (1949)

Cada conjunto consistente de oraciones tiene un modelo.

Un conjunto de oraciones Γ es consistente si no se puede probar una contradicción de esas hipótesis, es decir, no puede ocurrir que $\Gamma \vdash \phi$ y $\Gamma \vdash \neg\phi$.

A partir de este teorema, es fácil deducir el **teorema de completitud**. Pero, primero necesitamos el siguiente Lema:

Lema

Si $\Gamma \cup \{\neg\phi\}$ es inconsistente, entonces $\Gamma \vdash \phi$.

Demostración: Asuma que $\Gamma \cup \{\neg\phi\}$ es inconsistente. Luego, existe una fórmula ψ tal que

$$\Gamma \cup \{\neg\phi\} \vdash \psi \quad \text{y} \quad \Gamma \cup \{\neg\phi\} \vdash \neg\psi.$$

Es decir, $\Gamma, \neg\phi \vdash \bot$, lo que implica:

$$\Gamma \vdash \neg\phi \to \bot \quad \text{(regla de introducción de implicancia)}$$

$$\Gamma \vdash \phi \quad \text{(contrapositiva de implicancia)}$$

Teorema de Completitud:

$\Gamma \models \phi \;\Rightarrow\; \Gamma \vdash \phi$ o equivalentemente,

$$\Gamma \nvdash \phi \quad \Rightarrow \quad \Gamma \not\models \phi$$

Demostración:

1	$\Gamma \nvdash \phi$	(supuesto)
2	$\Gamma \cup \neg\phi$ es consistente	(contraposición del lema)
3	$\mathcal{M} \models \Gamma \cup \neg\phi$	(Teorema de Henkin)
4	$\mathcal{M} \models \Gamma$ y $\mathcal{M} \models \neg\phi$	
5	$\Gamma \not\models \phi$	(definición de consecuencia lógica)

Parte II

Técnicas de Demostración

Capítulo 3

Inducción Matemática

3.1. Introducción a la Demostración

Una **demostración** es un argumento lógico que establece la verdad de un enunciado matemático. Se construye a partir de axiomas, definiciones y teoremas previamente probados.

En clases anteriores, hemos explorado las **reglas de inferencia** en lógica, que nos permiten derivar conclusiones a partir de premisas utilizando razonamiento formal. Estas reglas, como el *Modus Ponens*, *Modus Tollens*, la transitividad de la implicación, entre otras, constituyen la base fundamental de las demostraciones matemáticas. Ahora, aplicaremos estas reglas de manera menos estructurada en distintas estrategias de demostración para establecer formalmente la validez de proposiciones matemáticas.

Las demostraciones pueden ser formales o informales. Las **demostraciones formales** detallan cada paso con precisión matemática, pero suelen ser largas y difíciles de seguir. En la práctica, las demostraciones suelen ser más informales y omiten detalles obvios para los lectores familiarizados con la teoría.

3.1.1. Clasificación de Teoremas y Resultados Matemáticos

En matemáticas, los resultados se clasifican en:

- **Teorema**: Un resultado matemático importante con una demostración no trivial.
- **Lema**: Un resultado auxiliar que facilita la demostración de un teorema.
- **Proposición**: Un resultado matemático relevante pero de menor importancia que un teorema.
- **Corolario**: Un resultado que sigue inmediatamente de un teorema.
- **Conjetura**: Una afirmación que se cree verdadera pero que aún no ha sido probada.

3.1.2. Métodos de Demostración

Muchos teoremas tienen la forma $\forall x\ (P(x) \to Q(x))$, donde P representa la hipótesis y Q la conclusión. Para probar esta afirmación, existen diversas estrategias:

Demostración Directa

Se parte de la hipótesis P y se deduce la conclusión Q mediante una secuencia lógica de pasos.

Ejemplo: Demostrar que si n es impar, entonces n^2 también es impar.

Demostración. Si n es impar, entonces se puede escribir como $n = 2k+1$ para algún entero k. Elevando al cuadrado:

$$n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1.$$

Como el resultado es de la forma $2m + 1$, donde m es un entero, n^2 es impar. □

Demostración por Contraposición

En lugar de demostrar $P \to Q$, se demuestra su contrapartida lógica: $\neg Q \to \neg P$.

Ejemplo: Si $3n + 2$ es impar, demostrar que n es impar.

Demostración. Demostramos la contraposición: si n es par, entonces $3n + 2$ es par. Si $n = 2k$, entonces $3n + 2 = 3(2k) + 2 = 6k + 2 = 2(3k + 1)$, que es par. Por lo tanto, si $3n + 2$ es impar, entonces n es impar. □

Demostración por Análisis de Casos

Se divide el problema en varios casos y se demuestra el resultado en cada uno de ellos.

Ejemplo: Probar que si n no es múltiplo de 3, entonces $3 \mid (n^2 - 1)$.

Demostración. Si n no es múltiplo de 3, puede ser de la forma $3k + 1$ o $3k + 2$.

- Si $n = 3k + 1$, entonces $n^2 = 9k^2 + 6k + 1$, y $n^2 - 1 = 3(3k^2 + 2k)$, múltiplo de 3.
- Si $n = 3k + 2$, entonces $n^2 = 9k^2 + 12k + 4$, y $n^2 - 1 = 3(3k^2 + 4k + 1)$, múltiplo de 3.

En ambos casos, $3 \mid (n^2 - 1)$. □

Demostración por Contradicción

Se asume que la afirmación es falsa y se llega a una contradicción. Es decir, en lugar de demostrar P, se demuestra $\neg P \to \bot$

Ejemplo: Demostrar que no existe un número racional menor que todos los racionales positivos.

Demostración. Supongamos que existe un número racional r tal que r es el menor de todos los racionales positivos. Entonces, $r/2$ también es un número racional positivo y es menor que r, lo que contradice la suposición. Por lo tanto, tal número no puede existir. □

Una proposición de la forma $p \to q$ también puede ser probada por contradicción. Basta derivar una contradicción a partir de $p \wedge \neg q$.

3.1.3. Conclusión

Las demostraciones matemáticas son fundamentales en la matemática discreta y en toda la disciplina matemática. Cada técnica tiene su lugar y su utilidad dependiendo de la estructura del teorema a demostrar. La práctica en la construcción de demostraciones ayuda a mejorar la capacidad de razonamiento lógico y la creatividad matemática.

3.1.4. Problemas Propuestos

1. Probar que la suma de dos números impares siempre es par.
2. Demuestre por contraposición que si $n = ab$, donde ambos a y b son reales positivos, entonces $a \leq \sqrt{n}$ o $b \leq \sqrt{n}$.
3. Usar el método de contradicción para demostrar que $\sqrt{3}$ es irracional.
4. Probar por contradicción que si a y b son reales positivos, entonces $a + b \geq 2\sqrt{ab}$
5. Probar que para todo entero n, si n^2 es par, entonces n es par.
6. Probar por análisis de casos que el producto de dos enteros consecutivos es siempre par.

3.2. Principio de Inducción

El **principio de inducción** es una técnica poderosa para demostrar propiedades sobre los números naturales. Se basa en la idea de que si una propiedad se cumple para un número base y se mantiene al pasar de un número al siguiente, entonces la propiedad se cumple para todos los naturales.

3.2.1. Definición del Principio de Inducción

El principio de inducción establece que una propiedad $P(n)$ se cumple para todos los números naturales si se cumplen las siguientes condiciones:

Principio de Inducción

Una propiedad vale para todos los números naturales si:

- **Caso base:** Se demuestra que la propiedad es cierta para $n = 0$.
- **Caso inductivo:** Se asume que la propiedad es cierta para un número n arbitrario (hipótesis inductiva) y se demuestra que también es cierta para $n + 1$.

Simbólicamente,

$$P(0) \land \forall n \ (P(n) \to P(n+1)) \Rightarrow \forall n.P(n).$$

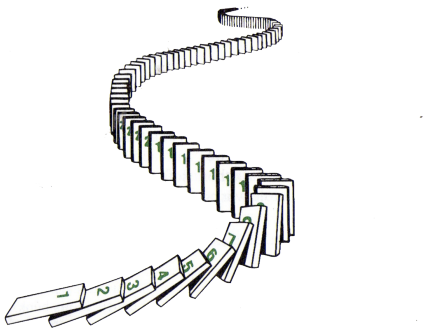


Figura 3.1: Ilustración del principio de inducción.

Esta técnica se puede visualizar como una fila de fichas de dominó: si la primera ficha cae y si al caer una ficha hace caer la siguiente, entonces todas las fichas caerán.

Ejemplo: Suma de los primeros números naturales

Probaremos por inducción que:

$$0 + 1 + \cdots + n = \frac{n(n+1)}{2}.$$

Caso base: Para $n = 0$, tenemos $0 = \frac{0(0+1)}{2}$, que es cierto.
Caso inductivo: Supongamos que la fórmula es válida para algún n, es decir,

$$0 + 1 + \cdots + n = \frac{n(n+1)}{2}.$$

Queremos demostrar que también se cumple para $n + 1$:

$$0 + 1 + \cdots + n + (n+1) = \frac{n(n+1)}{2} + (n+1) \quad = \frac{n(n+1) + 2(n+1)}{2} \quad = \frac{(n+1)(n+2)}{2}.$$

Por lo tanto, la propiedad se mantiene y queda demostrada por inducción.

Limitaciones del principio de inducción

La inducción es una técnica muy útil para **probar** versiones cerradas de fórmulas que involucran e.g. sumatorias, productorias o recurrencias, pero no nos dice nada sobre cómo **inferir** dichas formas cerradas.

3.2.2. Generalización del Principio de Inducción

En algunas ocasiones, la propiedad que queremos demostrar no necesariamente comienza en $n = 0$, sino en un valor inicial diferente, digamos n_0. En estos casos, podemos formular una versión más general del principio de inducción:

Principio de Inducción Generalizado

Si queremos demostrar que una propiedad $P(n)$ es válida para todos los números naturales mayores o iguales a un cierto n_0, basta con probar:

- **Caso base:** Se demuestra que la propiedad es cierta para $n = n_0$.
- **Caso inductivo:** Se asume que la propiedad es cierta para un número arbitrario $n \geq n_0$ (hipótesis inductiva) y se demuestra que también es cierta para $n + 1$.

Simbólicamente,

$$P(n_0) \land \forall n \geq n_0 \, (P(n) \to P(n+1)) \Rightarrow \forall n \geq n_0 . P(n).$$

Esta generalización es útil cuando la propiedad que queremos demostrar solo tiene sentido a partir de cierto número natural. Por ejemplo, si estamos probando una afirmación sobre

la factorización de números naturales en factores primos, podríamos necesitar que el primer caso considerado sea $n_0 = 2$, ya que el número 1 no es primo ni compuesto.

Ejercicio: Lanzamiento de Pelotas en el Parque

Un número impar de personas está de pie en un parque, cada una con una pelota en la mano. La distancia entre cualquier par de personas es distinta.

Cada persona lanza su pelota a la persona que está más cerca de ella. Queremos demostrar por inducción que **siempre existe al menos una persona que no recibe ninguna pelota**.

Usando inducción, demostraremos la siguiente proposición $P(n)$ sobre los números naturales mayores o iguales a 1:

$P(n) \equiv$ Cuando en el parque se paran $2n+1$ personas, existe una que no recibe ninguna pelota.

Caso base: $P(1)$

Entre las tres personas, sean A y B las que están separadas por la menor distancia y C la persona restante, es decir:

$$d(A,B) < d(A,C) \quad \text{y} \quad d(A,B) < d(B,C).$$

Como consecuencia, A y B se lanzan la pelota mutuamente, y por lo tanto, C no recibe ninguna pelota.

Caso inductivo

Supongamos por hipótesis inductiva que en un grupo de $2n+1$ personas siempre hay alguien que no recibe ninguna pelota. Queremos demostrar que lo mismo ocurre para $2n+3$ personas.

Sean A y B las personas que están separadas por la menor distancia. Como en el caso base, A y B se lanzan la pelota mutuamente.

Ahora analizamos los siguientes casos:

- Si alguien más le lanza su pelota a A o a B, entonces el máximo de pelotas recibidas entre los restantes $2n+1$ participantes es $2n$, y por tanto, al menos uno de ellos no recibe ninguna pelota.
- Supongamos que la única pelota que recibe A es la de B, y viceversa. El resto del grupo es de $2n+1$ personas, y todas las distancias entre ellas son distintas. Por hipótesis inductiva, alguien en este grupo no recibe ninguna pelota.

Por lo tanto, la proposición se mantiene para $2n+3$, y el principio de inducción nos permite concluir que siempre habrá una persona que no recibe ninguna pelota en cualquier número impar de participantes.

3.2.3. Ejercicios propuestos

- Demuestre que si $h > -1$, entonces para todo entero no negativo n se tiene que $1+nh \leq (1+h)^n$.
- Demuestre que para todo entero positivo n se tiene que $\sum_{i=1}^{n} \frac{1}{\sqrt{i}} > 2(\sqrt{n+1}-1)$.
- Sean $a_1, a_2, \ldots, a_n$ números reales positivos. Probar que

$$\frac{1}{n}(a_1 + a_2 + \cdots + a_n) \geq (a_1 a_2 \cdots a_n)^{\frac{1}{n}}.$$

3.3. Principio de Inducción Fuerte

En el caso inductivo, para probar $P(n+1)$ sólo podemos emplear

$$P(n)$$

Sin embargo, muchas pruebas se simplifican significativamente si pudiésemos emplear

$$P(0) \ \wedge \ P(1) \ \wedge \ \ldots \ \wedge \ P(n)$$

Este último argumento de prueba es válido y se conoce como **principio de inducción fuerte**.

Principio de Inducción Fuerte

Una propiedad vale para todos los números naturales si:

Caso base: la propiedad vale para 0.

Caso inductivo: cualquiera sea el natural n, la propiedad vale para $n+1$ si vale para todos los naturales menores o iguales a n.

Simbólicamente,

$$P(0) \ \wedge \ \forall n \ (P(0) \wedge P(1) \wedge \ldots \wedge P(n)) \rightarrow P(n+1) \quad \rightarrow \quad \forall n.\ P(n)$$

3.3.1. Ejercicio: Descomposición en Factores Primos

Pruebe por inducción fuerte que todo natural mayor que 1 puede escribirse como producto (posiblemente unitario) de factores primos.

Caso base: Trivial ya que 2 puede escribirse como el producto unitario 2.

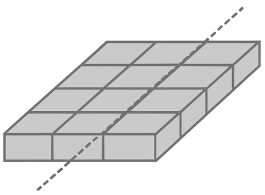
Caso inductivo: Considere el natural $n+1$ y asuma por hipótesis inductiva que todo natural en el intervalo $[2, n]$ puede escribirse como producto de factores primos. Ahora analicemos si $n+1$ es primo o compuesto.

- Si $n+1$ es primo, la propiedad es trivial.
- Si $n+1$ es compuesto, entonces existen naturales j, k en el intervalo $[2, n]$ tales que $n+1 = j \cdot k$. Por hipótesis inductiva, j y k pueden escribirse como producto de factores primos, y por lo tanto $n+1 = j \cdot k$ también.

Observación: Para esta prueba usamos la variante del principio de inducción fuerte que establece la propiedad respectiva a partir de 2 (no a partir de 0).

3.3.2. Ejercicio: Cortando Barras de Chocolate

Una barra de chocolate consiste en $m \times n$ cuadrados dispuestos en m filas y n columnas. Uno puede partir la barra en cuadrados cortándola a lo largo de sus "lineas". Asumiendo que las lineas no se pueden cortar parcialmente, demuestre que $mn - 1$ cortes alcanzan para obtener los mn cuadrados.



Haremos la demostración por inducción fuerte sobre el número $k = mn$ de cuadrados de la barra de chocolate.

Caso base: Una barra de $k = 1$ cuadrado trivialmente puede ser cortada en cuadrados con $1 - 1$ cortes.

Caso inductivo: Consideremos una barra de $k+1$ cuadrados. Hagamos sobre la barra un corte arbitrario. Éste divide a la barra en dos subbarras de k_1 y k_2 cuadrados respectivamente, donde $k_1 + k_2 = k + 1$.

Por hipótesis de inducción estas dos barras pueden ser cortadas con

$$k_1 - 1 \quad \text{y} \quad k_2 - 1$$

cortes. Por tanto, la barra original (de $k+1$ cuadrados) puede cortarse con:

$$\begin{aligned} 1 \ + \ k_1 - 1 \ + \ k_2 - 1 &= k_1 + k_2 - 1 \\ &= k \end{aligned}$$

cortes. Así se demuestra que cualquier barra de chocolate de mn cuadrados puede ser cortada en unidades utilizando exactamente $mn - 1$ cortes.

3.4. Comparando los dos principios de inducción

Los dos principios de inducción estudiados (el estándar—también llamado *débil*—y el fuerte) son **igualmente expresivos**. Esto significa que una propiedad puede ser establecida usando uno si y sólo si puede ser establecida usando el otro.

Ambos principios permiten demostrar afirmaciones sobre los números naturales, pero en algunos casos el uso del principio de inducción fuerte simplifica considerablemente la demostración, ya que en lugar de depender solo de $P(n)$ para demostrar $P(n+1)$, permite utilizar todas las propiedades anteriores $P(0), P(1), ..., P(n)$.

Ejercicio: Derive el principio de inducción fuerte a partir del principio de inducción estándar. *Hint:* Proceda por inducción.

3.4.1. Principio del Buen Orden

El principio del buen orden es un concepto fundamental en la teoría de números y es equivalente al principio de inducción. Nos dice que cualquier subconjunto no vacío de los números naturales tiene un elemento mínimo. Esto significa que no existen secuencias infinitamente decrecientes en los naturales.

Principio del Buen Orden

Sea S un subconjunto no vacío de los números naturales $\mathbb{N}$. Entonces, existe un elemento mínimo $m \in S$, es decir, un número m tal que para todo $x \in S$, se cumple que $m \leq x$.

El principio del buen orden es la base lógica de la inducción matemática, ya que podemos utilizarlo para demostrar el principio de inducción estándar por reducción al absurdo. La idea detrás de esta prueba es suponer que existe un número natural para el cual no se cumple la propiedad que queremos demostrar, y mostrar que esto lleva a una contradicción con el principio del buen orden.

Ejercicio: Derive el principio de inducción estándar a partir del principio del buen orden. *Hint:* Proceda por reducción al absurdo.

Es más, puede probarse que el principio de inducción débil, el principio de inducción fuerte y el principio del buen orden resultan **equivalentes**. En otras palabras, si se asume cualquiera de estos principios como cierto, se pueden demostrar los otros dos.

3.5. Introducción a la Inducción Estructural

La inducción es una técnica poderosa utilizada para demostrar propiedades de estructuras definidas recursivamente. Aunque la inducción matemática es útil para propiedades de los números naturales, la **inducción estructural** extiende esta idea a conjuntos definidos recursivamente.

A veces es difícil definir un objeto explícitamente, pero no lo es tanto si lo definimos en términos de objetos de su mismo tipo. A este proceso lo llamamos **recursión**. Mediante recursión, podemos definir secuencias, funciones y conjuntos, especificando un caso base y reglas que permiten construir nuevos elementos a partir de los ya existentes.

3.5.1. Definiciones Recursivas

Definición Recursiva

Un objeto puede definirse recursivamente especificando:

- **Caso base**: Elementos iniciales del conjunto.
- **Regla recursiva**: Forma de construir nuevos elementos a partir de los ya definidos.

Ejemplo: Secuencia de Potencias de 2

- Caso base: $a_0 = 1$.
- Regla recursiva: $a_{n+1} = 2a_n$.

Aplicando esta definición recursiva, obtenemos los términos de la secuencia que generan las potencias de 2: $1, 2, 4, 8, 16, 32, \ldots$.

La ventaja de definir una secuencia recursivamente es que no necesitamos conocer una fórmula explícita general desde el principio. En muchos casos, esto facilita la comprensión y el diseño de algoritmos que dependen de estructuras previas, como en el caso de la programación dinámica o la resolución de problemas combinatorios.

3.5.2. Definición Recursiva de Funciones

Factorial

Un ejemplo clásico de función definida recursivamente es el factorial. La función $f(n) = n!$ puede ser definida recursivamente como:

$$f(0) = 1, \quad f(n+1) = (n+1)f(n).$$

Ejercicio: Defina recursivamente la función $S(n) = \sum_{k=0}^{n} a_k$.

Números de Fibonacci

Los números de Fibonacci están definidos por:

$$\begin{aligned} f_0 &= 0, \quad f_1 = 1, \\ f_n &= f_{n-1} + f_{n-2}, \quad \text{para } n \geq 2. \end{aligned}$$

Ejemplo: $0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots$

Ejercicio: Propiedad de Crecimiento de Fibonacci

Demuestre que para todo $n \geq 3$, si $\alpha = \frac{1+\sqrt{5}}{2}$, entonces $f_n > \alpha^{n-2}$.

Demostración por inducción:

Caso Base: Debemos demostrar el caso base tanto para $n = 3$ como para $n = 4$ (porque no podemos demostrar que $f_4 > \alpha^2$ desde la hipótesis inductiva).

Esto no es difícil puesto que:

$$f_3 = 2 > \alpha = \frac{1+\sqrt{5}}{2},$$
$$f_4 = 3 > \alpha^2 = \frac{3+\sqrt{5}}{2}.$$

Caso Inductivo: Considere un entero $n+1$ tal que $n \geq 4$. Por hipótesis inductiva, $f_j > \alpha^{j-2}$ para todo $j \in [3, n]$.

Dado que $f_{n+1} = f_n + f_{n-1}$, se tiene:

$$f_{n+1} > \alpha^{n-2} + \alpha^{n-3} = \alpha^{n-3}(1+\alpha).$$

Por tanto, para demostrar que $f_{n+1} > \alpha^{n-1}$ basta demostrar que:

$$(1+\alpha) \geq \alpha^2.$$

Esto es fácil, ya que:

$$(1+\alpha) = \frac{3+\sqrt{5}}{2} = \alpha^2.$$

Así, la desigualdad se cumple y la demostración por inducción queda completada.

Ejercicio: Demuestre que para todo entero positivo n,

$$f_0 f_1 + f_1 f_2 + \cdots + f_{2n-1} f_{2n} = f_{2n}^2.$$

3.5.3. Conjuntos Definidos Recursivamente

Los conjuntos también pueden definirse de manera recursiva, siguiendo una estructura similar a la de las funciones. Un conjunto recursivamente definido se construye especificando un conjunto base y una regla que permite generar nuevos elementos a partir de los que ya están en el conjunto.

La definición recursiva de conjuntos sigue la estructura:

- **Caso base**: Elementos iniciales del conjunto.
- **Regla recursiva**: Forma de construir nuevos elementos.

Definición de Σ^*

El conjunto de strings sobre un alfabeto Σ:

- **Caso base**: La cadena vacía ϵ pertenece a Σ^*.
- **Regla recursiva**: Si $w \in \Sigma^*$ y $x \in \Sigma$, entonces $wx \in \Sigma^*$.

Este conjunto, denotado como Σ^*, representa todas las posibles cadenas (incluyendo la cadena vacía) que pueden formarse con los símbolos de Σ.

Definiciones Recursivas de Operaciones sobre Strings

Al igual que los conjuntos, las operaciones sobre strings pueden definirse de manera recursiva. Definir operaciones de esta manera nos permite formalizar algoritmos y probar propiedades sobre strings de manera estructurada.

Una de las operaciones más importantes sobre strings es la concatenación, que une dos strings en un solo. La concatenación de dos strings w_1 y w_2 se define recursivamente de la siguiente manera:

Definición de concatenación

- **Regla base**: Si $w \in \Sigma^*$, entonces $w \cdot \epsilon = w$ (concatenar con la cadena vacía no cambia la cadena original).
- **Regla inductiva**: Si $w_1, w_2 \in \Sigma^*$ y $x \in \Sigma$, entonces $w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x$ (concatenar un carácter a una cadena es equivalente a concatenar primero el prefijo y luego agregar el carácter final).

Por ejemplo, si tenemos el alfabeto $\Sigma = \{a, b\}$ y las cadenas $w_1 = ab$, $w_2 = ba$, la concatenación $w_1 \cdot w_2$ sigue:

$$\begin{aligned} ab \cdot ba &= (ab \cdot b)a \\ &= ((ab \cdot \epsilon)b)a \\ &= (ab)b)a \\ &= abba. \end{aligned}$$

Otra propiedad importante de los strings es su longitud. Podemos definir recursivamente la función $l(w)$, que calcula la longitud de un string w:

Definición de longitud

- **Regla base**: $l(\epsilon) = 0$.
- **Regla inductiva**: $l(wx) = l(w) + 1$, para $x \in \Sigma$ (cada vez que agregamos un carácter a la cadena, su longitud aumenta en uno).

Por ejemplo, si $w = ab$:

$$\begin{aligned} l(\epsilon) &= 0, \\ l(a) &= l(\epsilon) + 1 = 1, \\ l(ab) &= l(a) + 1 = 2. \end{aligned}$$

3.6. Inducción Estructural sobre Estructuras Recursivas

La **inducción estructural** es un método para demostrar propiedades de conjuntos definidos de manera recursiva. Dado que estos conjuntos están construidos a partir de reglas que generan nuevos elementos a partir de los anteriores, la inducción estructural nos permite razonar sobre ellos en una forma natural y efectiva. La inducción estructural permite demostrar que una propiedad se cumple para todos los elementos del conjunto sin necesidad de verificarlos uno por uno.

El método de inducción estructural sigue estos pasos:

Inducción Estructural

- **Caso base**: Demostrar que la propiedad se cumple para cada caso base de la definición recursiva.
- **Paso inductivo**: Se asume que la propiedad es cierta para los elementos ya definidos (hipótesis inductiva) y se prueba que debe ser cierta para los nuevos elementos generados por la regla recursiva.

3.6.1. Ejemplo: Longitud de Cadenas

Ejercicio: Demuestre que para todo par de strings $w_1, w_2 \in \Sigma^*$, se cumple:

$$l(w_1 w_2) = l(w_1) + l(w_2).$$

Demostración

Definimos la proposición $P(y)$ como:

$$l(xy) = l(x) + l(y) \quad \text{para cualquier } x \in \Sigma^*.$$

Queremos probar que $P(y)$ es cierta para todo $y \in \Sigma^*$ usando inducción estructural.

Caso base: Si $y = \epsilon$ (cadena vacía), entonces:

$$l(x\epsilon) = l(x) = l(x) + 0 = l(x) + l(\epsilon).$$

Esto muestra que la propiedad se cumple para el caso base.

Paso inductivo: Supongamos que la propiedad es cierta para una cadena w, es decir, que:

$$l(xw) = l(x) + l(w).$$

Ahora, consideremos una cadena extendida con un símbolo adicional $a \in \Sigma$, es decir, wa. Queremos demostrar que:

$$l(xwa) = l(x) + l(wa).$$

Por la definición recursiva de la longitud de un string, sabemos que:

$$l(wa) = l(w) + 1.$$

Aplicando la hipótesis inductiva:

$$\begin{aligned} l(xwa) &= l(xw) + 1 \\ &= (l(x) + l(w)) + 1 \\ &= l(x) + (l(w) + 1) \\ &= l(x) + l(wa). \end{aligned}$$

Dado que hemos demostrado que la propiedad se mantiene en el caso base y en el paso inductivo, por el principio de inducción estructural, concluimos que la propiedad es cierta para toda cadena $y \in \Sigma^*$.

3.6.2. Definición Recursiva de Árboles Binarios

Árbol Binario

Un árbol binario completo se define como:

- **Caso base:** Un árbol con un único nodo raíz r.
- **Regla recursiva:** Si T_1 y T_2 son árboles binarios completos, entonces $T = T_1 \cdot T_2$ es un árbol binario completo que consiste de una raíz r junto con arcos que unen a r con la raíz del sub-árbol izquierdo T_1 y con la raíz del sub-árbol derecho T_2.

Ejercicio: Número de Vértices en un Árbol Binario

Defina recursivamente la función $n(T)$ que entrega el número de vértices de un árbol binario T.

Solución:

- **Caso base**: Si T está formado sólo por una raíz r, entonces $n(T) = 1$.
- **Regla inductiva**: Si T_1 y T_2 son árboles binarios completos, entonces el árbol binario completo $T = T_1 \cdot T_2$ tiene:

$$n(T) = 1 + n(T_1) + n(T_2).$$

Ejercicio: Altura de un Árbol Binario

Defina la función $h(T)$ que entrega la distancia máxima entre la raíz de un árbol binario T y una de sus hojas.

Solución:

- **Caso base**: Si T está formado sólo por una raíz r, entonces $h(T) = 0$.
- **Regla inductiva**: Si T_1 y T_2 son árboles binarios completos, entonces el árbol binario completo $T = T_1 \cdot T_2$ tiene:

$$h(T) = 1 + \text{máx}(h(T_1), h(T_2)).$$

Ejercicio: Relación entre $n(T)$ y $h(T)$

Demuestre que para todo árbol binario completo T, se cumple que:

$$n(T) \leq 2^{h(T)+1} - 1.$$

Demostración por Inducción Estructural:

Caso base: Si T está formado sólo por una raíz r, entonces $n(T) = 1$ y $h(T) = 0$, por lo que:

$$n(T) = 1 \leq 2^{0+1} - 1 = 1.$$

Paso inductivo: Supongamos que la propiedad se cumple para los subárboles T_1 y T_2, es decir,

$$n(T_1) \leq 2^{h(T_1)+1} - 1, \quad n(T_2) \leq 2^{h(T_2)+1} - 1.$$

Utilizando la hipótesis inductiva:

$$\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) \\
&\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) \quad \text{por H.I.} \\
&\leq 2 \cdot \text{máx}(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 \quad \text{suma es como máx 2 veces el mayor} \\
&= 2 \cdot 2^{max(h(T_1),h(T_2))+1} - 1 \quad max(2^x, 2^y) = 2^{max(x,y)} \\
&= 2 \cdot 2^{h(T)+1} - 1 \quad \text{por def. de } h(T) \\
&= 2^{h(T)+1} - 1.
\end{aligned}$$

Por lo tanto, la propiedad se cumple para todo árbol binario completo T, completando la demostración.

3.6.3. Ejercicios Propuestos:

- Definir recursivamente la función $rev(w)$ que invierte un string en Σ^*, y demostrar por inducción estructural que $rev(xy) = rev(y)rev(x)$ para todo $x, y \in \Sigma^*$.

- Sea A el conjunto definido recursivamente por: **(i)** $\epsilon \in A$; **(ii)** si $w \in A$ entonces $awa \in A$, para $a \in \Sigma$. Demostrar que toda cadena en A tiene longitud par.
- Definir recursivamente la función que cuenta el número de hojas en un árbol binario completo.
- Probar por inducción estructural que el número de hojas en un árbol binario completo de altura h es exactamente 2^h.
- Demostrar que en un árbol binario completo con n nodos internos, hay exactamente $n + 1$ hojas.
- Probar que la altura de un árbol binario completo con n nodos es al menos $\log_2(n+1)-1$.

3.7. Ejercicios Resueltos

3.7.1. Ejercicio: Propiedad de la reversa de strings

Definir recursivamente la función $rev(w)$ que invierte un string $w \in \Sigma^*$. Luego, demostrar por inducción estructural que:

$$\text{rev}(xy) = \text{rev}(y)\text{rev}(x)$$

para todo $x, y \in \Sigma^*$.

Definición recursiva de rev(w):

- Caso base: $\text{rev}(\epsilon) = \epsilon$
- Paso recursivo: Si $w \in \Sigma^*$ y $a \in \Sigma$, entonces $\text{rev}(wa) = a \cdot \text{rev}(w)$

Demostración: Fijamos un string $x \in \Sigma^*$. Demostraremos por inducción estructural sobre y que:

$$\text{rev}(xy) = \text{rev}(y) \cdot \text{rev}(x)$$

Caso base: $y = \epsilon$

$$\text{rev}(x\epsilon) = \text{rev}(x) = \epsilon \cdot \text{rev}(x) = \text{rev}(\epsilon) \cdot \text{rev}(x)$$

Paso inductivo: Supongamos que $\text{rev}(xw) = \text{rev}(w)\text{rev}(x)$ para una cadena $w \in \Sigma^*$. Queremos demostrar que también se cumple para wa, donde $a \in \Sigma$.

$$\begin{aligned}
\text{rev}(xwa) &= a \cdot \text{rev}(xw) \\
&= a \cdot (\text{rev}(w)\text{rev}(x)) \\
&= (a \cdot \text{rev}(w)) \cdot \text{rev}(x) \\
&= \text{rev}(wa) \cdot \text{rev}(x)
\end{aligned}$$

Por lo tanto, se cumple la propiedad para todo $y \in \Sigma^*$.

3.7.2. Ejercicio: Longitud par en un conjunto definido recursivamente

Sea el conjunto A definido por:

- Caso base: $\epsilon \in A$
- Paso recursivo: Si $w \in A$ y $a \in \Sigma$, entonces $awa \in A$

Demostrar que toda cadena en A tiene longitud par.

Demostración por inducción estructural:

Caso base: $\epsilon \in A$, y su longitud es 0, que es par.

Paso inductivo: Supongamos que $|w|$ es par. Entonces $|awa| = |w| + 2$. Como $|w|$ es par, $|w| = 2k$ para algún k, por lo tanto:

$$|awa| = 2k + 2 = 2(k + 1)$$

que es par.

Por lo tanto, toda cadena en A tiene longitud par.

Parte III

Relaciones y Funciones

Capítulo 4

Relaciones

4.1. Definición de relaciones

4.1.1. ¿Qué es una relación?

En matemáticas, una de las formas más naturales de establecer conexiones entre elementos de distintos conjuntos es a través de las relaciones.

En ciencias de la computación, las relaciones juegan un rol fundamental en múltiples contextos. Permiten modelar datos y sus vínculos, como en bases de datos relacionales; representan estructuras como grafos en algoritmos y redes; y son esenciales para describir transiciones de estado, dependencias, ordenamientos y equivalencias. Comprender las propiedades de las relaciones permite, por ejemplo, optimizar consultas en SQL, verificar invariantes en programas o formalizar sistemas lógicos complejos.

Relación binaria

Una **relación binaria** entre dos conjuntos A y B es un **subconjunto del producto cartesiano** $A \times B$. Es decir, es un conjunto de pares ordenados (a, b) donde $a \in A$ y $b \in B$.

Notación. Si $R \subseteq A \times B$, entonces decimos que los elementos $a_0 \in A$ y $b_0 \in B$ están relacionados por R si $(a_0, b_0) \in R$. Esto lo denotamos como:

$$a_0 \, R \, b_0.$$

Si no están relacionados, usamos:

$$a_0 \not R \; b_0 \quad \text{o} \quad (a_0, b_0) \notin R.$$

Ejemplo: Relación de divisibilidad.

Consideremos los conjuntos $\mathbb{N}$ (naturales) y $\mathbb{N}_0$ (naturales con cero). Podemos definir la relación de divisibilidad como:

$$| = \{(a, k \cdot a) \mid a \in \mathbb{N},\ k \in \mathbb{N}_0\}.$$

Otra forma equivalente de definirla:

$$a \mid b \quad \Leftrightarrow \quad \exists k \in \mathbb{N}_0 \text{ tal que } b = k \cdot a.$$

Ejemplo: Base de datos relacional.

En una base de datos relacional, una relación puede modelar la asignación de estudiantes a cursos como un subconjunto del producto cartesiano entre el conjunto de estudiantes E y el conjunto de cursos C. Por ejemplo, si $(e, c) \in R$ indica que el estudiante e está inscrito en el curso c.

4.1.2. Representación de relaciones

Cuando los conjuntos involucrados son finitos, podemos representar gráficamente una relación como un grafo dirigido. Una flecha desde $a \in A$ hacia $b \in B$ indica que $a\,R\,b$.

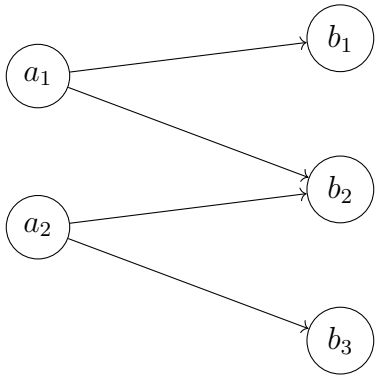


Figura 4.1: En esta ilustración, los nodos del lado izquierdo representan elementos de un conjunto A, los del lado derecho a un conjunto B, y las flechas indican los pares relacionados en $R \subseteq A \times B$.

¿Cuántas relaciones existen entre dos conjuntos finitos A y B?

Si A tiene m elementos y B tiene n, entonces el conjunto $A \times B$ tiene $m \cdot n$ elementos. Por lo tanto, existen 2^{mn} relaciones distintas entre A y B.

Esto muestro cómo una definición tan simple como "subconjunto del producto cartesiano" puede esconder una riqueza combinatoria enorme. Incluso con conjuntos pequeños, la cantidad de posibles relaciones crece exponencialmente. Esto tiene consecuencias profundas en

ciencias de la computación. Por ejemplo, en **bases de datos**, cada relación representa una tabla posible; entender su espacio total ayuda a evaluar restricciones, integridad referencial y optimización.

4.1.3. Relaciones sobre un conjunto

En el caso particular en que una relación R relaciona elementos de un conjunto consigo mismo, es decir, $R \subseteq A \times A$, decimos que R es una **relación sobre el conjunto** A.

Ejemplos.

- $R = \{(a, b) \mid a \leq b\}$ sobre $\mathbb{N}$.
- $R = \{(a, b) \mid a + b \leq 3\}$ también sobre $\mathbb{N}$.

4.1.4. Propiedades de las relaciones

Al estudiar relaciones sobre un conjunto A, es común interesarse por ciertas propiedades estructurales que éstas pueden tener. De hecho, cada una de estas propiedades ya las expresamos usando lógica de predicados.

Propiedades fundamentales

Sea $R \subseteq A \times A$ una relación sobre A. Decimos que:

- R es **reflexiva** si $(a, a) \in R$ para todo $a \in A$.
- R es **simétrica** si $(a, b) \in R$ implica $(b, a) \in R$.
- R es **antisimétrica** si $(a, b) \in R$ y $(b, a) \in R$ implica $a = b$.
- R es **transitiva** si $(a, b) \in R$ y $(b, c) \in R$ implica $(a, c) \in R$.

Ejemplo: Sobre el conjunto de enteros $\mathbb{Z}$:

- $R_1 = \{(a, b) \mid a \leq b\}$ es reflexiva, antisimétrica y transitiva.
- $R_2 = \{(a, b) \mid a \mid b\}$ también cumple las tres propiedades anteriores.
- $R_3 = \{(a, b) \mid a + b \leq 3\}$ es simétrica, pero no transitiva ni reflexiva.

4.2. Relaciones de orden

Las relaciones de orden nos permiten capturar de manera formal la noción de jerarquía, precedencia o comparación entre elementos. Son fundamentales para modelar estructuras como cronogramas de tareas, jerarquías de clases, árboles de decisión, relaciones de dependencia

en grafos y muchas otras situaciones donde ciertos elementos están "por encima.º .ªntes"que otros.

Relaciones de orden

Una relación $R \subseteq A \times A$ se llama:

- **Orden parcial** si es reflexiva, antisimétrica y transitiva.
- **Orden total** si además, para todo $a, b \in A$, se cumple que $(a, b) \in R$ o $(b, a) \in R$.

Ejemplo.

- La relación $\leq$ sobre $\mathbb{N}$ es un orden total.
- La inclusión $\subseteq$ sobre 2^S es un orden parcial, pero no total si $S \neq \emptyset$.

4.3. Relaciones de equivalencia

Las relaciones de equivalencia permiten formalizar la idea de que dos elementos son esencialmente "lo mismo" bajo cierto criterio. Estas relaciones son centrales para definir particiones de conjuntos, clases de objetos indistinguibles y estructuras algebraicas como clases de congruencia, espacios cociente o tipos abstractos de datos.

En ciencias de la computación, se usan para identificar elementos equivalentes en optimización de código, hashing, agrupamiento de datos y en la definición de semánticas de programas (por ejemplo, dos expresiones que se evalúan igual en todo contexto son consideradas equivalentes).

Relación de equivalencia

Una relación R sobre un conjunto A es una **relación de equivalencia** si es reflexiva, simétrica y transitiva.

Ejemplo: congruencia módulo p. Para $p \geq 2$ y $a, b \in \mathbb{Z}$,

$$a \equiv_p b \quad \Leftrightarrow \quad p \mid (a - b)$$

es una relación de equivalencia.

4.3.1. Clases de equivalencia

Una de las consecuencias más interesantes de tener una relación de equivalencia es que nos permite agrupar los elementos del conjunto en subconjuntos disjuntos llamados **clases**

Tipo de relación	Reflexiva	Simétrica	Antisimétrica	Transitiva	Ejemplo
Relación de equivalencia	✓	✓	–	✓	Congruencia módulo n
Orden parcial	✓	–	✓	✓	$\subseteq$, $\leq$ en $\mathbb{N}$
Orden total	✓	–	✓	✓	$\leq$ en $\mathbb{Z}$

Tabla 4.1: Relación entre propiedades y tipos de relaciones comunes

de equivalencia. Cada clase está formada por todos los elementos que son equivalentes entre sí respecto a la relación R. Esta idea permite clasificar elementos según algún criterio de equivalencia y es muy útil tanto en matemáticas como en ciencias de la computación, por ejemplo, al agrupar datos, detectar redundancias o definir estados equivalentes en autómatas.

Clase de equivalencia

Dada una relación de equivalencia R sobre un conjunto A y un elemento $a \in A$, la **clase de equivalencia** de a, denotada $[a]_R$, es:

$$[a]_R = \{b \in A \mid (a, b) \in R\}$$

Ejemplo: módulo 3.

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

Lema: Caracterizaciones de equivalencia

Si R es una relación de equivalencia sobre A, entonces para cualquier $a, b \in A$ se cumple:

$$(a, b) \in R \quad \Leftrightarrow \quad [a]_R = [b]_R \quad \Leftrightarrow \quad [a]_R \cap [b]_R \neq \emptyset$$

Demostración. Demostremos las implicaciones una por una:

$(1 \Rightarrow 2)$ Supongamos que $(a, b) \in R$. Queremos probar que $[a]_R = [b]_R$. Sea $x \in [a]_R$, es decir, $(a, x) \in R$. Como $(a, b) \in R$ y R es simétrica y transitiva, se sigue que $(b, x) \in R$, por lo tanto $x \in [b]_R$, y así $[a]_R \subseteq [b]_R$. Análogamente, $(b, a) \in R$ (por simetría), y el mismo argumento muestra que $[b]_R \subseteq [a]_R$. Entonces $[a]_R = [b]_R$.

$(2 \Rightarrow 3)$ Supongamos que $[a]_R = [b]_R$. Entonces tienen al menos un elemento en común (por ejemplo $a \in [a]_R$), por lo tanto $[a]_R \cap [b]_R \neq \emptyset$.

$(3 \Rightarrow 1)$ Supongamos que $[a]_R \cap [b]_R \neq \emptyset$. Entonces existe $x \in A$ tal que $(a, x) \in R$ y $(b, x) \in R$. Como R es simétrica, $(x, b) \in R$ y luego por transitividad, $(a, b) \in R$.

Por lo tanto, todas las implicaciones se cumplen y las tres condiciones son equivalentes. □

4.4. Relaciones de equivalencia y particiones

Vimos cómo una relación de equivalencia nos permite agrupar los elementos de un conjunto en clases de equivalencia, donde todos los elementos de una clase están relacionados entre sí. Esta estructura naturalmente nos lleva a la noción de **partición**, que es una forma de dividir un conjunto en subconjuntos que no se superponen y que cubren completamente al conjunto original.

Las clases de equivalencia inducidas por una relación de equivalencia forman una partición del conjunto. Inversamente, toda partición de un conjunto define una relación de equivalencia entre elementos.

Partición

Una **partición** de un conjunto A es una colección de subconjuntos no vacíos, disjuntos dos a dos, cuya unión es A.

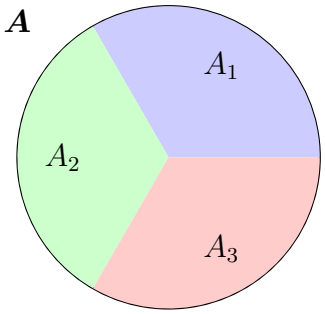


Figura 4.2: En esta ilustración, el conjunto A está dividido en tres subconjuntos disjuntos A_1, A_2 y A_3, que juntos forman una partición de A. Cada elemento del conjunto A pertenece a uno y solo uno de los subconjuntos.

Ejemplo.

- $\{\{2\}, \{1, 3, 4, 5, \ldots\}\}$ es una partición de $\mathbb{N}$.
- $\{\text{pares}, \text{impares}\}$ también lo es.

Teorema: Relaciones de equivalencia inducen particiones

Si R es una relación de equivalencia sobre A, entonces:

$$\{[a]_R \mid a \in A\}$$

es una partición de A. Esta colección se llama **conjunto cociente**, denotado $A/_R$.

Demostración. Sea R una relación de equivalencia sobre A. Debemos probar que el conjunto de clases de equivalencia $\{[a]_R \mid a \in A\}$ cumple con las tres condiciones de una partición:

- **No vacías:** Por reflexividad, todo $a \in A$ pertenece a su propia clase $[a]_R$, ya que $(a, a) \in R$. Por lo tanto, cada clase $[a]_R$ contiene al menos un elemento.
- **Disjuntas dos a dos:** $[a]_R \cap [b]_R \neq \emptyset \Leftrightarrow [a]_R = [b]_R$. Por lo tanto, dos clases de equivalencia o son idénticas o son disjuntas.
- **Cubren A:** Para todo $a \in A$, $a \in [a]_R$ por reflexividad. Entonces cada elemento pertenece a alguna clase.

Por lo tanto, $\{[a]_R \mid a \in A\}$ es una partición de A. □

Ejemplo. Dado la relación de congruencia módulo 3, la siguiente es una partición de $\mathbb{Z}$:

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

Teorema: Particiones inducen relaciones de equivalencia

Dada una partición $\{A_i | i \in I\}$ de un conjunto A, existe una relación de equivalencia R sobre A cuyas clases de equivalencia son precisamente los conjuntos A_i.

Demostración. Definamos una relación R sobre A de la siguiente manera: para $x, y \in A$, decimos que $(x, y) \in R$ si y sólo si existen $i \in I$ tal que $x \in A_i$ y $y \in A_i$, es decir, si x y y pertenecen al mismo conjunto de la partición.

Vamos a demostrar que R es una relación de equivalencia:

- **Reflexiva:** Como $x \in A_i$ para algún i, se tiene que $(x, x) \in R$ porque x y x pertenecen al mismo conjunto.
- **Simétrica:** Si $(x, y) \in R$, entonces $x, y \in A_i$ para algún i, por lo tanto también $(y, x) \in R$.
- **Transitiva:** Si $(x, y) \in R$ y $(y, z) \in R$, entonces $x, y \in A_i$ y $y, z \in A_j$ para algunos $i, j \in I$. Pero como y pertenece a ambos y los A_i son disjuntos dos a dos, se deduce que $i = j$, y por lo tanto $x, z \in A_i$, así que $(x, z) \in R$.

Así, R es una relación de equivalencia y sus clases de equivalencia son exactamente los conjuntos A_i. □

4.4.1. Relaciones n-arias

Hasta ahora nos hemos centrado en las relaciones binarias, es decir, aquellas que vinculan pares de elementos. Sin embargo, en muchas aplicaciones necesitamos expresar relaciones

entre tres o más elementos. Esto nos lleva al concepto de **relaciones n-arias**, que generalizan la noción de relación binaria a tuplas de longitud arbitraria.

Estas relaciones aparecen frecuentemente en bases de datos relacionales (donde una tabla puede tener múltiples columnas) y en especificaciones lógicas de sistemas (como restricciones que involucran varios objetos).

Relación n-aria

Una **relación n-aria** sobre un conjunto A es un subconjunto de $A^n = A \times A \times \cdots \times A$ (n veces).

Ejemplos.

- La propiedad de ser primo es una relación unaria sobre $\mathbb{N}$.
- La suma puede verse como una relación ternaria: $(a, b, c) \in \mathbb{N}^3$ tal que $a + b = c$.
- En una base de datos, una tabla con atributos (ID, nombre, curso) puede verse como una relación ternaria entre personas y cursos.
- En programación lógica, una relación cuaternaria puede expresar que un algoritmo transforma una entrada y un estado inicial en una salida y un nuevo estado.

¿Cuántas relaciones n-arias hay sobre un conjunto finito A?

Primero observemos que una relación n-aria sobre A es simplemente un subconjunto del conjunto de todas las n-uplas posibles de elementos de A, es decir, de A^n. Si el conjunto A tiene k elementos (es decir, $|A| = k$), entonces, para cada elementos se puede decidir si está o no en la relación, el número total de relaciones n-arias posibles sobre A es: 2^{k^n}

Capítulo 5

Funciones

5.1. Operaciones sobre Relaciones

En esta sección abordaremos cómo se pueden manipular las relaciones mediante operaciones, algunas heredadas de la teoría de conjuntos y otras específicas de las relaciones. Esto permite construir relaciones más complejas a partir de otras más simples, analizar sus propiedades y establecer conexiones entre distintas estructuras.

5.1.1. Operaciones Clásicas: Unión, Intersección y Diferencia

Dado que una relación R sobre un conjunto A es un subconjunto de $A \times A$, podemos aplicar las operaciones usuales de conjuntos para combinarlas:

- **Unión:** $R \cup R'$ contiene todos los pares que pertenecen a R, a R' o a ambos.
- **Intersección:** $R \cap R'$ contiene solo los pares comunes a ambas relaciones.
- **Diferencia:** $R \setminus R'$ contiene los pares que están en R pero no en R'.

Estas operaciones nos permiten combinar o contrastar relaciones distintas, como por ejemplo, intersectar relaciones de amistad y colaboración en redes sociales.

5.1.2. Operaciones Específicas sobre Relaciones

Además de las operaciones anteriores, existen operaciones específicas para relaciones:

- **Inversión:** La relación inversa de R se denota R^{-1} y consiste en invertir el orden de todos los pares: $R^{-1} = \{(b, a) \mid (a, b) \in R\}$.
- **Composición:** Dadas dos relaciones $R \subseteq A \times B$ y $R' \subseteq B \times C$, su composición es:

$$R' \circ R = \{(a, c) \mid \exists b.(a, b) \in R \wedge (b, c) \in R'\}.$$

  Esta operación se interpreta como "relación indirecta" a través de un intermediario.

- **Potencia:** Si R es una relación sobre un conjunto A, definimos inductivamente:

$$R^1 = R, \quad R^{n+1} = R^n \circ R.$$

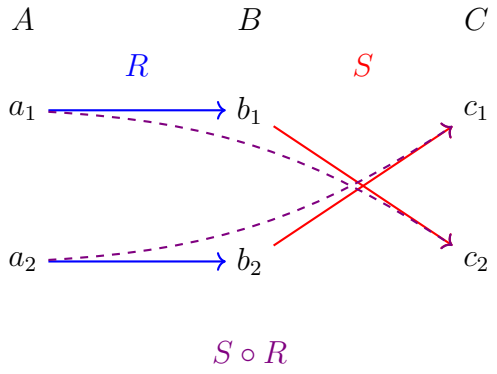


Figura 5.1: Composición de relaciones. Las flechas azules representan la relación $R \subseteq A \times B$ y las rojas la relación $S \subseteq B \times C$. La composición $S \circ R \subseteq A \times C$ se indica con flechas punteadas moradas, que conectan elementos de A con C cuando existe un elemento intermedio en B que los vincula a través de R y S.

Ejercicios resueltos

1. Transitividad y potencias:

Demostrar que R es transitiva si y solo si $R^n \subseteq R$ para todo $n \geq 1$.
(⇒) Supongamos que R es transitiva. Usamos inducción sobre n:
Caso base: $n = 1$. Claramente $R^1 = R \subseteq R$.
Paso inductivo: Supongamos que $R^k \subseteq R$. Entonces:

$$R^{k+1} = R^k \circ R \subseteq R \circ R \subseteq R$$

por transitividad. Así, por inducción, $R^n \subseteq R$ para todo $n \geq 1$.
(⇐) Supongamos que $R^n \subseteq R$ para todo $n \geq 1$. En particular, $R^2 \subseteq R$ implica que $(a, b) \in R$ y $(b, c) \in R$ lleva a $(a, c) \in R^2 \subseteq R$, es decir, R es transitiva.

2. Simetría:

Demostrar que R es simétrica si y solo si $R = R^{-1}$.
(⇒) Si R es simétrica, entonces $(a, b) \in R$ implica $(b, a) \in R$, por lo tanto $R \subseteq R^{-1}$ y también $R^{-1} \subseteq R$, ya que si $(a, b) \in R^{-1}$ entonces $(b, a) \in R$ y por simetría $(a, b) \in R$. Así, $R = R^{-1}$.
(⇐) Si $R = R^{-1}$, entonces todo $(a, b) \in R$ satisface que $(b, a) \in R$, lo cual es precisamente la definición de simetría.

3. Potencias de relación simétrica:

Sea R simétrica. Demostrar que R^n es simétrica para todo $n \geq 1$.
Demostramos por inducción sobre n que R^n es simétrica.
Caso base: $n = 1$. $R^1 = R$ es simétrica por hipótesis.
Paso inductivo: Supongamos que R^k es simétrica. Consideremos $R^{k+1} = R^k \circ R$. Tomemos $(a, c) \in R^{k+1}$, entonces existe b tal que $(a, b) \in R^k$ y $(b, c) \in R$. Por hipótesis, $(b, a) \in R^k$ y $(c, b) \in R$. Entonces $(c, a) \in R^{k+1}$, y se concluye que R^{k+1} también es simétrica.
Así, R^n es simétrica para todo $n \geq 1$.

5.2. Clausuras

Una clausura de una relación consiste en extenderla para que satisfaga una propiedad deseada (reflexividad, simetría o transitividad), agregando la menor cantidad posible de pares.

Clausura

Dada una relación R sobre un conjunto A, su clausura reflexiva/simétrica/transitiva es la *menor* relación (en términos de inclusión) que contiene a R y que es reflexiva/simétrica/transitiva, respectivamente.

5.2.1. Clausura Reflexiva

Se obtiene agregando todos los pares (a, a) para cada $a \in A$ que no estaban ya en R.

Ejemplo

La clausura reflexiva de la relación $<$ sobre $\mathbb{N}$ es $\leq$.

5.2.2. Clausura Simétrica

Para obtener la clausura simétrica de una relación R, se agregan todos los pares (b, a) tales que $(a, b) \in R$ pero $(b, a) \notin R$.

Ejemplo

La clausura simétrica de $R = \{(a, b) \mid a > b\}$ es $R \cup R^{-1} = \{(a, b) \mid a \neq b\}$.

5.2.3. Clausura Transitiva

La clausura transitiva de R incluye todos los pares que se pueden alcanzar transitivamente:

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

La clausura reflexiva-transitiva se denota $R^* = R^+ \cup \{(a, a) \mid a \in A\}$.

Ejemplo

La clausura transitiva de la relación "sucesor" en $\mathbb{N}$ es la relación de orden $>$.

Lema

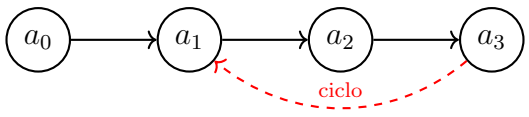
Sea R una relación sobre un conjunto finito A con $|A| = n$. Demuestre que:

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

Demostración. La clausura transitiva R^+ contiene todos los pares (a, b) tales que existe un camino desde a hasta b siguiendo pasos de la relación R.

En un conjunto finito con n elementos, cualquier secuencia de pasos que conecta dos elementos no necesita tener más de n pasos: si un camino tiene más de n nodos, entonces alguno se repite (por el principio del palomar que veremos en la próxima clase), y el camino puede acortarse eliminando ciclos.

Por lo tanto, para encontrar todos los pares conectados transitivamente, basta con considerar composiciones de R hasta longitud n. □



Se repite el nodo a_1: el camino contiene un ciclo

Figura: Un camino de más de n pasos en un conjunto de n elementos debe repetir nodos.

Ejemplo: Clausura transitiva en planificación de tareas

Supongamos que tenemos un conjunto de tareas $A = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ en un proyecto. La relación $R \subseteq A \times A$ indica dependencias directas: si $(x, y) \in R$, entonces la tarea x debe completarse antes de comenzar la tarea y.
Sean las dependencias directas:

$$R = \{(\mathrm{A}, \mathrm{B}), (\mathrm{B}, \mathrm{C}), (\mathrm{C}, \mathrm{D})\}$$

Objetivo: Calcular la clausura transitiva R^+, que representa todas las tareas que deben realizarse antes de otra, directa o indirectamente.
Cálculo de R^+:

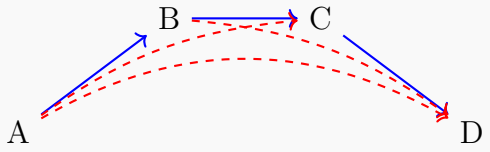
$$R^1 = R = \{(\mathrm{A}, \mathrm{B}), (\mathrm{B}, \mathrm{C}), (\mathrm{C}, \mathrm{D})\}$$

$$R^2 = \{(\mathrm{A}, \mathrm{C}), (\mathrm{B}, \mathrm{D})\}$$

$$R^3 = \{(\mathrm{A}, \mathrm{D})\}$$

$$R^+ = R \cup R^2 \cup R^3 = \{(\mathrm{A}, \mathrm{B}), (\mathrm{B}, \mathrm{C}), (\mathrm{C}, \mathrm{D}), (\mathrm{A}, \mathrm{C}), (\mathrm{B}, \mathrm{D}), (\mathrm{A}, \mathrm{D})\}$$

Representación gráfica:



Azul: dependencias directas. Rojo punteado: dependencias inferidas transitivamente.

5.3. Funciones

Función

Una función $f : A \to B$ es una relación que a cada elemento $a \in A$ le asigna *un único* elemento $b \in B$. Se denota $f(a) = b$.

Dado $a \in A$ escribimos $f(a)$ para denotar al *único* $b \in B$ tal que $(a, b) \in f$. En este caso decimos que b es la **imagen** de a (o que a es una **preimagen** de b).

El conjunto A se llama *dominio* y el conjunto $\{b \in B \mid \exists a \in A, f(a) = b\}$ se llama *rango*.

5.3.1. Clasificación de funciones

- **Inyectiva (uno a uno):** Si $a_1 \neq a_2$ implica $f(a_1) \neq f(a_2)$.
- **Sobreyectiva (sobre):** Si todo $b \in B$ tiene una preimagen en A.
- **Biyectiva:** Si es inyectiva y sobreyectiva.

Ejemplos

La función $f : \mathbb{N}_0 \to \mathbb{N}_0$ dada por $f(n) = 2n$ es inyectiva pero no sobreyectiva.
La función $f : \mathbb{N}_0 \to \mathbb{N}_0$ dada por $f(0) = 1$, $f(n) = n - 1$ para $n > 0$, es sobreyectiva pero no inyectiva.

5.3.2. Operaciones con funciones

- **Composición:** Dados $f : A \to B$ y $g : B \to C$, se define $g \circ f : A \to C$ por $g(f(a))$.
- **Inversa:** Si f es biyectiva, entonces existe $f^{-1} : B \to A$ tal que $f^{-1}(f(a)) = a$.

Ejercicio Propuesto

Demuestre que si f y g son inyectivas (resp. sobreyectivas), entonces $g \circ f$ también lo es.

5.4. Crecimiento de Funciones

Para comparar el comportamiento asintótico de funciones, usamos la notación O:

Notación O

Dadas funciones f y g, decimos que $f(x)$ es $O(g(x))$ si existen constantes $c, k > 0$ tales que:

$$f(x) \leq c \cdot g(x), \quad \forall x > k.$$

Ejercicios Propuestos

- Demuestre que $f(x) = x^2 + 2x + 1$ es $O(x^2)$.
- Demuestre que todo polinomio de grado n con coeficientes reales es $O(x^n)$.

5.4.1. Composición de funciones en términos de O

Teorema

Si $f_1 = O(g_1(x))$ y $f_2 = O(g_2(x))$, entonces:

- $f_1(x) + f_2(x) = O(\text{máx}\{g_1(x), g_2(x)\})$.
- $f_1(x) \cdot f_2(x) = O(g_1(x) \cdot g_2(x))$.

Demostración. Por hipótesis, existen constantes positivas c_1, k_1 tales que

$$f_1(x) \leq c_1 \cdot g_1(x), \quad \forall x > k_1,$$

y constantes c_2, k_2 tales que

$$f_2(x) \leq c_2 \cdot g_2(x), \quad \forall x > k_2.$$

Sea $k = \text{máx}\{k_1, k_2\}$. Entonces para todo $x > k$, ambas desigualdades se cumplen.

1. Suma:

$$f_1(x) + f_2(x) \leq c_1 g_1(x) + c_2 g_2(x) \leq (c_1 + c_2) \cdot \text{máx}\{g_1(x), g_2(x)\}.$$

Por lo tanto, $f_1(x) + f_2(x) = O(\text{máx}\{g_1(x), g_2(x)\})$.

2. Producto:

$$f_1(x) \cdot f_2(x) \leq (c_1 g_1(x)) \cdot (c_2 g_2(x)) = (c_1 c_2) \cdot g_1(x) g_2(x).$$

Por lo tanto, $f_1(x) \cdot f_2(x) = O(g_1(x) \cdot g_2(x))$. □

Ejercicios Propuestos

- Demuestre que $(x + 1)\log(x^2 + 1) + 3x^2$ es $O(x^2)$.

Parte IV

Combinatoria

5.5. Principios de Conteo

En muchos contextos cotidianos, nos enfrentamos a situaciones en las que debemos contar posibilidades: ¿cuántas combinaciones de ingredientes puede tener un sándwich? ¿cuántas claves distintas puede tener un sistema? ¿de cuántas maneras puedo organizar un grupo de estudiantes?

A este tipo de preguntas responde la **combinatoria**, una rama de la matemática dedicada al estudio de métodos sistemáticos de conteo. En general, busca responder preguntas del tipo:

- ¿*De cuántas maneras* se puede llevar a cabo un procedimiento dado?
- ¿*Cuántas configuraciones* distintas admite una estructura bajo ciertas restricciones?

5.5.1. Principios de conteo básico

Para abordar problemas de conteo, existen dos principios fundamentales: la **regla del producto** y la **regla de la suma**. Ambas nos permiten descomponer problemas complejos en pasos más simples y sistemáticos.

Regla del Producto

La regla del producto se utiliza cuando una tarea se puede descomponer en múltiples tareas secuenciales (independientes).

Regla del Producto:

Si un procedimiento puede ser dividido en una secuencia de m tareas $T_1, \ldots, T_m$, y cada tarea T_i puede ser realizada de n_i maneras, sin importar cómo se han realizado las tareas anteriores, entonces el procedimiento puede ser realizado de $n_1 n_2 \cdots n_m$ maneras.

Esta regla refleja lo que hacemos al tomar decisiones sucesivas, cada una con varias opciones. Por ejemplo, si queremos formar un código con una letra seguida de un número, y hay 26 letras y 10 números posibles, entonces hay $26 \cdot 10 = 260$ combinaciones posibles.

Importante: Para que la regla del producto sea válida, es indispensable que cada etapa pueda ser realizada de n_i maneras **independientemente** de cómo se realizaron las tareas anteriores. Es decir, la elección en una etapa no debe restringir o afectar las posibilidades en las siguientes.

Regla de la Suma

Regla de la Suma:

Si que una tarea puede realizarse de n_1 o de n_2 ... o de n_m maneras, donde ninguna de las n_i maneras coincide con alguna de las n_j maneras (para $i \neq j$ en $1 \dots m$). Entonces la tarea puede realizarse de $n_1 + n_2 + \cdots + n_m$ maneras.

Esta regla corresponde a una elección entre alternativas mutuamente excluyentes. Es importante que los casos no se superpongan para evitar contar dos veces.

5.5.2. Aplicaciones de la Regla del Producto

Uno de los usos más inmediatos de esta regla es el conteo de permutaciones.

Permutaciones de n elementos:

Al ordenar una lista de n objetos distintos, el primer lugar puede ser ocupado por cualquiera de los n elementos, el segundo por cualquiera de los $n-1$ restantes, y así sucesivamente. Entonces:

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

Cantidad de funciones:

¿Cuántas funciones existen desde $A = \{a_1, \dots, a_n\}$ hacia $B = \{b_1, \dots, b_m\}$?

Para cada elemento a_i de A, podemos elegir libremente una imagen en B, es decir, hay m opciones. Como hay n elementos en A, aplicamos la regla del producto:

$$\text{Total de funciones} = m^n.$$

Cantidad de subconjuntos:

Para cada elemento de un conjunto de n elementos, podemos decidir si está o no en un subconjunto. Esta es una decisión binaria independiente por cada elemento, por lo tanto:

$$2^n \text{ subconjuntos en total.}$$

5.5.3. Forma Alternativa de la Regla del Producto

La regla del producto también puede verse como una propiedad de los productos cartesianos, lo que permite extender su aplicación desde procedimientos secuenciales hasta estructuras matemáticas más abstractas.

Regla del Producto

Dado m conjuntos finitos $A_1, \ldots, A_m$:

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

Para conectar esta formulación de la regla del producto con la definición original basada en tareas, debemos interpretar a cada conjunto A_i como el **conjunto de maneras de realizar la tarea** T_i. Así, el producto cartesiano $A_1 \times \cdots \times A_m$ representa todas las combinaciones posibles de realizar las tareas T_1 hasta T_m, y su cardinalidad corresponde exactamente al número total de procedimientos posibles.

5.5.4. Aplicaciones de la Regla de la Suma

Ejemplo:

¿Cuántos números de tres cifras comienzan con 3 o con 4?
Los números de tres cifras que comienzan con 3 son 100 en total, y lo mismo los que comienzan con 4. Como no hay solapamiento entre ambos grupos:

$$\text{Total} = 100 + 100 = 200.$$

Ejemplo: Subcomités de senadores

Supongamos que hay n senadores en un comité y queremos contar cuántos subcomités distintos pueden formarse.
Podemos tener subcomités de tamaño 0 (ningún miembro) hasta tamaño n (todos los senadores). La cantidad de subcomités de tamaño k está dada por:

$$\binom{n}{k}$$

Por la regla de la suma, el número total de subcomités posibles es:

$$\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{n} = 2^n$$

Esto coincide con el número total de subconjuntos del conjunto de senadores. Cada subconjunto representa un subcomité posible, lo cual establece una conexión directa entre la teoría de conjuntos y la combinatoria.

Observación: También podríamos haber aplicado la regla del producto. Cada senador tiene dos opciones: o bien pertenece al subcomité o no. Por lo tanto, hay $2 \cdot 2 \cdot \cdots \cdot 2 = 2^n$ maneras de tomar esa decisión para todos, lo que nuevamente nos da 2^n subcomités posibles.

5.5.5. Forma Alternativa de la Regla de la Suma

Una manera de entender la regla de la suma es a través de conjuntos disjuntos. En lugar de pensar en tareas o casos, podemos considerar colecciones de elementos que no se superponen.

Forma alternativa de la regla de la suma

Si $A_1, A_2, \ldots, A_m$ es una colección de conjuntos finitos y disjuntos dos a dos ($A_i \cap A_j = \emptyset$ para $i \neq j$), entonces

$$|A_1 \cup A_2 \cup \ldots \cup A_m| \quad == \quad |A_1| + |A_2| + \cdots + |A_m|$$

Esta formulación es equivalente a la regla de la suma que ya discutimos, pero expresada en términos de conjuntos. Así como antes sumábamos la cantidad de maneras de realizar tareas disjuntas, ahora sumamos la cantidad de elementos de conjuntos disjuntos.

Importante:

La regla de la suma solo aplica directamente si los casos son disjuntos. Si pueden solaparse, necesitamos una corrección: el principio de inclusión-exclusión.

5.6. Principio de Inclusión-Exclusión

Principio de Inclusión-Exclusión (dos conjuntos):

Sean A y B conjuntos finitos:

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

Este principio corrige la doble contabilidad de los elementos comunes al sumar $|A| + |B|$. Luego generalizaremos el principio de inclusión-exclusión para más de dos conjuntos.

5.6.1. Aplicaciones del Principio de Inclusión-Exclusión

Ejemplo: ¿Cuántos números de tres dígitos comienzan con 3 o terminan en 4?

Sea A el conjunto de números de tres dígitos que comienzan con 3, y B el conjunto de números de tres dígitos que terminan en 4.

- Hay 100 números que comienzan con 3: desde 300 hasta 399. Entonces $|A| = 100$.
- Hay 90 números que terminan en 4: desde 104 hasta 994, en pasos de 10 (por ejemplo: 104, 114, ..., 994). Entonces $|B| = 90$.
- Hay 10 números que comienzan con 3 y terminan en 4: 304, 314, 324, ..., 394. Entonces $|A \cap B| = 10$.

Aplicando el principio de inclusión-exclusión:

$$|A \cup B| = |A| + |B| - |A \cap B| = 100 + 90 - 10 = 180.$$

Por lo tanto, hay 180 números de tres dígitos que comienzan con 3 o terminan en 4.

Ejemplo: ¿Cuántos números entre 1 y 100 no son múltiplos de 3 ni de 5?

Sea U el conjunto de números del 1 al 100. Queremos contar los que no pertenecen al conjunto de múltiplos de 3 ni de 5.

- Hay $\left\lfloor \frac{100}{3} \right\rfloor = 33$ múltiplos de 3.
- Hay $\left\lfloor \frac{100}{5} \right\rfloor = 20$ múltiplos de 5.
- Hay $\left\lfloor \frac{100}{15} \right\rfloor = 6$ múltiplos de 3 y 5, es decir, múltiplos de 15.

Entonces el número total de múltiplos de 3 o 5 es:

$$|A \cup B| = 33 + 20 - 6 = 47.$$

Por lo tanto, el número de enteros entre 1 y 100 que no son múltiplos de 3 ni de 5 es:

$$|U \setminus (A \cup B)| = 100 - 47 = 53.$$

Extensión a tres conjuntos

Demostremos usando el principio de inclusión-exclusión para dos conjuntos, la siguiente extensión a tres conjuntos:

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

Demostración. Llamemos $D = A \cup B$. Entonces podemos aplicar el principio de inclusión-exclusión para dos conjuntos dos veces: Primero aplicamos inclusión-exclusión para obtener $|D| = |A \cup B|$:

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

Ahora consideramos $D \cup C = (A \cup B) \cup C = A \cup B \cup C$, y aplicamos nuevamente inclusión-exclusión:

$$|A \cup B \cup C| = |A \cup B| + |C| - |(A \cup B) \cap C|$$

Para calcular $|(A \cup B) \cap C|$, usamos la propiedad distributiva de la intersección:

$$(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$$

Entonces, aplicamos inclusión-exclusión a esta unión:

$$|(A \cup B) \cap C| = |A \cap C| + |B \cap C| - |A \cap B \cap C|$$

Sustituyendo todo en la fórmula general $|A \cup B \cup C|$, da el resultado esperado. Esto completa la demostración utilizando únicamente la versión del principio de inclusión-exclusión para dos conjuntos. □

Ejercicio: combinaciones de billetes

Supongamos que existen sólo billetes de \$2, \$3 y \$5. Queremos saber cuántas cantidades enteras distintas entre \$1 y \$100 inclusive pueden pagarse usando sólo un único tipo de billete.

Solución:

- A: cantidades divisibles por 2: $\$2, \$4, ..., \$100$. Hay $\left\lfloor \frac{100}{2} \right\rfloor = 50$.
- B: cantidades divisibles por 3: $\$3, \$6, ..., \$99$. Hay $\left\lfloor \frac{100}{3} \right\rfloor = 33$.
- C: cantidades divisibles por 5: $\$5, \$10, ..., \$100$. Hay $\left\lfloor \frac{100}{5} \right\rfloor = 20$.
- $|A \cap B| = \left\lfloor \frac{100}{6} \right\rfloor = 16$
- $|A \cap C| = \left\lfloor \frac{100}{10} \right\rfloor = 10$
- $|B \cap C| = \left\lfloor \frac{100}{15} \right\rfloor = 6$
- $|A \cap B \cap C| = \left\lfloor \frac{100}{30} \right\rfloor = 3$

Aplicando la fórmula de inclusión-exclusión para tres conjuntos:

$$|A \cup B \cup C| = 50 + 33 + 20 - 16 - 10 - 6 + 3 = 74.$$

Hay 74 cantidades distintas entre \$1 y \$100 que pueden pagarse usando sólo un tipo de billete.

Ejercicios Propuestos

- ¿Cuántos strings de largo 7 sobre $\{0, 1\}$ existen?
- ¿Cuántos números de tres dígitos son pares?
- ¿Cuántas funciones inyectivas existen desde un conjunto de n elementos a otro de m elementos (con $m \geq n$)?
- ¿Cuántos strings de largo 10 sobre el alfabeto $\{0, 1\}$ contienen 5 consecutivos 0s?
- ¿Cuántos palíndromos de longitud n existen sobre $\{0, 1\}$?
- ¿Cuántos strings de largo 10 sobre el alfabeto $\{0, 1\}$ contienen 5 consecutivos 0s o 5 consecutivos 1s?
- ¿Cuántos strings de largo 8 sobre el alfabeto $\{0, 1\}$ contienen 3 consecutivos 0s o 4 consecutivos 1s?

5.7. Principio del Palomar

El *principio del palomar* es una de las ideas más simples y poderosas de la combinatoria. También conocido como el *principio de las cajas de Dirichlet* , establece que si tratamos de colocar más objetos que contenedores disponibles, al menos uno de los contenedores deberá tener más de un objeto.

Esta idea, aparentemente trivial, tiene una sorprendente variedad de aplicaciones en matemáticas, informática, teoría de números y lógica. Nos permite llegar a conclusiones inevitables sobre repeticiones, coincidencias o superposiciones, incluso cuando no tenemos información específica sobre la distribución.

Más allá de su simplicidad, el principio del palomar juega un rol clave en la teoría de la computación. En particular, aparece en formulaciones fundamentales sobre la complejidad de ciertos problemas lógicos. Como lo explica Ben Brubaker en un artículo de *Quanta Magazine*, el principio ha servido para definir nuevas clases de problemas computacionales y ha sido una herramienta esencial para mostrar que ciertos problemas no sólo tienen solución, sino que su existencia está garantizada de manera *no constructiva*, es decir, sin que sepamos necesariamente cómo encontrarlas.

Este tipo de razonamientos ha permitido a investigadores como Christos Papadimitriou y Oliver Korten explorar límites fundamentales de la teoría de la complejidad, revelando profundas conexiones entre el principio del palomar y el estudio formal de la dificultad computacional[1].

[1]Ver Ben Brubaker, "How a Problem About Pigeons Powers Complexity Theory,"Quanta Magazine (2025). Disponible en: `https://www.quantamagazine.org/how-a-problem-about-pigeons-powers-complexity-theory-20250404/`

Principio del palomar

Sea $k \in \mathbb{N}$. Si $k+1$ objetos tienen que colocarse en k cajas, entonces habrá al menos una caja que contenga al menos dos objetos.

Ejemplo intuitivo

Si una bandada de 10 palomas vuela hasta un palomar que contiene sólo 9 agujeros, entonces al menos un agujero contendrá a dos palomas.

Aplicaciones básicas

Cumpleaños compartido

En un grupo de 367 personas debe haber al menos dos con el mismo cumpleaños, ya que hay solo 366 días posibles (incluyendo años bisiestos).

Diferencia múltiplo de 10

Dados 11 números enteros, siempre hay dos cuya diferencia es múltiplo de 10.

Considere los restos de la division entera de los 11 números por 10. Como sólo hay 10 restos posibles (0,1,...,9), habrá (al menos) dos números que tienen el mismo resto, y su diferencia será múltiplo de 10.

Divisibilidad entre números enteros

En cualquier conjunto de 51 números enteros entre 1 y 100, existe uno que divide a otro.

Recordar que todo natural puede escribirse de la forma $2^k \cdot b$ para algún $k \in \mathbb{N}_0$ y algún impar $b \in \mathbb{N}$.
Como entre 1 y 100 hay sólo 50 números impares, dentro del conjunto de los 51 enteros debe haber (al menos) dos comparten el mismo b en la descomposición arriba descrita. Luego el que tiene asociado el menor k divide al que tiene asociado el mayor k.

Principio del palomar generalizado

La versión clásica del principio del palomar nos asegura que si hay más objetos que recipientes, entonces alguno de ellos debe contener al menos dos objetos. Pero ¿qué sucede si tenemos aún más objetos, o si queremos estimar no sólo la colisión mínima sino cuántos deben coincidir como mínimo en un contenedor?

Aquí entra en juego el *principio del palomar generalizado*, que nos permite estimar el mínimo número de objetos que necesariamente compartirán al menos una caja en una partición.

Principio generalizado

Si n objetos se colocan en k cajas, entonces existe al menos una caja que contiene al menos $\lceil n/k \rceil$ objetos.

Este resultado es útil cuando el número de objetos y contenedores es arbitrario, y queremos una cota garantizada sobre cuántos deben agruparse juntos. Se usa, por ejemplo, en problemas de distribución, hash, codificación, pruebas de colisiones y razonamiento lógico.

Ejemplo: Personas por mes de nacimiento

Entre 100 personas hay al menos $\lceil 100/12 \rceil = 9$ personas que nacieron en el mismo mes.

Ejercicios Propuestos

- Sea $\{(x_i, y_i, z_i) \mid i \in [1, 9]\}$ un conjunto de 9 puntos distintos con coordenadas enteras en el espacio tridimensional. Demuestre que el punto medio de al menos un par de estos puntos tiene coordenadas enteras.
- ¿Cuál es el mínimo número de alumnos que un curso debe tener para asegurarse de que al menos 6 alumnos reciban la misma nota, si hay 5 posibles notas?
- ¿Cuál es el mínimo número de cartas que se deben tomar de un mazo estándar para asegurarse que al menos 3 tengan la misma pinta?
- (Ramsey) En un grupo de seis personas, cada par de personas son o amigos o enemigos. Demuestre que existen tres mutuos amigos o tres mutuos enemigos.

  Sugerencia: Fije una persona y observe las relaciones con las otras cinco. Luego use el principio del palomar para agruparlas como amigos o enemigos.
- En cualquier fiesta de al menos dos personas, demuestre que existen dos personas que conocen a la misma cantidad de gente en la fiesta.

  Sugerencia: Piense en los posibles valores que puede tomar el número de conocidos de cada persona, y aplique el principio del palomar considerando restricciones por simetría.

5.8. Combinatoria

Muchos problemas cotidianos e incluso científicos se reducen a contar de cuántas formas pueden ocurrir ciertos eventos. Desde generar contraseñas, planificar entrevistas, hasta calcular probabilidades: todos requieren herramientas de conteo.

La **combinatoria** es la rama de las matemáticas que se encarga de estudiar las distintas formas en las que podemos seleccionar o disponer objetos de un conjunto. Dos conceptos fundamentales en este estudio son las **permutaciones** y las **combinaciones**.

Comenzaremos explorando la diferencia entre estas dos ideas fundamentales.

5.8.1. Permutaciones y combinaciones

Permutaciones

Una *r-permutación* es una lista **ordenada** de r objetos distintos tomados de un conjunto de n elementos. El número de r-permutaciones se denota por:

$$\mathsf{P}(n,r) = n \cdot (n-1) \cdot \ldots \cdot (n-r+1) = \frac{n!}{(n-r)!}$$

Ejemplo

¿Cuántos "podios" distintos pueden haber en una competencia con 10 participantes, premiando 1º, 2º y 3º lugar?
Aquí importa el orden, por lo que se trata de una permutación:

$$\mathsf{P}(10,3) = 10 \cdot 9 \cdot 8 = 720$$

Combinaciones

Una *r-combinación* es un subconjunto **no ordenado** de r objetos tomados de un conjunto de n elementos. El número de combinaciones se denota por:

$$\mathsf{C}(n,r) = \frac{n!}{r!\,(n-r)!}$$

Relación entre permutaciones y combinaciones

Cada combinación de r elementos puede ordenarse de $r!$ maneras, por lo que:

$$\mathsf{P}(n,r) = \mathsf{C}(n,r) \cdot r! \quad \Rightarrow \quad \mathsf{C}(n,r) = \frac{\mathsf{P}(n,r)}{r!}$$

Aplicaciones de Permutaciones

Ejemplo: ¿Cuántos números de tres dígitos con cifras impares distintas se pueden formar?

Tenemos 5 dígitos impares: 1, 3, 5, 7, 9. Buscamos permutaciones de 3 elementos:

$$\mathsf{P}(5,3) = 5 \cdot 4 \cdot 3 = 60$$

Ejemplo: ¿Cuántas placas de auto con 4 letras distintas se pueden formar?

Hay 27 letras disponibles. Buscamos $\mathsf{P}(27, 4)$:

$$\mathsf{P}(27, 4) = 27 \cdot 26 \cdot 25 \cdot 24$$

Ejemplo:

¿Cuántas permutaciones de las letras A, B, C, D, E, F, G, H contienen la subcadena ABC?

Aquí el truco está en considerar a ABC como un único elemento. Luego el problema se reduce a contar el número de 6-permutaciones sobre los elementos ABC,D,E,F,G,H, que son $\mathsf{P}(6, 6) = 6!$.

Aplicaciones de Combinaciones

Ejemplo: ¿Cuántos grupos distintos de 3 personas pueden formarse entre 10 participantes?

Como el orden no importa, se trata de una combinación:

$$\mathsf{C}(10, 3) = \frac{10 \cdot 9 \cdot 8}{3 \cdot 2 \cdot 1} = 120$$

Ejemplo: ¿Cuántas manos distintas de póker (5 cartas) pueden formarse desde un mazo de 52 cartas?

$$\mathsf{C}(52, 5) = \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5!}$$

Ejemplo: ¿Cuántos bitstrings de largo n tienen exactamente r 1s?

Esto corresponde a contar de cuántas maneras puedo elegir las r posiciones para los 1s, ya que el bitstring queda completamente definido al rellenar las restantes posiciones con 0s. Éstas son

$$\mathsf{C}(n, r)$$

5.8.2. Permutaciones y combinaciones con repetición

Hasta ahora hemos considerado casos donde los elementos no se repiten. Sin embargo, en muchos contextos (como contraseñas, combinaciones de sabores, códigos, etc.) es posible o incluso necesario permitir repeticiones. Estos casos también pueden analizarse de forma sistemática.

Permutaciones con repetición

Cuando los elementos pueden repetirse, usamos directamente la regla del producto:

$$n^r$$

que representa la cantidad de r-permutaciones con repetición de un conjunto de n elementos.

Ejemplo

¿Cuántas cadenas de 5 bits existen?
Cada posición puede tomar dos valores (0 o 1), así que el total es:

$$2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$$

Combinaciones con repetición

En un conjunto de n elementos, la cantidad de r-combinaciones con repetición está dada por:

$$\mathsf{C}\,(n + r - 1, r) = \frac{(n + r - 1)!}{r!(n - 1)!}$$

Demostración. Para contar cuántas formas hay de elegir r elementos con repetición permitida entre n tipos distintos (sin importar el orden), usamos una representación mediante el método de *barras y estrellas*.

Pensamos en el problema como distribuir r bolas indistinguibles (los elementos elegidos) en n cajas (los tipos). Cada configuración corresponde a una partición de r elementos en n grupos, donde los grupos pueden tener tamaño cero (porque podemos no elegir cierto tipo).

Para representar esto visualmente, usamos:

- r estrellas $*$ para representar los elementos seleccionados.
- $n - 1$ barras verticales $|$ para separar los tipos.

Por ejemplo, la secuencia

$$\star\star\,|\,\star\,||\,\star\star\star$$

representa la combinación con repetición donde se eligen 2 del primer tipo, 1 del segundo, 0 del tercero y 4 del cuarto (si $n = 4$ y $r = 7$).

debemos contar cuántas maneras hay de reordenar 3 estrellas y 7 barras. En total, hay 3 + 7 = 10 3+7=10 símbolos, y debemos elegir cuáles 3 de ellos serán las estrellas

Una configuración es una secuencia de estos símbolos. Debemos contar cuántas maneras hay de reordenar las estrellas y las barras. En total, hay $r + n - 1$ símbolos, y debemos elegir cuáles 3 de ellos serán las estrellas La cantidad total de secuencias distintas de r estrellas y $n - 1$ barras es:

$$\mathsf{C}\,(r + (n - 1), r) = \mathsf{C}\,(n + r - 1, r)$$

□

Ejemplo: Helados

Ejemplo: Una tienda vende 8 sabores de helado. Queremos formar un cono con 3 bolas, permitiendo repeticiones. Entonces:

- $n = 8$ tipos (sabores)
- $r = 3$ bolas a elegir

Una configuración posible podría representarse así:

$$|**||*|||||$$

Esto indica 0 bolas del primer sabor, 2 del segundo, 0 del tercero, 1 del cuarto, y 0 de los sabores restantes.
La cantidad total de configuraciones es:

$$\mathsf{C}(8+3-1,3) = \mathsf{C}(10,3) = 120$$

Por tanto, hay 120 formas distintas de elegir 3 bolas de helado entre 8 sabores, permitiendo repeticiones.

5.9. Pruebas combinatoriales

En combinatoria, existen dos enfoques principales para justificar fórmulas o identidades:

- **Demostraciones algebraicas:** se basan en manipular expresiones simbólicas, factoriales y fórmulas usando álgebra básica. Son precisas y sintéticas, pero pueden no entregar intuición sobre el significado de la identidad.
- **Demostraciones combinatorias:** explican la identidad interpretándola como una forma de contar elementos de un conjunto desde dos perspectivas distintas. Este tipo de demostración suele ser más intuitiva, ya que da sentido a la fórmula en términos de objetos concretos y elecciones.

Ambos métodos son válidos y muchas veces se complementan: el algebraico da verificación formal, el combinatorio aporta comprensión.

Identidades Combinatorias

Simetría del binomio

$$\mathsf{C}(n,r) = \mathsf{C}(n,n-r)$$

Demostración algebraica:

$$\mathsf{C}(n,r) = \frac{n!}{r!\,(n-r)!} = \frac{n!}{(n-r)!\,\underbrace{(n-(n-r))!}_{=r}} = \mathsf{C}(n,n-r)$$

Demostración combinatoria:

Elegir r elementos de un conjunto de n es equivalente a elegir cuáles $n-r$ se descartan. Por lo tanto, hay la misma cantidad de subconjuntos de tamaño r que de tamaño $n-r$.

Suma binomial

$$\sum_{k=0}^{n} \mathsf{C}(n,k) = 2^n$$

Demostración algebraica: Esta identidad se deduce evaluando la expansión binomial con $x = y = 1$:

$$(x+y)^n = \sum_{k=0}^{n} \mathsf{C}(n,k)\, x^k y^{n-k} \quad \Rightarrow \quad (1+1)^n = \sum_{k=0}^{n} \mathsf{C}(n,k)$$

Demostración combinatoria: Cada término $\mathsf{C}(n,k)$ cuenta los subconjuntos de tamaño k de un conjunto de n elementos. La suma total cuenta todos los subconjuntos posibles, es decir, 2^n subconjuntos.

Identidad de Pascal

$$\mathsf{C}(n+1,k) = \mathsf{C}(n,k-1) + \mathsf{C}(n,k)$$

Demostración algebraica: Aplicando la fórmula:

$$\mathsf{C}(n,k-1) + \mathsf{C}(n,k) = \frac{n!}{(k-1)!\,(n-k+1)!} + \frac{n!}{k!\,(n-k)!} = \frac{(n+1)!}{k!\,(n+1-k)!} = \mathsf{C}(n+1,k)$$

Demostración combinatoria: Para formar un subconjunto de k elementos desde $n+1$ elementos, se puede:

- incluir un elemento fijo (e.g. el último), y elegir los $k-1$ restantes entre los primeros n, o
- no incluirlo, y elegir todos los k entre los primeros n.

La cantidad total es la suma de ambas posibilidades.

Ejercicios Propuestos

- Dadas 12 personas, ¿cuántos grupos de 5 personas se pueden formar si hay dos personas, llamemoslas A y B, que insisten en trabajar juntas? (En otros palabras, un grupo de 5 personas es válido si contiene a ambas o a ninguna).

- Demuestre la identidad de Vandermonde:

$$\binom{m+n}{r} = \sum_{k=0}^{r} \binom{m}{r-k}\binom{n}{k}$$

- Reflexiona por qué $\mathsf{C}(n,i)$ es el coeficiente de $x^i y^{n-i}$ en $(x+y)^n$.

5.10. Relaciones de Recurrencia

5.10.1. Introducción

Las **relaciones de recurrencia** son una herramienta poderosa en matemáticas discretas para definir y analizar secuencias de manera recursiva. Estas relaciones aparecen frecuentemente en análisis de algoritmos, conteo, estructuras de datos y resolución de problemas combinatorios.

Definición: Relación de recurrencia

Una relación de recurrencia para una secuencia $\langle a_n \rangle_{n\geq 0}$ es una ecuación que expresa a_n en términos de sus predecesores $a_{n-1}, a_{n-2}, \ldots, a_0$.

Para determinar unívocamente la secuencia debemos dar explícitamente los valores de un número finito de los primeros términos de la secuencia. Éstos se conocen como las **condiciones iniciales** de la recurrencia.

En otras palabras, una relación de recurrencia no es nada más que una **definición recursiva** de una secuencia $\langle a_n \rangle_{n\geq 0}$.

Ejemplo: crecimiento de un sueldo

Laura recibe un sueldo inicial de 2000 y un aumento del 10 % mensual. Sea a_n el sueldo en el mes n:

$$\begin{aligned} a_1 &= 2000 \\ a_n &= 1{,}1 \cdot a_{n-1} \quad \forall n \geq 2 \end{aligned}$$

La forma cerrada es: $a_n = 2000 \cdot 1{,}1^{n-1}$

Ejemplo: Fibonacci

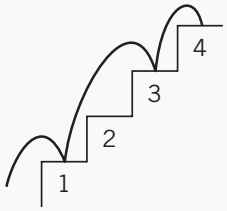
La sucesión de Fibonacci es un ejemplo clásico:

$$f_0 = 0, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2}, \text{ para } n \geq 2.$$

Aplicaciones en problemas de conteo

Problema: Subiendo la escalera

Se desea subir una escalera de n escalones. En cada paso se puede subir uno o dos escalones. ¿De cuántas maneras distintas puede se llegar al escalón n?



Solución:
Para $n = 1$ hay una única forma: dar un paso de un escalón.
Para $n = 2$ hay dos formas:

- dar dos pasos de un escalón,
- dar un solo paso de dos escalones.

Para ver de cuántas maneras se puede subir una escalera de $n \geq 3$ escalones hacemos un análisis de casos:

- Si el primer paso es de un escalón, restan $n - 1$: hay a_{n-1} formas de completarlo.
- Si el primer paso es de dos escalones, restan $n - 2$: hay a_{n-2} formas.

Por la regla de la suma:

$$a_n = a_{n-1} + a_{n-2} \quad \text{para } n \geq 3 \tag{5.1}$$

Resumiendo, tenemos la siguiente relación de recurrencia para calcular el número de formas a_n en las que Juan puede subir una escalera de n escalones:

$$\begin{aligned} a_1 &= 1 \\ a_2 &= 2 \\ a_n &= a_{n-1} + a_{n-2} \quad \forall n \geq 3 \end{aligned}$$

Esta es la **sucesión de Fibonacci**, y tiene muchas aplicaciones en problemas de conteo.

Ejercicios: Palabras binarias sin 00 consecutivos

Sea a_n el número de palabras binarias de largo n sin dos

Solución:
Para construir una palabra binaria válida de largo n (es decir, sin ocurrencias de 00), consideramos los posibles caracteres finales:

- Si la palabra termina en `1`, entonces el prefijo de largo $n-1$ puede ser cualquier palabra válida. Hay a_{n-1} de estas.

- Si termina en `0`, entonces el penúltimo carácter debe ser `1` (para evitar 00). Por lo tanto, el prefijo de largo $n-2$ puede ser cualquier palabra válida, seguida de `10`. Hay a_{n-2} de estas.

Por la regla de la suma:

$$a_n = a_{n-1} + a_{n-2} \quad \forall n \geq 3$$

Condiciones iniciales:

$$\begin{aligned} a_1 &= 2 \quad (\texttt{0}, \texttt{1}) \\ a_2 &= 3 \quad (\texttt{01}, \texttt{10}, \texttt{11}) \end{aligned}$$

Esta es nuevamente la **sucesión de Fibonacci**, pero con condiciones iniciales distintas.

Ejercicio: Conexiones entre personas

Cada vez que una nueva persona entra a una reunión, saluda a todas las personas que ya están. ¿Cuántos apretones de manos se han dado después de que han llegado n personas (una por una)?

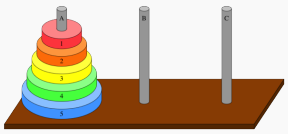
Solución:
Sea a_n el número total de apretones de manos después de que han llegado n personas. Cada nueva persona saluda a todas las que ya llegaron, es decir, realiza $n-1$ apretones. Entonces:

$$\begin{aligned} &a_1 = 0 \\ &a_n = a_{n-1} + (n-1) \quad \text{para } n \geq 2 \end{aligned}$$

Problema: Torres de Hanoi

Las Torres de Hanoi son un puzzle que consiste de 3 barras montadas en un tablero y una serie de discos de diferente tamaño. Inicialmente los discos estás dispuestos como se muestran a continuación: El objetivo del puzzle es pasar todos los discos a otra de las barras con las siguientes restricciones:

- Los discos deben quedar dispuestos de la misma manera (en orden)
- Los discos se pueden pasar entre las barras, de a uno (por lo que en cada paso se puede mover sólo el disco superior de cada barra)
- Al depositar un disco en otra barra, no puede quedar sobre otro de menor tamaño.



¿Cuál es la menor cantidad de movimientos h_n que debo hacer para resolver el puzzle desde una configuración inicial con n discos?

Solución:
La clave está en observar la estructura recursiva del problema.

- Para mover una torre de n discos desde la barra origen a la barra destino:
  1. Primero movemos los $n-1$ discos superiores a la barra auxiliar (requiere h_{n-1} movimientos).
  2. Luego movemos el disco más grande directamente a la barra destino (1 movimiento).
  3. Finalmente, movemos los $n-1$ discos desde la barra auxiliar a la barra destino (otros h_{n-1} movimientos).

Esto da lugar a la siguiente relación de recurrencia:

$$\begin{aligned} h_n &= 2h_{n-1}+1 \quad \forall n \geq 2 \\ h_1 &= 1 \end{aligned}$$

Puede verificarse fácilmente que

$$h_n = 2^n - 1$$

es la forma cerrada(o solución) de la recurrencia.

Asociación de productos (Números de Catalán)

Sea c_n el número de formas de parentizar n factores $a_1 \cdot a_2 \cdot \cdots \cdot a_{n-1} \cdot a_n$. Demostrar la siguiente recurrencia:

$$c_1 = 1$$
$$c_n = \sum_{i=1}^{n-1} c_i \cdot c_{n-i} \quad (n \geq 2)$$

Solución:
Si se agrupa el producto como $(A) \cdot (B)$, donde:

- A es un subproducto de los primeros k factores,
- B es el subproducto de los $n - k$ restantes,

entonces hay c_k formas de parentizar A y c_{n-k} formas de parentizar B, lo que da $c_k \cdot c_{n-k}$ formas para esa partición. Sumando sobre todas las posibles particiones (de $k = 1$ a $n - 1$) obtenemos la recurrencia.

Esta sucesión es conocida como los **números de Catalán**, denotados habitualmente por C_{n-1}. **Fórmula cerrada:**

$$c_n = \frac{1}{n}\binom{2n-2}{n-1} = \frac{(2n-2)!}{n!(n-1)!}$$

5.10.2. Sistemas de relaciones de recurrencia

En muchos problemas de conteo, la cantidad total de soluciones no depende solamente de un valor anterior en la secuencia, sino que involucra el resultado de varias secuencias interrelacionadas. En estos casos, hablamos de **sistemas de relaciones de recurrencia**. A diferencia de las relaciones simples, estos sistemas describen conjuntos de secuencias que se definen mutuamente.

Ejercicio: Piezas de dominó y cuadrículas

Enunciado: ¿De cuántas maneras, F_n, se puede llenar una cuadrícula de $3 \times n$ con piezas de dominó de 2×1?

Solución:
Para determinar F_n analizamos cómo puede completarse la *última columna* de la cuadrícula. Hay tres configuraciones típicas (ver Figura 5.3):

Notar que la cantidad de maneras de llenar la cuadrícula en los últimos dos casos es la misma, ya que son simétricas. Llamemos G_{n-1} a ese número. Luego, por la regla de la suma tenemos que

$$F_n \;=\; F_{n-2} + 2G_{n-1}$$

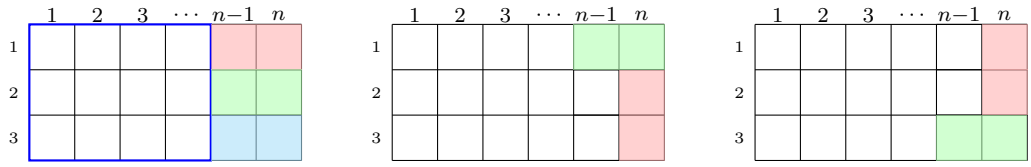


Figura 5.2: Tres formas posibles de completar la última columna de una cuadrícula $3 \times n$ usando piezas de dominó.

Nuevamente, hacemos análisis de casos sobre cómo puede rellenarse la columna más pequeña de G_n. Vemos que hay dos maneras (ver Figura 5.3). En el primer caso, la forma de cubrir la subestructura es equivalente a F_{n-1} y en el segundo caso, a G_{n-2}. Por lo tanto: y por la regla de la suma resulta que

$$G_n \;=\; F_{n-1} + G_{n-2}$$

Agregando las condiciones iniciales tenemos que

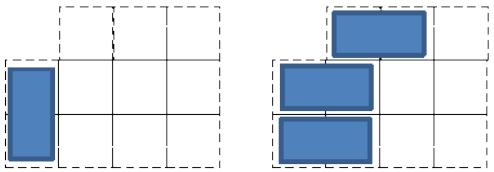


Figura 5.3: Dos formas posibles de completar la última columna de una cuadrícula G_n usando piezas de dominó.

$$\begin{aligned} F_n &= F_{n-2} + 2G_{n-1} \quad \forall n \geq 2 \\ F_0 &= 1 \qquad F_1 = 0 \\ G_n &= F_{n-1} + G_{n-2} \quad \forall n \geq 2 \\ G_0 &= 0 \qquad G_1 = 1 \end{aligned}$$

lo que nos da un sistema de recurrencias para definir F_n.

5.10.3. Ejercicios propuestos

- **Ejercicio:** Un string de números decimales es válido si contiene un número par de ceros. Use una relación de recurrencia para definir c_n, el número de strings válidos de largo n.

- **Ejercicio:** Defina recursivamente la sucesión a_n que da la cantidad de strings de largo n que no tienen dos ceros consecutivos.

- **Ejercicio:** Supongamos que dibujamos n rectas en una hoja de papel de manera que todo par de líneas se intersectan (pero no hay tres líneas que se intersectan en un único punto). ¿En cuántas regiones queda dividido el plano?

5.11. Recurrencias Lineales

5.11.1. Definición General

Recurrencia lineal homogénea con coeficientes constantes

Una relación de recurrencia es **lineal, homogénea y con coeficientes constantes** si tiene la forma:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k},$$

donde $c_1, c_2, \ldots, c_k \in \mathbb{R}$ y $c_k \neq 0$. Decimos que la recurrencia es de **grado** k.

Ejemplos

- $a_n = 3a_{n-1} + 2a_{n-3}$ 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.

5.11.2. Estrategia de Resolución

La idea central para resolver estas recurrencias es asumir que las soluciones tienen la forma $a_n = r^n$. Sustituyendo esta forma en la ecuación de recurrencia se obtiene la llamada **ecuación característica**:

$$\begin{aligned} & r^n = c_1 r^{n-1} + c_2 r^{n-2} + \cdots + c_k r^{n-k} \\ \Leftrightarrow & \\ & r^k = c_1 r^{k-1} + c_2 r^{k-2} + \cdots + c_k \end{aligned}$$

Resolver esta ecuación polinomial nos da las raíces r_i, y la solución general de la recurrencia se construye a partir de estas raíces, dependiendo de su multiplicidad.

Caso de Grado 2 con Raíces Distintas

Solución con raíces distintas

Sea la recurrencia:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2}$$

y supongamos que la ecuación característica $r^2 = c_1 r + c_2$ tiene dos raíces reales distintas r_1 y r_2. Entonces, la solución general es:

$$a_n = \alpha_1 r_1^n + \alpha_2 r_2^n,$$

donde α_1 y α_2 se determinan usando las condiciones iniciales.

Demostración. Por hipótesis, sabemos que $r_1^2 = c_1 r_1 + c_2$ y $r_2^2 = c_1 r_2 + c_2$. Entonces:

$$\begin{aligned} c_1 a_{n-1} + c_2 a_{n-2} &= c_1(\alpha_1 r_1^{n-1} + \alpha_2 r_2^{n-1}) + c_2(\alpha_1 r_1^{n-2} + \alpha_2 r_2^{n-2}) \\ &= \alpha_1 r_1^{n-2}(c_1 r_1 + c_2) + \alpha_2 r_2^{n-2}(c_1 r_2 + c_2) \\ &= \alpha_1 r_1^n + \alpha_2 r_2^n \\ &= a_n \end{aligned}$$

Concluimos que $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n$ es solución para cualquier par de constantes α_1, α_2.

Si además se conocen las condiciones iniciales a_0 y a_1, entonces las constantes α_1, α_2 se determinan unívocamente resolviendo el sistema lineal:

$$\begin{cases} a_0 = \alpha_1 r_1^0 + \alpha_2 r_2^0 \\ a_1 = \alpha_1 r_1^1 + \alpha_2 r_2^1 \end{cases}$$

□

Ejemplo

Considere la relación de recurrencia,

$$\begin{cases} a_n = 2a_{n-1} + 3a_{n-2}, & \text{para } n \geq 2 \\ a_0 = a_1 = 1 \end{cases}$$

La ecuación característica es $r^2 = 2r + 3 \Rightarrow r^2 - 2r - 3 = 0$, con raíces $r_1 = 3$, $r_2 = -1$.
Solución general:

$$a_n = \alpha_1 3^n + \alpha_2 (-1)^n$$

Usamos condiciones iniciales para determinar:

$$\begin{cases} 1 = \alpha_1 + \alpha_2 \\ 1 = 3\alpha_1 - \alpha_2 \end{cases} \Rightarrow \alpha_1 = \frac{1}{2}, \quad \alpha_2 = \frac{1}{2}$$

Solución final:

$$a_n = \tfrac{1}{2} 3^n + \tfrac{1}{2} (-1)^n$$

5.11.3. Raíces Dobles

Solución con raíz doble

Si la ecuación característica tiene una raíz doble r_*, es decir, $r^2 = c_1 r + c_2$ con $r_1 = r_2 = r_*$, entonces la solución general es:

$$a_n = \alpha_1 r_*^n + \alpha_2 n r_*^n.$$

Demostración. Por hipótesis, sabemos que $r_*^2 = c_1 r_* + c_2$. Por lo tanto:

$$\begin{aligned} c_1 a_{n-1} + c_2 a_{n-2} &= c_1(\alpha_1 r_*^{n-1} + \alpha_2(n-1)r_*^{n-1}) + c_2(\alpha_1 r_*^{n-2} + \alpha_2(n-2)r_*^{n-2}) \\ &= \alpha_1 r_*^{n-2}(c_1 r_* + c_2) + \alpha_2 r_*^{n-2}((n-1)c_1 r_* + (n-2)c_2) \\ &= \alpha_1 r_*^n + \alpha_2 r_*^{n-2}(nc_1 r_* - c_1 r_* + nc_2 - 2c_2) \\ &= \alpha_1 r_*^n + \alpha_2 r_*^{n-2}(n(c_1 r_* + c_2) - c_1 r_* - 2c_2) \\ &= \alpha_1 r_*^n + \alpha_2 n r_*^n - \alpha_2 r_*^{n-2}(c_1 r_* + 2c_2) \\ &= a_n - \alpha_2 r_*^{n-2}(c_1 r_* + 2c_2) \end{aligned}$$

Para que la expresión sea igual a a_n, basta establecer que $c_1 r_* + 2c_2 = 0$.
Nota: Como r_* es raíz doble, esto significa que la ecuación característica

$$r^2 - c_1 r - c_2 = 0$$

tiene un discriminante nulo:

$$\Delta = c_1^2 + 4c_2 = 0.$$

Aplicando la fórmula general de las raíces cuadráticas:

$$r = \frac{c_1 \pm \sqrt{c_1^2 + 4c_2}}{2},$$

obtenemos que:

$$r_* = \frac{c_1}{2}$$

cuando el discriminante es cero. Reemplazando en $c_1 r_* + 2c_2$ se obtiene:

$$c_1 r_* + 2c_2 = \frac{c_1^2}{2} + 2c_2 = 2\Delta = 0,$$

por lo tanto, se verifica la igualdad y concluimos que

$$a_n = \alpha_1 r_*^n + \alpha_2 n r_*^n$$

es solución para cualquier par de constantes α_1, α_2. □

5.11.4. Caso General: Grado Arbitrario

Solución general de grado k

Sea

$$a_n = c_1 a_{n-1} + \cdots + c_k a_{n-k}$$

una recurrencia con ecuación característica que tiene t raíces distintas $r_1, \ldots, r_t$ con multiplicidades $m_1, \ldots, m_t$. Entonces, la solución general es:

$$a_n = \sum_{i=1}^{t} \left(\sum_{j=0}^{m_i - 1} \alpha_{i,j} n^j \right) r_i^n$$

5.11.5. Recurrencias No Homogéneas

Recurrencia lineal no homogénea

Una relación de recurrencia lineal, a coeficientes constantes de grado k, se dice **no homogénea** si es de la forma:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n),$$

donde $c_1, c_2, \ldots, c_k$ son números reales, $c_k \neq 0$, y $f(n)$ es una función arbitraria. Definimos la **recurrencia homogénea asociada**:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

Ejemplos

- $a_n = 2a_{n-1} + 1$ (no homogénea de grado 1)
- $a_n = 2a_{n-1} - 3a_{n-3} + 5n^2 + 2^n$ (no homogénea de grado 3)

Teorema de la solución general

La solución general de una recurrencia no homogénea 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 no homogénea.

Una **solución particular** $a_n^{(p)}$ es una función concreta que satisface la relación de recurrencia completa, es decir, incluyendo el término no homogéneo $f(n)$. A diferencia de la solución general, que contiene constantes libres, una solución particular es un caso específico. Se utiliza como complemento a la solución general de la parte homogénea para construir la solución

total.
Por ejemplo, si tenemos:

$$a_n = 2a_{n-1} + 1,$$

la recurrencia homogénea asociada es $a_n^{(g)} = \alpha 2^n$. Una solución particular válida es $a_n^{(p)} = -1$, ya que satisface:

$$-1 = 2(-1) + 1.$$

Entonces, la solución general completa es:

$$a_n = \alpha 2^n - 1.$$

Demostración. Como $a_n^{(p)}$ es una solución particular de la relación de recurrencia no homogénea, se cumple que:

$$a_n^{(p)} = c_1 a_{n-1}^{(p)} + c_2 a_{n-2}^{(p)} + \cdots + c_k a_{n-k}^{(p)} + f(n)$$

Supongamos ahora que b_n es otra solución de la misma relación no homogénea, entonces también se cumple:

$$b_n = c_1 b_{n-1} + c_2 b_{n-2} + \cdots + c_k b_{n-k} + f(n)$$

Restando ambas expresiones:

$$b_n - a_n^{(p)} = c_1(b_{n-1} - a_{n-1}^{(p)}) + c_2(b_{n-2} - a_{n-2}^{(p)}) + \cdots + c_k(b_{n-k} - a_{n-k}^{(p)})$$

Esto implica que la diferencia $b_n - a_n^{(p)}$ satisface la relación de recurrencia homogénea asociada. Es decir, $b_n - a_n^{(p)} = a_n^{(g)}$ para alguna solución homogénea $a_n^{(g)}$. Por lo tanto:

$$b_n = a_n^{(p)} + a_n^{(g)},$$

como se quería demostrar. □

Ejemplo: Torres de Hanoi

La mínima cantidad de movimientos h_n necesarios para resolver el problema de las Torres de Hanoi con n discos satisface:

$$\begin{cases} h_1 = 1 \\ h_n = 2h_{n-1} + 1 \quad (n \geq 2) \end{cases}$$

Paso 1: Se trata de una recurrencia no homogénea de primer orden con $f(n) = 1$.
La recurrencia homogénea asociada es:

$$h_n = 2h_{n-1},$$

cuyo comportamiento general es:

$$h_n^{(g)} = \alpha 2^n.$$

Paso 2: Buscamos una solución particular $h_n^{(p)}$ tal que:

$$h_n^{(p)} = 2h_{n-1}^{(p)} + 1.$$

Probamos con una constante $h_n^{(p)} = C$:

$$C = 2C + 1 \Rightarrow C = -1.$$

Entonces, una solución particular es $h_n^{(p)} = -1$.
Paso 3: La solución general es:

$$h_n = h_n^{(g)} + h_n^{(p)} = \alpha 2^n - 1.$$

Usamos la condición inicial $h_1 = 1$:

$$1 = \alpha 2^1 - 1 \Rightarrow 1 = 2\alpha - 1 \Rightarrow \alpha = 1.$$

Solución cerrada:

$$\boxed{h_n = 2^n - 1}$$

5.11.6. Soluciones Particulares: Método de coeficientes indeterminados

Obtener la solución general $a_n^{(g)}$ de la recurrencia homogénea asociada es fácil (vimos cómo hacerlo). Sin embargo no hay ningún mecanismo general para obtener una solución particular $a_n^{(p)}$ de una recurrencia no homogénea. Para una clase particular de f's tenemos el siguiente teorema:

Teorema

Sea la recurrencia $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k} + f(n)$, donde

$$f(n) = (b_t n^t + b_{t-1} n^{t-1} + \cdots + b_1 n + b_0) s^n,$$

y $b_0, b_1, \ldots, b_t$ y s son números reales. Entonces:

- Si s **no es raíz** de la ecuación característica asociada, entonces existe una solución particular de la forma:

$$(p_t n^t + p_{t-1} n^{t-1} + \cdots + p_1 n + p_0) s^n.$$

- Si s **es raíz** de la ecuación característica con multiplicidad m, entonces existe una solución particular de la forma:

$$n^m (p_t n^t + p_{t-1} n^{t-1} + \cdots + p_1 n + p_0) s^n.$$

Ejercicio resuelto

Ejercicio: recurrencia con término lineal

Encuentre la solución general de la recurrencia:

$$a_n = 3a_{n-1} - 4n$$

Paso 1: Estudiamos la recurrencia homogénea asociada:

$$a_n^{(g)} = 3a_{n-1}^{(g)} \Rightarrow a_n^{(g)} = \alpha 3^n$$

Paso 2: Como el término no homogéneo es $f(n) = -4n$, un polinomio de grado 1, proponemos una solución particular de la forma:

$$a_n^{(p)} = cn + d$$

Sustituimos en la ecuación original:

$$cn + d = 3(c(n-1) + d) - 4n$$

Expandimos el lado derecho:

$$cn + d = 3cn - 3c + 3d - 4n$$

Agrupamos términos:

$$cn + d = (3c - 4)n + (3d - 3c)$$

Igualando coeficientes de ambos lados:

$$\begin{cases} c = 3c - 4 \\ d = 3d - 3c \end{cases} \Rightarrow \begin{cases} 2c = 4 \Rightarrow c = 2 \\ 3d - 3c = d \Rightarrow 2d = 3c \Rightarrow d = \frac{3c}{2} = 3 \end{cases}$$

Paso 3: Por lo tanto, una solución particular es:

$$a_n^{(p)} = 2n + 3$$

Solución general:

$$a_n = a_n^{(g)} + a_n^{(p)} = \alpha 3^n + 2n + 3$$

Nota: Para determinar α se requiere una condición inicial adicional, como a_0 o a_1.

5.11.7. Ejercicios Propuestos

- Resolver $a_n = a_{n-1} + 2a_{n-2}$, con $a_0 = 2$, $a_1 = 7$.
- Encontrar una fórmula explícita para los números de Fibonacci.
- Encontrar una solución particular para: $a_n = 6a_{n-1} - 9a_{n-2} + n3^n$ y $a_n = 6a_{n-1} - 9a_{n-2} + (n^2 + 1)2^n$.
- Defina una recurrencia s_n para la suma de los primeros n naturales, y resuélvala.
- Resolver el sistema:

$$\begin{cases} a_n = 3a_{n-1} + 2b_{n-1} \\ b_n = a_{n-1} + 2b_{n-1} \\ a_0 = 1,\ b_0 = 2 \end{cases}$$

5.12. Funciones Generadoras

5.12.1. Introducción

En muchas ocasiones, necesitamos describir secuencias de números, por ejemplo, el número de formas de resolver un problema de conteo o los términos definidos por una relación de recurrencia. Existen varias formas de describir una secuencia:

- Por extensión: listar los primeros términos, como $0, 1, 3, 7, 15, 31, \ldots$
- Por una relación de recurrencia: por ejemplo, $a_n = 2a_{n-1} + 1$
- Por una fórmula cerrada: como $a_n = 2^n - 1$

Sin embargo, existe una herramienta aún más poderosa: asociar a la secuencia una *función generadora*, que encapsula toda la información de la secuencia en una única expresión formal. Las funciones generadoras no sólo permiten representar una secuencia, sino también manipularla algebraicamente para obtener nuevas propiedades y resolver recurrencias.

5.12.2. Definición de Función Generadora

Función Generadora

Sea $\{a_n\}_{n\geq 0}$ una secuencia de números reales o complejos. Su **función generadora ordinaria** (o simplemente función generadora) es la serie formal:

$$A(x) = \sum_{n\geq 0} a_n x^n$$

Aquí, x se interpreta como un marcador formal, no como una variable real: lo que nos interesa no es el valor de la suma sino los coeficientes a_n.

5.12.3. Ejemplos Iniciales

Ejemplo secuencia constante:

La secuencia constante $a_n = 3$, para todo $n \geq 0$, tiene como función generadora:

$$A(x) = \sum_{n\geq 0} 3x^n = 3(1 + x + x^2 + x^3 + \cdots)$$

Pero puede escribirse en forma más sucinta: $A(x)$ es una **serie geométrica**, cuya suma conocemos si $|x| < 1$:

$$A(x) = \frac{3}{1-x}$$

Ejemplo secuencia lineal:

Para $a_n = n + 1$, tenemos:

$$A(x) = \sum_{n\geq 0} (n+1)x^n = \frac{1}{(1-x)^2}$$

Ejemplo secuencia exponencial:

Para $a_n = c^n$, con c constante:

$$A(x) = \sum_{n\geq 0} (cx)^n = \frac{1}{1-cx}$$

Ejemplo secuencia binomio:

Para $a_n = \binom{m}{n}$ con m fijo, entonces:

$$A(x) = \sum_{n\geq 0} \binom{m}{n} x^n = (1+x)^m$$

5.12.4. Resolución de Recurrencias

Una de las principales aplicaciones de las funciones generadoras es resolver relaciones de recurrencia. Veremos cómo transformar una recurrencia en una expresión cerrada usando funciones generadoras.

Ejemplo

Sea la secuencia definida recursivamente por $a_0 = 0$ y $a_{n+1} = 2a_n + 1$ para todo $n \geq 0$.

Queremos encontrar una forma cerrada para a_n. Para ello, multiplicamos ambos lados de la recurrencia por x^n:

$$a_{n+1}x^n = 2a_n x^n + x^n$$

y luego sumamos para todo $n \geq 0$:

$$\sum_{n\geq 0} a_{n+1}x^n = \sum_{n\geq 0}(2a_n + 1)x^n$$

En el lado izquierdo tenemos *casi* $A(x)$, excepto por el índice $n+1$ en vez de n. Podemos re-armar $A(x)$ "multiplicando y dividiendo por x":

$$\sum_{n\geq 0} a_{n+1}x^n = \frac{1}{x}\sum_{n\geq 0} a_{n+1}x^{n+1} = \frac{A(x) - a_0}{x} = \frac{A(x)}{x}$$

En el lado derecho tenemos

$$\sum_{n\geq 0}(2a_n x^n + x^n) \;=\; 2\sum_{n\geq 0} a_n x^n + \sum_{n\geq 0} x^n \;=\; 2A(x) + \sum_{n\geq 0} x^n \;=\; 2A(x) + \frac{1}{1-x}$$

donde usamos la forma cerrada para la serie geométrica (válida para $|x| < 1$, lo cual supondremos siempre en estas manipulaciones).
Igualando ambas expresiones:

$$\frac{A(x)}{x} = 2A(x) + \frac{1}{1-x}$$

Despejando $A(x)$:

$$A(x) = \frac{x}{(1-x)(1-2x)}$$

¿Cómo seguimos?

Si podemos transformar $A(x)$ de vuelta en una serie tipo $\sum_{n\geq 0} a_n x^n$, basta con identificar el coeficiente de x^n para obtener una fórmula explícita para a_n.

Para ello, realizamos una expansión en fracciones parciales. Notamos que:

$$\frac{x}{(1-x)(1-2x)} = x\left(\frac{2}{1-2x} - \frac{1}{1-x}\right)$$

Y sabemos las series que representan las dos fracciones de la derecha:

$$x \cdot \left(\frac{2}{1-2x}\right) - x \cdot \left(\frac{1}{1-x}\right) = 2x \cdot \sum_{n\geq 0}(2x)^n - x \cdot \sum_{n\geq 0} x^n$$

Obtenemos:

$$2x \cdot \sum_{n\geq 0}(2^n x^n) = \sum_{n\geq 0} 2^{n+1} x^{n+1} = \sum_{n\geq 1} 2^n x^n$$

$$x \cdot \sum_{n\geq 0} x^n = \sum_{n\geq 0} x^{n+1} = \sum_{n\geq 1} x^n$$

Restando las dos series:

$$\sum_{n\geq 1}(2^n - 1)x^n = \sum_{n\geq 0}(2^n - 1)x^n - (2^0 - 1) = \sum_{n\geq 0}(2^n - 1)x^n$$

Por lo tanto, **la solución de la recurrencia es**:

$$a_n = 2^n - 1$$

5.12.5. Resolución de recurrencias: Receta General

La técnica de funciones generadoras es especialmente útil para resolver relaciones de recurrencia y obtener formas cerradas de sus soluciones. A continuación presentamos una receta paso a paso que resume el proceso:

1. **Transformar la relación de recurrencia en una función generadora:**
   a) Asegúrese de que la recurrencia esté bien definida, incluyendo condiciones iniciales claras y los valores de n para los cuales se aplica.
   b) Dé un nombre a la función generadora: por ejemplo, $A(x) = \sum_{n\geq 0} a_n x^n$.
   c) Multiplique ambos lados de la relación por x^n.
   d) Sume ambos lados para todos los valores de n donde la relación es válida.
   e) Exprese los resultados en términos de la función generadora $A(x)$.
   f) Despeje $A(x)$.

2. **Transformar $A(x)$ en una serie de potencias:**

   a) Si $A(x)$ es una función racional (es decir, cociente entre dos polinomios), intente expandirla mediante *fracciones parciales*.

   b) Utilice identidades conocidas de series geométricas para reescribir cada término como una serie de potencias.

3. **Leer la solución:**

   a) Identifique el coeficiente de x^n en la expansión final: eso corresponde al valor de a_n en forma cerrada.

Este procedimiento ha sido ilustrado en el ejemplo del caso $a_{n+1} = 2a_n + 1$, donde finalmente llegamos a la forma cerrada $a_n = 2^n - 1$ a través de funciones generadoras.

5.12.6. Propiedades y ejemplos clásicos de funciones generadoras

Las funciones generadoras no solo sirven para resolver recurrencias, sino que también poseen propiedades algebraicas que facilitan su manipulación y permiten construir nuevas funciones a partir de otras ya conocidas. En esta sección reunimos varios resultados útiles y ejemplos clásicos.

Operaciones básicas entre funciones generadoras

Teorema (Serie de Potencias)

Si $A(x) = \sum_{n\geq 0} a_n x^n$ y $B(x) = \sum_{n\geq 0} b_n x^n$ son funciones generadoras y ambas series convergen, entonces:

- Suma:
$$A(x) + B(x) = \sum_{n\geq 0} (a_n + b_n) x^n$$
- Producto (convolución):
$$A(x) \cdot B(x) = \sum_{n\geq 0} \left(\sum_{k=0}^{n} a_k b_{n-k} \right) x^n$$
  Esta operación se conoce como *convolución*.

Ejemplo: convolución de series geométricas

Sea $A(x) = \frac{1}{(1-x)^2}$ una función generadora. Queremos obtener su expansión como serie de potencias.
Primero, recordemos que:

$$\frac{1}{1-x} = \sum_{n\geq 0} x^n$$

Multiplicando dos veces esta serie:

$$\left(\frac{1}{1-x}\right)^2 = \left(\sum_{n\geq 0} x^n\right)^2 = \sum_{n\geq 0}(n+1)x^n$$

Esto corresponde a la función generadora de la secuencia $1, 2, 3, 4, \ldots$.

5.12.7. Catálogo clásico de funciones generadoras

Secuencia	Función Generadora	Forma cerrada
$1, 0, 0, 0, \ldots$	$\sum_{n\geq 0}[n=0]x^n$	1
$1, 1, 1, 1, \ldots$	$\sum_{n\geq 0} x^n$	$\frac{1}{1-x}$
$1, -1, 1, -1, \ldots$	$\sum_{n\geq 0}(-1)^n x^n$	$\frac{1}{1+x}$
$1, c, c^2, c^3, \ldots$	$\sum_{n\geq 0} c^n x^n$	$\frac{1}{1-cx}$
$1, 2, 3, 4, \ldots$	$\sum_{n\geq 0}(n+1)x^n$	$\frac{1}{(1-x)^2}$
$\binom{c}{0}, \binom{c}{1}, \ldots$	$\sum_{n\geq 0}\binom{c}{n}x^n$	$(1+x)^c$
$\binom{c-1}{0}, \binom{c}{1}, \ldots$	$\sum_{n\geq 0}\binom{c+n-1}{n}x^n$	$\frac{1}{(1-x)^c}$

Estas funciones generadoras permiten reconocer patrones y resolver rápidamente problemas de conteo o recurrencias.

5.12.8. Ejemplo: sucesión de Fibonacci

Paso 1. A partir de la relación de recurrencia $F_{n+1} = F_n + F_{n-1}$, con $F_0 = 0$ y $F_1 = 1$, queremos obtener la función generadora $F(x) = \sum_{n\geq 0} F_n x^n$.

Multiplicamos ambos lados por x^n y sumamos desde $n = 1$:

$$\sum_{n\geq 1} F_{n+1}x^n = \sum_{n\geq 1} F_n x^n + \sum_{n\geq 1} F_{n-1}x^n$$

El lado izquierdo se transforma en:

$$\sum_{n\geq 1} F_{n+1}x^n = \frac{1}{x}(F(x) - x)$$

El lado derecho:

$$\sum_{n\geq 1} F_n x^n + \sum_{n\geq 1} F_{n-1} x^n = F(x) + xF(x)$$

Igualando ambos lados:

$$\frac{1}{x}(F(x) - x) = F(x) + xF(x)$$

Despejando:

$$F(x) = \frac{x}{1 - x - x^2}$$

Paso 2. Recuperamos la serie de potencias de:

$$F(x) = \frac{x}{1 - x - x^2}$$

Esto puede hacerse mediante fracciones parciales. Para ello factorizamos el denominador:

$$1 - x - x^2 = (1 - r_0 x)(1 - r_1 x)$$

donde $r_0 = \frac{1+\sqrt{5}}{2}$ y $r_1 = \frac{1-\sqrt{5}}{2}$. Así:

$$F(x) = \frac{x}{(1 - r_0 x)(1 - r_1 x)} = \frac{1}{r_0 - r_1}\left(\frac{1}{1 - r_0 x} - \frac{1}{1 - r_1 x}\right)$$

Como sabemos que:

$$\frac{1}{1 - ax} = \sum_{n\geq 0} a^n x^n$$

Podemos escribir:

$$F(x) = \sum_{n\geq 0} \frac{1}{\sqrt{5}}(r_0^n - r_1^n)x^n$$

Por lo tanto,

$$F_n = \frac{1}{\sqrt{5}}(r_0^n - r_1^n)$$

Esta es la conocida fórmula cerrada de los números de Fibonacci, también llamada fórmula de Binet.

Referencias

Libros de referencia:

- "Concrete Mathematics" de R. Graham, D. Knuth, y O. Patashnik.

5.13. Más sobre funciones generadoras

5.13.1. Coeficientes Binomiales

Las funciones generadoras no solo sirven para encontrar soluciones cerradas a recurrencias, sino que también son *herramientas poderosas para resolver problemas de conteo sofisticados*. Para esto, necesitamos extender la noción de coeficientes binomiales a casos más generales.

A continuación definimos los coeficientes binomiales extendidos, los cuales permiten trabajar con exponentes reales y negativos en funciones generadoras.

Coeficiente Binomial Extendido

Sea r cualquier número real y k un entero. Se define:

$$\binom{r}{k} = \begin{cases} \dfrac{r(r-1)\cdots(r-k+1)}{k!} & \text{si } k \geq 0, \\ 0 & \text{si } k < 0. \end{cases}$$

Ejemplos:

Veamos algunos ejemplos ilustrativos del uso de esta definicion en casos no enteros o negativos.

- $\binom{-2}{3} = \dfrac{-2 \cdot -3 \cdot -4}{3!} = -4$
- $\binom{1/2}{3} = \dfrac{1/2 \cdot (-1/2) \cdot (-3/2)}{6} = 1/16$

Finalmente, resumimos algunas propiedades algebraicas importantes de estos coeficientes.

Propiedades de los Coeficientes Binomiales

- $\binom{-r}{k} = (-1)^k \binom{r+k-1}{k}$, si $k, r \geq 0$.
- $\binom{r}{k} = \dfrac{r}{k}\binom{r-1}{k-1}$, si $k \neq 0$.
- Identidad de simetría: $\binom{r}{k} = \binom{r}{r-k}$, si r, k son enteros con $r \geq 0$.

5.13.2. Teorema del Binomio Extendido

El siguiente resultado extiende el teorema binomial a exponentes reales.

Teorema del Binomio Extendido

Sea $r \in \mathbb{R}$ y x, y reales tales que $|x/y| < 1$, entonces:

$$(x+y)^r = \sum_{k=0}^{\infty} \binom{r}{k} x^k y^{r-k}.$$

Notar que, si r es un entero no negativo, entonces $\binom{r}{k} = 0$ para todo $k > r$, por lo que la serie se reduce a una suma finita. Esto se debe a que el numerador del coeficiente binomial incluye un término nulo al multiplicar más allá de r. Por ejemplo, el término $(r-k+1)$ será cero o negativo si $k > r$, lo que hace que el producto total se anule y, por tanto, $\binom{r}{k} = 0$.

En cambio, si r no es un entero no negativo, la serie es infinita y requiere condiciones de convergencia, como que $|x/y| < 1$. Intuitivamente, cuando x es suficientemente pequeño en relación a y, los términos de la serie se atenúan y la suma infinita converge.

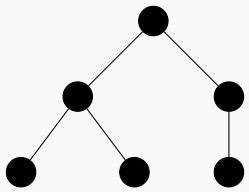
Un caso especial ocurre cuando $y = 1$, obteniendo:

$$(1+x)^r = \sum_{k=0}^{\infty} \binom{r}{k} x^k.$$

5.13.3. Aplicación: Números de Catalán

Los números de Catalán surgen en una gran variedad de contextos combinatorios. Uno de los ejemplos clásicos es el cómputo del número de *árboles binarios diferentes con n vértices*. A continuación desarrollamos la función generadora de los números de Catalán paso a paso, partiendo desde su definición recursiva.

Ejemplo: Árboles binarios



Para contar los árboles binarios con n vértices, comenzamos con un nodo raíz y distribuimos los restantes $n-1$ nodos entre el subárbol izquierdo (con k vértices) y el derecho (con $n-1-k$ vértices). Esto produce la recurrencia:

$$C_n = \sum_{k=0}^{n-1} C_k C_{n-k-1}, \quad C_0 = 1.$$

Ejemplo: Secuencias de paréntesis balanceados

Otra interpretación clásica de los números de Catalán es contar cuántas secuencias de paréntesis bien balanceadas existen con n pares de paréntesis. Por ejemplo, para $n = 3$ hay exactamente 5 secuencias válidas: (()()); ()()(); (())(); (()()); ((())). Esto también se modela con la misma recurrencia de Catalán.

Fórmula cerrada de los números de Catalán

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Demostración. Definimos la función generadora:

$$C(x) = \sum_{n\geq 0} C_n x^n.$$

Reescribiendo la recurrencia como $C_{n+1} = \sum_{k=0}^{n} C_k C_{n-k}$, y multiplicando ambos lados por x^n y sumando:

$$\sum_{n\geq 0} C_{n+1}x^n = \sum_{n\geq 0}\left(\sum_{k=0}^{n} C_k C_{n-k}\right)x^n.$$

Al lado derecho, obtenemos la convolución resultante de multiplicar $C(x)$ por sí misma:

$$\sum_{n=0}^{\infty}\left(\sum_{k=0}^{n} C_k C_{n-k}\right)x^n \;=\; \left(\sum_{n\geq 0} C_n x^n\right)^2 \;=\; C^2(x)$$

El lado izquierdo es $\frac{1}{x}(C(x)-1)$, por lo tanto:

$$\frac{1}{x}(C(x)-1) = C(x)^2 \quad \Rightarrow \quad xC(x)^2 - C(x) + 1 = 0.$$

Resolvemos la ecuación cuadrática:

$$C(x) = \frac{1 \pm \sqrt{1-4x}}{2x}.$$

¿cuál raíz es la correcta? Seleccionamos la raíz negativa ya que debe cumplirse $C(0) = \sum_{n\geq 0} C_n 0^n = C_0 = 1$, y la otra opción diverge. Obtenemos:

$$C(x) = \frac{1 - \sqrt{1-4x}}{2x}.$$

Aplicando el teorema del binomio extendido a $\sqrt{1-4x}$, derivamos la serie de potencias:

$$\sqrt{1-4x} = \sum_{n=0}^{\infty}(-4)^n\binom{1/2}{n}x^n.$$

Cambiando el indice inicial de la serie de potencia, dado que $\binom{1/2}{0} = 1$, obtenemos:

$$1 - \sqrt{1-4x} = 1 - \left(\sum_{n=1}^{\infty} (-4)^n \binom{1/2}{n} x^n + 1 \right) = \sum_{n=1}^{\infty} (-4)^n \binom{1/2}{n} x^n$$

Con eso obtenemos

$$\begin{aligned} C(x) &= \frac{1}{2x}(1 - \sqrt{1-4x}) = \frac{1}{2x} \sum_{n=1}^{\infty} (-4)^n \binom{1/2}{n} x^n \\ &= \frac{1}{2} \sum_{n=1}^{\infty} (-4)^n \binom{1/2}{n} x^{n-1} = \frac{1}{2} \sum_{n=0}^{\infty} (-4)^{n+1} \binom{1/2}{n+1} x^n \end{aligned}$$

Por lo tanto,

$$C_n = \frac{1}{2}(-4)^{n+1} \binom{1/2}{n+1}$$

Simplificación del coeficiente.

$$\begin{aligned} \binom{0{,}5}{n}(-4)^n &= \frac{0{,}5(0{,}5-1)\cdots(0{,}5-n+1)}{n!} \cdot (-4)^n = \frac{2(2-4)\cdots(2-4n+4)}{n!}(-1)^n \\ &= -\frac{2(2)(6)\cdots(4n-6)}{n!} = -\frac{2^n(1)(3)\cdots(2n-3)}{n!} \\ &= -\frac{(n! \cdot 2^n)\cdot(1)(3)\cdots(2n-3)}{n!n!} = -\frac{(2n)!}{n!n!(2n-1)} \\ &= -\frac{1}{2n-1}\binom{2n}{n} \end{aligned}$$

Por lo tanto,

$$C_n = \frac{1}{2}(-4)^{n+1} \binom{1/2}{n+1} = \frac{1}{2(n+2)} \binom{2(n+1)}{n+1} = \frac{1}{n+1} \binom{2n}{n}.$$

□

5.13.4. Problemas de Conteo y Funciones Generadoras

Ejemplo: suma con restricciones

¿Cuántas soluciones existen de $e_1 + e_2 = 8$ sujeto a: $\begin{cases} 2 \leq e_1 \leq 5, \\ 3 \leq e_2 \leq 6 \end{cases}$?

Construimos los polinomios generadores asociados a los posibles valores de e_1 y e_2:

$$f_1(x) = x^2 + x^3 + x^4 + x^5, \quad f_2(x) = x^3 + x^4 + x^5 + x^6.$$

El producto $f_1(x) \cdot f_2(x)$ representa todas las combinaciones posibles de e_1 y e_2, y el coeficiente de x^8 en su expansión indica cuántas sumas satisfacen $e_1 + e_2 = 8$ bajo las restricciones dadas.

$$f_1(x)f_2(x) = x^5 + 2x^6 + 3x^7 + 4x^8 + 3x^9 + 2x^{10} + x^{11}.$$

Por lo tanto, hay **4 maneras** de formar x^8, combinando exponentes del primer polinomio con los del segundo de modo que sus sumas den 8. Las combinaciones que generan este término son:

$$2+6, \quad 3+5, \quad 4+4, \quad 5+3.$$

Ejemplo: suma generalizada

¿Cuántas soluciones existen de $e_1 + e_2 + e_3 = 14$ sujeto a: $\begin{cases} 2 \leq e_1 \leq 5, \\ 3 \leq e_2 \leq 6 \\ 4 \leq e_3 \leq 7 \end{cases}$?

Usamos:

$$f(x) = (x^2 + x^3 + x^4 + x^5)(x^3 + x^4 + x^5 + x^6)(x^4 + x^5 + x^6 + x^7).$$

El número de soluciones es el coeficiente de x^{14} en $f(x)$. Esto se debe a que al multiplicar los tres polinomios, el término $x^i x^j x^k$ aparece en el producto cuando $i + j + k = 14$, es decir, cuando las elecciones de e_1, e_2 y e_3 suman 14.

Ejemplo: Distribución de objetos con cotas

¿De cuántas maneras se pueden distribuir 8 galletas idénticas entre 3 perritos distintos, si cada uno recibe entre 2 y 4 galletas?
Las posibilidades para cada perrito son $2 \leq p_i \leq 4$, y pueden capturarse en 3 exponentes: $(x^2 + x^3 + x^4)$
Todas las combinaciones de 3 perritos se capturan con la **función generadora**:

$$G(x) = (x^2 + x^3 + x^4)^3.$$

Expandiendo:

$$G(x) = x^{12} + 3x^{11} + 6x^{10} + 7x^9 + \boxed{6x^8} + 3x^7 + x^6.$$

La respuesta está en el coeficiente de x^8, esto es, 6.

Formas de pagar con monedas

Supongamos que tenemos monedas de 1 euro, 2 euros, y 5 euros (representadas por ①, ②, y ⑤ respectivamente), y queremos contar todas las maneras de pagar un total de r euros en una máquina.
Considere dos casos: cuando el orden **sí importa** y cuando el orden **no importa**.

Ejemplo: si $r = 5$, entonces:
Orden SÍ importa:

1. $5 \times$ ①: 1 forma: ①①①①①
2. $3 \times$ ① + ②: 4 formas:

   ①①①②, ①①②①, ①②①①, ②①①①
3. $1 \times$ ① + $2 \times$ ②: 3 formas:

   ①②②, ②①②, ②②①
4. ⑤: 1 forma.

Total: 9 maneras

Orden NO importa:

1. $5 \times$ ①
2. $3 \times$ ① + ②
3. ① + $2 \times$ ②
4. ⑤

Total: 4 maneras

Solución: Formas de pagar con monedas (caso orden no importa)

Este problema se modela usando una función generadora infinita, en la cual consideramos que podemos usar cualquier cantidad (incluyendo cero) de cada tipo de moneda:

$$F(x) = \underbrace{(1 + x + x^2 + x^3 + \dots)}_{\text{monedas de 1 euro}} \cdot \underbrace{(1 + x^2 + x^4 + x^6 + \dots)}_{\text{monedas de 2 euros}} \cdot \underbrace{(1 + x^5 + x^{10} + x^{15} + \dots)}_{\text{monedas de 5 euros}}.$$

Este producto genera un término x^r por cada forma de sumar exactamente r euros con las monedas disponibles, sin importar el orden. Esto, porque cada combinación de monedas genera un producto $x^{i+2j+5k}$, donde i, j, y k son la cantidad de monedas de 1, 2 y 5 euros, respectivamente. El coeficiente de x^r en $F(x)$ cuenta cuántas combinaciones de este tipo suman $r = 5$.

Podemos calcular el coeficiente de x^r descomponiendo la función generadora en fracciones parciales, multiplicando manualmente los polinomios, o utilizando herramientas computacionales como `Matlab` o `Wolfram Alpha`.

$$F(x) = \frac{1}{(1-x)(1-x^2)(1-x^5)}.$$

Su expansión comienza como:

$$F(x) = 1 + x + 2x^2 + 2x^3 + 3x^4 + \boxed{4x^5} + 5x^6 + 6x^7 + \dots$$

Por lo tanto, hay **4 maneras** de pagar 5 euros con monedas de 1, 2 y 5 euros si el orden no importa.

Solución: Formas de pagar con monedas (caso orden SÍ importa)

Supongamos que tenemos monedas de 1, 2 y 5 euros, y queremos pagar un total de r euros usando exactamente n monedas. Cuando el orden **sí importa**, esto se modela con la función generadora:

$$G(x) = (x + x^2 + x^5)^n.$$

El coeficiente de x^r en este polinomio cuenta las maneras distintas de elegir n monedas (considerando el orden) cuya suma sea r.

Ejemplo:

Si $n = 2$, entonces:

$$G(x) = (x + x^2 + x^5)^2 = x^2 + 2x^3 + x^4 + 2x^6 + 2x^7 + x^{10}.$$

El coeficiente de x^3 es 2, pues existen dos formas ordenadas de pagar 3 euros con 2 monedas:

$$①②, \quad ②①.$$

Pero queremos *cualquier número de monedas* que juntas den r euros, no sólo n de ellas, por eso debemos considerar más valores de n. Si se permite cualquier cantidad de monedas,

la función generadora total es una suma infinita:

$$\begin{aligned} F(x) &= \sum_{i=0}^{\infty}(x+x^2+x^5)^i \\ &= \frac{1}{1-(x+x^2+x^5)} = \frac{1}{1-x-x^2-x^5} \end{aligned}$$

La expansión comienza como:

$$F(x) = 1 + x + 2x^2 + 3x^3 + 5x^4 + \boxed{9x^5} + 15x^6 + 26x^7 + 44x^8 + 75x^9 + \ldots$$

Por lo tanto, hay **9 maneras** de pagar 5 euros con monedas de 1, 2 y 5 euros si el orden sí importa y se pueden usar tantas monedas como se desee.

Ejercicios propuestos

- ¿De cuántas maneras se pueden entregar 12 peluches idénticos a 5 niños, si cada uno recibe a lo más 3 peluches?
- ¿Cuántas maneras hay de pagar 8 euros con monedas de 1, 2, 3, y 4 euros si el orden **no importa**?
- ¿Y si el orden **sí importa** para pagar 7 euros con las mismas monedas?

Parte V

Teoría de grafos

5.14. Introducción a los Grafos

5.14.1. Motivación y Aplicaciones

Los grafos son estructuras matemáticas utilizadas para modelar relaciones entre objetos. Son fundamentales en ciencias de la computación, matemática discreta, teoría de redes, y muchas otras disciplinas.

Definición: Grafo

Un *grafo* $G = (V, E)$ está compuesto por un conjunto no vacío V de vértices (o nodos) y un conjunto E de aristas (o arcos) que conectan pares de vértices.

Notación: Usaremos (v_1, v_2) para denotar una arista (o arco) entre los nodos v_1 y v_2, donde $v_1, v_2 \in V$. Esta notación también será válida para representar la dirección en el caso de grafos dirigidos.

5.14.2. Historia

Historia: Los Puentes de Königsberg

El matemático Leonhard Euler resolvió en 1736 el famoso problema de los puentes de Königsberg, una ciudad atravesada por un río y dividida en varias islas conectadas por siete puentes. El desafío consistía en encontrar un recorrido que cruzara cada puente exactamente una vez.

Para abordar el problema, Euler realizó una abstracción revolucionaria: representó cada porción de tierra como un nodo y cada puente como una arista que conecta dos nodos. Esta representación marcó el nacimiento formal de la teoría de grafos.

Su demostración mostró que no es posible recorrer todos los puentes sin repetir alguno, y estableció principios fundamentales sobre conectividad y estructura. La enseñanza clave es que muchos problemas complejos se simplifican cuando se modelan como grafos.

Abstracción del problema de los Puentes de Königsberg: desde el mapa original hasta su representación como grafo. Imagen: Bogdan Giuşcă.

5.14.3. Ejemplos de aplicaciones

Los grafos no solo son una herramienta matemática elegante, sino también un lenguaje poderoso para representar sistemas complejos en el mundo real. A continuación, se presentan algunas aplicaciones concretas y ampliamente estudiadas del uso de grafos en distintas áreas del conocimiento y la tecnología [2]

- Modelar redes de comunicación: cada dispositivo es un nodo, y una conexión directa entre dos dispositivos (como un cable o enlace inalámbrico) se modela como una arista. Esto permite representar arquitecturas de red como topologías en estrella, anillo o malla, facilitando el análisis de eficiencia y tolerancia a fallos.
- Representar relaciones sociales: personas como nodos y las amistades, interacciones o vínculos (como seguir a alguien en redes sociales) como aristas. Esta abstracción permite estudiar propiedades como centralidad, comunidades, y difusión de información.
- Representar colaboraciones científicas: se construyen grafos donde cada nodo es un autor, y dos autores están conectados si han publicado juntos. Un ejemplo contemporáneo es el uso de bases de datos académicas para modelar la evolución de la colaboración interdisciplinaria.
- Modelar el transporte: estaciones, paraderos o aeropuertos son nodos, y las rutas (como líneas de metro, vuelos o carreteras) son aristas. Esto permite optimizar rutas, estudiar congestión o conectividad.

5.14.4. Clasificación de Grafos

Tipos de grafos

- **Grafo simple**: sin lazos ni aristas paralelas.
- **Multigrafo**: permite aristas paralelas.
- **Grafo con lazos**: permite aristas de un nodo a sí mismo.
- **Grafos dirigidos**: las aristas tienen dirección.

Nombre	Aristas	Aristas paralelas	Lazos
Grafo simple	No dirigidas	No	No
Multigrafo	No dirigidas	Sí	No
Grafo	No dirigidas	Sí	Sí
Grafo dirigido simple	Dirigidas	No	No
Multigrafo dirigido	Dirigidas	Sí	No
Grafo dirigido	Dirigidas	Sí	Sí

[2]Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

5.14.5. Terminología

Grafos no dirigidos

- Dos nodos son **adyacentes** si comparten una arista.
- Una arista es **incidente** a un nodo si el nodo es uno de sus extremos.
- El **grado** de un nodo v, denotado $\deg(v)$, es el número de aristas incidentes (los lazos cuentan doble).
- Un nodo es **aislado** si $\deg(v) = 0$.

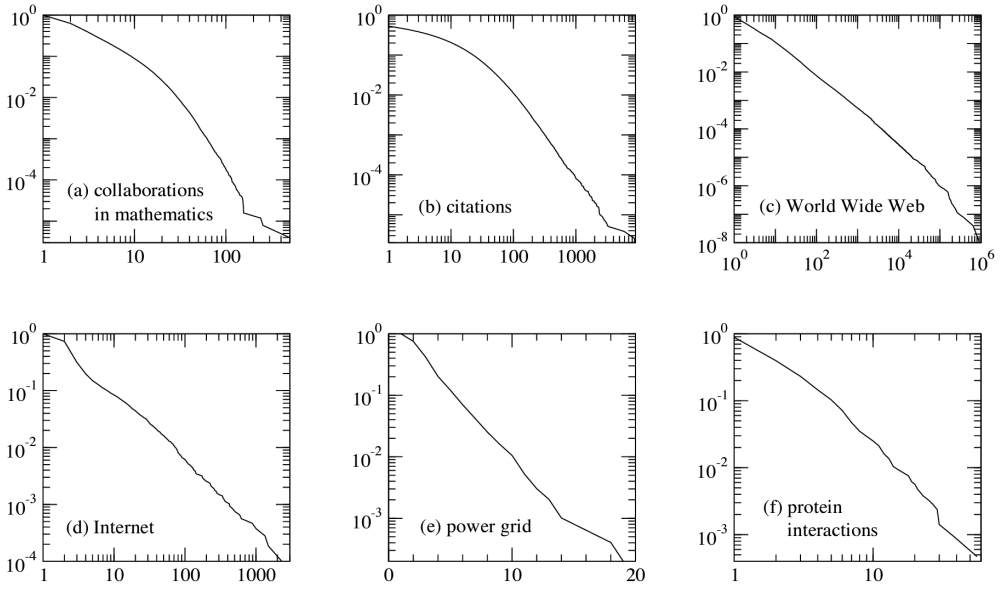
Grafos dirigidos

- En una arista (u, v), u es el **origen** y v el **destino**.
- **Grado de entrada**: $\deg^-(v)$ es el número de aristas que llegan a v.
- **Grado de salida**: $\deg^+(v)$ es el número de aristas que salen de v.

5.14.6. Ejemplos empíricos: Distribución de grado en redes reales

Una propiedad estructural fundamental de un grafo es su **distribución de grado**, que indica cuántos nodos tienen k conexiones para cada posible valor de k. Esta distribución revela la organización global de la red y es especialmente importante en grafos grandes como los que aparecen en contextos reales.

En la siguiente figura se observan distribuciones de grado para diferentes redes reales, muchas de las cuales exhiben comportamientos que siguen una ley de potencias:



Distribuciones de grado para diversas redes: colaboración matemática, citas, web, internet, red eléctrica y proteínas. Fuente: Newman, M.E.J. (2003). *The Structure and Function of Complex Networks. SIAM Review.*

Características comunes

- En redes como la web, el internet o las redes sociales, la mayoría de los nodos tiene pocos enlaces, mientras que unos pocos tienen muchos (nodos *hub*).
- Este comportamiento es típico de una **distribución de grado sesgada** o de tipo *escala libre*, en contraste con una red aleatoria.
- En estos casos, la probabilidad de que un nodo tenga grado k decrece como $P(k) \sim k^{-\gamma}$ para alguna constante γ.

Estas propiedades tienen consecuencias importantes: los hubs pueden facilitar la difusión rápida de información o infecciones, pero también pueden ser puntos vulnerables si fallan.

5.15. Propiedades de los Grafos

Una de las propiedad de los grafos no dirigidos es la relación entre el número total de aristas y los grados de los vértices. Esta relación se conoce comúnmente como el *Teorema de los saludos*, ya que puede interpretarse en un contexto cotidiano: El número total de apretones de manos en una fiesta es igual a la suma de las manos estrechadas por cada persona, y por

lo tanto, siempre es par. En consecuencia, el número de personas que da un número impar de apretones es par.

Teorema de los saludos

Sea $G = (V, E)$ un grafo no dirigido. Entonces:

$$\sum_{v \in V} \deg(v) = 2|E|$$

En particular, **el número de nodos de grado impar es par**.

Demostración. El resultado es inmediato ya que cada arista $e \in E$ aporta dos unidades a la suma $\sum_{v \in V} deg(v)$. (Puede probarse más formalmente haciendo inducción sobre el número de aristas del grafo.)

Para probar que el número de nodos de grado impar es par, consideremos la suma total $S = \sum_{v \in V} \deg(v)$ como una suma de enteros. Dado que S es par (igual a $2|E|$), y que la suma de números impares sólo puede ser par si hay una cantidad par de sumandos impares, se deduce que debe haber un número par de vértices con grado impar. □

Lema de grados para grafos dirigidos

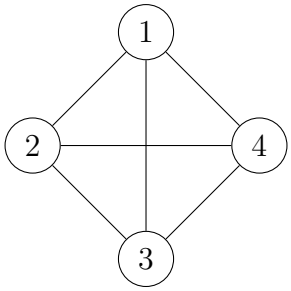
Sea $G = (V, E)$ un grafo dirigido. Entonces:

$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$$

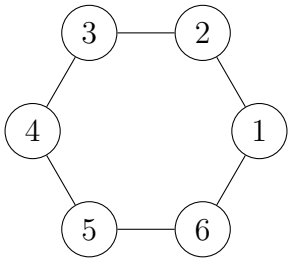
Demostración. Cada arista dirigida $(u, v) \in E$ tiene un nodo origen u y un nodo destino v. Por definición, cada arista incrementa en uno el grado de salida de u y el grado de entrada de v. Por lo tanto, al sumar los grados de salida y de entrada sobre todos los vértices, cada arista contribuye una única vez a cada suma: $\sum_{v \in V} \deg^-(v) = |E|$, $\sum_{v \in V} \deg^+(v) = |E|$. □

5.15.1. Ejemplos Clásicos de Grafos

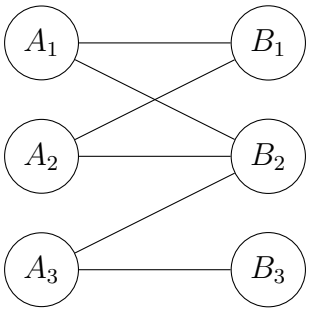
- **Cliques** (K_n): Todos los nodos están conectados entre sí.



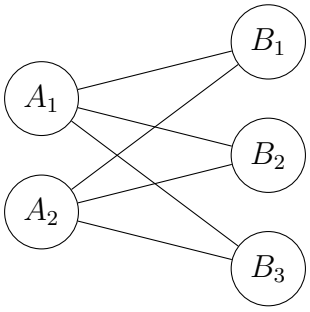
- **Ciclos** (C_n): Un ciclo de $n \geq 3$ vértices $v_1, v_2, \ldots, v_n$ es aquel grafo simple que contiene las aristas $\{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n), (v_n, v_1)\}$. Usamos C_n para denotar al ciclo de n vértices.



- **Grafos bipartitos**: Un grafo $G = (V, E)$ es bipartito sii V puede ser particionado en dos conjuntos V_1 y V_2 tal que cada arco del grafo conecta un nodo de V_1 con uno de V_2, y no hay arcos dentro de un mismo conjunto.conjuntos.



- **Grafos bipartitos completos** ($K_{m,n}$): Un grafo bipartito se dice además **completo** si existe un arco entre *cada* par de nodos (u, v) tal que $u \in V_1$ y $v \in V_2$. Usamos $K_{|V_1|,|V_2|}$ para denotar a dicho grafo.



Caracterización de grafos bipartitos

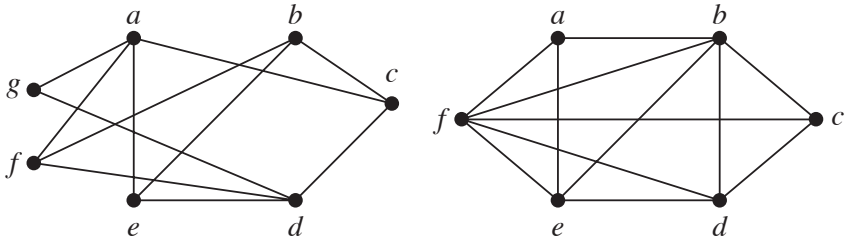
Un grafo simple es bipartito si y solo si sus vértices pueden colorearse con dos colores de forma que ningún par de nodos adyacentes tenga el mismo color.

Demostración. ($\Rightarrow$) Suponga que $G = (V, E)$ es bipartito y que la partición de V está dada por V_1 y V_2. Pinte los vértices en V_1 de azul y los vértices en V_2 de rojo. Como en un grafo bipartito todas las aristas conectan vértices de conjuntos distintos, ningún par de vértices adyacentes estará pintado del mismo color.

($\Leftarrow$) Suponga ahora que es posible pintar de rojo o azul cada vértice de G de tal forma que si dos nodos son adyacentes, entonces reciben colores distintos. Defina V_1 como el conjunto de nodos coloreados de azul y V_2 como el conjunto de nodos coloreados de rojo. Entonces $V = V_1 \cup V_2$ y toda arista conecta un nodo de V_1 con uno de V_2, lo que demuestra que G es bipartito. □

5.15.2. Ejercicios Propuestos

1. Use el teorema anterior para determinar si los siguientes grafos son bipartitos:



2. Demuestre que el ciclo C_n es bipartito si y solo si n es par.

3. Sea $G = (V, E)$ un grafo simple. Demuestre que

$$\text{mín}\,\{deg(v) \mid v \in V\} \;\leq\; \frac{2\,|E|}{|V|} \;\leq\; \text{máx}\,\{deg(v) \mid v \in V\}$$

4. Sea $G = (V, E)$ un grafo bipartito simple. Demuestre que $|E| \leq |V|^2/4$.

5.16. Representación de grafos

Una vez que comprendemos la noción abstracta de grafo, surge una pregunta natural: ¿cómo podemos representar un grafo en un computador de manera que sea útil para analizarlo y procesarlo? Existen varias formas de representar un grafo, y la elección depende del tipo de operaciones que queremos realizar sobre él.

La representación elegida puede afectar considerablemente la eficiencia de los algoritmos. Por ejemplo, encontrar los vecinos de un nodo es muy eficiente en una lista de adyacencia, pero verificar si dos nodos están conectados es inmediato en una matriz de adyacencia.

Consideremos el siguiente grafo simple, no dirigido, con cuatro vértices:

Este grafo tiene:

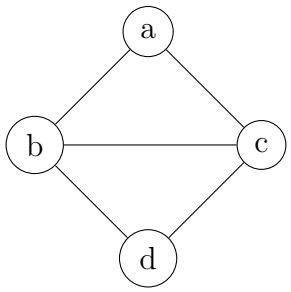


Figura 5.4: Grafo base utilizado en las representaciones

- Vértices: $V = \{a, b, c, d\}$
- Aristas: $E = \{(a, b), (a, c), (b, d), (c, d), (b, c)\}$

A continuación presentamos tres formas comunes para representar un grafo.

5.16.1. Listas de adyacencia

Una de las representaciones más comunes es la de **listas de adyacencia**, donde para cada vértice se mantiene una lista de sus vértices adyacentes.

Lista de adyacencia

Dado un grafo $G = (V, E)$, una lista de adyacencia asocia a cada vértice $v \in V$ un conjunto o lista de vértices adyacentes a v, es decir, aquellos vértices $u \in V$ tales que $(v, u) \in E$.

Ejemplo: Lista de adyacencia

Lista de adyacencia correspondiente al grafo de la Figura 5.4:

Vértice	Adyacentes
a	b, c
b	a, c, d
c	a, b, d
d	b, c

Esta representación es eficiente en espacio cuando el grafo es *disperso* (pocas aristas en relación al número de vértices).

5.16.2. Matriz de adyacencia

Otra forma clásica es la **matriz de adyacencia**, una matriz cuadrada de tamaño $n \times n$, donde $n = |V|$, que indica si existe una arista entre dos vértices.

Matriz de adyacencia

Sea $G = (V, E)$ con $V = \{v_1, v_2, \ldots, v_n\}$. La matriz de adyacencia $A = [a_{ij}]$ se define como:

$$a_{ij} = \begin{cases} 1 & \text{si } (v_i, v_j) \in E \\ 0 & \text{si } (v_i, v_j) \notin E \end{cases}$$

Ejemplo: Matriz de adyacencia

Matriz de adyacencia del grafo de la Figura 5.4, en el orden a, b, c, d:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Una propiedad interesante es que la matriz depende del orden de los vértices. Por tanto, un mismo grafo puede tener $n!$ matrices distintas asociadas.

5.16.3. Matriz de incidencia

La tercera representación que estudiaremos es la **matriz de incidencia**, que vincula vértices con aristas.

Matriz de incidencia

Sea $G = (V, E)$ con $V = \{v_1, \ldots, v_n\}$ y $E = \{e_1, \ldots, e_m\}$. La matriz de incidencia $B = [b_{ij}]$ es una matriz $n \times m$ donde:

$$b_{ij} = \begin{cases} 1 & \text{si el vértice } v_i \text{ es incidente en la arista } e_j \\ 0 & \text{si no} \end{cases}$$

Ejemplo: Matriz de incidencia

Matriz de incidencia correspondiente al grafo de la Figura 5.4, ordenando las aristas como: $e_1 = (a, b), e_2 = (a, c), e_3 = (b, d), e_4 = (c, d), e_5 = (b, c)$:

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

Esta representación se utiliza frecuentemente en teoría de flujo y optimización lineal sobre redes.

Resumen y comparación

Representación	Espacio	Éficiente para grafos densos?	Éficiente para grafos dispersos?
Lista de adyacencia	$O(n+m)$	No	**Sí**
Matriz de adyacencia	$O(n^2)$	**Sí**	No
Matriz de incidencia	$O(n \cdot m)$	Intermedia	Intermedia

5.17. Subgrafos e Isomorfismo

5.17.1. Subgrafos

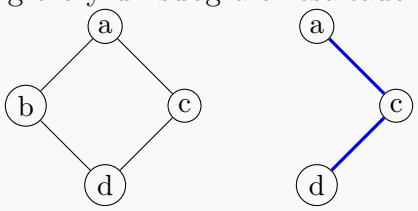
En muchos casos, no es necesario considerar la totalidad de un grafo para resolver un problema. Basta con trabajar con una parte o "fragmento" de este.

Subgrafo

Sea $G = (V, E)$ un grafo. Un **subgrafo** de G es un grafo $G' = (V', E')$ tal que $V' \subseteq V$ y $E' \subseteq E$. Decimos que el subgrafo es **propio** si $G' \neq G$.

Ejemplo:

En la Figura se muestra un grafo y un subgrafo resaltado en azul a su derecha.



5.17.2. Isomorfismo de grafos

A veces dos grafos tienen distinta *apariencia*, pero representan esencialmente la misma estructura. En tales casos decimos que los grafos son **isomorfos**.

Isomorfismo

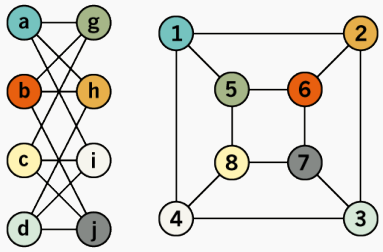
Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos simples. Decimos que G y G' son **isomorfos** si existe una biyección $f\colon V \to V'$ tal que para todo par de vértices $v_1, v_2 \in V$, se cumple:

$$(v_1, v_2) \in E \quad \text{si y sólo si} \quad \big(f(v_1), f(v_2)\big) \in E'$$

A la función f se le llama un **isomorfismo** entre G y G'.

Ejemplo:

A continuación se muestran dos grafos que, aunque visualmente diferentes, son isomorfos.



Pregunta: *Cuántas posibles correspondencias (biyecciones) existen entre dos grafos simples con n nodos?*

5.17.3. Demostrando isomorfismo y no isomorfismo

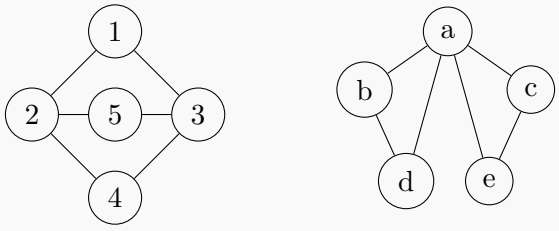
Demostrar que dos grafos son isomorfos implica encontrar una biyección entre sus vértices que preserve las adyacencias. Aunque conceptualmente directo, este problema es **computacionalmente costoso**. Los mejores algoritmos actuales para grafos generales tienen una complejidad **exponencial** en el número de vértices. Por ello, es un problema central en el *Graph Isomorphism Problem*, uno de los pocos problemas importantes cuya complejidad aún no se conoce con certeza.

La dificultad inherente en el problema de grafos isomorfos lo hacen un entretenido y desafiante puzzle: `https://aabeliuk.github.io/graph-isomorphism-game/`

Demostrando que dos grafos no son isomorfos

Ejemplo

¿Són los siguientes grafos isomorfos?



Respuesta: No, porque uno de los grafos tiene un vértice de grado 4 (el vértice a), mientras que el otro grafo no tiene ningún vértice con ese grado.

Para demostrar que dos grafos **no** son isomorfos, basta encontrar una **propiedad invariante** bajo isomorfismos que difiera entre ambos. Algunas propiedades invariantes comunes son:

- Número de vértices y aristas.
- Grado de los vértices.
- Existencia de ciclos de ciertas longitudes.
- Distribución de componentes conexas.

Estas invariantes permiten descartar el isomorfismo sin necesidad de revisar todas las permutaciones posibles.

Clases de isomorfismo

Isomorfismo es clase de equivalencia

Demostrar que la relación de isomorfía entre grafos define una relación de equivalencia en la colección de todos los grafos.
Hint: Demostrar que la relación es reflexiva, simétrica y transitiva.

Dado que el isomorfismo de grafos es una relación de equivalencia, divide el conjunto de todos los grafos en **clases de equivalencia**. No importa cuál de los grafos isomórficos de una clase de equivalencia se escoja: todos comparten la misma estructura esencial.

Aplicaciones: Esta idea de clasificar grafos en clases de isomorfismo tiene múltiples aplicaciones en distintas áreas. Permite representar moléculas como grafos y detectar compuestos equivalentes. En **bases de datos de grafos**, se utiliza para identificar duplicados estructurales, incluso si los nombres de los nodos difieren. En **reconocimiento de patrones**, ayuda a comparar estructuras como árboles sintácticos o redes neuronales. Finalmente, en **algoritmos**, permite optimizar procedimientos al operar sobre representaciones canónicas de clases de grafos equivalentes.

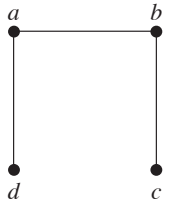
Ejercicios sobre isomorfismo

El **complemento** de un grafo simple $G = (V, E)$ es el grafo $\overline{G} = (V, \overline{E})$, tal que

$$(v_1, v_2) \in \overline{E} \Leftrightarrow (v_1, v_2) \notin E.$$

Un grafo simple es **auto-complementario** si G es isomorfo a $\bar{G}$.

- **Ejercicio:** Demuestre que el siguiente grafo es auto-complementario.



- **Ejercicio:** Encuentre un grafo auto-complementario con 5 vértices.
- **Ejercicio:** Demuestre que si G es un grafo auto-complementario simple con n vértices, entonces $n \equiv 0$ o $n \equiv 1$ módulo 4.
- **Ejercicio:** ¿Cuántos grafos no isomorfos de 4 vértices existen? Dibújelos todos y justifique por qué no hay más.

5.18. Caminos en grafos

5.18.1. Navegando grafos

Muchos problemas pueden modelarse y resolverse **navegando grafos**. La noción de **camino** nos permite formalizar preguntas relacionadas con conectividad, eficiencia de recorrido y planificación de rutas.

- **Ejemplo:** ¿Es posible mandar un mensaje entre dos computadoras usando las conexiones intermedias? Este problema puede resolverse explorando los caminos en una red.
- **Ejemplo:** ¿Cuál es la manera más rápida de salir de una granja, pasar por todos los establecimientos, y regresar al punto de partida? Esta es una instancia clásica del *Problema del Viajante* (TSP).

5.18.2. Definición de camino

Camino

Sea $G = (V, E)$ un grafo no dirigido y sea $u, u' \in V$. Un **camino** (de largo n) entre u y u' es una secuencia de n aristas $e_1, e_2, \ldots, e_n$ tal que existen nodos $v_0, v_1, v_2, \ldots, v_n$ que satisfacen:

- $v_0 = u$ y $v_n = u'$, y
- cada arista e_i conecta v_{i-1} y v_i, para $1 \leq i \leq n$

- Si el camino no tiene aristas repetidas, se dice que es un **camino simple**.
- Si $u = u'$, entonces el camino se denomina **circuito** o **ciclo**.

Notación: Si el grafo no contiene aristas paralelas, podemos denotar un camino como la secuencia de vértices: $u, v_1, v_2, \ldots, v_{n-1}, u'$.

5.18.3. Conectividad de grafos no dirigidos

En algunos grafos siempre es posible ir (i.e. conectar mediante un camino) desde un vértice hasta cualquier otro. En otros no.

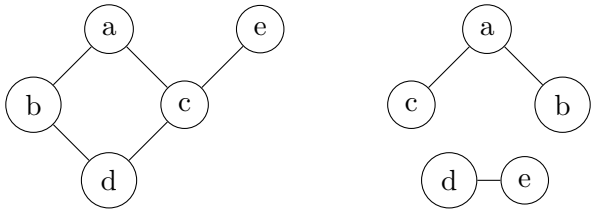


Figura 5.5: Ejemplos de grafo conexo (izquierda) y no conexo (derecha)

Conectividad

Un grafo no dirigido se llama **conexo** sii existe un camino entre cada par de vértices distintos del grafo.

Propiedades e implicancias

Si existe un camino de u a v en un grafo $G = (V, E)$, entonces existe también un **camino simple** entre estos vértices. Esto es importante porque muchos algoritmos y teoremas operan sobre caminos simples, ya que descartan redundancias o ciclos intermedios.

Además, los caminos permiten definir **invariantes estructurales**, como la conectividad entre pares de vértices o la cantidad de componentes conexas. Estas invariantes pueden utilizarse para demostrar que dos grafos **no** son isomorfos, ya que todo isomorfismo debe preservar tales propiedades. Por ejemplo, si un grafo es conexo y otro no, no pueden ser isomorfos.

5.18.4. Componentes conexas

Si un grafo es no conexo, entonces está formado por la unión disjunta de sus **componentes conexas**.

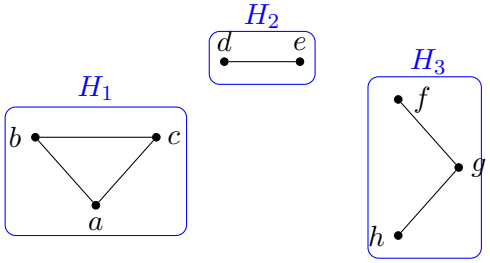


Figura 5.6: Ejemplo de grafo no conexo con tres componentes conexas H_1, H_2 y H_3

Componente conexa

Sea G un grafo no dirigido. Una **componente conexa** de G es un subgrafo G' de G tal que:

- G' es conexo, y
- G' no es un subgrafo propio de otro subgrafo conexo de G.

En otras palabras, una componente conexa de G es un subgrafo conexo **maximal**.

5.18.5. Conectividad de grafos dirigidos

Hay dos variantes de la noción de conectividad sobre grafos dirigidos: una considerando la dirección de los arcos y otra no. Para esta última, necesitamos la noción de grafo no-dirigido **subyacente**.

Grafo no-dirigido subyacente

Dado un grafo dirigido $G = (V, E)$, el **grafo no-dirigido subyacente** G' de G se obtiene a partir de G, computando la clausura simétrica de su conjunto de aristas E.

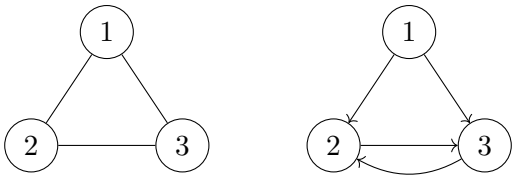
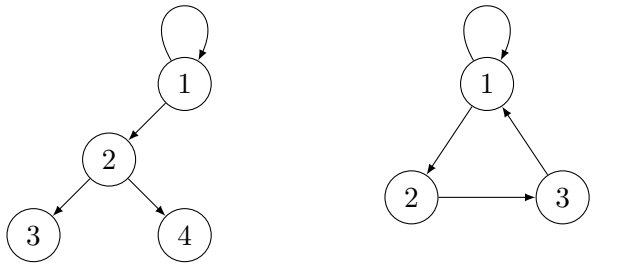


Figura 5.7: Ejemplo de grafo no dirigido (izquierda) y grafo dirigido (derecha) con arcos entre nodos

Conectividad fuerte y débil en grafos dirigidos

Conectividad en grafos dirigidos

- Un grafo dirigido es **fuertemente conexo** si para todo par de vértices v, v', existe un camino dirigido de v a v' y viceversa.
- Un grafo dirigido es **débilmente conexo** si su grafo no dirigido subyacente es conexo, es decir, si para todo par v, v' existe un camino entre ellos ignorando las direcciones.



Débilmente conexo **Fuertemente conexo**

Figura 5.8: Comparación entre grafo débilmente conexo (izquierda) y fuertemente conexo (derecha)

Componentes fuertemente conexas

Componente fuertemente conexa

Sea G un grafo dirigido. Una **componente fuertemente conexa** de G es un subgrafo dirigido G' tal que:

- G' es fuertemente conexo, y
- G' no es un subgrafo propio de otro subgrafo fuertemente conexo de G.

5.18.6. Caminos e isomorfismo

La existencia de ciertos caminos también puede usarse como **invariante** para distinguir grafos no isomorfos.

Ejemplo

Utilice un invariante de caminos para demostrar que los grafos de la Figura 5.9 no son isomorfos.

Respuesta: Aunque los grafos tienen la misma cantidad de nodos y aristas, así como los mismos grados, el grafo H tiene un circuito simple de largo 3 (v_1, v_2, v_6, v_1), mientras que G no posee ninguno. Esto muestra que no pueden ser isomorfos.

5.18.7. Ejercicios

- **Ejercicio:** Demuestre que para todo grafo conexo simple, todo nodo de grado impar está unido mediante un camino a algún otro nodo de grado impar.
- **Ejercicio:** Demuestre que todo grafo conexo con n vértices debe tener al menos $n - 1$ aristas.

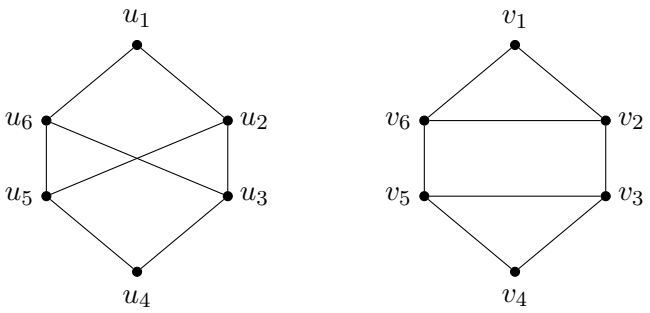


Figura 5.9: Dos grafos G y H

- **Ejercicio:** Demuestre que un grafo simple G es bipartito si y sólo si no contiene circuitos de largo impar.

5.19. Circuitos Eulerianos

5.19.1. El Problema de los Puentes de Königsberg

En el siglo XVIII, en la ciudad prusiana de Königsberg, se planteó una pregunta curiosa sobre la posibilidad de recorrer la ciudad cruzando todos sus puentes exactamente una vez. La ciudad estaba dividida por el río Pregel en cuatro regiones conectadas por siete puentes.

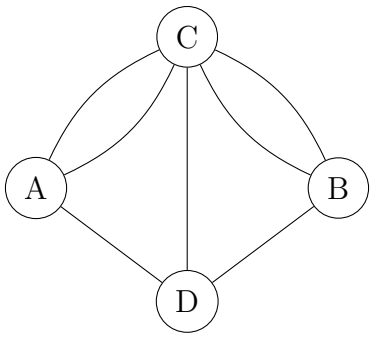
Abstracción del problema de los Puentes de Königsberg: desde el mapa original hasta su representación como grafo. Imagen: Bogdan Giuşcă.

> **Problema:** ¿Es posible comenzar en una región cualquiera, recorrer todos los puentes exactamente una vez, y volver al punto de partida?

Este problema intrigó a los matemáticos de la época, y fue Leonhard Euler quien, en 1736, ofreció una solución formal, dando origen a la teoría de grafos.

5.19.2. Modelado con Grafos

Podemos modelar este problema mediante un **grafo**, donde cada región se representa con un vértice y cada puente con una arista que conecta dos vértices.



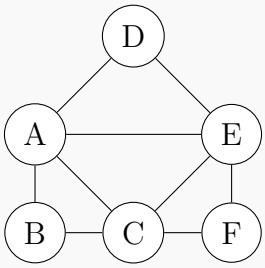
El problema se convierte ahora en determinar si existe un **circuito euleriano** en este grafo.

Definición: Circuito Euleriano

Sea G un grafo. Un *circuito euleriano* es un circuito simple que recorre cada arista de G exactamente una vez.

Ejemplo

El siguiente grafo posee múltiples circuitos eulerianos:



Algunos circuitos eulerianos posibles (empezando en A) son:

$$ADEACEFCBA \quad \text{y} \quad AECABCFEDA$$

Teorema Clásico de Euler

Un multigrafo conexo tiene un circuito euleriano si y sólo si todos sus vértices tienen grado par.

5.19.3. Solución al problema de los puentes de Königsberg

El problema de los puentes de Königsberg no tiene solución: el grafo que lo representa tiene cuatro vértices de grado impar, lo cual impide la existencia de un circuito euleriano. Este resultado marcó el inicio del estudio sistemático de grafos, un área fundamental en la matemática discreta y la informática.

5.19.4. Demostración del Teorema de Euler

($\Rightarrow$) Supongamos que G tiene un circuito euleriano. Cada vez que el circuito pasa por un vértice, debe entrar y luego salir por aristas distintas, lo que contribuye con dos al grado de ese vértice. Como el circuito pasa por todas las aristas del grafo (por definición), y G es conexo, todos los vértices deben tener grado par.

($\Leftarrow$) Supongamos ahora que G es conexo y que todos sus vértices tienen grado par. Construiremos un circuito euleriano de forma constructiva:

- Elegimos un vértice cualquiera v_0 y un arco (v_0, v_1) incidente.
- Continuamos extendiendo un camino $v_0, v_1, v_2, \ldots, v_n$ de forma que no repita aristas y se detenga sólo cuando no sea posible continuar.

Proposición

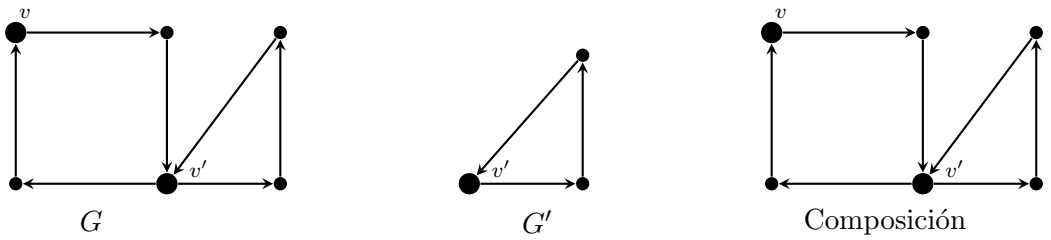
En el procedimiento anterior, necesariamente se cumple $v_0 = v_n$, es decir, se obtiene un circuito.

Demostración. Supongamos lo contrario: que $v_0 \neq v_n$. Entonces el camino termina en un vértice v_n con grado impar (ya que entró una sola vez). Pero esto contradice la hipótesis de que todos los vértices tienen grado par. Por lo tanto, el circuito debe cerrar. □

Por lo tanto tenemos un circuito simple

$$v_0, v_1, v_2, \ldots, v_{n-1}, v_0$$

Si el circuito contiene todos los arcos, la prueba está terminada. Si no, eliminamos sus aristas del grafo G y obtenemos un subgrafo G' con vértices de grado par (aunque G' puede ser disconexo). Como G es conexo, existe al menos un vértice, v', en común entre G' y el circuito original. A partir de dicho vértice se puede construir un nuevo circuito en G' y componerlo con el anterior.



Reiterando este proceso, se obtiene un circuito euleriano que recorre todas las aristas de G. ■

5.19.5. Caminos eulerianos

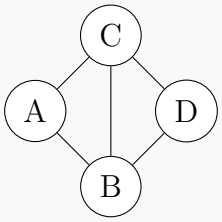
La noción de *camino euleriano* generaliza la de circuito euleriano: en lugar de requerir que el recorrido comience y termine en el mismo vértice, basta con que recorra todas las aristas del grafo sin repetir ninguna.

Definición: Camino euleriano

Sea G un grafo. Un camino de G se dice que es **euleriano** si es simple y pasa por cada arista de G exactamente una vez.

Ejemplo

El siguiente grafo admite múltiples caminos eulerianos:



Uno de ellos es el camino $C \to A \to B \to D \to C \to B$.

Sin embargo, este grafo **no** admite ningún circuito euleriano, ya que no es posible recorrer todas sus aristas y regresar al vértice de origen sin repetir alguna.

Caracterización de caminos eulerianos

Sea G un multigrafo conexo con al menos dos vértices. Entonces G tiene un camino euleriano (pero no un circuito) si y sólo si tiene exactamente **dos vértices de grado impar**.

Demostración.

$\Rightarrow$ Sean u y v los extremos del camino euleriano. Como el camino recorre todas las aristas, debe contener todos los vértices del grafo. Para cualquier vértice intermedio t, cada vez que se entra a t, también se debe salir, lo que implica que su grado es par. Los extremos u y v, en cambio, tienen grado impar, ya que sólo se entra o se sale una única vez más que el resto de las veces. Por lo tanto, G tiene exactamente dos vértices de grado impar.

$\Leftarrow$ Supongamos ahora que G tiene exactamente dos vértices de grado impar, digamos u y v. Construimos un nuevo grafo G' añadiendo una arista artificial entre u y v. Este nuevo grafo G' tiene todos los vértices de grado par y es conexo, por lo que admite un circuito euleriano. Al eliminar la arista artificial, obtenemos un camino euleriano en G que comienza en u y termina en v.

□

5.19.6. Caminos y circuitos eulerianos en grafos dirigidos

En el caso de los grafos dirigidos, las condiciones para la existencia de un camino o circuito euleriano cambian ligeramente respecto del caso no dirigido.

Caracterización en grafos dirigidos

Sea G un multigrafo dirigido sin vértices aislados.

- G contiene un **circuito euleriano** si y sólo si:
  - Es **débilmente conexo**, y
  - El grado de entrada coincide con el grado de salida en cada vértice.
- G contiene un **camino euleriano** (pero no un circuito) si y sólo si:
  - Es **débilmente conexo**, y
  - Todos los vértices tienen el mismo grado de entrada y de salida, excepto dos vértices v_1 y v_2, tales que:

$$deg^{in}(v_1) = deg^{out}(v_1) + 1 \quad \text{y} \quad deg^{out}(v_2) = deg^{in}(v_2) + 1.$$

Notar que la existencia de un **circuito euleriano** en un grafo dirigido no implica que el grafo sea *fuertemente conexo*. En la Figura 5.10, el grafo tiene un circuito que recorre todas las aristas exactamente una vez, pero no existe camino dirigido entre algunos pares de vértices (por ejemplo, de C a A), por lo tanto no es fuertemente conexo.

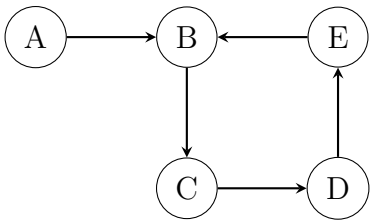


Figura 5.10: Grafo dirigido con circuito euleriano $A \to B \to C \to D \to E \to B$, pero que no es fuertemente conexo (por ejemplo, no hay camino dirigido de C a A).

5.20. Circuitos Hamiltonianos

5.20.1. Introducción

Hasta ahora hemos estudiado los **circuitos eulerianos**, recorridos que atraviesan todas las *aristas* de un grafo exactamente una vez. Esto es útil para problemas donde lo importante es cubrir todas las conexiones, como por ejemplo: Inspeccionar todos los cables de una red eléctrica.

En contraste, en muchas aplicaciones lo que se desea es visitar cada **vértice** exactamente una vez, sin repetir lugares:

- Un viajante que quiere visitar cada ciudad una vez y volver al punto de origen,
- Un robot que debe recorrer todas las ubicaciones de una bodega sin repetir ninguna,
- Problemas de planificación o exploración sin retrocesos.

Este tipo de problema da origen a la noción de **circuito hamiltoniano**, llamado así en honor a William Rowan Hamilton, quien lo introdujo en el siglo XIX al estudiar recorridos sobre los vértices de un dodecaedro (poliedro de doce caras).

¿Cuándo es posible visitar todos los vértices de un grafo sin repetir ninguno y volver al origen?

En esta sección exploraremos esta pregunta y veremos cómo, a diferencia del caso euleriano, determinar la existencia de un circuito hamiltoniano es un problema mucho más difícil, sin caracterización general conocida hasta la fecha.

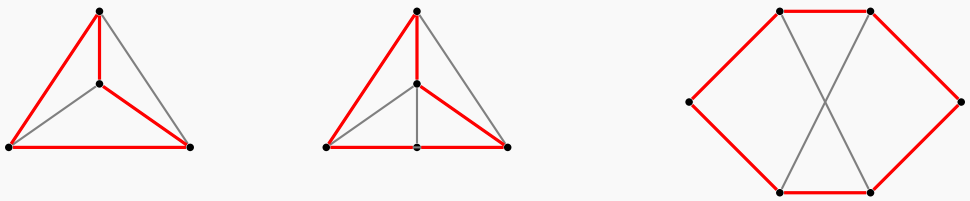
5.20.2. Definición

Circuito hamiltoniano

Sea G un grafo. Un circuito de G se dice que es **hamiltoniano** si es simple y pasa por cada vértice de G exactamente una vez.

Ejemplo

Los siguientes grafos admiten un circuito hamiltoniano:



A diferencia de los circuitos eulerianos, **no se conoce una caracterización eficiente** que permita determinar en general si un grafo tiene o no un circuito hamiltoniano.

5.20.3. Condiciones necesarias y suficientes

No existe una caracterización completa de los grafos que admiten circuitos hamiltonianos. Sin embargo, se conocen algunas condiciones útiles:

- **Condiciones necesarias:** permiten descartar la existencia de un circuito hamiltoniano en algunos casos.
- **Condiciones suficientes:** garantizan su existencia en clases específicas de grafos (ver, por ejemplo, el teorema de Dirac y Ore).

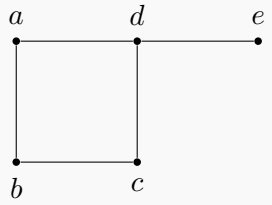
Condiciones necesarias para descartar hamiltonicidad

Las siguientes observaciones permiten probar que un grafo **no** tiene circuitos hamiltonianos:

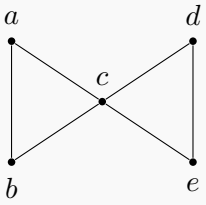
- Si un grafo tiene un vértice de grado 1, no puede tener un circuito hamiltoniano.
- Si un vértice tiene grado 2, entonces **ambas aristas** incidentes deben formar parte de cualquier circuito hamiltoniano.
- Si un vértice tiene grado mayor a 2, y se sabe que **dos de sus aristas** están en un circuito hamiltoniano, entonces el **resto de las aristas incidentes no pueden estar** en dicho circuito.

Ejercicio: demostración por contradicción

Utilice las condiciones anteriores para demostrar que los siguientes grafos **no** tienen circuitos hamiltonianos:



G



H

Solución:

- El grafo G no tiene circuito hamiltoniano porque contiene un vértice de grado 1.
- Para el grafo H, supongamos por contradicción que tiene un circuito hamiltoniano. Como los vértices a, b, d, e tienen grado 2, las dos aristas incidentes en cada uno deben pertenecer al circuito. Por lo tanto, las 4 aristas que inciden en el vértice c también deberían pertenecer al circuito, lo cual es imposible (ya que solo puede tener 2 aristas en un camino simple cerrado). Esto contradice la tercera observación.

Condiciones suficientes para circuitos hamiltonianos

A diferencia de los circuitos eulerianos, no existe una caracterización general y eficiente que determine cuándo un grafo posee un circuito hamiltoniano. Sin embargo, existen ciertos resultados clásicos que establecen **condiciones suficientes** para garantizar su existencia.

A continuación presentamos dos de los más conocidos: el **Teorema de Ore** y el **Teorema de Dirac**.

Informalmente, si fijamos el conjunto de vértices de un grafo, en cuanto más artistas tenga, mayor potencial tendrá de contener caminos hamiltonianos.

El siguiente teoremas establece que para que un grafo admita circuitos hamiltonianos es condición suficiente que los vertices tengan grado "lo suficientemente grande".

Teorema de Ore

Sea $G = (V, E)$ un grafo simple con $n \geq 3$ vértices. Si para todo par de vértices no adyacentes $u, v \in V$ se cumple que

$$\deg(u) + \deg(v) \geq n,$$

entonces G posee un circuito hamiltoniano.

Un corolario del Teorema de Ore, más simple de aplicar, es el siguiente:

Teorema de Dirac

Sea $G = (V, E)$ un grafo simple con $n \geq 3$ vértices. Si

$$\deg(v) \geq \frac{n}{2} \quad \text{para todo } v \in V,$$

entonces G posee un circuito hamiltoniano.

Demostración. El Teorema de Dirac es un caso particular del Teorema de Ore. En efecto, si todos los vértices tienen grado al menos $\frac{n}{2}$, entonces para cualquier par de vértices no adyacentes u, v,

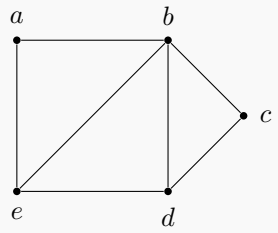
$$\deg(u) + \deg(v) \geq \frac{n}{2} + \frac{n}{2} = n,$$

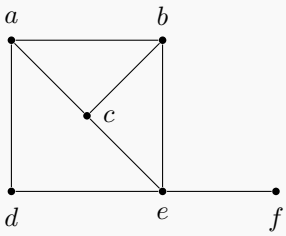
cumpliendo con la hipótesis del Teorema de Ore. □

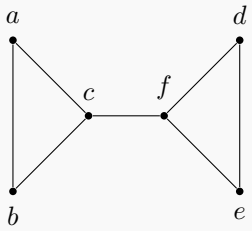
Ejercicios

Ejercicio

Determinar cuáles de los siguientes grafos tienen un circuito hamiltoniano, justificando la respuesta:







Ejercicio: grafo bipartito

Sea $K_{m,n}$ el grafo bipartito completo con $m, n \geq 2$. Probar que $K_{m,n}$ tiene un circuito hamiltoniano si y sólo si $m = n$.
Hint:

- Para el $\Rightarrow$: usar reducción al absurdo, suponiendo $m \neq n$.
- Para el $\Leftarrow$: aplicar el Teorema de Dirac.

Demostración.

$\Rightarrow$ Supongamos que $K_{m,n}$ tiene un circuito hamiltoniano y que $m \neq n$. Sin pérdida de generalidad, supongamos que $m < n$. Sea A y B la partición del conjunto de vértices con $|A| = m$ y $|B| = n$, y supongamos que el recorrido parte desde $a_1 \in A$. Como el grafo es bipartito, el circuito debe alternar entre vértices de A y B. Luego de $2m$ pasos, se han recorrido todos los vértices de A y vuelto a a_1, pero aún quedarían $n - m > 0$ vértices en B sin visitar. Esto contradice que el recorrido sea hamiltoniano.

$\Leftarrow$ Si $m = n$, entonces el grafo tiene $2m$ vértices y cada vértice tiene grado m, cumpliendo:

$$\deg(v) = m \geq \frac{2m}{2} = m.$$

Por el Teorema de Dirac, $K_{m,m}$ posee un circuito hamiltoniano.

□

5.21. Caminos más cortos sobre grafos

5.21.1. Motivación

En muchas aplicaciones prácticas, como el diseño de redes de transporte o de comunicación, necesitamos encontrar el camino más corto entre dos puntos. Este problema puede

modelarse usando grafos con pesos, donde los vértices representan ubicaciones y las aristas representan las conexiones con sus respectivos costos o distancias.

5.22. Grafos con pesos

Grafo con pesos

Un **grafo con pesos** es una terna (V, E, w) donde:

- V es el conjunto de vértices,
- $E \subseteq V \times V$ es el conjunto de aristas,
- $w : E \to \mathbb{R}_{\geq 0}$ es una función que asigna un peso a cada arista.

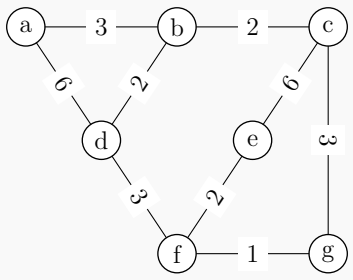
Sea $P = (v_0, v_1, \ldots, v_k)$ un camino en un grafo con pesos. Su **longitud** es la suma de los pesos de sus aristas, es decir:

$$\ell(P) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$$

Un **camino más corto** entre dos vértices u y v es aquel camino P que parte en u, termina en v, y tiene longitud mínima entre todos los posibles caminos entre ellos. Este camino no necesariamente es único.

Ejemplo

La siguiente red representa distancias entre ciudades:



Pregunta: ¿Cuál es la ruta más corta desde a hasta g?

5.22.1. Algoritmo de Dijkstra

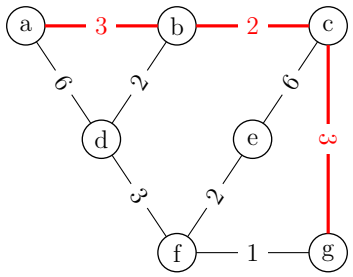
El algoritmo de Dijkstra es uno de los más importantes en teoría de grafos y fue propuesto por Edsger W. Dijkstra en 1956. Su objetivo es encontrar el camino más corto desde un vértice de origen hasta todos los demás vértices de un grafo con pesos no negativos.

El procedimiento sigue una estrategia codiciosa (greedy): comienza con el nodo de origen y explora iterativamente el vértice adyacente más cercano, acumulando las distancias mínimas conocidas. A cada paso, actualiza las distancias si encuentra un camino más corto que los previamente registrados.

Este algoritmo ha sido fundamental en el desarrollo de sistemas de navegación, planificación de rutas y optimización de redes, y es una pieza clave en muchos cursos de algoritmos y ciencias de la computación teórica.

Ejemplo

Aplicaremos el algoritmo de Dijkstra desde el vértice a para encontrar el camino más corto hacia todos los demás vértices.



Resolución del algoritmo de Dijkstra:

Iteración	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**Procesado**
1	0_a	3_a	∞	6_a	∞	∞	∞	a
2	0_a	3_a	5_b	5_b	∞	∞	∞	b
3	0_a	3_a	5_b	5_b	11_c	∞	8_c	c
4	0_a	3_a	5_b	5_b	11_c	8_d	8_c	d
5	0_a	3_a	5_b	5_b	11_c	8_d	8_c	f
6	0_a	3_a	5_b	5_b	10_f	8_d	8_c	e

Camino más corto desde a **hasta** g: $a \to b \to c \to g$, con costo total 8.

- La celda x_y indica la distancia más corta encontrada hasta el nodo x, con y indicando el nodo anterior en el camino más corto.
- Se termina cuando todos los nodos están procesados.
- El camino más corto a un nodo se puede reconstruir siguiendo los subíndices hacia atrás.

Pseudocódigo comentado

El siguiente pseudocódigo describe el algoritmo de Dijkstra para encontrar caminos más cortos desde un nodo origen hacia todos los demás nodos de un grafo con pesos no negativos.

```
function Dijkstra(Graph, source):
  // Inicialización: asigna distancia infinita a todos los nodos y sin predecesor conoci
  for each vertex v in Graph:
    dist[v] := inf              // Distancia estimada desde el origen a v
    prev[v] := undefined        // Nodo anterior en el camino más corto encontrado

  dist[source] := 0             // La distancia del nodo origen a sí mismo es 0
  Q := all nodes in Graph       // Conjunto de nodos no visitados

  while Q is not empty:
    u := node in Q with min dist[u]   // Selecciona el nodo más cercano aún no procesado
    remove u from Q

    for each neighbor v of u:
      alt := dist[u] + length(u, v)   // Calcula la distancia alternativa a través de u
      if alt < dist[v]:
        dist[v] := alt                // Se encontró un camino más corto hacia v
        prev[v] := u                  // u se convierte en el predecesor de v

  return dist[], prev[]               // dist contiene las distancias mínimas; prev los
```

5.22.2. Observaciones importantes

El algoritmo de Dijkstra es versátil y puede aplicarse tanto en grafos no dirigidos como en **grafos dirigidos**, lo cual amplía significativamente su utilidad práctica en modelar distintos tipos de redes.

Restricción de pesos no negativos: Es importante destacar que este algoritmo requiere que todos los pesos de las aristas sean no negativos, ya que su lógica de actualización de distancias mínimas depende de esta propiedad. En presencia de pesos negativos, el algoritmo puede fallar al no detectar caminos más cortos correctamente.

Tratamiento de múltiples aristas: Si el grafo tiene múltiples aristas entre un mismo par de nodos, se recomienda conservar únicamente aquella con menor peso, ya que siempre será preferible en la búsqueda de caminos mínimos.

Más recursos

Para ver el algoritmo en funcionamiento, sobre un grafo ingresado por el usuario, ver **demo interactiva:** `https://algorithms.discrete.ma.tum.de/graph-algorithms/spp-dijkstra/index_en.html`

5.23. Colorabilidad de grafos

En muchos problemas del mundo real, como la asignación de frecuencias a antenas, la programación de horarios escolares o la resolución de sudokus, surge la necesidad de distinguir elementos conectados entre sí mediante çolores". Esta idea intuitiva está en el corazón de la teoría de coloración de grafos.

El objetivo es sencillo de enunciar: queremos asignar colores a los vértices de un grafo de forma que vértices adyacentes (es decir, conectados por una arista) no compartan el mismo color. Pero como veremos, este problema encierra una profunda riqueza matemática y computacional.

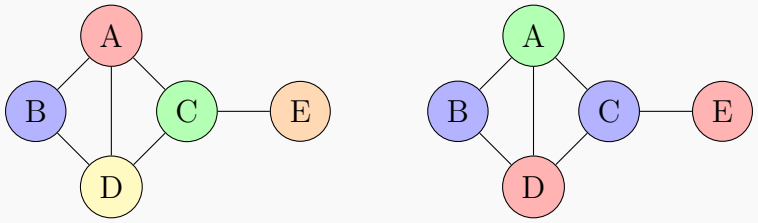
Coloración de un grafo

Una **coloración** de un grafo simple $G = (V, E)$ es una asignación de colores a los vértices de G tal que ningún par de vértices adyacentes recibe el mismo color.

Es evidente que podemos colorear cualquier grafo simple con n vértices utilizando n colores diferentes, asignando un color distinto a cada vértice. No obstante, en la mayoría de los casos, esta cantidad resulta excesiva: muchos grafos permiten una coloración válida con un número significativamente menor de colores.

A continuación se muestran dos ejemplos de coloración:

Ejemplos de coloraciones



En el primer caso se utilizan cinco colores distintos; en el segundo, solo tres. Esto nos lleva a una noción central en teoría de grafos: el **número cromático**.

5.23.1. Número cromático

Número cromático

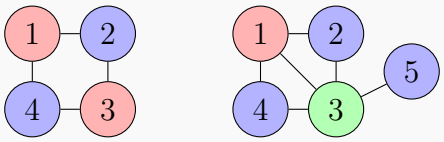
El **número cromático** de un grafo G, denotado $\chi(G)$, es el menor número de colores necesario para colorear G de modo que vértices adyacentes tengan colores distintos.

El número cromático permite clasificar grafos según su complejidad estructural. Por ejemplo:

- Un grafo sin aristas tiene número cromático $\chi = 1$.

- Un grafo bipartito tiene $\chi = 2$.
- Un ciclo impar tiene $\chi = 3$.
- Un grafo completo K_n tiene $\chi = n$.

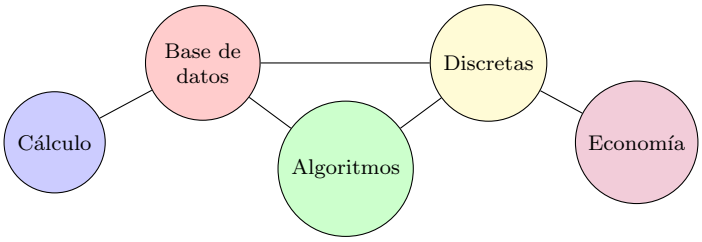
Ejemplos de número cromático



En el primer grafo (un ciclo par), el número cromático es $\chi = 2$. En el segundo grafo, se requieren tres colores: $\chi = 3$.

5.23.2. Aplicación: Programación de exámenes

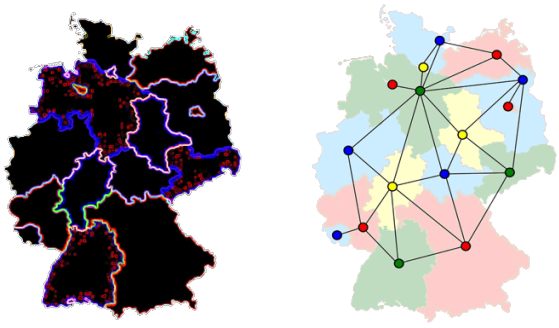
La Escuela asigna un bloque de tres horas para el examen final de cada ramo. ¿Cuál es la menor cantidad de bloques que puede asignar, de tal manera que los alumnos no tengan tope entre los exámenes de los distintos ramos en los que están inscritos?



La respuesta está dada por el número cromático del grafo de incompatibilidad, donde los nodos representan los ramos, y dos nodos son adyacentes si y solo si tienen algún alumno en común.

5.23.3. Aplicación: Pintando mapas

¿Cuál es la mínima cantidad de colores que son necesarios para pintar el siguiente mapa si las regiones limítrofes deben tener distinto color?



La respuesta está dada por el número cromático del grafo de la derecha, donde cada vértice representa una región y dos vértices están conectados si las regiones que representan son limítrofes.s.

5.24. Grafos planares

El estudio de coloraciones mínimas está fuertemente ligado a las propiedades geométricas de los grafos, en particular, a su representación en el plano.

Grafo planar

Un grafo es **planar** si y solo si puede dibujarse en el plano sin que sus aristas se crucen.

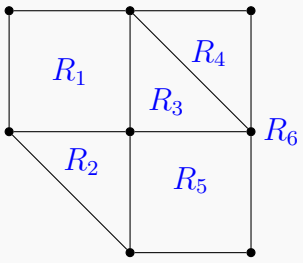
Una representación planar de un grafo divide al plano en **regiones**, que son las zonas delimitadas por las aristas del grafo cuando este se dibuja en el plano sin cruces. Entre estas regiones siempre se encuentra una región no acotada o exterior, que rodea completamente al dibujo.

Cada vez que una arista o conjunto de aristas encierran una parte del plano sin cruzarse entre sí, forman una región distinta. El número de regiones en una representación planar depende de cómo se conecta el grafo y cuántas aristas y vértices contiene, pero no de cómo se dibuje, siempre que sea una representación planar válida.

Estas regiones son clave en el estudio topológico de grafos, ya que permiten aplicar la fórmula de Euler. También son útiles en aplicaciones como mapas o circuitos, donde evitar cruces mejora la claridad y funcionalidad del diseño.

Ejemplo

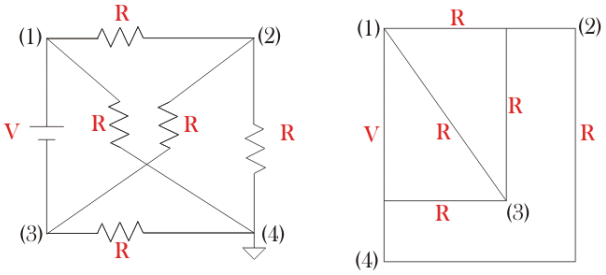
La siguiente representación planar divide el plano en 6 regiones:



5.24.1. Aplicación: Diseño de circuitos eléctricos

Una aplicación fundamental de los grafos planares se encuentra en el diseño de circuitos eléctricos y microprocesadores. En estos contextos, los componentes electrónicos y sus conexiones se modelan mediante grafos, donde los vértices representan nodos de conexión y las aristas los conductores o enlaces físicos.

Para minimizar costos y evitar interferencias, es crucial que el diseño del circuito sea **planar**, es decir, que las conexiones puedan trazarse en una sola capa sin que se crucen. Si un grafo no es planar, se requieren capas adicionales o el uso de tecnología más costosa para separar conexiones cruzadas.



Este problema motiva algoritmos para el reconocimiento de grafos planares y su embebido en el plano.

5.24.2. Fórmula de Euler

Leonhard Euler descubrió una relación entre el número de vértices, aristas y regiones en cualquier grafo planar, conexo y sin múltiples aristas ni lazos. Esta relación es conocida como la **fórmula de Euler**.

Fórmula de Euler

Sea $G = (V, E)$ un grafo simple, conexo y planar. Sea r el número de regiones en una representación planar de G. Entonces:

$$r = |E| - |V| + 2$$

Esta fórmula es válida para cualquier representación planar del grafo, sin importar su disposición geométrica específica.

Demostración. La demostración se realiza por inducción sobre el número de aristas. Construimos una secuencia de subgrafos planares:

$$G_1, G_2, \ldots, G_{|E|}$$

donde cada G_{j+1} se obtiene al agregar una arista de G a G_j, de forma que se mantenga la conexidad. Como G es conexo, siempre podemos agregar una arista que se conecte con los vértices ya presentes.

Demostraremos por inducción que todo $G_j = (V_j, E_j)$ cumple:

$$r_j = |E_j| - |V_j| + 2$$

siendo r_j el número de regiones de su representación planar.

Caso base: G_1 tiene una sola arista y dos vértices. Como no encierra ninguna región, $r_1 = 1$, y:

$$r_1 = 1 = 1 - 2 + 2 = 1\checkmark$$

Paso inductivo: Al agregar una nueva arista, se presentan dos casos:

(1) Si une dos vértices ya presentes, crea una nueva región: $r_{j+1} = r_j + 1$, $E_{j+1} = E_j + 1$ y $V_{j+1} = V_j$.

$$\therefore\ r_{j+1} = |E_{j+1}| - |V_{j+1}| + 2.$$

(2) Si introduce un nuevo vértice, no crea regiones: $r_{j+1} = r_j$, $E_{j+1} = E_j + 1$ y $V_{j+1} = V_j + 1$.

$$\therefore\ r_{j+1} = |E_{j+1}| - |V_{j+1}| + 2.$$

En ambos casos se mantiene:

$$r_{j+1} = |E_{j+1}| - |V_{j+1}| + 2 \quad \checkmark$$

Por lo tanto, por inducción, la fórmula se cumple para G. □

Ejemplo de Fórmula de Euler

En el grafo del ejemplo anterior:

- Número de vértices: $|V| = 8$
- Número de aristas: $|E| = 12$
- Número de regiones: $r = 6$

Verifiquemos la fórmula:

$$r = |E| - |V| + 2 = 12 - 8 + 2 = 6 \quad \checkmark$$

La fórmula se cumple.

5.24.3. Corolarios de la fórmula de Euler

El primer corolario que estudiaremos establece una cota superior para el número de arcos de un grafo planar.

Corolario 1

Si G es un grafo simple, conexo y planar con $|V| \geq 3$, entonces:

$$|E| \leq 3|V| - 6$$

El siguiente corolario establece una condición necesaria de planaridad.

Corolario 2

Todo grafo simple, conexo y planar tiene al menos un vértice de grado a lo sumo 5.

Ejercicios propuestos

1. Demostrar que el grafo completo K_5 no es planar.

2. Sea G un grafo simple y planar con 9 vértices. ¿Cuál es el máximo número posible de aristas que puede tener?

3. Da un ejemplo de un grafo conexo que cumple $|E| \leq 3|V| - 6$ pero no es planar. ¿Es esto posible?

4. *Demuestra que el grafo bipartito completo $K_{3,3}$ no es planar, utilizando una variante del Corolario 1 para grafos con ciclos de longitud ≥ 4.

5.25. Teorema de los Cuatro Colores

Uno de los resultados más sorprendentes y célebres de la teoría de grafos es el llamado **Teorema de los Cuatro Colores**. Este surge al estudiar la coloración de grafos planares.

La respuesta, aunque simple, tardó más de un siglo en ser formalmente demostrada.

Teorema de los Cuatro Colores *(Appel & Haken, 1976)*

Sea G un grafo simple, conexo y planar. Entonces:

$$\chi(G) \leq 4$$

Es decir, todo grafo planar puede ser coloreado usando a lo sumo cuatro colores.

Este resultado implica que cualquier mapa, por complejo que sea, puede colorearse con cuatro colores de modo que regiones adyacentes no compartan el mismo color.

Breve historia del teorema

La conjetura fue planteada en 1852 por Francis Guthrie. A pesar de múltiples intentos fallidos de demostración, no fue hasta 1976 que Kenneth Appel y Wolfgang Haken lograron una prueba definitiva, aunque controversial, utilizando asistencia computacional.

La idea de la demostración. Appel y Haken demostraron que no puede existir un **contraejemplo minimal** al teorema. Para ello, redujeron la verificación a 1.936 configuraciones críticas, las cuales fueron examinadas mediante un algoritmo computacional que corrió durante cientos de horas.

La parte teórica de su demostración ocupa más de 500 páginas, con análisis manual de *contra-contra-ejemplos*. Muchos de estos fueron revisados por el propio hijo de Haken.

El método de demostración generó debate en la comunidad matemática por dos razones. Primero, una parte clave de la prueba fue verificada únicamente por computadora, lo que en su momento generaba desconfianza. Segundo, la porción escrita a mano era demasiado extensa y engorrosa para que alguien la verificara por completo.

Segunda prueba (Robertson et al., 1997) En 1997, Robertson, Sanders, Seymour y Thomas publicaron una nueva prueba del Teorema de los Cuatro Colores en la revista *Journal of Combinatorial Theory*. Esta sigue la misma estrategia general de Appel y Haken, pero con importantes mejoras, por ejemplo, el número de configuraciones se redujo a 633.

Estado actual. Hasta hoy, no se conoce una demostración del teorema que pueda ser completamente verificada sin asistencia computacional. Aun así, el resultado se acepta universalmente como verdadero, y marca un hito en el uso de computadoras para la demostración matemática.

5.25.1. Teoremas de coloración débil para grafos planares

Además del Teorema de los Cuatro Colores, existen versiones más accesibles que se pueden demostrar con técnicas elementales.

6-coloración

Sea G un grafo simple, conexo y planar. Entonces:

$$\chi(G) \leq 6$$

Demostración. Por inducción en el número de vértices n.

Caso base: Para $n \leq 6$, coloreamos cada vértice con un color distinto.

Hipótesis inductiva (HI): Supongamos que todo grafo planar con $n-1$ vértices es 6-colorable.

Sea G un grafo planar con n vértices. Por el Corolario del Teorema de Euler, existe un vértice v de grado a lo sumo 5.

Eliminamos v y sus aristas incidentes. El grafo restante tiene $n-1$ vértices y es 6-colorable por HI.

Entonces podemos colorear v con un color distinto al de sus (a lo sumo) 5 vecinos. □

5-coloración

Sea G un grafo simple, conexo y planar. Entonces:

$$\chi(G) \leq 5$$

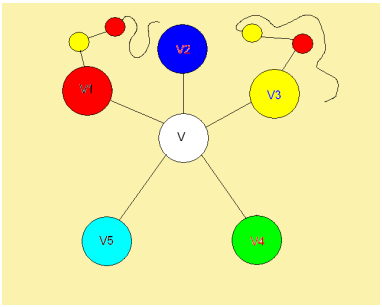
Demostración. La demostración también se basa en inducción, pero requiere análisis más delicado sobre cómo intercambiar colores entre vecinos de un nodo v de grado 5.

Caso base: Todo grafo planar con $n \leq 5$ es trivialmente 5-colorable.

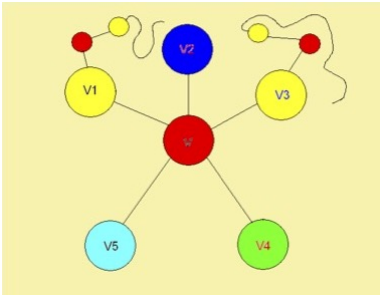
Hipótesis inductiva (HI): Todo grafo simple, conexo y planar con $n-1$ vértices es 5-colorable.

Sea G un grafo con n vértices. Por el Corolario del Teorema de Euler, existe un vértice v de grado a lo sumo 5. Eliminamos v y sus aristas incidentes. Por HI, el grafo restante se puede colorear con 5 colores.

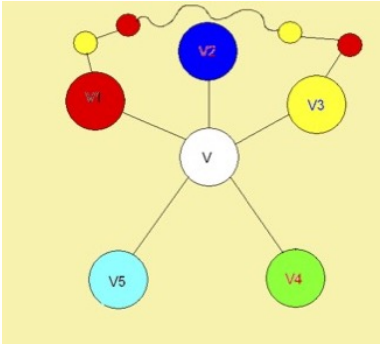
Si v tiene menos de 5 vecinos, al menos un color está disponible para colorearlo. El caso crítico ocurre cuando v tiene exactamente 5 vecinos con 5 colores distintos.



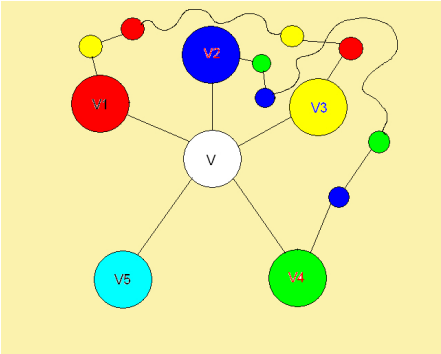
Sea v_1, v_2, v_3, v_4, v_5 los vecinos de v, cada uno con un color distinto. Si en el subgrafo inducido por los vecinos, dos vértices con colores *rojo* y *amarillo* (digamos v_1 y v_3) están en componentes diferentes, se puede intercambiar rojo y amarillo en el componente de v_1, liberando un color para v.



Si hay un camino entre v_1 y v_3, se puede intentar intercambiar otros pares de colores, como *verde* y *azul*:



En el caso extremo en que hay caminos entre v_1 y v_3, y también entre v_2 y v_4, los caminos deben cruzarse. Esto contradice la planaridad del grafo:



Por lo tanto, uno de los intercambios de colores es siempre posible, y v puede ser coloreado con un color distinto a sus vecinos. Esto completa la prueba. □

5.26. Árboles

5.26.1. Motivación e historia

Los árboles son estructuras fundamentales en matemáticas discretas y ciencias de la computación. Representan relaciones jerárquicas y son usadas en estructuras de datos, algoritmos, lenguajes de programación, bases de datos y más.

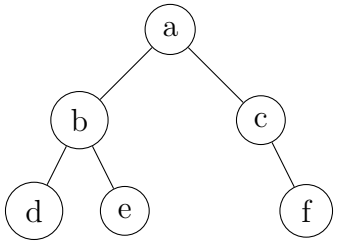
El término "árbol" fue introducido por Arthur Cayley en el siglo XIX para contar estructuras moleculares. Desde entonces, se ha convertido en un concepto central en teoría de grafos.

5.26.2. ¿Qué es un árbol?

Árbol

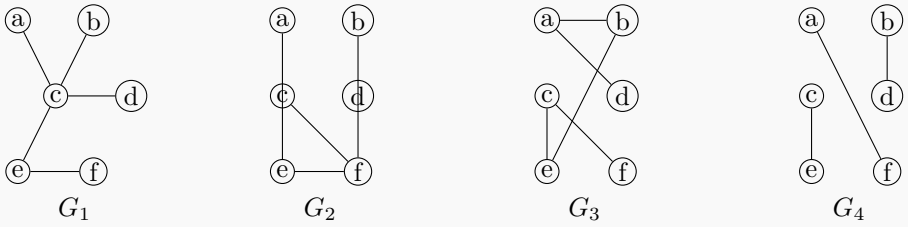
Un **árbol** es un grafo no dirigido que es *conexo* y *acíclico*. Es decir, existe un camino entre todo par de vértices, y no contiene ciclos simples.

Observación: Todo árbol debe ser necesariamente un grafo simple (es decir, sin lazos ni aristas paralelas).



5.26.3. Ejemplos y contraejemplos

¿Cuáles de los siguientes grafos son árboles?



5.26.4. Aplicaciones de los árboles

Los árboles son útiles en múltiples contextos, como se muestra a continuación:

- **Búsqueda y ordenamiento:** Los *árboles binarios de búsqueda* (BST) permiten organizar datos de forma que cada nodo tiene a lo sumo dos hijos. El hijo izquierdo contiene valores menores y el derecho, mayores. Esto permite búsquedas, inserciones y eliminaciones en tiempo promedio $O(\log n)$, si el árbol está balanceado. Son usados en estructuras de datos como sets, maps y en bases de datos.

- **Compresión de datos:** El algoritmo de *Huffman* construye un árbol binario donde cada hoja representa un símbolo, y la ruta desde la raíz define su código binario. Los símbolos más frecuentes tienen códigos más cortos, minimizando la longitud promedio del mensaje. Estos árboles son fundamentales en formatos como JPEG, MP3 y ZIP.

- **Modelado de juegos:** En teoría de juegos, los árboles modelan todos los posibles estados y jugadas. Cada nodo representa un estado, y las ramas, movimientos. Son esenciales en juegos como ajedrez o tic-tac-toe. Algoritmos como *minimax* y *poda alfa-beta* operan sobre estos árboles para decidir jugadas óptimas.

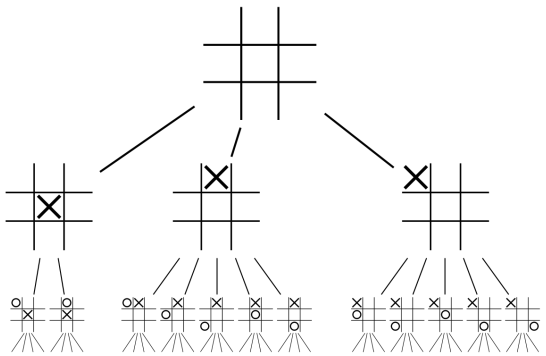


Figura: Árbol de decisiones del juego Tic-Tac-Toe.

- **Estructuración de información:** Formatos como *XML* y *JSON* representan información anidada de forma jerárquica usando árboles. Cada nodo es una etiqueta o clave, y sus hijos representan contenido anidado. Esto facilita la consulta (e.g. con XPath), la transformación, y la validación de documentos.

- **Representación de decisiones:** En *machine learning*, los árboles de decisión modelan decisiones mediante tests secuenciales sobre los datos. Cada nodo representa una pregunta; las ramas, respuestas posibles; y las hojas, una predicción. Son la base de modelos como CART, Random Forest y XGBoost, destacándose por su interpretabilidad.

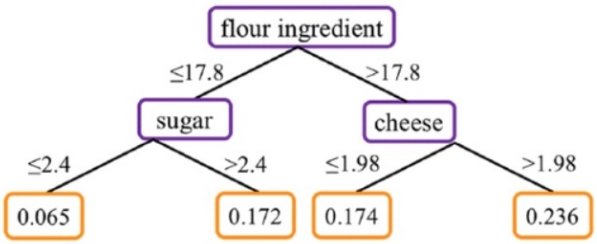


Figura: Árbol de regresión con tres predictores (harina, azúcar y queso), utilizado para clasificar países según su consumo y predecir prevalencia de obesidad en cuatro grupos distintos. Fuente: Dunstan, J., et al. (2020). Predicting nationwide obesity from food sales using machine learning.

5.26.5. Caracterizaciones alternativas

A primera vista, la definición de árbol puede parecer un tanto arbitraria: un grafo que es *conexo* y *acíclico*. Pero, ¿por qué esas dos condiciones en particular? Para responder esta pregunta, es útil considerar qué implican esas propiedades en términos de la estructura del grafo.

Conexión significa que hay un camino entre cada par de nodos. Es decir, el grafo debe tener suficientes aristas para mantener todos los vértices conectados en una sola componente. Por otro lado, **no tener ciclos** nos obliga a restringir el número de conexiones: no puede haber más aristas que las necesarias, porque cualquier exceso podría cerrar un ciclo.

Los árboles, entonces, se sitúan exactamente en el *punto de equilibrio* entre tener suficientes aristas para ser conexos, pero no tantas como para crear ciclos. Este balance permite que muchas propiedades estructurales equivalentes emerjan, como veremos en el siguiente teorema.

Teorema

Sea $G = (V, E)$ un grafo no dirigido. Las siguientes propiedades son equivalentes:

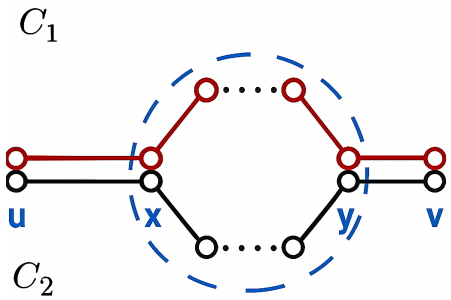
1. G es un árbol;
2. G contiene un único camino simple entre cada par de nodos distintos;
3. G es conexo y $|E| = |V| - 1$.

Demostraciones

1 ⇒ 2: Sea u y v vértices distintos de G. Como G es un árbol, es conexo y por lo tanto existe un camino simple que los conecta.

Supongamos que hay dos caminos simples distintos entre u y v: C_1 y C_2. Esto implicaría la existencia de un ciclo simple en G, contradiciendo la hipótesis.

Si divergen antes de finalizar, los caminos generan un ciclo entre el primer punto de divergencia x y el de reencuentro y:



2 ⇒ 1: Por hipótesis, existe un único camino simple entre cada par de nodos. Esto implica que G es conexo. Supongamos ahora que G contiene un ciclo simple. Entonces, entre dos nodos de ese ciclo hay más de un camino simple, contradiciendo la unicidad. □

1 ⇒ 3: Demostración por inducción fuerte sobre $|E|$.

HI: Para todo árbol con $|E| \leq d$, se cumple $|E| = |V| - 1$.

Consideremos un árbol cualquiera G con $|E| = d + 1$. Sea $G \setminus a$ el grafo que se forma quitando de G la arista a tal que el árbol no es conexo, i.e., $G \setminus a = G_1 \cup G_2$, donde G_1 y G_2 son árboles con no más de d aristas. Por la caracterización 2, toda arista es esencial: si se elimina, el grafo se desconecta.

Aplicamos a G_1 y G_2 la HI: $|E(G_i)| = |V_i| - 1$ y sumando obtenemos $|E| - 1 = (|V_1| - 1) + (|V_2| - 1) = |V| - 2$, por lo tanto $|E| = |V| - 1$. □

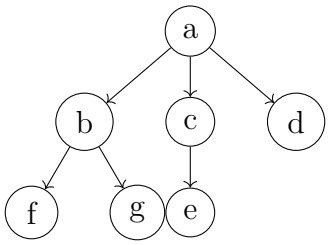
3 ⇒ 1: Supongamos que G es conexo y $|E| = |V|-1$, pero que contiene un ciclo. Eliminamos una arista del ciclo. El nuevo grafo G' sigue siendo conexo pero ahora tiene $|E| < |V| - 1$, lo

cual es imposible.

De hecho, **todo grafo conexo debe tener al menos** $|V|-1$ **aristas**, pues de lo contrario no es posible conectar todos los vértices. □

5.26.6. Árboles con raíz

En muchas aplicaciones es útil distinguir un nodo particular como la **raíz** del árbol. A menudo, en los **árboles con raíz** se orientan las aristas para que apunten *lejos* de la raíz.



Terminología básica

Sea T un árbol con raíz v:

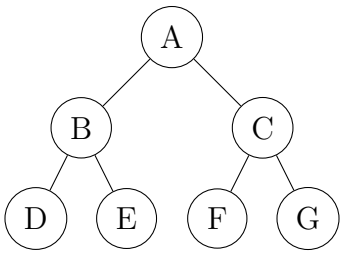
- El **padre** de un nodo $u \neq v$ es el vértice desde el cual se llega a u por una arista dirigida. A su vez, u es un **hijo** de ese vértice.
- Dos nodos son **hermanos** si tienen el mismo padre.
- Los **ancestros** de u son los nodos en el camino desde u hacia la raíz. Sus **descendientes** son los nodos que tienen a u como ancestro.
- Una **hoja** es un nodo sin hijos; los demás se llaman **internos**.
- La **altura** del árbol es la longitud del camino más largo desde la raíz hacia cualquier hoja.

5.26.7. Árboles m-arios

A veces interesa restringir la cantidad de hijos que puede tener un nodo.

Definición

Un árbol con raíz es m**-ario** ($m \geq 1$) si cada nodo interno tiene a lo más m hijos. Es **completo** si todos los nodos internos tienen exactamente m hijos.



Propiedades fundamentales

Teorema

- Si T es un árbol m-ario de altura h, entonces tiene a lo sumo m^h hojas.
- Si T es un árbol m-ario completo con i nodos internos, entonces tiene $m \cdot i + 1$ nodos en total.

Ejercicios

P1 Sea G un grafo no dirigido. Probar que G es un árbol si y sólo si G es conexo, pero sacarle cualquier arista lo vuelve disconexo.

Demostración.

$\Rightarrow$ Asuma que $G = (V, E)$ es un árbol. Queremos probar que quitar cualquier arista lo desconecta.

- Como G es un árbol, es conexo por definición.
- Supongamos por contradicción que existe una arista $e \in E$ tal que el grafo $G' = (V, E \setminus \{e\})$ sigue siendo conexo.
- Entonces G' sería un grafo conexo y sin ciclos (porque G no tenía ciclos y quitar una arista no puede crear uno), es decir, también un árbol.
- Pero entonces: $|E| = |V| - 1$ y $|E \setminus \{e\}| = |V| - 2$, lo que contradice la caracterización 3.

$\Leftarrow$ Asuma que $G = (V, E)$ es conexo, y que quitar cualquier arista lo desconecta. Queremos probar que G no tiene ciclos simples.

- Supongamos por contradicción que G contiene un ciclo simple C.
- Entonces podemos quitar una arista de C y el grafo seguiría siendo conexo, ya que los vértices del ciclo seguirían conectados por el resto del ciclo.

- Esto contradice la hipótesis de que quitar cualquier arista desconecta G.

□

P2 Demuestra que cualquier árbol es un grafo bipartito.

Demostración. Como T es un árbol, no contiene ciclos. Vamos a demostrar que T es bipartito, es decir, que su conjunto de vértices V se puede dividir en dos subconjuntos disjuntos V_1 y V_2 tal que ninguna arista conecta dos vértices dentro del mismo subconjunto.

Elegimos un vértice arbitrario $v \in V$ y definimos V_1 como el conjunto de vértices a distancia par de v y V_2 como aquellos a distancia impar.

Como T es un grafo sin ciclos, la distancia entre dos vértices cualesquiera está bien definida y única. Toda arista conecta dos vértices cuyas distancias respecto a v difieren en exactamente 1 (ya que no hay ciclos), por lo que nunca une dos vértices ambos en V_1 ni ambos en V_2.

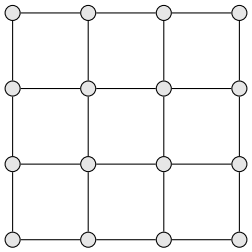
Por lo tanto, T es bipartito. □

P3 Ecuentre el número de árboles no isomórficos con 5 vértices

5.27. Árboles Generadores

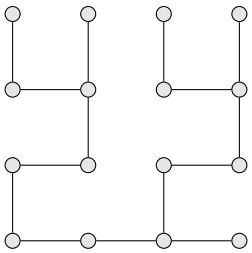
5.27.1. Motivación: Conectividad Mínima

Supongamos que tenemos una red vial entre varias ciudades, en la que todas las rutas son caminos de tierra. Queremos pavimentar la menor cantidad posible de caminos, de modo que sea posible viajar entre cualquier par de ciudades usando sólo caminos pavimentados.



Pregunta: ¿Cuál es la mínima cantidad de caminos que deben pavimentarse?

La respuesta está dada por encontrar un subgrafo que conecte todas las ciudades (vértices) y no tenga ciclos. Este subgrafo es un **árbol generador** del grafo original.



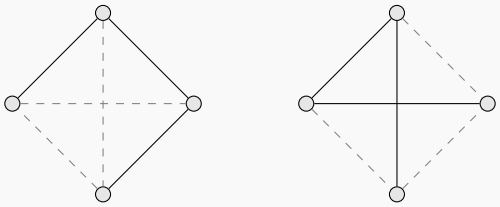
Árbol Generador

Un **árbol generador** de un grafo $G = (V, E)$ es un subgrafo $T = (V, E')$ tal que:

- T contiene todos los vértices de G, es decir, $V(T) = V(G)$.
- T es un árbol: está conectado y no contiene ciclos.

Este tipo de subgrafo no es único: un mismo grafo puede admitir múltiples árboles generadores diferentes, dependiendo de qué aristas se elijan para mantener la conectividad y evitar los ciclos.

Ejemplo: Dos árboles generadores distintos del mismo grafo.



Aplicación: Recorrido Eficiente

Una aplicación clave de los árboles generadores es la de recorrer todos los vértices de un grafo de forma eficiente. Por ejemplo, un *rastreador web* (web crawler), que recolecta datos para un motor de búsqueda, necesita visitar todos los documentos conectados por hipervínculos. Para lograrlo de manera eficiente, puede construirse un árbol generador del grafo de enlaces: este árbol permite visitar cada nodo exactamente una vez, garantizando así un recorrido completo en **tiempo lineal** respecto al número de vértices.

5.27.2. Caracterización de Grafos que Admiten Árbol Generador

Una vez comprendida la noción de árbol generador y su importancia para garantizar conectividad mínima sin redundancia, surge una pregunta natural: ¿qué condiciones debe cumplir un grafo para que sea posible extraerle un árbol generador?

Teorema

Un grafo G admite un árbol generador si y sólo si es conexo.

Demostración. ($\Rightarrow$) Si G tiene un árbol generador, entonces G debe ser conexo, porque un árbol conecta todos los vértices por definición.

($\Leftarrow$) Sea G un grafo conexo. Si G ya es un árbol, entonces el resultado es inmediato. Si G no es un árbol, entonces contiene al menos un ciclo simple. Podemos remover una arista de ese ciclo sin desconectar el grafo (porque sigue existiendo un camino entre los extremos de esa arista usando el resto del ciclo).

Repetimos este proceso mientras haya ciclos: eliminamos una arista de algún ciclo simple. Este proceso debe terminar porque G tiene un número finito de aristas, y cada vez eliminamos una. El resultado final es un subgrafo conexo sin ciclos, es decir, un árbol generador de G. □

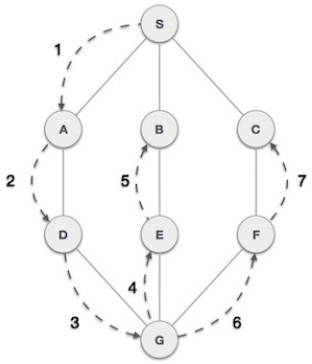
5.27.3. Construcción de Árboles Generadores

La prueba anterior nos entrega un procedimiento para construir árboles generadores. Sin embargo, este método no es muy eficiente, ya que requiere identificar ciclos simples dentro del grafo. Afortunadamente, existen algoritmos más eficientes y ampliamente utilizados en la práctica para este propósito. Los más conocidos son:

- **DFS (Depth First Search):** búsqueda en profundidad.
- **BFS (Breadth First Search):** búsqueda a lo ancho.

Búsqueda en Profundidad (DFS)

Dado un grafo no dirigido simple G, el algoritmo DFS parte desde un vértice inicial y avanza lo más lejos posible por cada rama antes de retroceder (*backtracking*). De esta forma, se construye un camino largo sin repetir vértices hasta que ya no se pueda continuar, y luego se regresa al nodo anterior para explorar otras ramas.



Algoritmo DFS

DFS(G grafo conectado con vértices $v_1, v_2, \ldots, v_n$):

$T :=$ árbol que contiene solo el vértice v_1

visita(v_1)

visita(v vértice de G):

Para cada vértice w adyacente a v y no en T hacer

agregar w y la arista (v, w) a T

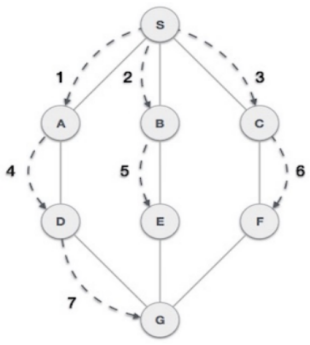
visita(w)

Complejidad del DFS

- Cada vértice v es visitado exactamente una vez: se llama al procedimiento `visita(`v`)` cuando v se encuentra por primera vez.
- Cada arista se examina como máximo dos veces: una desde cada uno de sus extremos.
- Por lo tanto, DFS tiene complejidad temporal $O(e)$ en representación por listas de adyacencia, o $O(n^2)$ en representación por matrices, donde n es el número de vértices y e el número de aristas.

Búsqueda a lo Ancho (BFS)

El algoritmo de búsqueda a lo ancho (BFS) también permite construir árboles generadores en grafos conexos. A diferencia del DFS, que explora lo más profundo posible antes de retroceder, el BFS explora todos los vecinos inmediatos de un nodo antes de continuar con los vecinos de esos vecinos, nivel por nivel.



Algoritmo BFS

BFS(G grafo conectado con vértices $v_1, v_2, \ldots, v_n$):

T := árbol que contiene solo el vértice v_1

Q := cola que contiene inicialmente solo v_1

Mientras Q no esté vacía hacer:

v := extraer primer elemento de Q

Para cada vértice w adyacente a v y no en T hacer

agregar w y la arista (v, w) a T

insertar w al final de Q

Complejidad del BFS

De manera análoga al análisis de DFS:

- Cada vértice se visita exactamente una vez.
- Cada arista se examina como máximo dos veces: una vez desde cada uno de sus extremos.
- Por lo tanto, el algoritmo BFS también tiene una complejidad de $O(e)$ en representación con listas de adyacencia, o $O(n^2)$ con matrices de adyacencia, siendo n el número de vértices y e el número de aristas.

5.27.4. Ejercicios Propuestos

- **Ejercicio:** Describa los árboles generadores producidos por los algoritmos DFS y BFS en una rueda W_n, comenzando por el vértice central.
  Una rueda W_n es un grafo con n vértices que se forma conectando un único vértice a todos los vértices de un ciclo-$(n-1)$.
- **Ejercicio:** Describa los árboles generadores producidos por los algoritmos DFS y BFS en un grafo completo K_n.
- **Ejercicio:** Sean T y T' dos árboles generadores del mismo grafo simple G. Asuma que existe un arco e en T que no pertenece a T'. Demuestre que existe un arco e' en T' tal que e' no está en T, y tal que T continúa siendo un spanning tree si le sacamos a e y le agregamos a e', y T' continúa siendo un spanning tree si le sacamos a e' y le agregamos a e.

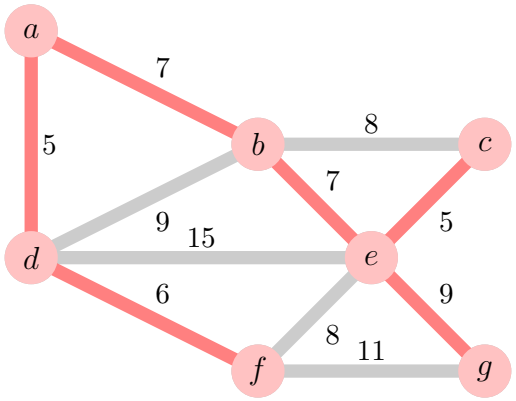
5.28. Árboles Generadores de Costo Mínimo

5.28.1. Definición y Motivación

Si un grafo G tiene costos asociados a cada arista, un problema natural es encontrar un árbol generador de **peso mínimo**. Esto significa hallar un subconjunto acíclico de aristas que conecte todos los vértices y cuya suma de pesos sea mínima.

Árbol Generador de Peso Mínimo

Dado un grafo simple G, un **árbol generador de peso mínimo** es un árbol generador de G tal que la suma de los pesos de sus aristas es mínima.



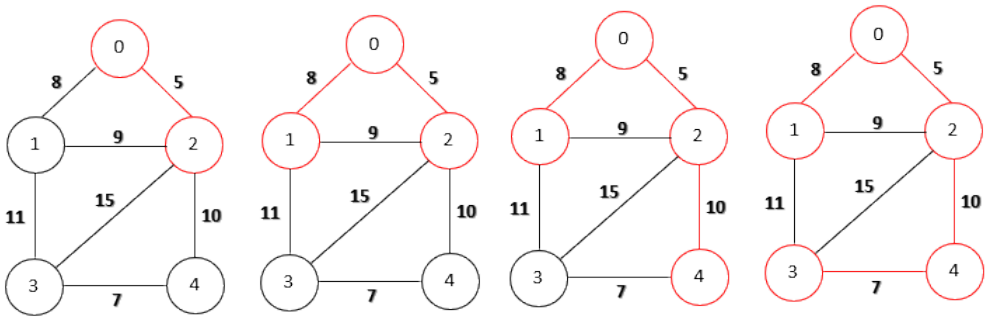
Hay dos algoritmos clásicos para encontrar árboles generadores mínimos: **Prim** y **Kruskal**.

5.28.2. Algoritmo de Prim

Algoritmo de Prim

Dado un grafo simple $G = (V, E)$, $n = |V|$, calcular el árbol de costo mínimo como sigue:

1. Escoger el arco de *menor peso* e incluirlo en el árbol.
2. Repetidamente escoger el arco de menor peso incidente a un vértice existente del árbol, sin formar ciclos.
3. Repetir hasta obtener $n - 1$ aristas.

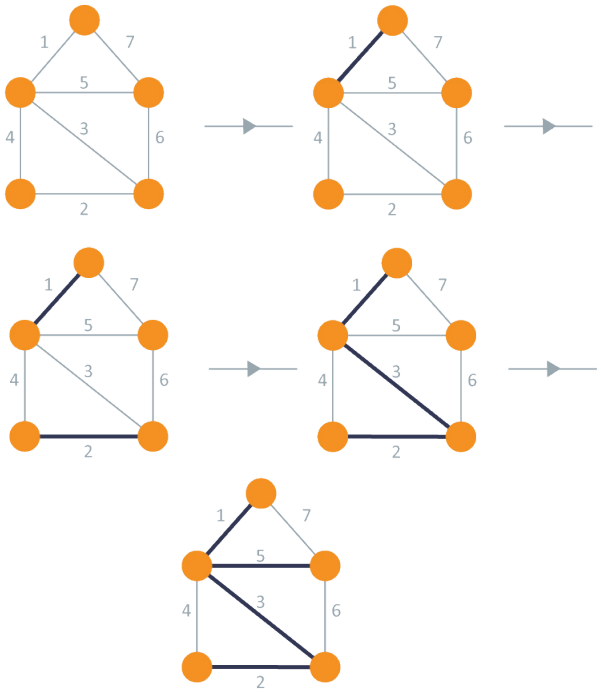


5.28.3. Algoritmo de Kruskal

Algoritmo de Kruskal

Dado un grafo simple $G = (V, E)$, $n = |V|$, calcular el árbol de costo mínimo como sigue:

1. Ordenar las aristas por peso creciente.
2. Añadir aristas una a una al árbol si no forman ciclos.
3. Detenerse al obtener $n - 1$ aristas.



5.28.4. Correctitud de los algoritmos

Tanto Prim como Kruskal garantizan encontrar un árbol generador de peso mínimo. Sus demostraciones se basan en el principio del intercambio: si una arista no está en el árbol mínimo pero puede reemplazar otra sin aumentar el peso total, entonces el nuevo árbol sigue siendo mínimo.

Teorema (correctitud algoritmo Prim)

Dado un grafo simple $G = (V, E)$ de $n = |V|$ vértices y $e = |E|$ arcos, el algoritmo de Prim encuentra un árbol generador de peso mínimo.

Demostración. Mostraremos la correctitud del algoritmo de Prim usando inducción matemática. Sea T el conjunto de aristas seleccionadas por el algoritmo de Prim en cada iteración, y sea M un árbol generador mínimo (MST).

Hipótesis inductiva: Después de cada iteración, el árbol parcial T es un subgrafo de algún MST M.

Caso base: Al inicio, $T = \emptyset$, por lo tanto T es un subgrafo de cualquier MST.

Paso inductivo: Supongamos que antes de una iteración, el árbol parcial T es subgrafo de un MST M. Sea $e = (u, v)$ la arista que el algoritmo de Prim selecciona para agregar a T en esta iteración. Como Prim solo añade aristas que conectan un vértice en T con uno fuera, entonces e conecta T con el resto del grafo.

- Si $e \in M$, entonces $T \cup \{e\}$ sigue siendo subgrafo de M, y la hipótesis se mantiene.
- Si $e \notin M$, entonces al agregar e a M se forma un ciclo, ya que M es un árbol.
- En ese ciclo, debe existir otra arista f que también conecta un vértice en T con otro fuera de T.
- Como Prim eligió e en vez de f, debe cumplirse que $w(e) \leq w(f)$.
- Al eliminar f del ciclo y agregar e, obtenemos un nuevo árbol generador M' cuyo peso total es menor o igual al de M, y que contiene a $T \cup \{e\}$.

Por lo tanto, por inducción, el árbol construido por Prim es un subgrafo de algún MST en cada paso, y al finalizar tiene $n - 1$ aristas y conecta todos los nodos. Así, el árbol resultante es un MST.

□

Teorema (correctitud algoritmo Kruskal)

Dado un grafo simple $G = (V, E)$ de $n = |V|$ vértices y $e = |E|$ arcos, el algoritmo de Kruskal encuentra un árbol generador de peso mínimo.