

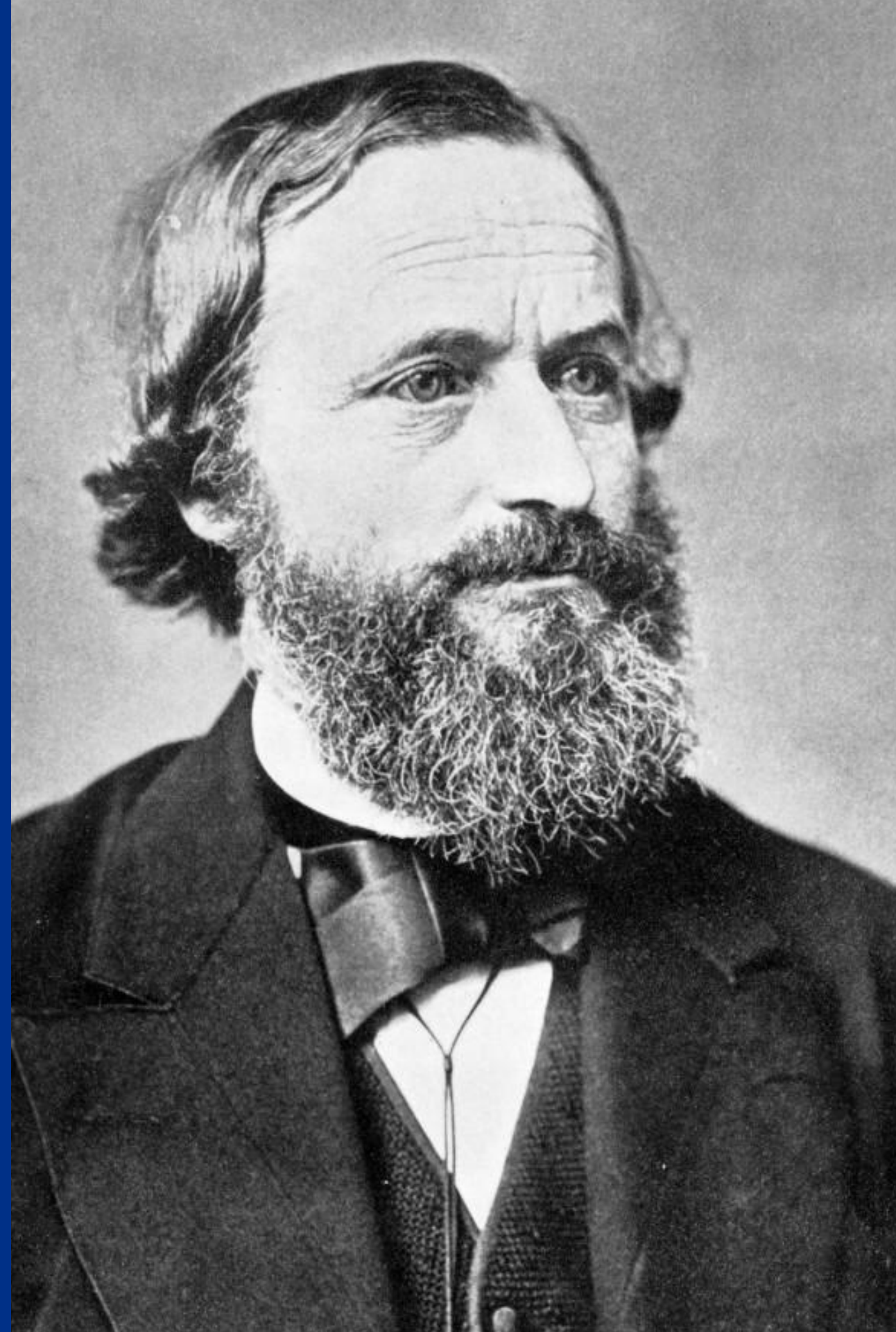
# Tema 4-3

## Conectividad y árboles



UNIVERSIDAD  
DE CHILE

Arturo Merino



**Recuerdo/Plan**

## ■ Recuerdo y plan

- **Recuerdo:**

## Recuerdo y plan

- **Recuerdo:**
  - Grafos, subgrafos y grados.

# Recuerdo y plan

- **Recuerdo:**

- Grafos, subgrafos y grados.
- Isomorfismos e invariantes.

# Recuerdo y plan

- **Recuerdo:**

- Grafos, subgrafos y grados.
- Isomorfismos e invariantes.

- **Plan:**

# Recuerdo y plan

- **Recuerdo:**

- Grafos, subgrafos y grados.
- Isomorfismos e invariantes.

- **Plan:**

- Conectividad y componentes.

# Recuerdo y plan

- **Recuerdo:**

- Grafos, subgrafos y grados.
- Isomorfismos e invariantes.

- **Plan:**

- Conectividad y componentes.
- Árboles y árboles enraizados.

# Recuerdo y plan

- **Recuerdo:**

- Grafos, subgrafos y grados.
- Isomorfismos e invariantes.

- **Plan:**

- Conectividad y componentes.
- Árboles y árboles enraizados.
- Conectividad en digrafos.

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

Def.

Sea  $G = (V, E)$  un grafo no dirigido.

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un *uv-camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un *uv-camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

- $v_0 = u$  y  $v_k = v$ ,

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un  *$uv$ -camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

- $v_0 = u$  y  $v_k = v$ ,
- $v_{i-1}v_i \in E$ .

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un  *$uv$ -camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

- $v_0 = u$  y  $v_k = v$ ,
- $v_{i-1}v_i \in E$ .

Diremos además que:

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un  *$uv$ -camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

- $v_0 = u$  y  $v_k = v$ ,
- $v_{i-1}v_i \in E$ .

Diremos además que:

- El camino es *simple* si todos sus vértices son distintos.

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un *uv-camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

- $v_0 = u$  y  $v_k = v$ ,
- $v_{i-1}v_i \in E$ .

Diremos además que:

- El camino es *simple* si todos sus vértices son distintos.
- Es *cerrado* si  $v_0 = v_k$ .

# Conectividad

- Invariante más fundamentales: puedo ir de cualquier punto a otro.
  - Rutas entre ciudades, comunicación entre servidores, estados de un programa.

## Def.

Sea  $G = (V, E)$  un grafo no dirigido.

Un  *$uv$ -camino* (o simplemente *camino*) de largo  $k$  es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $v_{i-1}v_i \in E$  para todo  $i = 1, \dots, k$ .

- $v_0 = u$  y  $v_k = v$ ,
- $v_{i-1}v_i \in E$ .

Diremos además que:

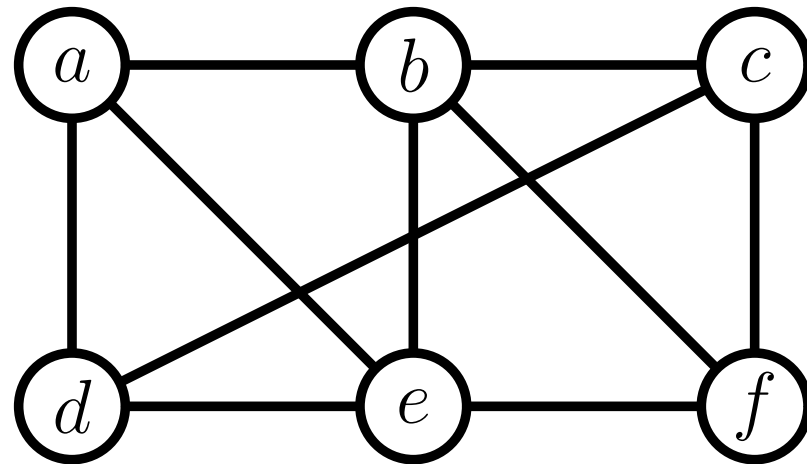
- El camino es *simple* si todos sus vértices son distintos.
- Es *cerrado* si  $v_0 = v_k$ .

- ¿Hay  $vv$ -caminos?

**Conectividad**

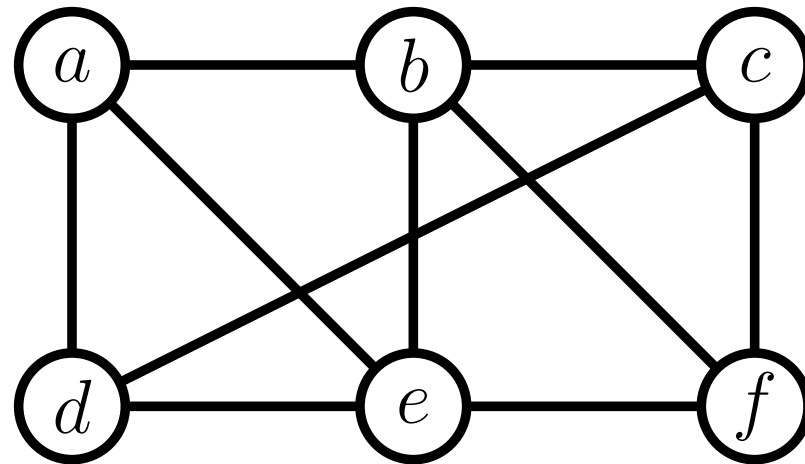
# Caminos

- Ej:



# Caminos

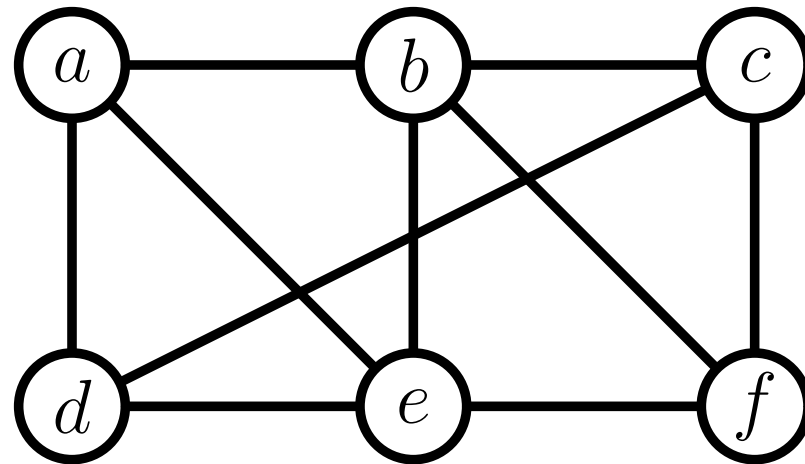
- Ej:



- **Conexión:** basta restringirnos a caminos simples.

# Caminos

- Ej:



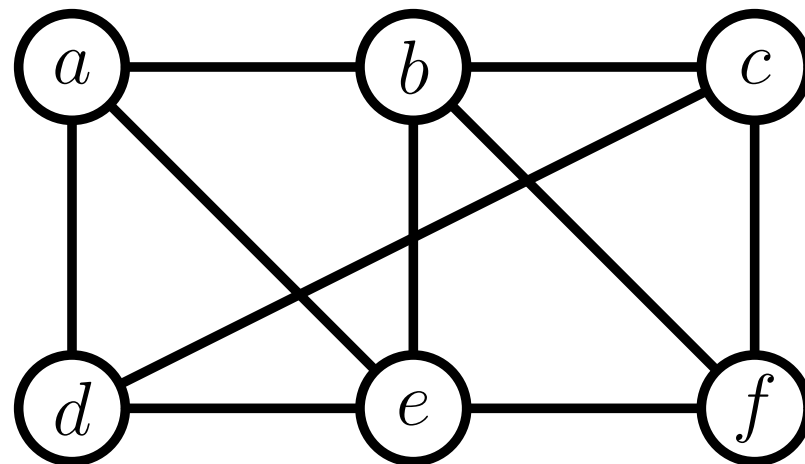
- **Conexión:** basta restringirnos a caminos simples.

Lem.

Sea  $G = (V, E)$  un grafo y sean  $u, v \in V$ .

# Caminos

- Ej:



- **Conexión:** basta restringirnos a caminos simples.

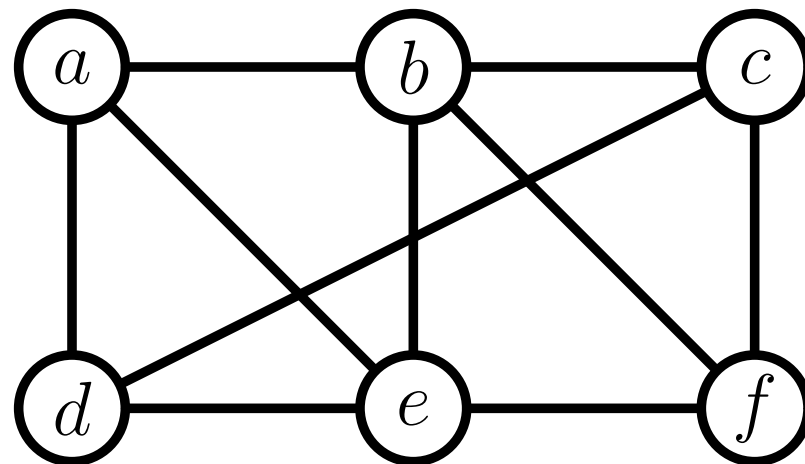
Lem.

Sea  $G = (V, E)$  un grafo y sean  $u, v \in V$ .

Si existe  $uv$ -camino, entonces existe un  $uv$ -camino simple.

# Caminos

- Ej:



- **Conexión:** basta restringirnos a caminos simples.

Lem.

Sea  $G = (V, E)$  un grafo y sean  $u, v \in V$ .

Si existe  $uv$ -camino, entonces existe un  $uv$ -camino simple.

- Dem:

# Conectividad

- **Conectividad/conexidad:** ir de cualquier vértice a cualquier otro.

# Conectividad

- **Conectividad/conexidad:** ir de cualquier vértice a cualquier otro.

Def.

Sea  $G = (V, E)$  un grafo, con  $n = |V|$ .

# Conectividad

- **Conectividad/conexidad:** ir de cualquier vértice a cualquier otro.

Def.

Sea  $G = (V, E)$  un grafo, con  $n = |V|$ .

Un grafo  $G = (V, E)$  es *conexo* si

# Conectividad

- **Conectividad/conexidad:** ir de cualquier vértice a cualquier otro.

Def.

Sea  $G = (V, E)$  un grafo, con  $n = |V|$ .

Un grafo  $G = (V, E)$  es *conexo* si

◦  $n \geq 1$  y

# Conectividad

- **Conectividad/conexidad:** ir de cualquier vértice a cualquier otro.

Def.

Sea  $G = (V, E)$  un grafo, con  $n = |V|$ .

Un grafo  $G = (V, E)$  es *conexo* si

- $n \geq 1$  y
- para todo par de vértices distintos  $u, v \in V$  existe un camino (simple) desde  $u$  hasta  $v$ .

# Conectividad

- **Conectividad/conexidad:** ir de cualquier vértice a cualquier otro.

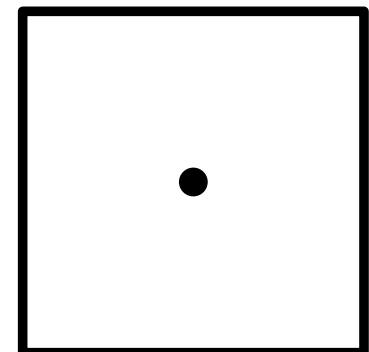
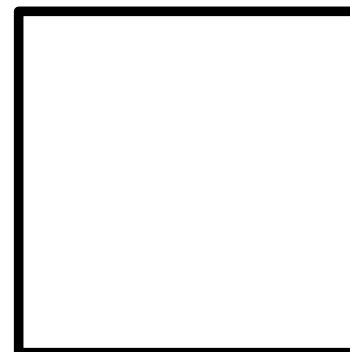
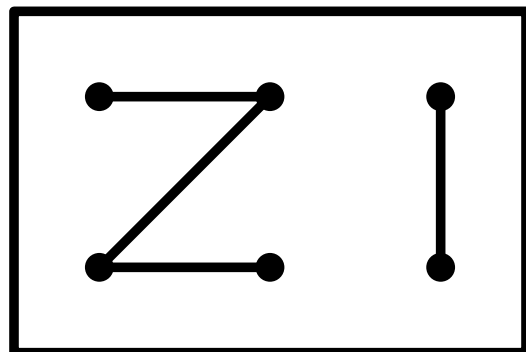
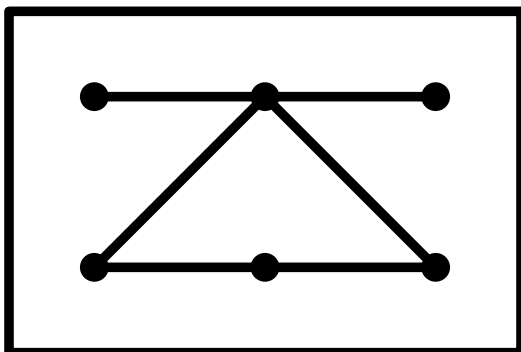
Def.

Sea  $G = (V, E)$  un grafo, con  $n = |V|$ .

Un grafo  $G = (V, E)$  es *conexo* si

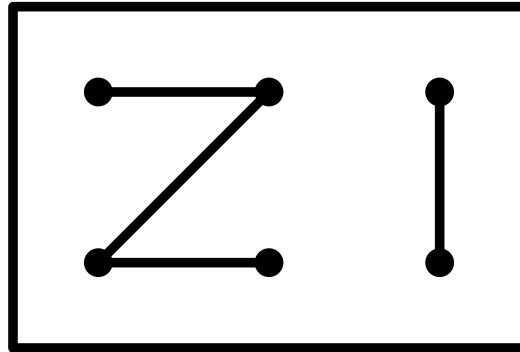
- $n \geq 1$  y
- para todo par de vértices distintos  $u, v \in V$  existe un camino (simple) desde  $u$  hasta  $v$ .

- Ej:



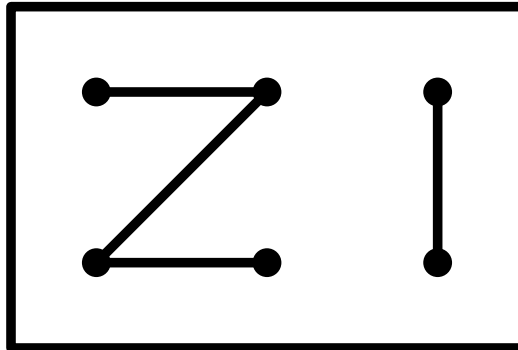
## Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



## Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?

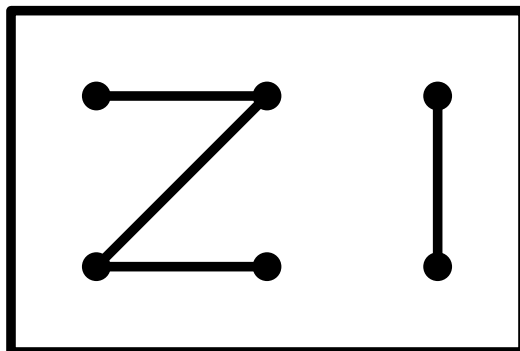


Def.

Sea  $G$  un grafo no dirigido.

# Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



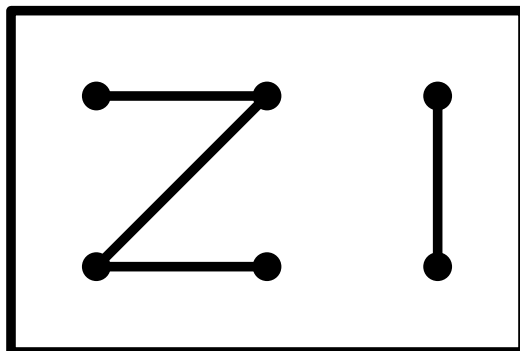
Def.

Sea  $G$  un grafo no dirigido.

Una *componente conexa* de  $G$  es un subgrafo  $H$  de  $G$  tal que:

# Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



Def.

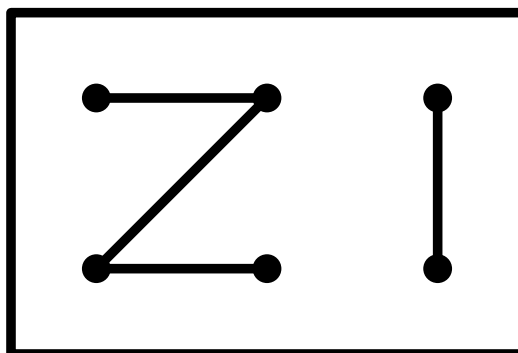
Sea  $G$  un grafo no dirigido.

Una *componente conexa* de  $G$  es un subgrafo  $H$  de  $G$  tal que:

- $H$  es conexo,

# Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



Def.

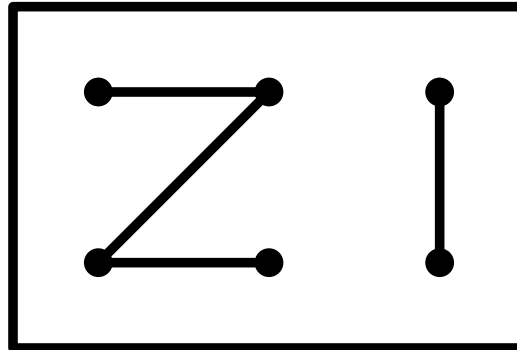
Sea  $G$  un grafo no dirigido.

Una *componente conexa* de  $G$  es un subgrafo  $H$  de  $G$  tal que:

- $H$  es conexo,
- $H$  no es subgrafo propio de ningún subgrafo conexo de  $G$ .

# Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



Def.

Sea  $G$  un grafo no dirigido.

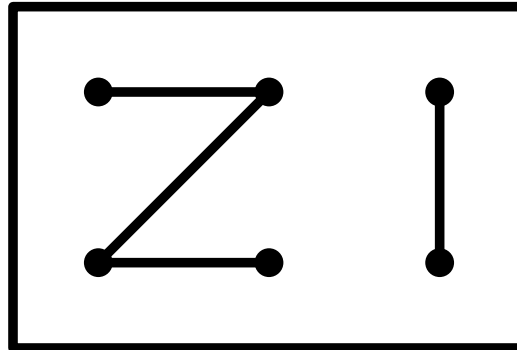
Una *componente conexa* de  $G$  es un subgrafo  $H$  de  $G$  tal que:

- $H$  es conexo,
- $H$  no es subgrafo propio de ningún subgrafo conexo de  $G$ .

Es decir, una componente conexa es un subgrafo conexo maximal.

# Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



Def.

Sea  $G$  un grafo no dirigido.

Una *componente conexa* de  $G$  es un subgrafo  $H$  de  $G$  tal que:

- $H$  es conexo,
- $H$  no es subgrafo propio de ningún subgrafo conexo de  $G$ .

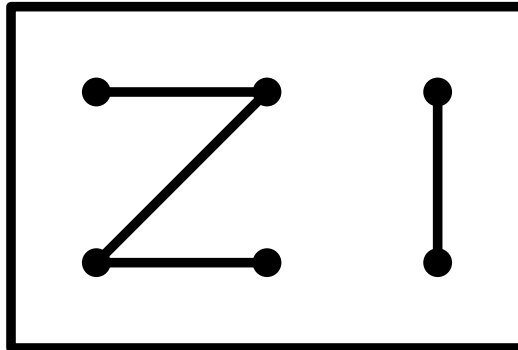
Es decir, una componente conexa es un subgrafo conexo maximal.

Teo.

Todo grafo es la unión disjunta de sus componentes conexas.

# Componentes conexas

- Si un grafo no es conexo: ¿cómo se quienes pueden llegar a quienes?



## Def.

Sea  $G$  un grafo no dirigido.

Una *componente conexa* de  $G$  es un subgrafo  $H$  de  $G$  tal que:

- $H$  es conexo,
- $H$  no es subgrafo propio de ningún subgrafo conexo de  $G$ .

Es decir, una componente conexa es un subgrafo conexo maximal.

## Teo.

Todo grafo es la unión disjunta de sus componentes conexas.

- **Dem:**

# Consecuencias

Prop.

Un grafo  $G$  es conexo si y sólo si tiene exactamente una componente conexa.

# Consecuencias

Prop.

Un grafo  $G$  es conexo si y sólo si tiene exactamente una componente conexa.

Prop.

Si dos grafos simples son isomorfos, entonces tienen la misma cantidad de componentes conexas.

# Consecuencias

Prop.

Un grafo  $G$  es conexo si y sólo si tiene exactamente una componente conexa.

Prop.

Si dos grafos simples son isomorfos, entonces tienen la misma cantidad de componentes conexas.

En particular, ser conexo es un invariante.

# Consecuencias

Prop.

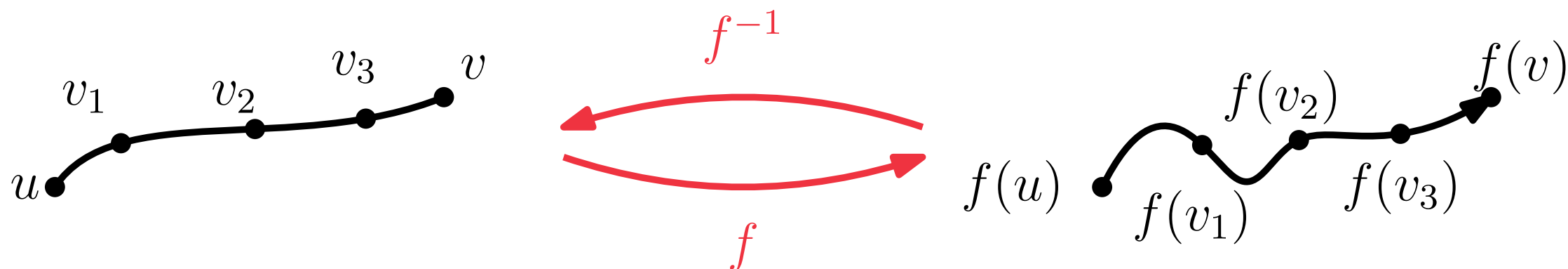
Un grafo  $G$  es conexo si y sólo si tiene exactamente una componente conexa.

Prop.

Si dos grafos simples son isomorfos, entonces tienen la misma cantidad de componentes conexas.

En particular, ser conexo es un invariante.

- **Idea:** el isomorfismo traslada los caminos.



Árboles

## Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?

## Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?
- Podemos describir un camino simple entre cada par de vértices.

## Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?
- Podemos describir un camino simple entre cada par de vértices.
- ¿Qué aristas remover?

## Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?
- Podemos describir un camino simple entre cada par de vértices.
- ¿Qué aristas remover?

Def.

Sea  $G$  un grafo simple.

## Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?
- Podemos describir un camino simple entre cada par de vértices.
- ¿Qué aristas remover?

Def.

Sea  $G$  un grafo simple.

En un grafo simple, un *ciclo* es un camino cerrado de largo  $k \geq 3$  en el que  $v_0, \dots, v_{k-1}$  son distintos.

## Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?
- Podemos describir un camino simple entre cada par de vértices.
- ¿Qué aristas remover?

### Def.

Sea  $G$  un grafo simple.

En un grafo simple, un *ciclo* es un camino cerrado de largo  $k \geq 3$  en el que  $v_0, \dots, v_{k-1}$  son distintos.

Un grafo es *acíclico* si no contiene ciclos.

# Certificar la conexidad

- ¿Cuántas aristas necesitamos para certificar que un grafo es conexo?
- Podemos describir un camino simple entre cada par de vértices.
- ¿Qué aristas remover?

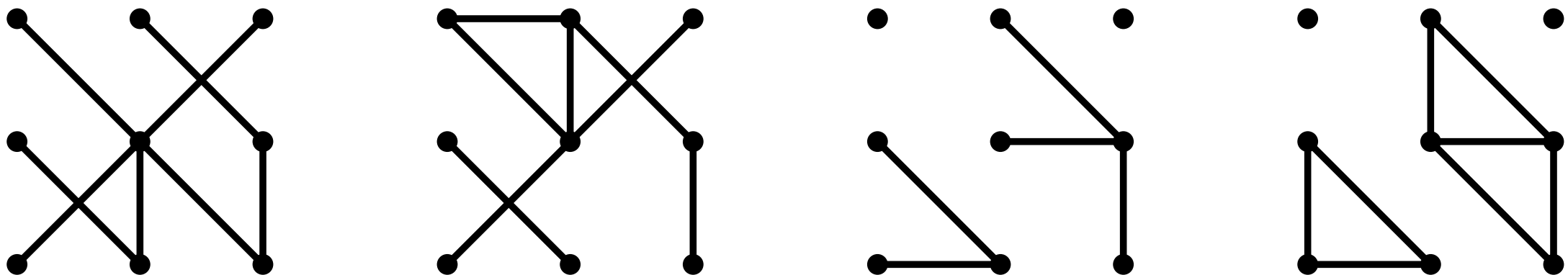
## Def.

Sea  $G$  un grafo simple.

En un grafo simple, un *ciclo* es un camino cerrado de largo  $k \geq 3$  en el que  $v_0, \dots, v_{k-1}$  son distintos.

Un grafo es *acíclico* si no contiene ciclos.

- Ej:





# Árboles

- Buscamos grafos que conecten sus vértices y no contengan ciclos.

# Árboles

- Buscamos grafos que conecten sus vértices y no contengan ciclos.

Def.

Sean  $T$  y  $F$  grafos simples.

# Árboles

- Buscamos grafos que conecten sus vértices y no contengan ciclos.

Def.

Sean  $T$  y  $F$  grafos simples.

Un grafo simple  $T$  se llama *árbol* si es conexo y acíclico, es decir, si no contiene ciclos.

# Árboles

- Buscamos grafos que conecten sus vértices y no contengan ciclos.

## Def.

Sean  $T$  y  $F$  grafos simples.

Un grafo simple  $T$  se llama *árbol* si es conexo y acíclico, es decir, si no contiene ciclos.

Un grafo simple  $F$  se llama también *bosque* si es acíclico.

# Árboles

- Buscamos grafos que conecten sus vértices y no contengan ciclos.

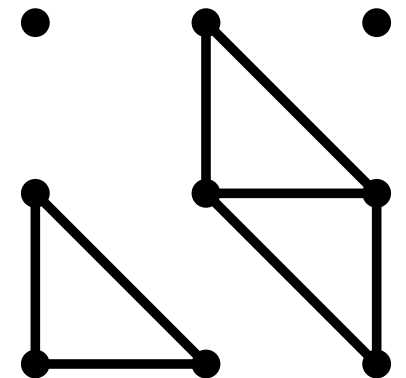
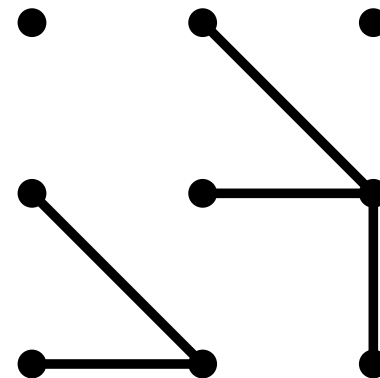
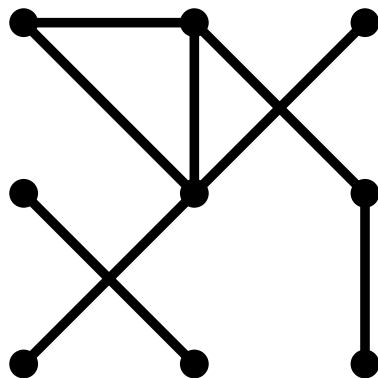
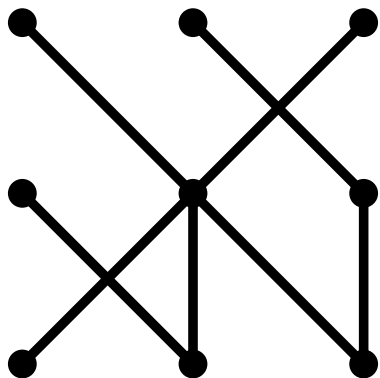
## Def.

Sean  $T$  y  $F$  grafos simples.

Un grafo simple  $T$  se llama *árbol* si es conexo y acíclico, es decir, si no contiene ciclos.

Un grafo simple  $F$  se llama también *bosque* si es acíclico.

- **Ej:** ¿Árbol/bosque?



# Estructura de los árboles

- Hay que certificar al menos un camino entre cada par.

## Estructura de los árboles

- Hay que certificar al menos un camino entre cada par.
- El árbol hace esto de manera **óptima**.

## Estructura de los árboles

- Hay que certificar al menos un camino entre cada par.
- El árbol hace esto de manera **óptima**.

Lem.

Sea  $T$  un árbol.

## Estructura de los árboles

- Hay que certificar al menos un camino entre cada par.
- El árbol hace esto de manera **óptima**.

Lem.

Sea  $T$  un árbol.

Entonces entre cada par de vértices distintos de  $T$  existe un único camino simple.

## Estructura de los árboles

- Hay que certificar al menos un camino entre cada par.
- El árbol hace esto de manera **óptima**.

Lem.

Sea  $T$  un árbol.

Entonces entre cada par de vértices distintos de  $T$  existe un único camino simple.

- **Dem:**

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

Una *hoja* de  $T$  es un vértice de grado 1.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

Una *hoja* de  $T$  es un vértice de grado 1.

- **Borrar** hojas preserva ser árbol.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

Una *hoja* de  $T$  es un vértice de grado 1.

- **Borrar** hojas preserva ser árbol.
- ¿Existencia?

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

Una *hoja* de  $T$  es un vértice de grado 1.

- **Borrar** hojas preserva ser árbol.
- ¿Existencia?

Lem.

Sea  $T$  un árbol con al menos dos vértices.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

Una *hoja* de  $T$  es un vértice de grado 1.

- **Borrar** hojas preserva ser árbol.
- ¿Existencia?

Lem.

Sea  $T$  un árbol con al menos dos vértices.

Entonces  $T$  tiene al menos dos hojas.

## Hojas de un árbol

- Muchas veces haremos razonamientos **inductivos** en árboles.
- Para esto queremos borrar un vértice y seguir siendo árbol.

Def.

Sea  $T$  un árbol.

Una *hoja* de  $T$  es un vértice de grado 1.

- **Borrar** hojas preserva ser árbol.
- ¿Existencia?

Lem.

Sea  $T$  un árbol con al menos dos vértices.

Entonces  $T$  tiene al menos dos hojas.

- **Dem:**

## Aristas de un árbol

- ¿Tamaño del árbol como certificado?

## Aristas de un árbol

- ¿Tamaño del árbol como certificado?

Prop.

Si  $T = (V, E)$  es un árbol con  $|V| = n$ , entonces  $|E| = n - 1$ .

## Aristas de un árbol

- ¿Tamaño del árbol como certificado?

Prop.

Si  $T = (V, E)$  es un árbol con  $|V| = n$ , entonces  $|E| = n - 1$ .

- **Dem:** Inducción en  $n$ .

## Aristas de un árbol

- ¿Tamaño del árbol como certificado?

Prop.

Si  $T = (V, E)$  es un árbol con  $|V| = n$ , entonces  $|E| = n - 1$ .

- **Dem:** Inducción en  $n$ .
- Uno puede mostrar que eso caracteriza los arboles.

## Aristas de un árbol

- ¿Tamaño del árbol como certificado?

Prop.

Si  $T = (V, E)$  es un árbol con  $|V| = n$ , entonces  $|E| = n - 1$ .

- **Dem:** Inducción en  $n$ .
- Uno puede mostrar que eso caracteriza los arboles.
- Hay **varias** caracterizaciones.

# Caraterización

Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

# Caraterización

Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

# Caraterización

Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

1.  $T$  es un árbol.

# Caraterización

Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

1.  $T$  es un árbol.
2.  $T$  es conexo y  $m = n - 1$ .

# Caraterización

## Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

1.  $T$  es un árbol.
2.  $T$  es conexo y  $m = n - 1$ .
3.  $T$  es conexo minimal: es conexo, y al borrar cualquier arista deja de ser conexo.

# Caraterización

## Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

1.  $T$  es un árbol.
2.  $T$  es conexo y  $m = n - 1$ .
3.  $T$  es conexo minimal: es conexo, y al borrar cualquier arista deja de ser conexo.
4.  $T$  es acíclico y  $m = n - 1$ .

# Caraterización

## Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

1.  $T$  es un árbol.
2.  $T$  es conexo y  $m = n - 1$ .
3.  $T$  es conexo minimal: es conexo, y al borrar cualquier arista deja de ser conexo.
4.  $T$  es acíclico y  $m = n - 1$ .
5.  $T$  es acíclico maximal: es acíclico, y al agregar cualquier arista entre dos vértices no adyacentes aparece un ciclo.

# Caraterización

## Teo.

Sea  $T = (V, E)$  un grafo simple con  $|V| = n \geq 1$  y  $|E| = m$ .

Las siguientes propiedades son equivalentes.

1.  $T$  es un árbol.
2.  $T$  es conexo y  $m = n - 1$ .
3.  $T$  es conexo minimal: es conexo, y al borrar cualquier arista deja de ser conexo.
4.  $T$  es acíclico y  $m = n - 1$ .
5.  $T$  es acíclico maximal: es acíclico, y al agregar cualquier arista entre dos vértices no adyacentes aparece un ciclo.
6. Entre cada par de vértices distintos de  $T$  existe un único camino simple.

# Árboles enraizados

# Árboles enraizados

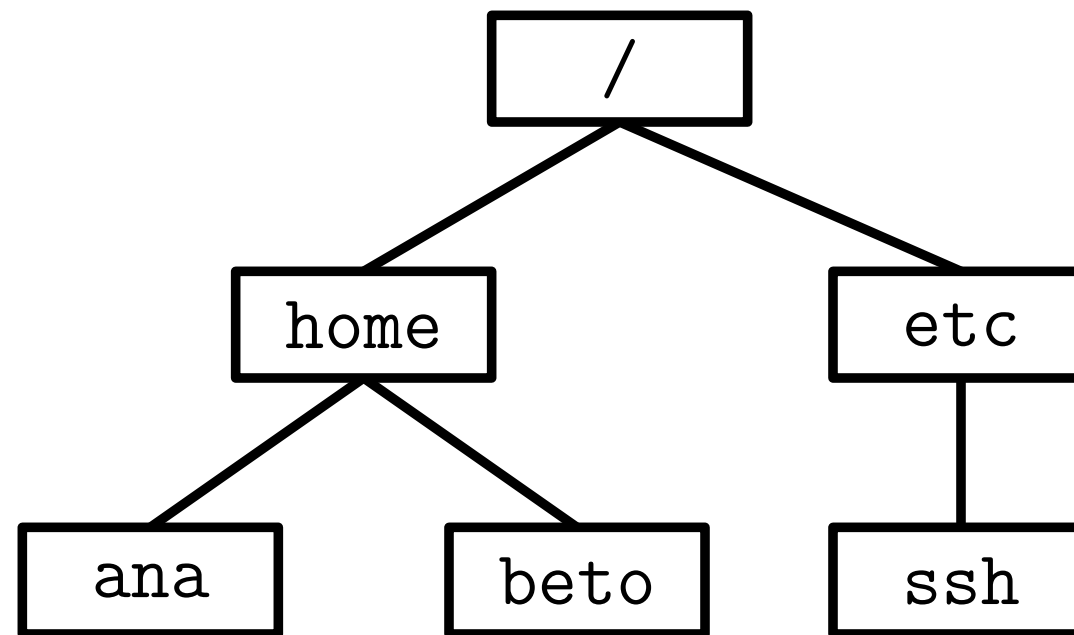
- Los árboles también sirven para modelar jerarquías.

# Árboles enraizados

- Los árboles también sirven para modelar jerarquías.
- Ej:

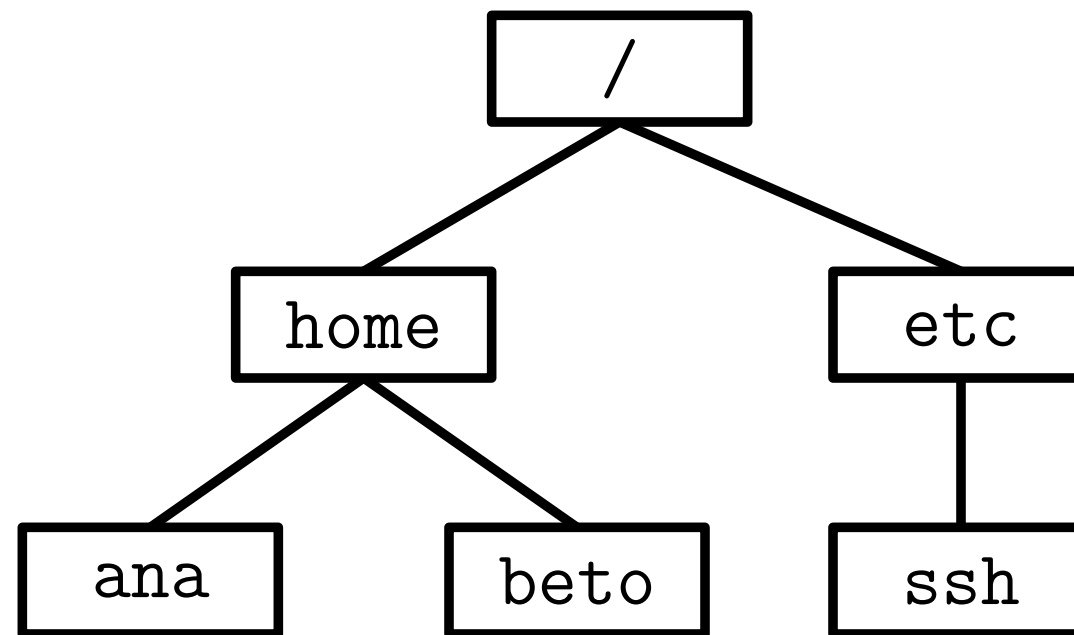
# Árboles enraizados

- Los árboles también sirven para modelar jerarquías.
- **Ej:**
  - Estructuras de directorios.



# Árboles enraizados

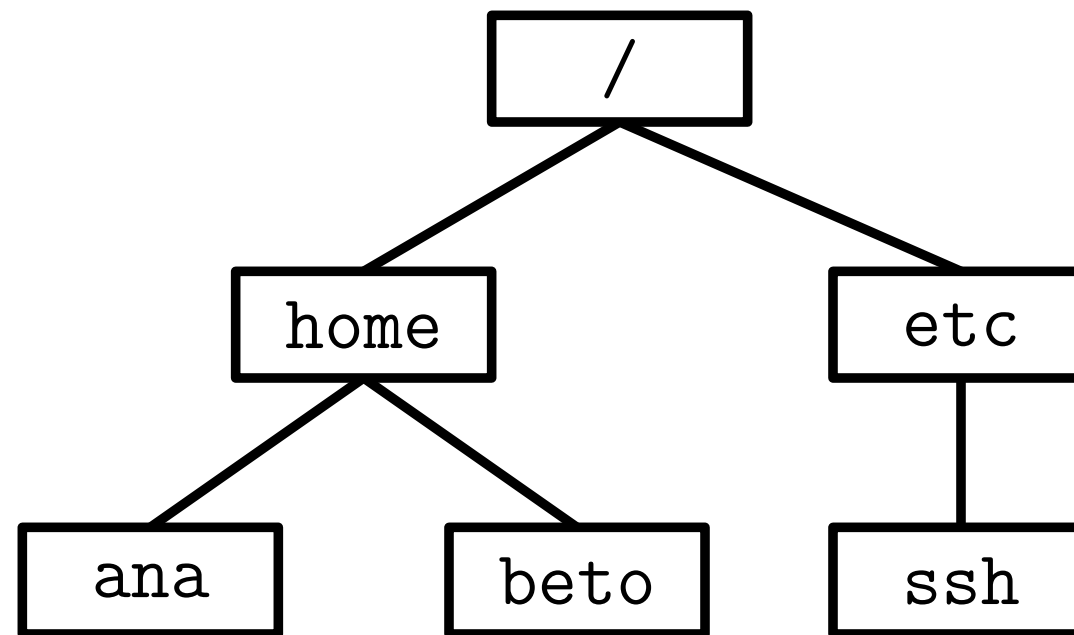
- Los árboles también sirven para modelar jerarquías.
- Ej:
  - Estructuras de directorios.



- ¿Otros ejemplos?

# Árboles enraizados

- Los árboles también sirven para modelar jerarquías.
- **Ej:**
  - Estructuras de directorios.



- ¿Otros ejemplos?
- ¿Diferencia con árboles?

# Árboles enraizados

Def.

Sea  $T$  un árbol.

# Árboles enraizados

Def.

Sea  $T$  un árbol.

Un *árbol enraizado* es un árbol  $T$  junto con un vértice distinguido  $r$ , llamado la *raíz*.

# Árboles enraizados

Def.

Sea  $T$  un árbol.

Un *árbol enraizado* es un árbol  $T$  junto con un vértice distinguido  $r$ , llamado la *raíz*.

- Ej:

# Árboles enraizados

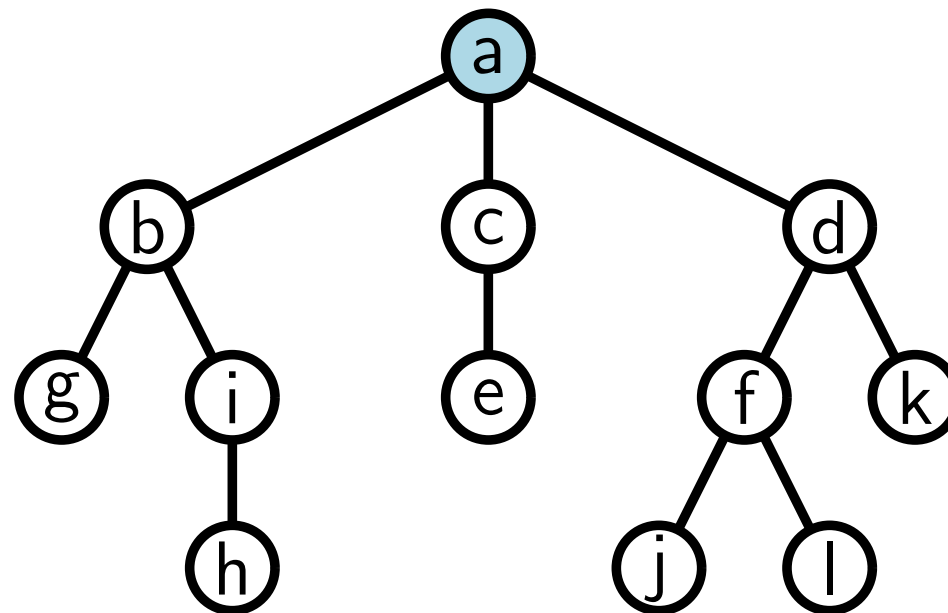
## Def.

Sea  $T$  un árbol.

Un *árbol enraizado* es un árbol  $T$  junto con un vértice distinguido  $r$ , llamado la *raíz*.

- Ej:

- Árbol enraizado en  $a$ .



# Árboles enraizados

- Ej:

# Árboles enraizados

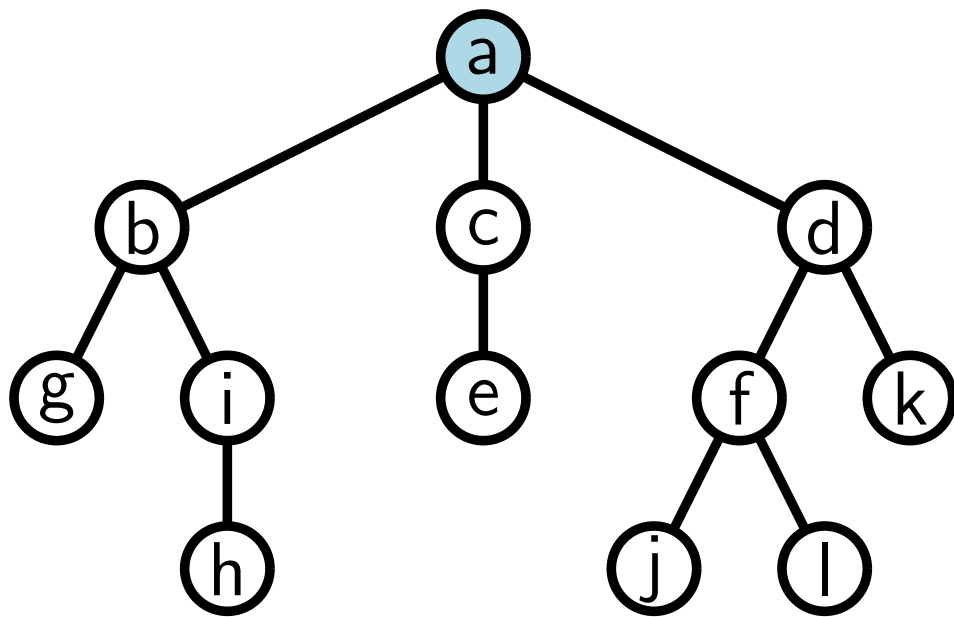
- Ej:

Def.

Sea  $T$  un árbol con raíz  $r$ .

# Árboles enraizados

- Ej:



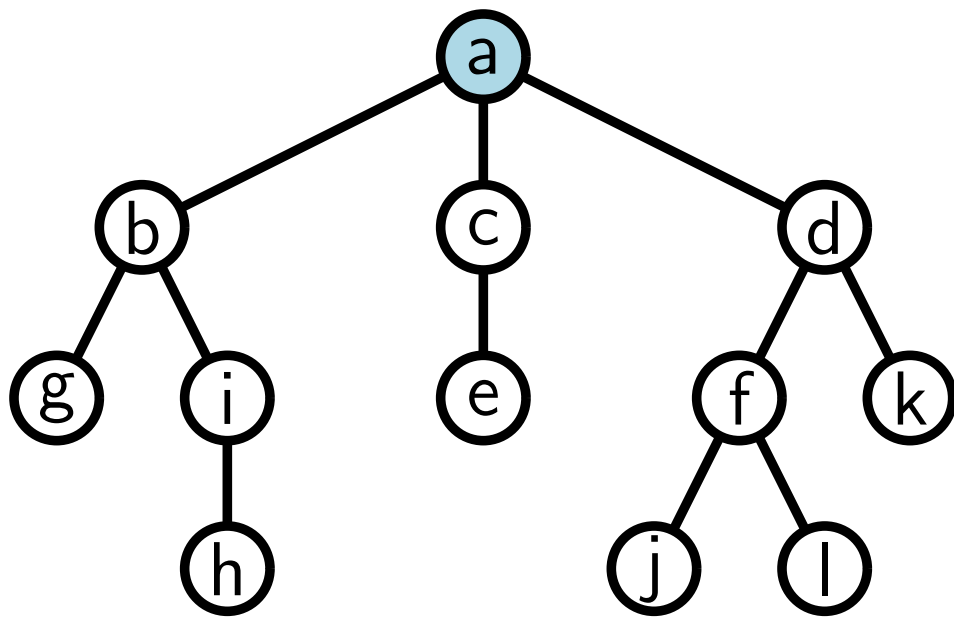
## Def.

Sea  $T$  un árbol con raíz  $r$ .

- Si  $v \neq r$ , el *padre* de  $v$  es el vértice anterior a  $v$  en el único camino simple desde  $r$  hasta  $v$ .

# Árboles enraizados

- Ej:



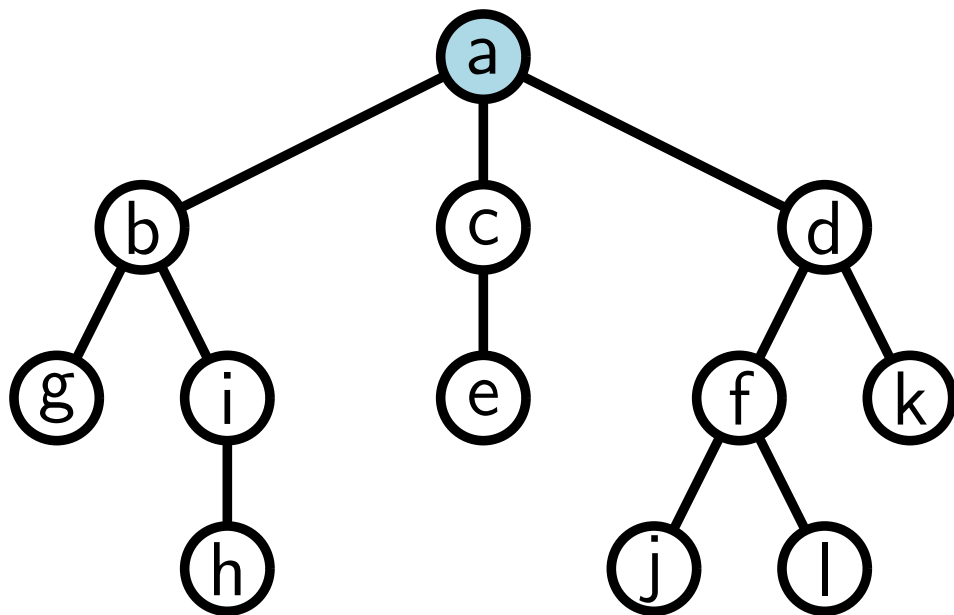
## Def.

Sea  $T$  un árbol con raíz  $r$ .

- Si  $v \neq r$ , el **padre** de  $v$  es el vértice anterior a  $v$  en el único camino simple desde  $r$  hasta  $v$ .
- Si  $u$  es el padre de  $v$ , decimos que  $v$  es un **hijo** de  $u$ .

# Árboles enraizados

- Ej:



## Def.

Sea  $T$  un árbol con raíz  $r$ .

- Si  $v \neq r$ , el *padre* de  $v$  es el vértice anterior a  $v$  en el único camino simple desde  $r$  hasta  $v$ .
- Si  $u$  es el padre de  $v$ , decimos que  $v$  es un *hijo* de  $u$ .
- Dos vértices distintos son *hermanos* si tienen el mismo padre.

# Árboles enraizados

- Ej:

# Árboles enraizados

- Ej:

Def.

Sea  $T$  un árbol con raíz  $r$ .

# Árboles enraizados

- Ej:
  - Árbol enraizado en  $a$ .

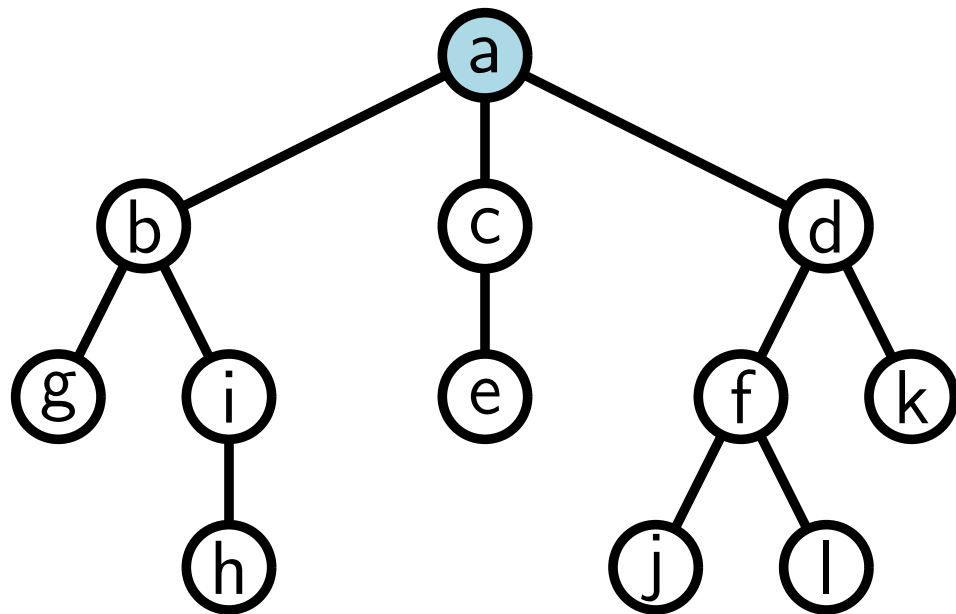
Def.

Sea  $T$  un árbol con raíz  $r$ .

# Árboles enraizados

- Ej:

- Árbol enraizado en  $a$ .



## Def.

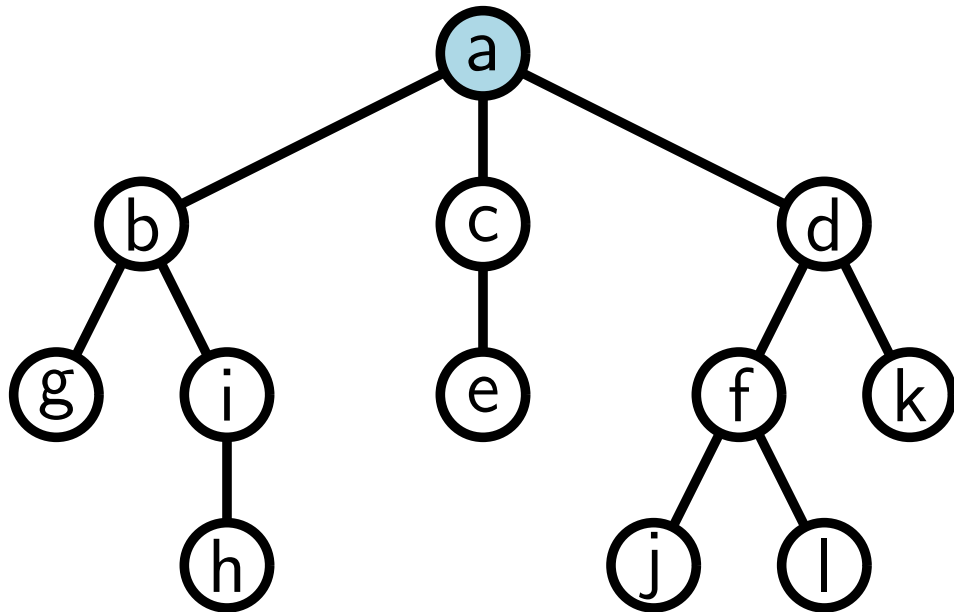
Sea  $T$  un árbol con raíz  $r$ .

- Los *ancestros* de  $v$  son los vértices que aparecen en el único camino simple desde  $r$  hasta  $v$ .

# Árboles enraizados

- Ej:

- Árbol enraizado en  $a$ .



## Def.

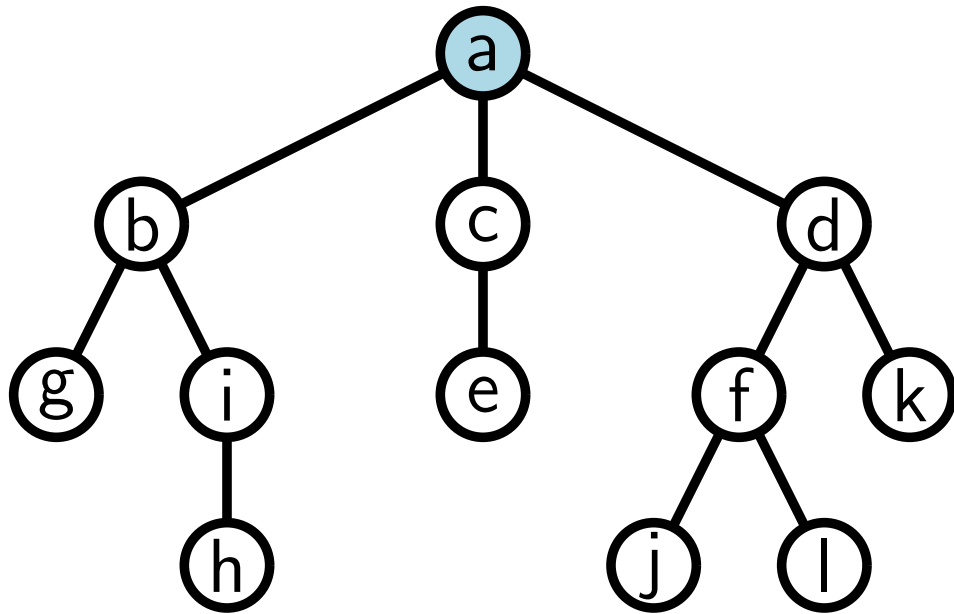
Sea  $T$  un árbol con raíz  $r$ .

- Los *ancestros* de  $v$  son los vértices que aparecen en el único camino simple desde  $r$  hasta  $v$ .
- Los *descendientes* de  $v$  son los vértices que tienen a  $v$  como ancestro.

# Árboles enraizados

- Ej:

- Árbol enraizado en  $a$ .



## Def.

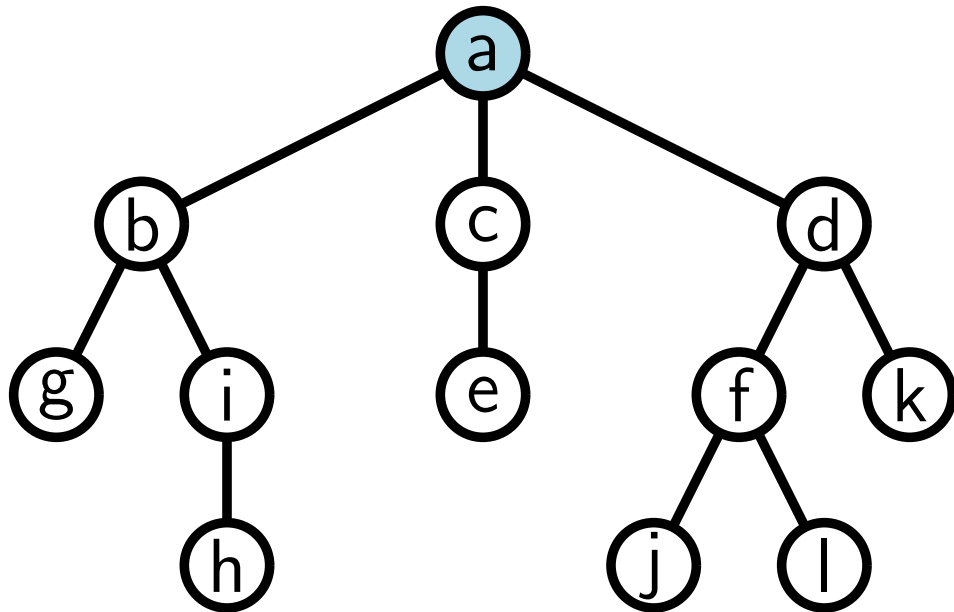
Sea  $T$  un árbol con raíz  $r$ .

- Los *ancestros* de  $v$  son los vértices que aparecen en el único camino simple desde  $r$  hasta  $v$ .
- Los *descendientes* de  $v$  son los vértices que tienen a  $v$  como ancestro.
- Una *hoja* es un vértice sin hijos.

# Árboles enraizados

- Ej:

- Árbol enraizado en  $a$ .



## Def.

Sea  $T$  un árbol con raíz  $r$ .

- Los *ancestros* de  $v$  son los vértices que aparecen en el único camino simple desde  $r$  hasta  $v$ .
- Los *descendientes* de  $v$  son los vértices que tienen a  $v$  como ancestro.
- Una *hoja* es un vértice sin hijos.
- Un *vértice interno* es un vértice con al menos un hijo.

## Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

## Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

Def.

Sea  $T$  un árbol enraizado.

## Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

Def.

Sea  $T$  un árbol enraizado.

La *altura* de  $T$  es el largo máximo de un camino simple desde la raíz hasta una hoja.

## Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

Def.

Sea  $T$  un árbol enraizado.

La *altura* de  $T$  es el largo máximo de un camino simple desde la raíz hasta una hoja.

La *aridad* de  $T$  es el máximo número de hijos que tiene un vértice de  $T$ .

# Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

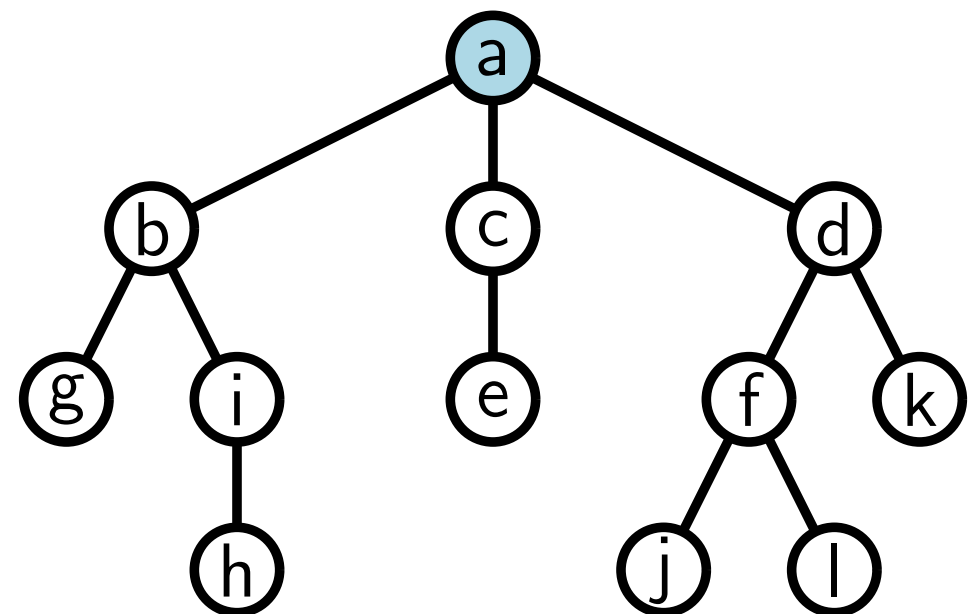
## Def.

Sea  $T$  un árbol enraizado.

La *altura* de  $T$  es el largo máximo de un camino simple desde la raíz hasta una hoja.

La *aridad* de  $T$  es el máximo número de hijos que tiene un vértice de  $T$ .

- Ej:



# Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

## Def.

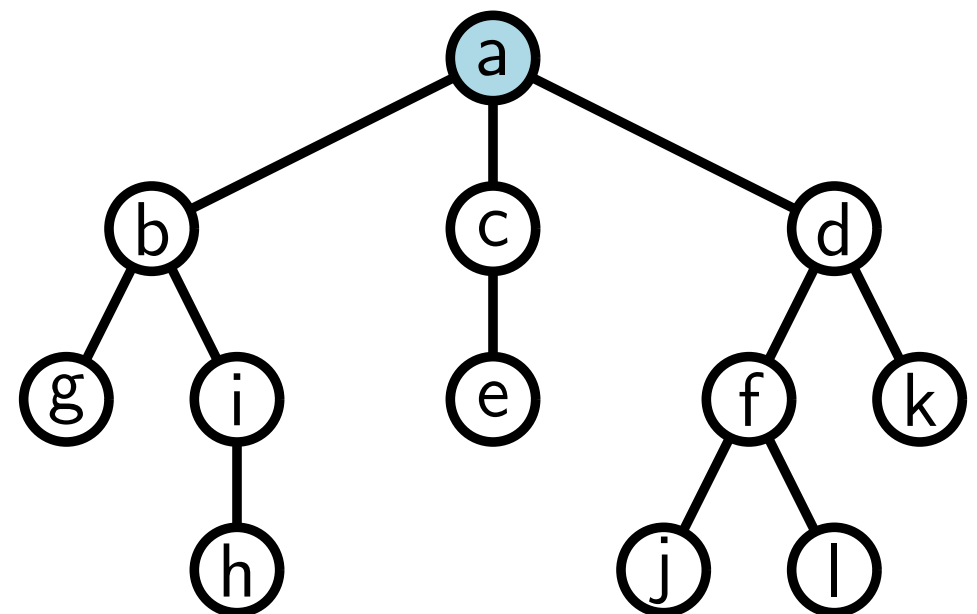
Sea  $T$  un árbol enraizado.

La *altura* de  $T$  es el largo máximo de un camino simple desde la raíz hasta una hoja.

La *aridad* de  $T$  es el máximo número de hijos que tiene un vértice de  $T$ .

## • Ej:

- ¿Altura? ¿Aridad?



# Altura y aridad

- ¿Qué tan profunda y ramificada es la jerarquía?

## Def.

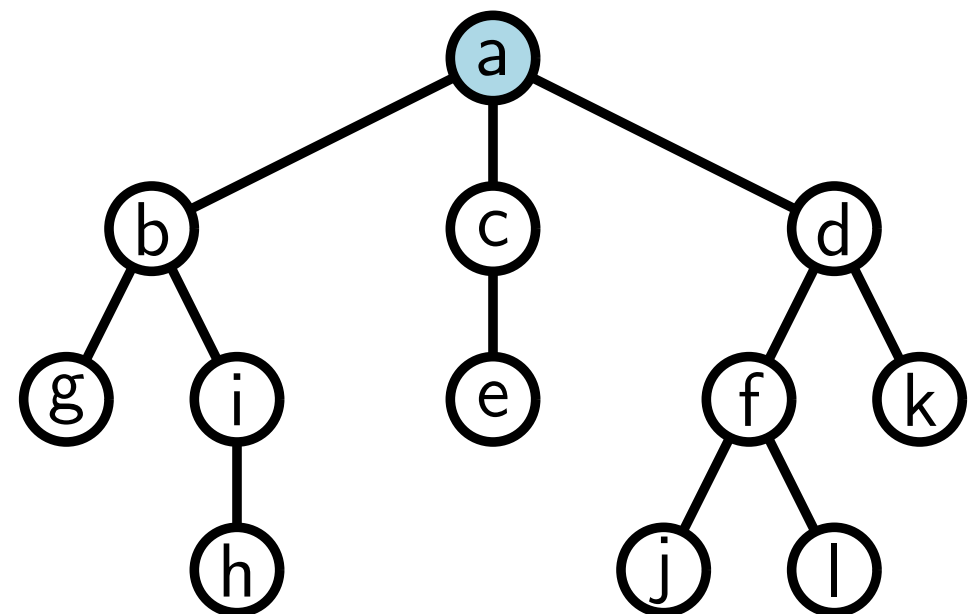
Sea  $T$  un árbol enraizado.

La *altura* de  $T$  es el largo máximo de un camino simple desde la raíz hasta una hoja.

La *aridad* de  $T$  es el máximo número de hijos que tiene un vértice de  $T$ .

## • Ej:

- ¿Altura? ¿Aridad?
- Cambiar raíz  $\rightarrow$  cambiar jerarquía.



# Árboles $k$ -arios

- Árboles con aridad restringida

# Árboles $k$ -arios

- Árboles con aridad restringida
  - ¿Qué restricción impone esto sobre sus hijos?

# Árboles $k$ -arios

- Árboles con aridad restringida
  - ¿Qué restricción impone esto sobre sus hijos?

Def.

Sea  $k \geq 1$  un entero.

# Árboles $k$ -arios

- Árboles con aridad restringida
  - ¿Qué restricción impone esto sobre sus hijos?

Def.

Sea  $k \geq 1$  un entero.

- Un árbol enraizado es  *$k$ -ario* si cada vértice interno tiene a lo más  $k$  hijos.

# Árboles $k$ -arios

- Árboles con aridad restringida
  - ¿Qué restricción impone esto sobre sus hijos?

## Def.

Sea  $k \geq 1$  un entero.

- Un árbol enraizado es  *$k$ -ario* si cada vértice interno tiene a lo más  $k$  hijos.
- Un árbol enraizado es  *$k$ -ario completo* si cada vértice interno tiene exactamente  $k$  hijos.

# Árboles $k$ -arios

- Árboles con aridad restringida
  - ¿Qué restricción impone esto sobre sus hijos?

## Def.

Sea  $k \geq 1$  un entero.

- Un árbol enraizado es  *$k$ -ario* si cada vértice interno tiene a lo más  $k$  hijos.
- Un árbol enraizado es  *$k$ -ario completo* si cada vértice interno tiene exactamente  $k$  hijos.

- Ej:

# Árboles $k$ -arios

- Árboles con aridad restringida
  - ¿Qué restricción impone esto sobre sus hijos?

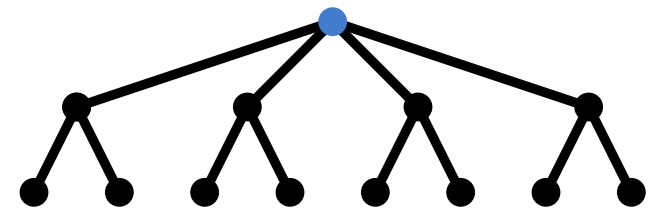
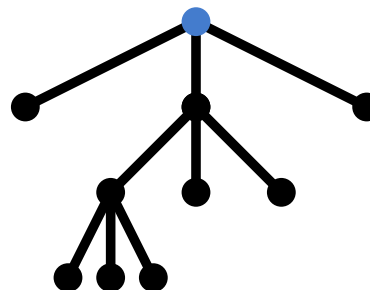
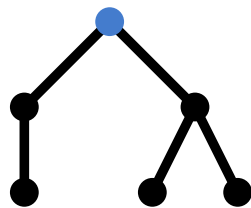
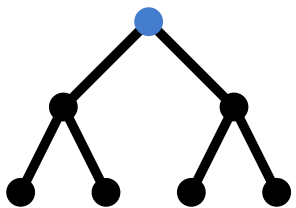
## Def.

Sea  $k \geq 1$  un entero.

- Un árbol enraizado es  *$k$ -ario* si cada vértice interno tiene a lo más  $k$  hijos.
- Un árbol enraizado es  *$k$ -ario completo* si cada vértice interno tiene exactamente  $k$  hijos.

## • Ej:

- ¿Aridad? ¿Completo?



# Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.

# Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**
- Si el árbol es  $k$ -ario completo, podemos calcular de manera más exacta.

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**
- Si el árbol es  $k$ -ario completo, podemos calcular de manera más exacta.

Prop.

Sea  $T$  un árbol  $k$ -ario completo con  $i$  vértices internos.

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**
- Si el árbol es  $k$ -ario completo, podemos calcular de manera más exacta.

Prop.

Sea  $T$  un árbol  $k$ -ario completo con  $i$  vértices internos.

Entonces

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**
- Si el árbol es  $k$ -ario completo, podemos calcular de manera más exacta.

Prop.

Sea  $T$  un árbol  $k$ -ario completo con  $i$  vértices internos.

Entonces

- El número total de vértices de  $T$  es  $ki + 1$ .

## Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**
- Si el árbol es  $k$ -ario completo, podemos calcular de manera más exacta.

Prop.

Sea  $T$  un árbol  $k$ -ario completo con  $i$  vértices internos.

Entonces

- El número total de vértices de  $T$  es  $ki + 1$ .
- El número de hojas de  $T$  es  $(k - 1)i + 1$ .

# Árboles $k$ -arios

- Dada altura y aridad, podemos acotar las hojas.
- Útil: en análisis de algoritmos.

Prop.

Sea  $T$  un árbol  $k$ -ario de altura  $h$ .

Entonces  $T$  tiene a lo más  $k^h$  hojas.

- **Dem:**
- Si el árbol es  $k$ -ario completo, podemos calcular de manera más exacta.

Prop.

Sea  $T$  un árbol  $k$ -ario completo con  $i$  vértices internos.

Entonces

- El número total de vértices de  $T$  es  $ki + 1$ .
- El número de hojas de  $T$  es  $(k - 1)i + 1$ .

- **Dem:**

# Conexidad en digrafos

# Conectividad en digrafos

- ¿Conexidad en digrafos?

## Conectividad en digrafos

- ¿Conexidad en digrafos?

Def.

Sea  $G = (V, E)$  un digrafo.

# Conectividad en digrafos

- ¿Conexidad en digrafos?

Def.

Sea  $G = (V, E)$  un digrafo.

Un  *$uv$ -dicamino* (o simplemente *dicamino*) de largo  $k$ , con  $k \geq 0$ , es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $(v_{i-1}, v_i) \in E$  para todo  $1 \leq i \leq k$ .

# Conectividad en digrafos

- ¿Conexidad en digrafos?

Def.

Sea  $G = (V, E)$  un digrafo.

Un *uv-dicamino* (o simplemente *dicamino*) de largo  $k$ , con  $k \geq 0$ , es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $(v_{i-1}, v_i) \in E$  para todo  $1 \leq i \leq k$ .

Diremos además que:

# Conectividad en digrafos

- ¿Conexidad en digrafos?

## Def.

Sea  $G = (V, E)$  un digrafo.

Un *uv-dicamino* (o simplemente *dicamino*) de largo  $k$ , con  $k \geq 0$ , es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $(v_{i-1}, v_i) \in E$  para todo  $1 \leq i \leq k$ .

Diremos además que:

- El dicamino es *simple* si todos sus vértices son distintos.

# Conectividad en digrafos

- ¿Conexidad en digrafos?

## Def.

Sea  $G = (V, E)$  un digrafo.

Un *uv-dicamino* (o simplemente *dicamino*) de largo  $k$ , con  $k \geq 0$ , es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $(v_{i-1}, v_i) \in E$  para todo  $1 \leq i \leq k$ .

Diremos además que:

- El dicamino es *simple* si todos sus vértices son distintos.
- Es *cerrado* si  $v_0 = v_k$ .

# Conectividad en digrafos

- ¿Conexidad en digrafos?

## Def.

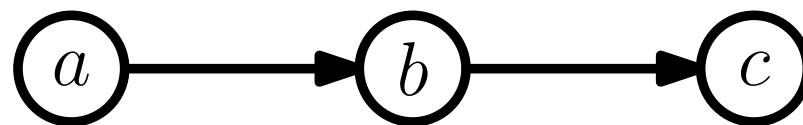
Sea  $G = (V, E)$  un digrafo.

Un *uv-dicamino* (o simplemente *dicamino*) de largo  $k$ , con  $k \geq 0$ , es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $(v_{i-1}, v_i) \in E$  para todo  $1 \leq i \leq k$ .

Diremos además que:

- El dicamino es *simple* si todos sus vértices son distintos.
- Es *cerrado* si  $v_0 = v_k$ .

- Ej:



# Conectividad en digrafos

- ¿Conexidad en digrafos?

## Def.

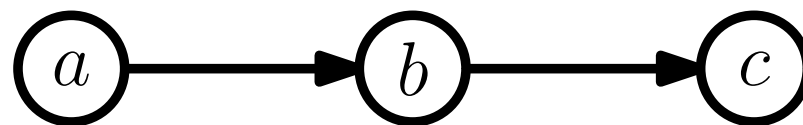
Sea  $G = (V, E)$  un digrafo.

Un *uv-dicamino* (o simplemente *dicamino*) de largo  $k$ , con  $k \geq 0$ , es una secuencia de vértices  $v_0, v_1, \dots, v_k$  tal que (1)  $v_0 = u$ , (2)  $v_k = v$ , y (3)  $(v_{i-1}, v_i) \in E$  para todo  $1 \leq i \leq k$ .

Diremos además que:

- El dicamino es *simple* si todos sus vértices son distintos.
- Es *cerrado* si  $v_0 = v_k$ .

- Ej:



- Como antes, nos podemos restringir a caminos simples.

## Conectividad fuerte

- Misma noción en base a dicaminos.

## Conectividad fuerte

- Misma noción en base a dicaminos.

Def.

Sea  $G = (V, E)$  un digrafo, con  $n = |V|$ .

## Conectividad fuerte

- Misma noción en base a dicaminos.

Def.

Sea  $G = (V, E)$  un digrafo, con  $n = |V|$ .

Un digrafo  $G = (V, E)$  es *fuertemente conexo* si

## Conectividad fuerte

- Misma noción en base a dicaminos.

Def.

Sea  $G = (V, E)$  un digrafo, con  $n = |V|$ .

Un digrafo  $G = (V, E)$  es *fuertemente conexo* si

◦  $n \geq 1$  y

## Conectividad fuerte

- Misma noción en base a dicaminos.

Def.

Sea  $G = (V, E)$  un digrafo, con  $n = |V|$ .

Un digrafo  $G = (V, E)$  es *fuertemente conexo* si

- $n \geq 1$  y
- para todo par de vértices distintos  $u, v \in V$  existe un dicamino desde  $u$  hasta  $v$  y existe un dicamino desde  $v$  hasta  $u$ .

# Conectividad fuerte

- Misma noción en base a dicaminos.

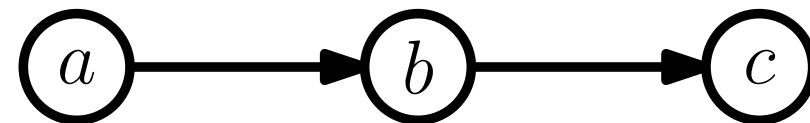
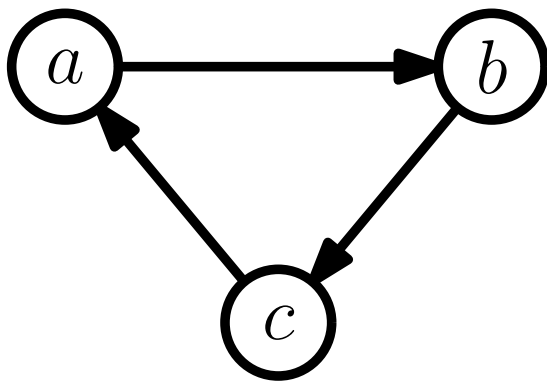
## Def.

Sea  $G = (V, E)$  un digrafo, con  $n = |V|$ .

Un digrafo  $G = (V, E)$  es *fuertemente conexo* si

- $n \geq 1$  y
- para todo par de vértices distintos  $u, v \in V$  existe un dicamino desde  $u$  hasta  $v$  y existe un dicamino desde  $v$  hasta  $u$ .

- Ej:



## Conectividad débil

- **Alternativa:** olvidar las direcciones.

## Conectividad débil

- **Alternativa:** olvidar las direcciones.

Def.

Sea  $G = (V, E)$  un digrafo.

## Conectividad débil

- **Alternativa:** olvidar las direcciones.

Def.

Sea  $G = (V, E)$  un digrafo.

El *grafo no dirigido subyacente* de  $G$  es el grafo no dirigido  $G^{\text{und}}$  con el mismo conjunto de vértices, donde  $u$  y  $v$  son adyacentes si y sólo si  $(u, v) \in E$  o  $(v, u) \in E$ .

## Conectividad débil

- **Alternativa:** olvidar las direcciones.

Def.

Sea  $G = (V, E)$  un digrafo.

El *grafo no dirigido subyacente* de  $G$  es el grafo no dirigido  $G^{\text{und}}$  con el mismo conjunto de vértices, donde  $u$  y  $v$  son adyacentes si y sólo si  $(u, v) \in E$  o  $(v, u) \in E$ .

Def.

Sea  $G$  un digrafo.

## Conectividad débil

- **Alternativa:** olvidar las direcciones.

Def.

Sea  $G = (V, E)$  un digrafo.

El *grafo no dirigido subyacente* de  $G$  es el grafo no dirigido  $G^{\text{und}}$  con el mismo conjunto de vértices, donde  $u$  y  $v$  son adyacentes si y sólo si  $(u, v) \in E$  o  $(v, u) \in E$ .

Def.

Sea  $G$  un digrafo.

Un digrafo  $G$  es *débilmente conexo* si su grafo no dirigido subyacente  $G^{\text{und}}$  es conexo.

# Conectividad débil

- **Alternativa:** olvidar las direcciones.

## Def.

Sea  $G = (V, E)$  un digrafo.

El *grafo no dirigido subyacente* de  $G$  es el grafo no dirigido  $G^{\text{und}}$  con el mismo conjunto de vértices, donde  $u$  y  $v$  son adyacentes si y sólo si  $(u, v) \in E$  o  $(v, u) \in E$ .

## Def.

Sea  $G$  un digrafo.

Un digrafo  $G$  es *débilmente conexo* si su grafo no dirigido subyacente  $G^{\text{und}}$  es conexo.

## Prop.

Sea  $G$  un digrafo.

# Conectividad débil

- **Alternativa:** olvidar las direcciones.

## Def.

Sea  $G = (V, E)$  un digrafo.

El *grafo no dirigido subyacente* de  $G$  es el grafo no dirigido  $G^{\text{und}}$  con el mismo conjunto de vértices, donde  $u$  y  $v$  son adyacentes si y sólo si  $(u, v) \in E$  o  $(v, u) \in E$ .

## Def.

Sea  $G$  un digrafo.

Un digrafo  $G$  es *débilmente conexo* si su grafo no dirigido subyacente  $G^{\text{und}}$  es conexo.

## Prop.

Sea  $G$  un digrafo.

Todo digrafo fuertemente conexo es débilmente conexo.

# Conectividad en digrafos

## Def.

Sea  $G$  un digrafo.

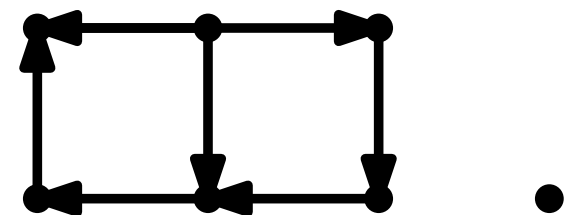
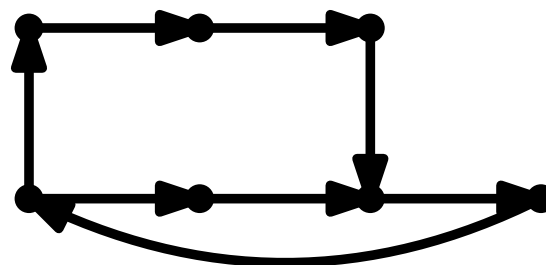
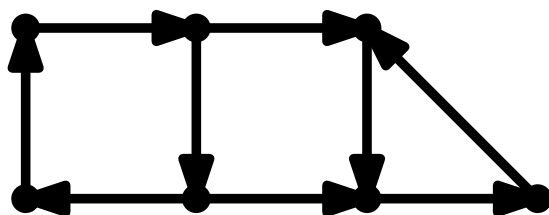
Un digrafo  $G$  es *débilmente conexo* si su grafo no dirigido subyacente  $G^{\text{und}}$  es conexo.

## Def.

Sea  $G = (V, E)$  un digrafo, con  $n = |V|$ .

Un digrafo  $G = (V, E)$  es *fuertemente conexo* si

- $n \geq 1$  y
- para todo par de vértices distintos  $u, v \in V$  existe un dicamino desde  $u$  hasta  $v$  y existe un dicamino desde  $v$  hasta  $u$ .



# Componentes fuertemente conexas

Def.

Sea  $G$  un digrafo.

# Componentes fuertemente conexas

Def.

Sea  $G$  un digrafo.

Una *componente fuertemente conexa* de  $G$  es un subgrafo  $H$  de  $G$ .

# Componentes fuertemente conexas

Def.

Sea  $G$  un digrafo.

Una *componente fuertemente conexa* de  $G$  es un subgrafo  $H$  de  $G$ .

- $H$  es fuertemente conexo;

# Componentes fuertemente conexas

Def.

Sea  $G$  un digrafo.

Una *componente fuertemente conexa* de  $G$  es un subgrafo  $H$  de  $G$ .

- $H$  es fuertemente conexo;
- $H$  no es subgrafo propio de otro subgrafo fuertemente conexo de  $G$ .

# Componentes fuertemente conexas

Def.

Sea  $G$  un digrafo.

Una *componente fuertemente conexa* de  $G$  es un subgrafo  $H$  de  $G$ .

- $H$  es fuertemente conexo;
- $H$  no es subgrafo propio de otro subgrafo fuertemente conexo de  $G$ .

Es decir, una componente fuertemente conexa es un subgrafo fuertemente conexo maximal.

# Componentes fuertemente conexas

## Def.

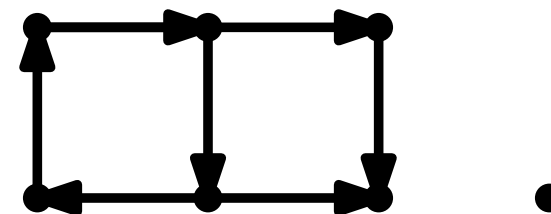
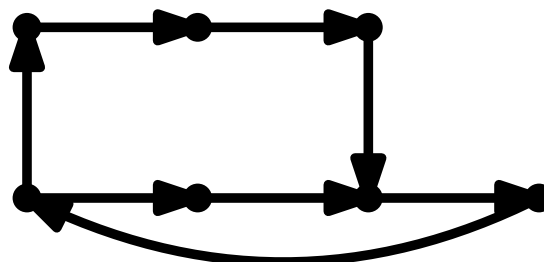
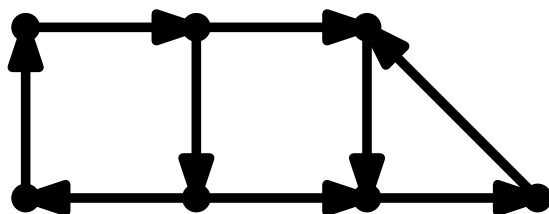
Sea  $G$  un digrafo.

Una *componente fuertemente conexa* de  $G$  es un subgrafo  $H$  de  $G$ .

- $H$  es fuertemente conexo;
- $H$  no es subgrafo propio de otro subgrafo fuertemente conexo de  $G$ .

Es decir, una componente fuertemente conexa es un subgrafo fuertemente conexo maximal.

- Las componentes particionan **sólo** los vértices.



**Dudas/Ejercicios**

## ■ Wrap-up

- **Hoy:**

# Wrap-up

- **Hoy:**
  - Conectividad y componentes.

# Wrap-up

- **Hoy:**
  - Conectividad y componentes.
  - Árboles y árboles enraizados.

# Wrap-up

- **Hoy:**

- Conectividad y componentes.
- Árboles y árboles enraizados.
- Conectividad en digrafos.

# Wrap-up

- **Hoy:**

- Conectividad y componentes.
- Árboles y árboles enraizados.
- Conectividad en digrafos.

- **Próx. clase:**

# Wrap-up

- **Hoy:**

- Conectividad y componentes.
- Árboles y árboles enraizados.
- Conectividad en digrafos.

- **Próx. clase:**

- Árboles generadores.