Árboles y distancia

Martín D. Safe

Instituto de Cálculo, Universidad de Buenos Aires

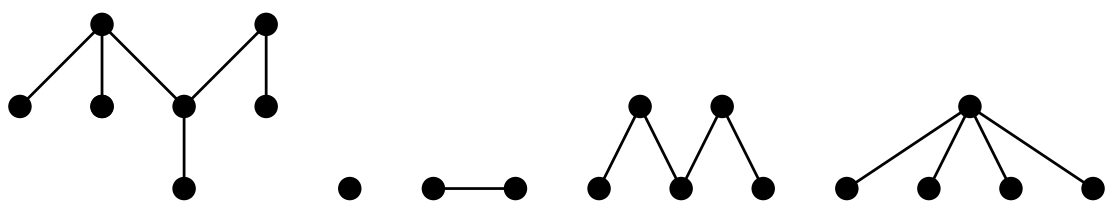
Tópicos Fundamentales en Teoría de Grafos
2.º semestre 2018

Árboles

La palabra "árbol" se usa en teoría de grafos para referirse a grafos que se ramifican y nunca forma ciclos. Estos árboles tienen muchas aplicaciones, especialmente en almacenamiento y recuperación de datos, búsqueda y comunicaciones.

Grafo acíclico. Bosque. Árbol. Hoja.

Un grafo acíclico es un grafo sin ciclos. (En particular sin bucles ni aristas múltiples.)
A los grafos acíclicos también se los llama bosques.
Un árbol es un grafo acíclico conexo.
Un vértice de grado 1 de un bosque (o árbol) se llama hoja



Árboles

Observaciones

1. Un árbol es un bosque conexo.
2. Cada componente conexa de un bosque es un árbol.
3. Como los árboles no tienen ciclos, en particular no tienen ciclos impares. Por lo tanto los árboles son grafos bipartitos.

Ejemplos

- Una estrella es un árbol que consiste en un vértice adyacente a todos los demás. La estrella de n vértices es $K_{1,n-1}$.
- Los caminos P_n también son árboles. Más precisamente, los caminos son exactamente los árboles con grado máximo a lo sumo 2.

Propiedades de los árboles

Lema

1. Todo árbol con al menos dos vértices tiene al menos dos hojas.
2. Si borramos una hoja de un árbol obtenemos otro árbol.

Demostración.

1. En un grafo acíclico, cada extremo de un camino maximal no trivial (i.e., con al menos una arista) tiene como único vecino a su vecino en el camino. Todo grafo conexo con al menos dos vértices tiene al menos una arista y, luego, algún camino maximal no trivial; los extremos de dicho camino son hojas.
2. Sea v una hoja de un árbol G y sea $G' = G - v$. Un vértice de grado 1 no pertenece a ningún camino que une a otros dos vértices. Por lo tanto, para todos $u, w \in V(G')$, todo camino con extremos u y w en G es también un camino en G'. Así que G' es conexo. Como borrando un vértice no se puede crear un ciclo, G' es también acíclico y por lo tanto un árbol. $\square$

Caracterizaciones de los árboles

Los árboles admiten muchas caracterizaciones equivalentes. Este tipo de caracterizaciones son útiles porque solo hace falta verificar que un grafo satisface una de ellas para probar que es un árbol y, por lo tanto, satisface todas las otras.

Teorema

Si G es un grafo con n vértices, las siguientes afirmaciones son equivalentes:

1. G es un árbol (es decir, es conexo y acíclico).
2. G es conexo y tiene $n-1$ aristas.
3. G tiene $n-1$ aristas y es acíclico.
4. Para cada $u, v \in V(G)$, existe un único camino en G con extremos u y v.

Demostración.

Primero probaremos la equivalencia entre 1, 2 y 3 mostrando que cualesquiera dos de las propiedades 'acíclico', 'conexo', 'tiene $n-1$ aristas' implica la tercera. (...)

Caracterizaciones de los árboles

Demostración (cont.)

1 $\Rightarrow$ 2 y 3: *Si* G *es conexo y acíclico entonces tiene* $n-1$ *aristas.*
Por inducción en n. Si $n = 1$, un grafo acíclico con un vértice no tiene aristas. Sea $n > 1$ y supongamos que la afirmación se cumple para todo los grafos con menos de n vértices. Dado un grafo acíclico y conexo G, el lema anterior nos asegura que tiene una hoja v y que $G' = G - v$ es acíclico y conexo. Por hipótesis inductiva, $|E(G')| = |V(G')| - 1 = n - 2$. Por lo tanto, $|E(G)| = |E(G')| + 1 = n - 1$.

2 $\Rightarrow$ 1 y 3: *Si* G *es conexo y tiene* $n-1$ *aristas entonces es acíclico.*
Si G tiene algún ciclo borramos una arista de ese ciclo. Repetimos esta operación hasta obtener un grafo acíclico G'. Como las aristas de ciclos no son aristas de corte, el grafo G' es también conexo. Por el inciso anterior, G' tiene $n-1$ aristas. Como G también tiene $n-1$ aristas entonces $G = G'$. En particular, G es acíclico. (...)

Caracterizaciones de los árboles

Demostración (cont.)

3 $\Rightarrow$ 1 y 2: *Si* G *tiene* $n-1$ *aristas y es acíclico entonces es conexo.*
Sean $G_1, \ldots, G_k$ las componentes de G. Como son conexas y acíclicas, $|E(G_i)| = |V(G_i)| - 1$ para cada $i \in \{1, \ldots, k\}$. Luego, $|E(G)| = \sum_{i=1}^{k} |E(G_i)| = \sum_{i=1}^{k} (|V(G_i)| - 1) = |V(G)| - k = n - k$. Como G tiene $n-1$ aristas, $k = 1$, es decir, G es conexo.

1 $\Rightarrow$ 4: *Si* G *es un árbol entonces para cada* $u, v \in V(G)$ *existe un único camino con extremos* u *y* v *en* G.
Como G es conexo, cada par de vértices está unido por un camino. Supongamos, por el absurdo, que hay al menos un par de vértices unidos por dos caminos distintos. Sean P y Q dos caminos distintos con los mismos extremos y que minimizan la suma de sus longitudes (es decir, $|E(P)| + |E(Q)|$). Por la minimalidad, P y Q comparten sólo sus extremos y por lo tanto forman un ciclo, contradicción. Esta contradicción provino de negar que cada par de vértices está unido por un único camino. (...)

Caracterizaciones de los árboles

Demostración (cont.)

4 $\Rightarrow$ 1: *Si cada par de vértices de* G *está unidos por un único camino entonces* G *es un árbol.*
Como todos los pares de vértices están unidos por caminos, G es conexo. Además, G es acíclico porque de lo contrario habría dos caminos distintos con los mismos extremos (incluso en el caso en que el ciclo fuera un bucle o dos aristas paralelas). □

Corolario

Cada arista de un árbol es una arista de corte.

Corolario

Agregando una arista a un árbol se crea exactamente un ciclo.

Árboles generadores

Subgrafo generador. Árbol generador.

Un subgrafo generador de G es un subgrafo de G con el mismo conjunto de vértices que G.
Un árbol generador es un subgrafo generador que es un árbol.

Lema

Todo grafo conexo posee un árbol generador.

Demostración.

Es similar a la prueba de (2) $\Rightarrow$ (1) y (3). Sea G un grafo conexo. Si tiene algún ciclo, removemos una arista de algún ciclo. Repetimos este procedimiento hasta obtener un grafo acíclico. Ese grafo acíclico es además conexo porque nunca se removieron aristas de corte. Como además tiene el mismo conjunto de vértices que G, es un árbol generador de G. $\square$

Subgrafos (no etiquetados)

Habíamos definido que H es un subgrafo de G si $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$ y cada arista $e \in E(H)$ tiene los mismos extremos en H que en G.

Vamos a relajar esta definición de la siguiente manera, que es consistente con la idea de no distinguir grafos isomorfos.

Subgrafo. Subgrafo inducido.

Un grafo H es un subgrafo de G si H es isomorfo a un grafo H' tal que $V(H') \subseteq V(G)$, $E(H') \subseteq V(G)$ y cada arista $e \in E(H')$ tiene los mismos extremos en H' que en G; en tal caso, H' se dice una copia de H contenida en G.
Si además

$$E(H') = \{uv \in E(G) \colon u, v \in V(H')\},$$

entonces H es un subgrafo inducido de G.

Árboles como subgrafos

Proposición

Si T es un árbol con k aristas y G es un grafo simple tal que $\delta(G) \geqslant k$ entonces T es un subgrafo de G.

Demostración.

Por inducción en k.
Si $k = 0$, todo grafo simple contiene el árbol sin aristas (K_1).
Sea $k > 0$ y supongamos que la afirmación es cierta para árboles con menos de k aristas. Como $k > 0$, T tiene al menos una hoja v y sea u su único vecino en T. Sea $T' = T - v$. Como T' tiene $k-1$ aristas y $\delta(G) \geqslant k > k-1$, la hipótesis inductiva nos asegura que T' es un subgrafo de G. Sea x el vértice correspondiente a u en una copia de T' contenida en G. Como T' tiene solo $k-1$ vértices distintos de u y $d_G(x) \geqslant k$ entonces x tiene un vecino y en G que no es un vértice de la copia de T' en G. Agregando la arista xy a la copia de T' en G se obtiene una copia de T en G. □

Distancias en árboles y en grafos

Cuando se usan grafos para modelar redes de comunicación, se desea que los vértices estén cerca entre sí para evitar demoras en la comunicación.

Distancia. Diámetro. Excentricidad. Radio

Si un grafo G tiene un camino con extremos u y v entonces la distancia entre u y v en G, denotada $d_G(u, v)$, es la menor de las longitudes entre todos los caminos con extremos u y v. Si G no tiene caminos con extremos u y v, se define $d_G(u, v) = \infty$.
El diámetro de G es $\operatorname{diam} G = \max_{u,v \in V(G)} d_G(u, v)$.
La excentricidad de un vértice u es $\epsilon_G(u) = \max_{v \in V(G)} d_G(u, v)$.
El radio de un grafo G es $\operatorname{rad} G = \min_{u \in V(G)} \epsilon_G(u)$.

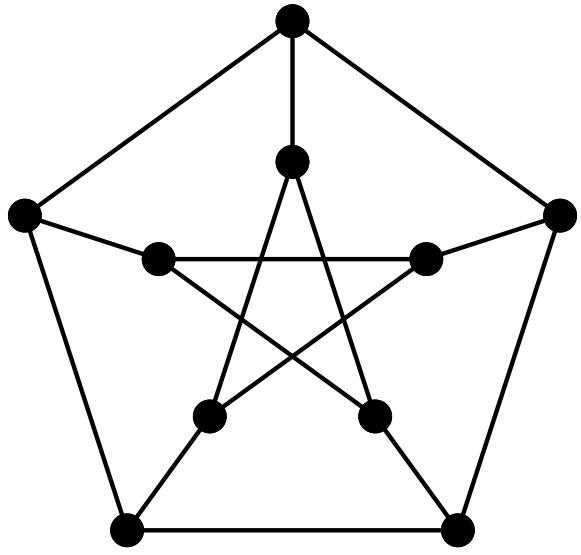
Observaciones

- El diámetro de un grafo es igual al máximo de las excentricidades de sus vértices.
- En un grafo disconexo, todas las excentricidades son ∞ y por lo tanto también el diámetro y el radio.

Diámetro, excentricidad y radio

Ejemplos

- El ciclo C_n tiene diámetro $\lfloor n/2 \rfloor$.



- El grafo de Petersen tiene diámetro 2. En efecto, recordemos que ya hemos probado que todo par de vértices no adyacentes tiene un vecino en común.
- El hipercubo Q_k tiene diámetro k.
- En estos estos tres ejemplos todos los vértices tienen la misma excentricidad y $\operatorname{diam} G = \operatorname{rad} G$.

Diámetro de un grafo y su complemento

Para tener diámetro alto muchas de las posibles aristas deben no estar presentes. Por lo tanto esperamos que el complemento de un grafo con diámetro alto tenga diámetro pequeño.

Teorema

Si G es un grafo simple y el $\operatorname{diam} G \geqslant 3$ entonces $\operatorname{diam} \overline{G} \leqslant 3$.

Demostración.

Como $\operatorname{diam} G > 2$, existen dos vértices no adyacentes $u, v \in V(G)$ que no tienen vecinos en común. Por lo tanto, cada vértice $x \in V(G) - \{u, v\}$ es no adyacente a u o no adyacente a v; y, en consecuencia, x es adyacente a u o a v en $\overline{G}$. Como además $uv \in E(\overline{G})$, entonces para cada de vértices x e y de G hay al menos un camino de longitud a lo sumo 3 entre x e y en $\overline{G}$ usando a u y v como posibles vértices internos. Concluimos que $\operatorname{diam} \overline{G} \leqslant 3$. □

Centro de un grafo

Centro

El centro de un grafo es el grafo inducido por sus vértices de mínima excentricidad.

El centro de un grafo es todo el grafo si y sólo si el radio y el diámetro son iguales. Esto sucede, como vimos, para los ciclos, el grafo de Petersen y los hipercubos.

El siguiente resultado describe los centros de los árboles. En el paso inductivo, borramos todas las hojas en lugar de solo una.

Centro de un árbol

Teorema (Jordan, 1869)

El centro de un árbol es un vértice o una arista.

Demostración.

Por inducción en el número n de vértices del árbol T. Si $n \leqslant 2$, el centro es todo el árbol. Supongamos que $n > 2$ y sea T' el grafo que se obtiene borrando todas las hojas de T. Por un lema anterior, T' es un árbol. Como no todo vértice de T es una hoja, T' tiene al menos un vértice.

Todo vértice de T que se encuentra a máxima distancia de algún vértice $v \in V(T)$ es una hoja. Como todas las hojas se eliminaron y una hoja no puede ser un vértice interior de un camino entre otros dos vértices, $\epsilon_{T'}(u) = \epsilon_T(u) - 1$ para todo vértice u de T que no es una hoja de T (Ejercicio). Además, la excentricidad de una hoja en T es mayor que la excentricidad de su único vecino en T. Luego, los vértices que minimizan ϵ_T son los mismos que minimizan $\epsilon_{T'}$. Es decir, los centros de T y T' coinciden. Por hipótesis inductiva, el centro de T' es un vértice o una arista. □

Índice de Wiener

En una red de comunicación un diámetro alto es aceptable si la mayoría de los pares se pueden comunicar a través de caminos cortos.
Esto lleva al estudio de las distancias medias en lugar de la máxima. Como el promedio es igual a la suma dividida por $\binom{n}{2}$, es equivalente a estudiar

$$D(G) = \sum_{u,v \in V(G)} d_G(u,v).$$

Índice de Wiener

La suma $D(G)$ es conocida como el índice de Wiener de G

Es uno de los primeros índices topológicos usados en química y fue introducido por Harry Wiener en 1947.

Mínimo del índice de Wiener en árboles

Teorema

Entre los árboles con n vértices, el índice de Wiener se minimiza en la estrella $K_{1,n-1}$ y solo en ella. Además $D(K_{1,n-1}) = (n-1)^2$.

Demostración.

Como un árbol tiene $n-1$ aristas, tiene $n-1$ pares de vértices a distancia 1 y todos los otros pares a distancia al menos 2. Las estrellas alcanzan este mínimo y por lo tanto minimizan $D(T)$. Para mostrar que son los únicos árboles que minimizan $D(T)$, debemos mostrar que no hay otros árboles en los que cada par de vértices no adyacentes están a distancia 2. Consideremos una hoja h y sea v su vecino. Si queremos que la distancia de h a todos los demás vértice sea 2 necesariamente deben ser vecinos a v, así que v debe ser el centro de una estrella. Por último verificamos que

$$D(K_{1,n-1}) = (n-1) + 2\binom{n-1}{2} = (n-1)^2. \qquad \square$$

Máximo del índice de Wiener en árboles

Teorema

Entre los árboles con n vértices, el índice de Wiener se maximiza en el camino P_n y solo en él. Además $D(P_n) = \binom{n+1}{3}$.

Demostración.

Por inducción en n. Si $n = 1$, es trivial. Sea $n > 1$ y sea h una hoja de un árbol T con n vértices. Tenemos que $D(T) = D(T - h) + \sum_{v \in V(T)} d_T(h, v)$. Por hipótesis inductiva, $D(T - h) \leqslant \binom{n}{3}$ con igualdad sólo si $T - h$ es un camino de longitud $n - 1$. Y $\sum_{v \in V(T)} d_T(h, v) \leqslant 1 + 2 + \cdots + (n - 1) = \binom{n}{2}$ con igualdad sólo si las distancias de h a todos los otros vértices son todas distintas. Luego, el máximo $\binom{n}{3} + \binom{n}{2} = \binom{n+1}{3}$ se alcanza si y sólo si $T - h$ es un camino y h es vecino a un extremo del mismo; es decir, $T = P_n$. $\square$

Índice de Wiener en general

Claramente entre los grafos con n vértices, $D(G)$ se minimiza en K_n. Veamos que entre los grafos conexos, $D(G)$ se maximiza en P_n.

Lema

Si H es un subgrafo generador de un grafo G entonces $d_G(u, v) \leqslant d_H(u, v)$.

Demostración.

Porque las distancia entre dos vértices en G es siempre menor o igual que la distancia entre los mismos en H. □

Corolario

Si G es un grafo conexo con n vértices entonces $D(G) \leqslant D(P_n)$.

Demostración.

Como G es conexo posee un árbol generador T. Por los resultados anteriores, $D(G) \leqslant D(T) \leqslant D(P_n)$. □

Árboles generadores y enumeración

¿Cuántos grafos simples hay que tienen $\{1, 2, \ldots, n\}$ como conjunto de vértices? Hay $2^{\binom{n}{2}}$ porque cada par de vértices distintos puede estar unido o no por una arista.

¿Cuántos de ellos son árboles? Esto equivale a contar el número de árboles generadores (etiquetados) del grafo completo con conjunto de vértices $\{1, 2, \ldots, n\}$. Vamos a resolver este problema de conteo.

Fórmula de Cayley

Notación

Si n es un natural, denotamos por $[n] = \{1, 2, \ldots, n\}$.

- Si $n = 1$ o $n = 2$, hay un único árbol con conjunto de vértices $[n]$.
- La única clase de isomorfismo de árboles con tres vértices es la de P_3. Luego, se pueden obtener 3 árboles distintos con conjunto de vértices $[3]$ de acuerdo a cómo se etiquete el vértice interior.
- Con 4 vértices, hay 4 formas de etiquetar los vértices de $K_{1,3}$ con $[4]$ y 12 formas de etiquetar los vértices de P_4 con $[4]$, lo que nos da en total 16 árboles con conjunto de vértices $[4]$.
- Hay 125 árboles distintos con conjunto de vértices $[5]$.

La fórmula de Cayley afirma que hay n^{n-2} árboles con conjunto de vértices $[n]$. Vamos a probar esta fórmula estableciendo una biyección entre los árboles y sus códigos de Prüfer.

Códigos de Prüfer

Dada un árbol con conjunto de vértices $S \subseteq \mathbb{N}$ tal que $|S| = n \geqslant 2$, vamos a asignarle una $(n-2)$-upla de valores en S.

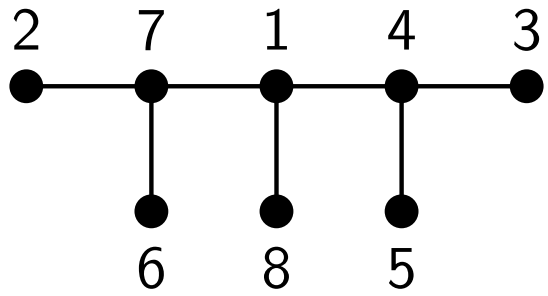
Algoritmo para determinar el código de Prüfer

Entrada: Un árbol T con conjunto de vértices $S \subseteq \mathbb{N}$ tal que $|S| = n$ y $n \geqslant 2$.
Salida: Una $(n-2)$-upla $(a_1, \ldots, a_{n-2})$.
Procedimiento: Hacemos $n-2$ pasos. En el i-ésimo paso, borramos la menor hoja aún presente y definimos a_i igual al vecino de dicha hoja.

Códigos de Prüfer

Ejemplo

El código de Prüfer del árbol



es $(7, 4, 4, 1, 7, 1)$ y los vértices que quedan al final son 1 y 8.

Fórmula de Cayley

Teorema (Cayley, 1889)

Si $S \subseteq \mathbb{N}$ tiene n elementos entonces hay n^{n-2} árboles con conjunto de vértices S.

Demostración (Prüfer, 1918).

Claramente vale si $n = 1$. Sea $n \geqslant 2$ y vamos a probar que la aplicación que asigna a cada árbol su código de Prüfer establece una biyección entre todos los árboles con conjunto de vértices S y las $(n-2)$-uplas de elementos de S. Para ello vamos a probar que para cada $(n-2)$-upla $a = (a_1, a_2, \ldots, a_{n-2})$ de elementos de S hay uno y sólo un árbol T cuyo código de Prüfer es a. La demostración es por inducción en n. $(\ldots)$

Fórmula de Cayley

Demostración (cont.)

Si $n = 2$, a es la tupla vacía y hay solo un árbol T con $V(T) = S$. Sea $n > 2$ y sea a una $(n-2)$-tupla de valores en S. Supongamos que T es un árbol con código de Prüfer a. Observamos que para computar a reducimos el grado de cada vértice de T a 1 y luego quizá también lo borramos. Por lo tanto, todos los vértices de T que no son hojas aparecen en a. Recíprocamente, ninguna hoja de T aparecen en a porque si se eliminara un vértice vecino a una hoja entonces se reduciría T a un sólo vértice, contradicción. Así que el primer vértice eliminado para el cálculo del código de Prüfer a de T es el mínimo elemento de S que no está presente en a. Llamamos x a dicho elemento. Luego, $T = T' + xa_1$ (i.e., se obtiene agregando una arista xa_1 a T') donde T' es un árbol con $V(T') = S - \{x\}$ y código de Prüfer $a' = (a_2, \ldots, a_{n-2})$. Por hipótesis, existe un único tal T'. Recíprocamente, dado tal T', el árbol $T = T' + xa_1$ tiene código de Prüfer a (porque ninguno de $a_1, \ldots, a_{n-2}$ es hoja de T, lo que incluye a todos los vértices menores que x). □

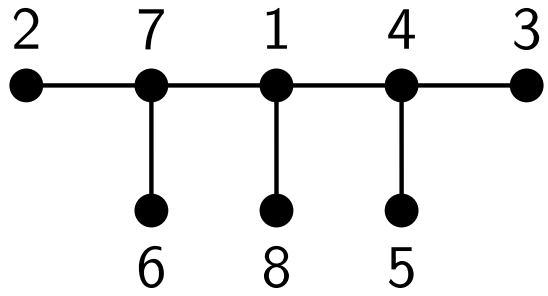
Fórmula de Cayley

Ejemplo

- Veamos como reconstruir el único árbol T cuyo conjunto de vértices es $[8]$ y cuyo código de Prüfer es $a = (7, 4, 4, 1, 7, 1)$.
- Como 2 es el mínimo vértice de T que no aparece en a entonces $T = T' + 27$ donde T' es el árbol con $V(T') = [8] - \{2\}$ y código $a' = (4, 4, 1, 7, 1)$.
- Como 3 es el mínimo vértice de T' que no aparece en a' entonces $T' = T'' + 34$ donde T'' es el árbol con $V(T'') = [8] - \{2, 3\}$ y código $a'' = (4, 1, 7, 1)$.
- Como 5 es el mínimo vértice de T'' que no aparece en a'' entonces $T'' = T^{(3)} + 54$ donde $T^{(3)}$ es el árbol con $V(T^{(3)}) = [8] - \{2, 3, 5\}$ y código $a^{(3)} = (1, 7, 1)$.
- Como 4 es el mínimo vértice de $T^{(3)}$ que no aparece en $a^{(3)}$ entonces $T^{(3)}) = T^{(4)} + 41$ donde $T^{(4)}$ es el árbol con $V(T^{(4)}) = [8] - \{2, 3, 4, 5\}$ y código $a^{(4)} = (7, 1)$.

Fórmula de Cayley

- Como 6 es el mínimo vértice de $T^{(4)}$ que no aparece en $a^{(4)}$ entonces $T^{(4)} = T^{(5)} + 67$ donde $T^{(5)}$ es el árbol con $V(T^{(5)}) = [8] - \{2, 3, 4, 5, 6\} = \{1, 7, 8\}$ y código $a^{(5)} = (1)$.
- Como 7 es el mínimo vértice de $T^{(5)}$ que no aparece en $a^{(5)}$ entonces $T^{(5)} = T^{(6)} + 71$ donde $T^{(6)}$ es el árbol con conjunto de vértices $\{1, 8\}$ con código de Prüfer $a^{(6)} = ()$, es decir, $E(T^{(6)}) = \{18\}$.
- Concluimos que $E(T) = \{27, 34, 54, 41, 67, 71, 18\}$.



Número de árboles con grados dados

La demostración original de Cayley contaba los árboles de acuerdo a los grados de sus vértices. La demostración de Prüfer también nos permite hacerlo.

Corolario (Cayley, 1889)

Si $d_1, \ldots, d_n$ son números naturales cuya suma es $2n-2$, existen exactamente $\frac{(n-2)!}{\prod_{i=1}^{n}(d_i-1)!}$ árboles con conjunto de vértices $[n]$ y tales que el vértice i tiene grado d_i para cada $i \in [n]$.

Demostración.

Cada vez que un vértice x aparece en el código de Prüfer significa que se ha borrado un vecino de él. Luego, como a cada vértice de T se le reduce progresivamente su grado hasta 1 (y eventualmente luego también se lo borra), cada vértice x de T aparece $d_T(x)-1$ veces en el código de Prüfer de T.

Por lo tanto podemos contar los árboles con estos grados como la cantidad de $(n-2)$-uplas tales que para cada i hay d_i-1 copias de i. Por la fórmula de combinaciones con repetición, esto es, $(n-2)!/\prod_{i=1}^{n}(d_i-1)!$. □

Número de árboles generadores

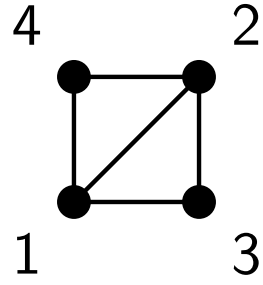
Como el grafo completo con conjunto de vértices $[n]$ tiene todas las aristas necesarias para construir todos los árboles con conjunto de vértices $[n]$, n^{n-2} es también la cantidad de árboles generadores del grafo completo con conjunto de vértices $[n]$.

Podemos considerar el problema más general de hallar el número de árboles generadores de un grafo G dado. Como G puede no tener tanta simetría como el grafo completo, no podemos esperar una fórmula tan simple como la de Cayley en general. Primero vamos a ver una forma recursiva para calcular el número de árboles generadores de G.

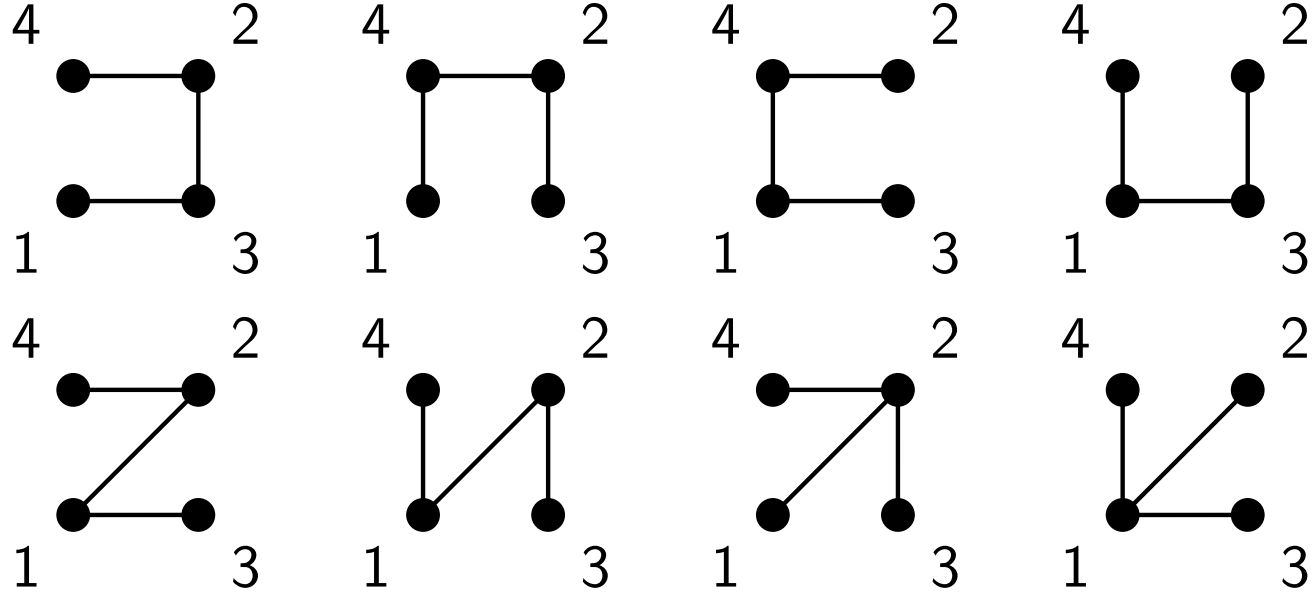
Número de árboles generadores

Ejemplo

¿Cuántos árboles generadores tiene el diamante?



Tiene 4 árboles generadores que son caminos alrededor del ciclo exterior y 4 cuatro árboles generadores que usan la diagonal:



Número de árboles generadores

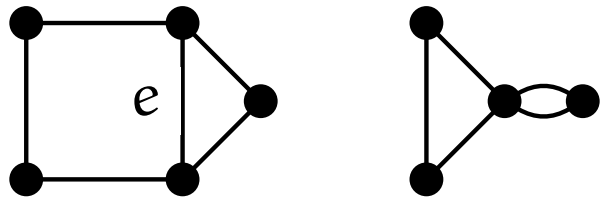
Es claro que los árboles generadores de G que no contienen una arista e son los árboles generadores de $G - e$. Pero, ¿cuántos árboles generadores de G contienen la arista e? La respuesta se expresa en términos de la siguiente operación.

Contracción de una arista

En un grafo G, la contracción de una arista e con extremos u y v consiste en reemplazar u y v con un solo vértice w en el que inciden las aristas distintas de e que eran incidentes en u o en v. Es decir,

$$V(G \cdot e) = (V(G) - \{u, v\}) \cup \{w\} \quad \text{y} \quad E(G \cdot e) = E(G) - \{e\}$$

donde los extremos de una arista $f \in E(G \cdot e)$ en $G \cdot e$ son sus extremos en G salvo por aquellos iguales a u o v, cada uno de los cuales pasa a ser w.



Número de árboles generadores

Observaciones

- La contracción de aristas puede introducir aristas múltiples y bucles.
- Para contar árboles generadores se pueden ignorar los bucles pero vamos a tomar en cuenta las aristas múltiples.
- Árboles generadores que tienen conjuntos de aristas distintos (aunque la diferencia sea solo cambiar una o más aristas por otras paralelas) se consideran distintos.

Número de árboles generadores

Teorema

Sea $\tau(G)$ el número de árboles generadores de un grafo G. Si $e \in E(G)$ no es un bucle entonces $\tau(G) = \tau(G - e) + \tau(G \cdot e)$.

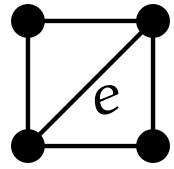
Demostración.

Los árboles generadores de G que no usan la arista e son exactamente $\tau(G - e)$. Mostraremos que la contracción de e define una biyección entre el conjunto de árboles generadores de G que usan e y el conjunto de árboles generadores de $G \cdot e$.
Cuando contraemos e en un árbol generador de G, obtenemos un árbol generador de $G \cdot e$ porque el subgrafo resultante de $G \cdot e$ es generador, conexo y tiene la cantidad correcta de aristas. Como las demás aristas son invariantes por la contracción, dos árboles generadores de G que usan e van a parar a árboles generadores distintos de $G \cdot e$. Recíprocamente, cada árbol generador de $G \cdot e$ da lugar a un único árbol generador de G que usa e precisamente agregándole la arista e. Queda probado que es una biyección. $\square$

Número de árboles generadores

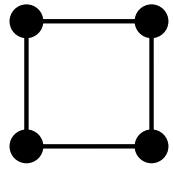
Ejemplo

El número de árboles generadores del diamante:

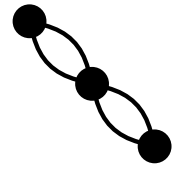


G

coincide con la suma de la cantidad de árboles generadores de los siguientes grafos:



$G - e$



$G \cdot e$

Observaciones

- Este método para computar $\tau(G)$ no es eficiente porque puede requerir sumar $2^{|E(G)|}$ términos.
- El Teorema de Kirchhoff nos da un método más eficiente.

Teorema de Kirchhoff

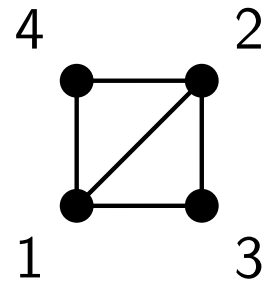
Matriz laplaciana

Una matriz laplaciana de un grafo es la resta $Q = D - A$ de una matriz de grados D (que es diagonal conteniendo en la misma los grados de los vértices) con la matriz adyacencia A de G siguiendo el mismo ordenamiento de los vértices usado en D.

Ejemplo

Una matriz laplaciana del diamante es

$$Q = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{pmatrix}$$



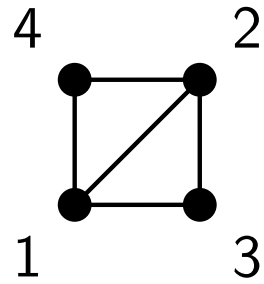
El teorema de Kirchhoff nos dice que si borramos la fila t-ésima y la columna t-ésima de una matriz laplaciana de un grafo sin bucles, el determinante de la submatriz que se obtiene es la cantidad de árboles generadores del grafo (independientemente de t).

Teorema de Kirchhoff

Ejemplo

Como una matriz laplaciana del diamante es

$$Q = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ -1 & -1 & 0 & 2 \end{pmatrix}$$



entonces, por ejemplo, el número de árboles generadores se puede obtener como el determinante de la submatriz Q^* que se obtiene borrando la primera fila y la primera columna, es decir,

$$\tau = \det Q^* = \begin{vmatrix} 3 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 2 \end{vmatrix} = 8.$$

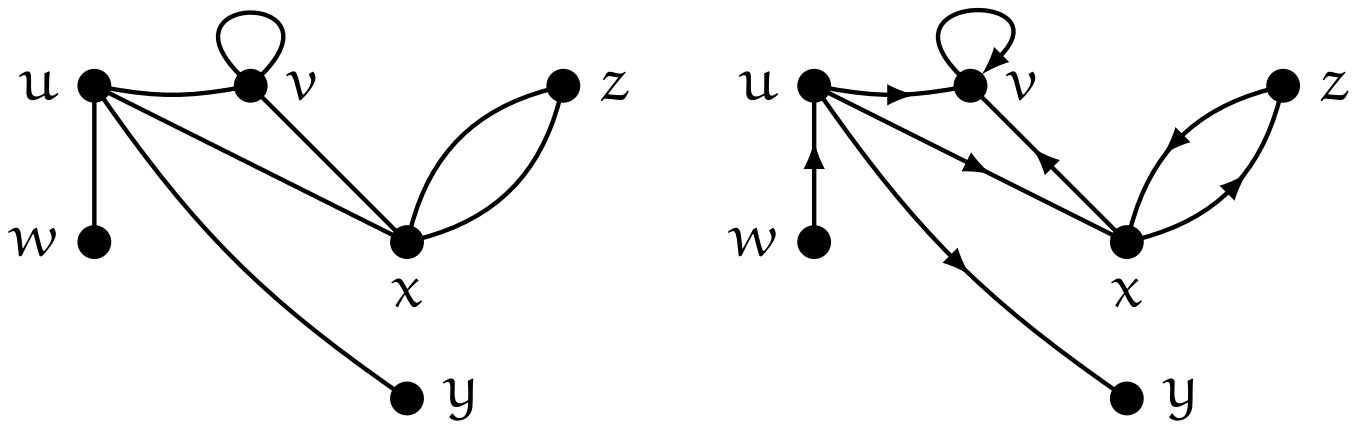
Digrafos y orientación de un grafo

Digrafo

Un grafo dirigido o digrafo D es una terna que consiste en un conjunto $V(D)$ de vértices, un conjunto $E(D)$ de aristas, y una correspondencia que asigna a cada arista un par ordenado de vértices. El primer elemento del par ordenado se llama la cola de la arista y el segundo la cabeza de la arista; ambos se llaman extremos de la arista.

Orientación

Una orientación de un grafo G es un digrafo D que se obtiene ordenando sus extremos de manera que uno de ellos sea la cola y el otro la cabeza.



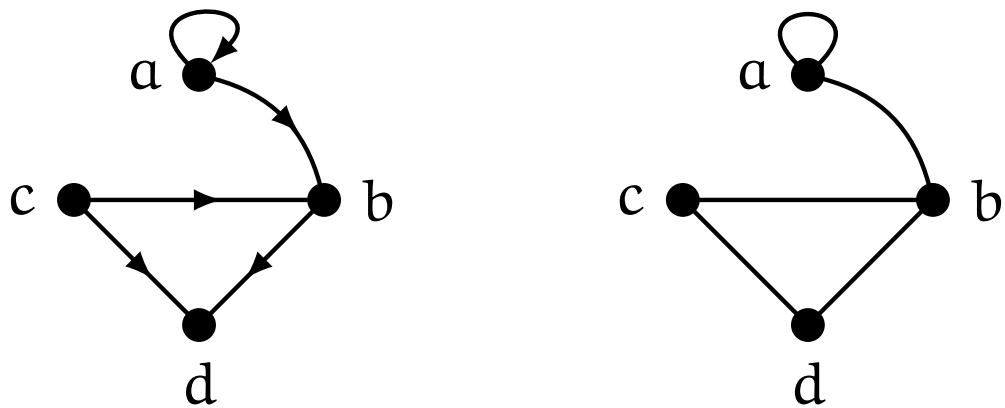
Digrafos y grafo subyacente

Observación

Seguimos la convención de West de utilizar tanto para grafos como para digrafos los nombres 'vértice' y 'arista'. Esto favorece las analogías entre grafos y digrafos. Sin embargo, en el contexto de digrafos es también común utilizar nodo en lugar de vértice y aún más común hablar de arcos en lugar de aristas.

Grafo subyacente

El grafo subyacente de un digrafo D es el grafo G cuyos vértices y aristas son los de D y los extremos de cada arista de G son los mismos que en D pero como par desordenado.



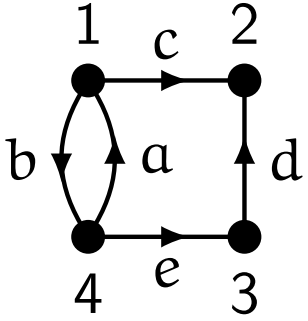
Matrices de adyacencia e incidencia de un digrafo

Matriz de adyacencia de un digrafo

Si D es un digrafo con vértices $v_1, \ldots, v_n$ entonces una matriz de adyacencia de D es la matriz $n \times n$ $A(D) = (a_{ij})$ donde a_{ij} es el número de aristas de v_i a v_j.

Matriz de incidencia de un digrafo

Si D es un digrafo sin bucles con vértices $v_1, \ldots, v_n$ y aristas $e_1, \ldots, e_m$ entonces una matriz de incidencia de D es la matriz $n \times m$ $M(D) = (m_{ij})$ donde m_{ij} es 1 si v_i es la cola de e_j, -1 si v_i es la cabeza de e_j y 0 en caso contrario.



$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \quad M(G) = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

Teorema de Kirchhoff

Para probar el teorema de Kirchhoff sobre el número de árboles generadores necesitamos el siguiente lema.

Lema

Sea G un grafo sin bucles. Si D es una orientación cualquiera de G, M es una matriz de incidencia de D y $Q = MM^T$, entonces Q es una la matriz laplaciana de G con las filas y columnas de Q ordenadas como las fila de M.

Demostración.

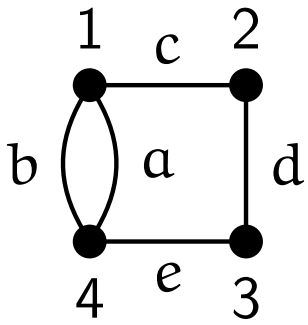
Sean D una orientación cualquiera de G. Sean $v_1, \ldots, v_n$ los vértices y $e_1, \ldots, e_m$ las aristas de D y sea M la matriz de incidencia correspondiente. Si $MM^T = (q_{ij})$ entonces q_{ij} es el producto escalar de las filas i y j de M. Cuando $i = j$ el producto cuenta 1 por cada arista incidente, lo que da el grado en G. Y cuando $i \neq j$, el producto escalar cuenta -1 para cada arista de G que une los dos vértices. □

Teorema de Kirchhoff

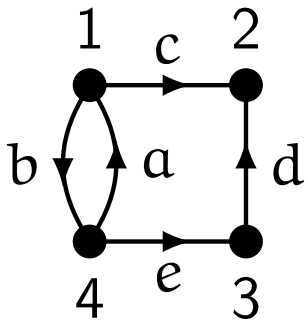
Ejemplo

Considerando un grafo G y una orientación D cualquiera del mismo

G:



D:



una matriz de incidencia es

$$M = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

y

$$Q = MM^{T} = \begin{pmatrix} 3 & -1 & 0 & -2 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -2 & 0 & -1 & 3 \end{pmatrix}$$

Teorema de Kirchhoff

Teorema (Kirchhoff, 1847)

Sea G un grafo sin bucles y sea Q una matriz laplaciana de G. Si Q^* es la matriz que se obtiene de Q removiendo la fila t-ésima y la columna t-ésima entonces $\tau(G) = \det Q^*$.

Demostración.

1. *Sea* M *una matriz de incidencia de una orientación de* G *y sea* B *una submatriz* $(n-1) \times (n-1)$ *de* M. *Si las aristas de las columnas de* B *no forman un árbol generador de* G, $\det B = 0$. Si las $n-1$ aristas correspondientes a las columnas de B no forman un árbol generador, forman un ciclo C al que damos un sentido de recorrido. Consideramos la siguiente combinación lineal de columnas de B: con coeficiente 0 si la arista correspondiente no está en C, 1 si está orientada en el mismo sentido que en el recorrido de C y -1 si está orientada en sentido opuesto. Como esta combinación lineal suma nulo, las columnas son linealmente dependientes y $\det B = 0$. $(\ldots)$

Teorema de Kirchhoff

Demostración (cont.)

2. *Si las aristas de* B *forman un árbol generador de* G *entonces* $\det B = \pm 1$.
   Si $n = 1$, por convención una matriz de tamaño 0×0 tiene determinante 1. Para $n > 1$, como T tiene al menos dos hojas y solo una fila se removió entonces B tiene una fila correspondiente a una hoja x de T. Esta fila tiene una única entrada no nula en B. Entonces al computar el determinante de B desarrollando por esa fila, la única submatriz con determinante no nulo en el desarrollo es la matriz B' correspondiente al árbol generador de $G - x$ que se obtiene de T borrando x y su única arista incidente. Como B' es una submatriz $(n-2) \times (n-2)$ de la matriz de incidencia de una orientación de $G - x$, la hipótesis inductiva asegura que $\det B' = \pm 1$. Como la entrada no nula de la fila de x es ± 1, obtenemos el mismo resultado para B. $(\ldots)$

Fórmula de Binet–Cauchy

El teorema de Binet–Cauchy expresa el determinante del producto de dos matrices cuyo producto es una matriz cuadrada, en términos de determinantes de submatrices de los factores. Este resultado generaliza el hecho de que el determinante del producto de dos matrices cuadradas es el producto de los determinantes de los factores.

Fórmula de Binet–Cauchy (1812)

Sean A y B dos matrices $n \times m$ y $m \times n$, respectivamente. Dado $S \in \binom{[m]}{n}$ (un subconjunto de $[m]$ con n elementos), sea A_S la submatriz $n \times n$ cuyas columnas son las columnas de A indexadas por S, y sea B_S la matriz cuyas filas son las filas de B indexadas por S. Entonces

$$\det(AB) = \sum_{S \in \binom{[m]}{n}} \det A_S \det B_S .$$

Fórmula de Binet–Cauchy

Si A es $n \times m$ y B es $m \times n$: $\det(AB) = \sum_{S \in \binom{[m]}{n}} \det A_S \det B_S$.

Demostración.

Por la multilinealidad del determinante, si $A_1, \ldots, A_m$ son las columnas de A y $B = (b_{ij})$ entonces

$$\det(AB) = \det\left(\sum_{j_1=1}^{m} b_{j_1 1} A_{j_1}, \ldots, \sum_{j_n=1}^{m} b_{j_n n} A_{j_n}\right)$$

$$= \sum_{j_1=1}^{m} \cdots \sum_{j_n=1}^{m} b_{j_1 1} \cdots b_{j_n n} \det(A_{j_1}, \ldots, A_{j_n}).$$

Notemos que solo hace falta sumar los términos donde $j_1, \ldots, j_n$ son todos distintos (pues los demás son nulos). $(\ldots)$

Fórmula de Binet–Cauchy

Demostración.

Sea $i = (i_1, \ldots, i_n)$ la tupla con los mismos elementos que $j = (j_1, \ldots, j_n)$ pero ordenados de menor a mayor. Y sea σ la permutación de $[n]$ que ordena j, es decir, $(j_{\sigma(1)}, \ldots, j_{\sigma(n)}) = i$. Tenemos que

$$b_{j_1 1} \cdots b_{j_n n} = b_{j_{\sigma(1)} \sigma(1)} \cdots b_{j_{\sigma(n)} \sigma(n)} = b_{i_1 \sigma(1)} \cdots b_{i_n \sigma(n)}$$

y $\det(A_{j_1}, \ldots, A_{j_n}) = \operatorname{sg}(\sigma) \det(A_{i_1}, \ldots, A_{i_n})$. Luego,

$$\begin{aligned}
\det(AB) &= \sum_{j_1=1}^{m} \cdots \sum_{j_n=1}^{m} b_{j_1 1} \cdots b_{j_n n} \det(A_{j_1}, \ldots, A_{j_n}) \\
&= \sum_{1 \leqslant i_1 < \cdots < i_n \leqslant m} \det(A_{i_1}, \ldots, A_{i_n}) \sum_{\sigma \in S_n} \operatorname{sg}(\sigma) b_{i_1 \sigma(1)} \cdots b_{i_n \sigma(n)} \\
&= \sum_{S \in \binom{[m]}{n}} \det A_S \det B_S .
\end{aligned}$$

□

Teorema de Kirchhoff

Demostración (cont.)

3. *Cálculo de* $\det Q^*$.
   Sea M^* el resultado de borrar la fila t-ésima de M, así que $Q^* = M^*(M^*)^T$. Cuando aplicamos la fórmula de Binet–Cauchy a $Q^* = M^*(M^*)^T$ nos queda que A_S es una submatriz $(n-1) \times (n-1)$ de M como la que veníamos analizando y que $B_S = A_S^T$. Por lo tanto, la fórmula

$$\det Q^* = \sum_{S \in \binom{[m]}{n}} (\det A_S)^2$$

   suma $(\pm 1)^2 = 1$ cada vez que las aristas de S forman un árbol generador y 0 en caso contrario. Concluimos que $\tau(G) = \det Q^*$. □

Teorema de Kirchhoff

Si A es una matriz cuadrada, denotamos por $\operatorname{Adj} A$ la adjunta de la matriz A; es decir, $\operatorname{Adj} A = (A_{ij})$ donde $A_{ij} = (-1)^{i+j} M_{ij}$ y M_{ij} es el menor de la matriz que resulta al eliminar la fila i-ésima y la columna j-ésima de A. Recordemos que $A(\operatorname{Adj} A)^T = (\det A)\,\mathrm{Id}$.

Proposición

Si A es una matriz cuadrada tal que cada una de sus filas suma 0 entonces las filas de $\operatorname{Adj} A$ son constantes.

Demostración.

Como cada una de sus columnas de A suma 0 entonces $\operatorname{rango} A < n$. Si $\operatorname{rango} A < n-1$ entonces $\operatorname{Adj} A$ es nula y la proposición vale. Por lo tanto, podemos suponer que $\operatorname{rango} A = n-1$. En particular, $\det A = 0$.
Como cada una de las filas de A suma 0 entonces $\mathbf{1} = (1, \ldots, 1)^T$ pertenece al espacio nulo de A. Y como $\operatorname{rango} A = n-1$, todos todos los vectores en el espacio nulo de A son múltiplos de $\mathbf{1}$.
Como $A(\operatorname{Adj} A)^T = (\det A)\,\mathrm{Id} = 0$, cada columna de $(\operatorname{Adj} A)^T$ está en el espacio nulo de A y por lo tanto es constante. $\square$

Teorema de Kirchhoff

Teorema (Kirchhoff, 1847)

Sea G un grafo sin bucles y sea Q una matriz laplaciana de G. Si Q^* es la matriz que se obtiene de Q removiendo la fila t-ésima y la columna t-ésima entonces $\tau(G) = \det Q^*$.

El teorema nos dice que todas las entradas en la diagonal de $\operatorname{Adj} Q$ son todas iguales a $\tau(G)$. Como cada una de las filas de Q suma 0 entonces, por la proposición anterior, las filas de $\operatorname{Adj} Q$ son constantes. Podemos entonces reformular el teorema de la siguiente forma.

Teorema (Kirchhoff, 1847)

Sea G un grafo sin bucles y sea Q una matriz laplaciana de G. Entonces todas las entradas de $\operatorname{Adj} Q$ son iguales a $\tau(G)$.

Grados de un digrafo

Grados. Grados mínimos y máximos. Vecindades.

Sea v un vértice de un digrafo D.

- El grado saliente de v, que se denota $d_D^+(v)$, es igual al número de aristas que tienen cola v.
- El grado entrante de v, que se denota $d_D^-(v)$, es igual al número de aristas que tienen cabeza v.
- El grado máximo saliente y el grado máximo entrante se denotan $\Delta^+(D)$ y $\Delta^-(D)$.
- El grado mínimo saliente y el grado mínimo entrante se denotan $\delta^+(D)$ y $\delta^-(D)$.
- La vecindad saliente de v, que se denota $N_D^+(v)$, es el conjunto $\{x \in V(D) \colon vx \in E(D)\}$.
- La vecindad entrante de v, que se denota $N_D^-(v)$ es el conjunto $\{x \in V(D) \colon xv \in E(D)\}$.

Grados en digrafos

Lema del apretón de manos para digrafos

Si D es un digrafo entonces
$\sum_{v \in V(D)} d_D^+(v) = |E(D)| = \sum_{v \in V(D)} d_D^-(v)$.

Una secuencia de pares de grados de un digrafo D con $V(D) = \{v_1, \ldots, v_n\}$ es $(d_D^+(v_1), d_D^-(v_1)), \ldots, (d_D^+(v_n), d_D^-(v_n))$.

Proposición

Una secuencia $d = (d_1^+, d_1^-), \ldots, (d_n^+, d_n^-)$ de pares de enteros no negativos es la secuencia de pares de grados de un digrafo si y sólo si $\sum_{i=1}^n d_i^+ = \sum_{i=1}^n d_i^-$.

Demostración.

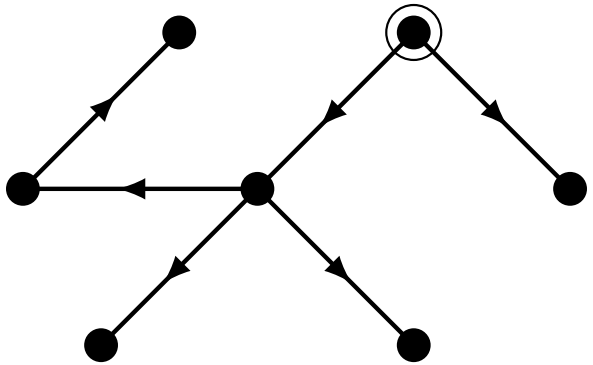
Por inducción en $m = \sum_{i=1}^n d_i^- = \sum_{i=1}^n d_i^+$. Si $m = 0$, es claro. Supongamos que $m > 0$. Luego, $d_i^+ > 0$ y $d_j^- > 0$ para algunos $i, j \in \{1, \ldots, n\}$. La secuencia d' que se obtiene restando 1 a d_i^+ y a d_j^- es realizada por algún digrafo (por hipótesis inductiva). Agregando una arista de v_i a v_j se obtiene un digrafo que realiza d. □

Teorema de Tutte

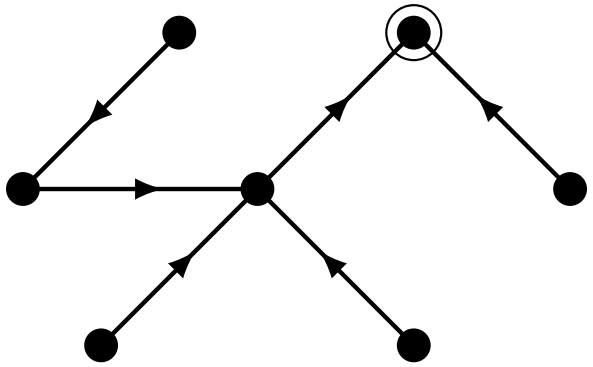
Tutte extendió el teorema de Kirchhoff para digrafos en términos de árboles generadores salientes y entrantes.

Árboles salientes y entrantes

Un árbol saliente (o ramificación o arborescencia) es una orientación de un árbol que tiene una raíz de grado entrante 0 y todos los otros vértices de grado entrante 1.

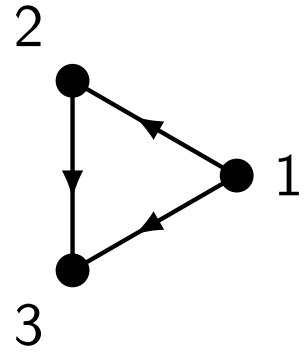


Un árbol entrante es todo aquel que se obtiene de un árbol saliente revirtiendo la orientación de cada una de las aristas.

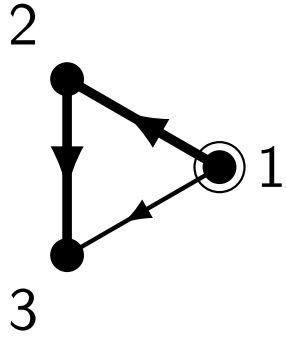


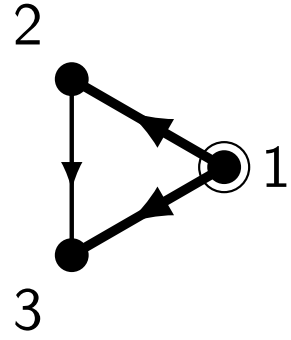
Número de árboles generadores salientes y entrantes

Consideremos el digrafo

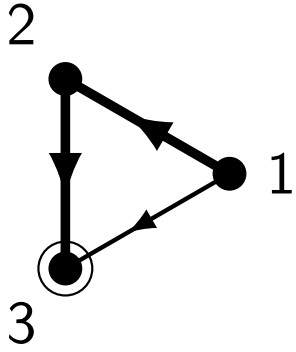


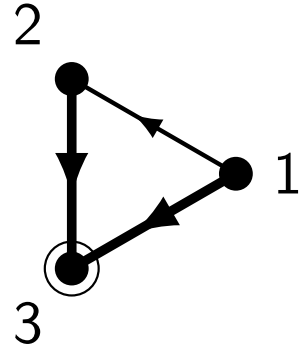
¿Cuántos árboles generadores salientes tiene? 2, ambos con raíz 1:





¿Cuántos árboles generadores entrantes tiene? 2, ambos con raíz 3:



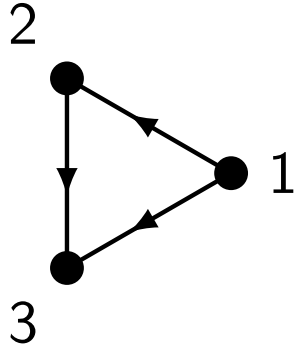


Matrices laplacianas de digrafos

Matrices laplacianas de un digrafo

Dado un digrafo D sin bucles, definimos:

- $\mathrm{Q}^- = \mathrm{D}^- - A$ donde D^- es la matriz diagonal de los grados entrantes de D y A es la matriz de adyacencia de D (para un mismo ordenamiento de los vértices); y
- $\mathrm{Q}^+ = \mathrm{D}^+ - A$ donde D^+ es la matriz diagonal de los grados salientes de D y A es la matriz de adyacencia de D (para un mismo ordenamiento de los vértices).

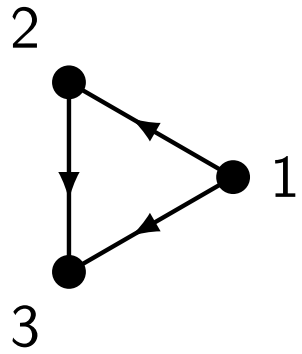


$$\mathrm{Q}^- = \begin{pmatrix} 0 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \qquad \mathrm{Q}^+ = \begin{pmatrix} 2 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

Teorema de Tutte

Teorema (Tutte, 1948)

Sea D un digrafo sin bucles con vértices $v_1, \ldots, v_n$. El número de árboles generadores salientes (resp. entrantes) con raíz en v_i es el determinante de la submatriz de Q^- (resp. de Q^+) que se obtiene eliminando la i-ésima fila y la i-ésima columna.



$$Q^- = \begin{pmatrix} 0 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \qquad Q^+ = \begin{pmatrix} 2 & -1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix}$$

Podemos contar los árboles generadores salientes calculando los cofactores de la diagonal de Q^-. Como son 2, 0 y 0, hay 2 árboles generadores salientes con raíz 1 y ninguno con raíz 2 o 3.
Parar los árboles generadores entrantes calculamos los cofactores de la diagonal de Q^+. Como son 0, 0 y 2, no hay árboles generadores entrantes con raíz 0 o 1, pero hay 2 con raíz 3.

Conexión entre los teoremas de Kirchhoff y Tutte

El teorema de Tutte se reduce al Teorema de Kirchhoff cuando el digrafo es simétrico (es decir, cuando su matriz de adyacencia es simétrica).

Sea G un grafo conexo sin bucles con conjunto de vértices V y sea D el digrafo con el mismo conjunto de vértices y tal que, por cada arista de G, contiene dos aristas que unen los mismos extremos, con orientaciones opuestos entre sí.

Si v es un vértice arbitrario de G, claramente existe un biyección entre los árboles generadores de G y los árboles generadores salientes (resp. entrantes) con raíz v de D. Además, por construcción,

$$\mathrm{d}_{\mathrm{D}}^{+}(v) = \mathrm{d}_{\mathrm{D}}^{-}(v) = \mathrm{d}_{\mathrm{G}}(v) \quad \text{para cada } v \in V(\mathrm{G})$$

y la matriz de adyacencia A de D coincide con la matriz de adyacencia de G.

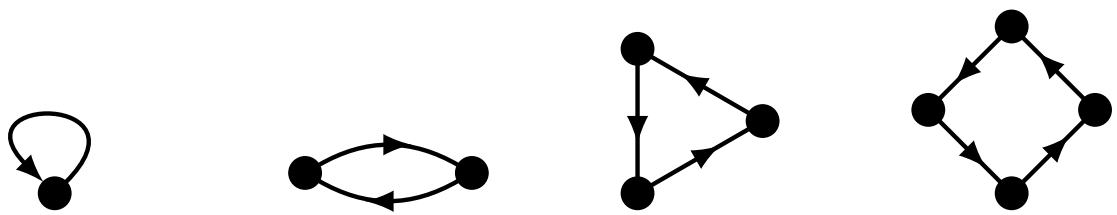
Por lo tanto, las matrices $\mathrm{Q}^{-} = \mathrm{D}^{-} - A$ y $\mathrm{Q}^{+} = \mathrm{D}^{+} - A$ coinciden con la matriz laplaciana de G.

Caminos, ciclos y paseos en digrafos

Digrafos caminos y ciclos

Un digrafo es un camino si sus vértices se pueden ordenar linealmente de manera que posee una arista de u a v si y sólo si v es el sucesor inmediato de u en el ordenamiento.

Un digrafo es un ciclo si sus vértices se pueden ordenar circularmente de manera que posee una arista de u a v si y sólo si v es el sucesor inmediato de u en el ordenamiento circular.



Las definiciones de paseo, recorrido y circuito para digrafos son análogas que para grafos. Por ejemplo, una secuencia $v_0, e_1, v_1, \ldots, e_k, v_k$ es un paseo de un digrafo D si, para cada $i \in \{1, \ldots, k\}$, e_i es una arista de D con cola v_{i-1} y cabeza v_i.

Digrafos fuertemente conexos

Digrafo fuertemente conexo

Un digrafo D es fuertemente conexo, o simplemente fuerte, si para cada par ordenado (u, v) de vértices existe un camino de u a v.

Lema

En todo digrafo fuerte, cada vértice es raíz de un árbol generador saliente.

Demostración.

Sea v un vértice arbitrario de un digrafo D. Agregaremos una arista por vez. Sea S_i el conjunto de vértices que se alcanzan desde v con las primeras i aristas agregadas. Al inicio, $S_0 = \{v\}$. Si $S_i \neq V(D)$ entonces, como el digrafo es fuerte, existe al menos una arista de S_i a $V(D) - S_i$. Si agregamos una tal arista, S_{i+1} contiene un vértice más que S_i. De esta forma, al agregar la $(n-1)$-ésima arista se obtiene un árbol generador saliente. En efecto, todos los vértices tienen grado entrante 1 salvo v y su grafo subyacente es un árbol porque que es conexo, tiene n vértices y $n-1$ aristas. □

Digrafos eulerianos

Digrafos eulerianos

Un recorrido euleriano de un digrafo D es un recorrido de D que atraviesa todas las aristas.
Un digrafo es euleriano si admite un recorrido euleriano cerrado.

La caracterización es análoga que para grafos. Para su demostración nos basamos en el siguiente lema.

Lema

Si un digrafo D satisface $\delta^+(D) \geqslant 1$ (o $\delta^-(D) \geqslant 1$) entonces D contiene un ciclo.

Teorema

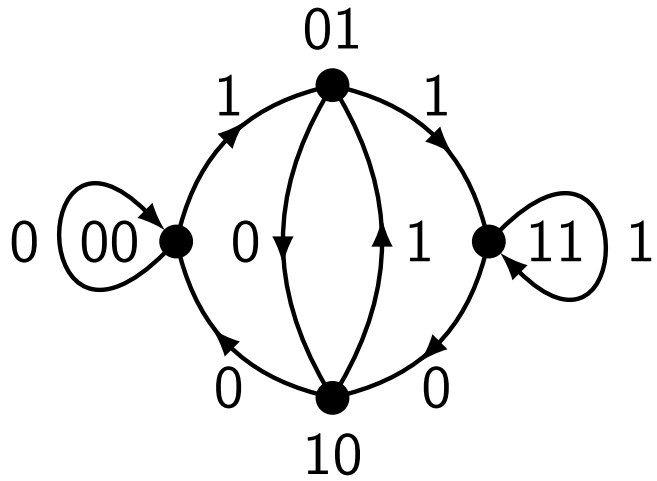
Un digrafo D es euleriano si y sólo si $d^+(v) = d^-(v)$ para todo $v \in V(D)$ y el grafo subyacente tiene a lo sumo una componente no trivial.

La demostración del lema y del teorema son análogas a las dadas para grafos.

Secuencias de De Bruijn

El número de secuencias binarias de longitud n es 2^n. ¿Existe un ordenamiento cíclico de 2^n dígitos binarios tales que las 2^n secuencias de n dígitos consecutivos son todas distintas? Por ejemplo, para $n = 2$ existe la secuencia 0011. Para $n = 3$ existe la secuencia 00010111.

Para estudiar estas secuencias, definimos el grafo de De Bruijn de orden n, denotado D_n, que es un digrafo cuyos vértices son las secuencias binarias de longitud $n-1$ y tal que hay una arista del vértice a al vértices b si y sólo si los últimos $n-2$ dígitos de a coinciden con los primeros $n-2$ dígitos de b. Etiquetamos cada arista de a a b con el último dígito de b. Por ejemplo, D_3 es:



Secuencias de De Bruijn

Teorema

El digrafo D_n es euleriano y las etiquetas de las aristas de un circuito euleriano de D_n forman una ordenamiento cíclico de longitud 2^n tal que cada uno de los 2^n segmento de n elementos consecutivos son distintos.

Demostración.

Por construcción, cada vértice de D_n tiene dos aristas salientes y también dos aristas entrantes (alguna arista puede ser entrante y saliente a la vez). Además, el grafo subyacente de D_n es conexo porque para llegar a un vértice arbitrario $\mathrm{b} = (\mathrm{b}_1, \ldots, \mathrm{b}_{n-1})$ desde cualquier otro vértice solo hace falta seguir una secuencia de aristas con las etiquetas $\mathrm{b}_1, \mathrm{b}_2, \ldots, \mathrm{b}_{n-1}$. Por la caracterización de digrafos eulerianos, D_n es un digrafo euleriano. $(\ldots)$

Secuencias de De Bruijn

Demostración (cont.)

Sea C un circuito euleriano de D_n. Cada vez que C llega a un vértice $a = (a_1, \ldots, a_{n-1})$, las etiquetas de las últimas $n-1$ aristas recorridas fueron $a_1, a_2, \ldots, a_{n-1}$, en ese orden. Si a continuación C prosigue un paso más por una arista etiquetada con a_n entonces las etiquetas de las últimas n aristas recorridas fueron las de la secuencia $a_1, \ldots, a_n$.
Como los 2^{n-1} vértices son distintos y las dos aristas salientes de cada vértice tienen etiquetas distintas, entonces las secuencias de etiquetas de las n aristas recorridas inmediatamente antes de cada paso forman 2^n secuencias binarias distintas de longitud n (es decir, todas las posibles). □

Circuitos eulerianos en digrafos

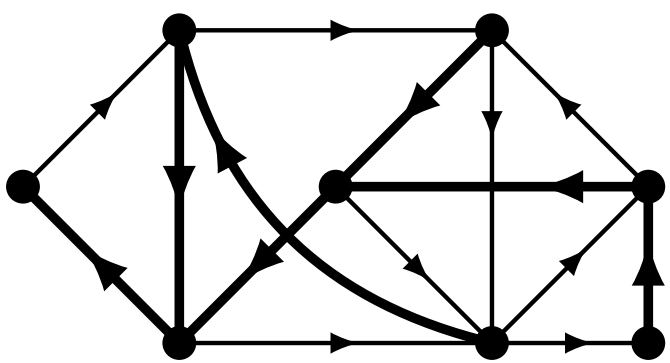
Algoritmo para circuitos eulerianos en digrafos

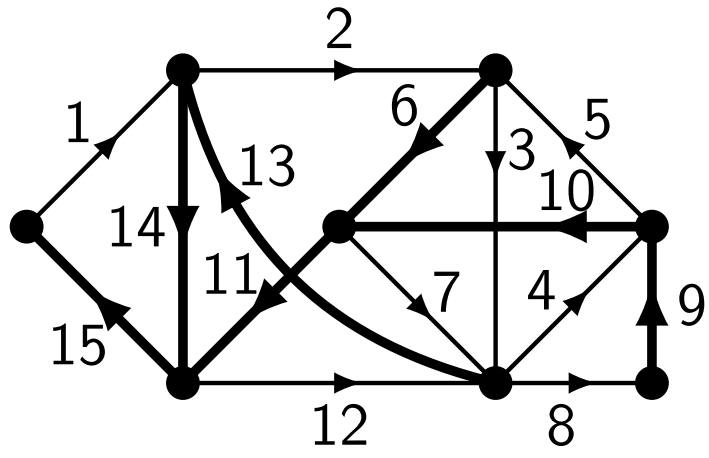
Entrada: Un digrafo euleriano D y un árbol generador entrante T de D con raíz v.

Paso 1: Para cada $u \in V(D)$, establecer un ordenamiento de las aristas que dejan u tal que si $u \neq v$ entonces la arista que deja u en T queda última.

Paso 2: Comenzando en v, construimos un paseo saliendo de cada vértice por la siguiente arista no utilizada (de acuerdo al orden fijado en el Paso 1 entre las aristas que dejan cada vértice), hasta llegar a un vértice en el cual no hay más aristas salientes sin usar.

Paso 3: Devolver el paseo construido en el paso anterior.





Circuitos eulerianos en digrafos

Teorema

Este algoritmo produce siempre un circuito euleriano.

Demostración.

Como el Paso 2 nunca repite aristas, el algoritmo se detiene y devuelve un recorrido. Notemos que cada vez que entra en un vértice u distinto de v, la arista que deja u en T todavía no fue usada porque $d_D^-(u) = d_D^+(v)$. Por lo tanto, el Paso 2 se detiene en la raíz v de T. Hemos probado que el algoritmo produce un recorrido cerrado que empieza y termina en v.
Solo nos falta probar que dicho recorrido atraviesa todas las aristas. (...)

Circuitos eulerianos en digrafos

Demostración (cont.)

Cuando el algoritmo se detiene, todas las aristas que dejan v fueron atravesadas y, como se regresó a v, también todas las aristas que ingresan en v fueron atravesadas. Afirmamos que cuando el algoritmo se detiene, el algoritmo ya atravesó también cada una de las aristas que salen de cada uno de los vértices de D distintos v. Supongamos por el absurdo que el algoritmo se detiene sin haber atravesado alguna arista e que sale de un vértice u distinto de v. Sea $P = u_1, e_1, u_2, e_2, \ldots, e_k, u_k$ el camino en T tal que $u_1 = u$ y $u_k = v$. Como todas las aristas que ingresan en v fueron atravesadas, la arista e_k fue atravesada. Luego, cada una de las aristas $e_{k-1}, e_{k-2}, \ldots, e_1$ fueron atravesadas. Pero entonces e_1 que es una arista de T y deja u fue atravesada sin haber atravesado antes la arista e que también deja u, lo que contradice el orden establecido en el Paso 1. Esta contradicción prueba la afirmación. □

Número de circuitos eulerianos en digrafos

Teorema (Van Aardenne-Ehrenfest y De Bruijn, 1951)

Sea D un digrafo euleriano con vértices $v_1, \ldots, v_n$ y sea $d_i = d_D^+(v_i) = d_D^-(v_i)$. Entonces el número de circuitos eulerianos de D es $c \prod_{i=1}^{n} (d_i - 1)!$ donde c es el número de árboles generadores entrantes con raíz en v_i para cualquier $i \in \{1, \ldots, n\}$.

Demostración.

Fijamos e una arista cualquiera de D y sea v su cola. Como dos circuitos son iguales si siguen el mismo orden cíclico de las aristas que atraviesan, alcanza con probar que el número de recorridos eulerianos cerrados que comienzan atravesando la arista e es $c \prod_{i=1}^{n} (d_i - 1)!$ donde c es el número de árboles generadores entrantes con raíz en v. $(\ldots)$

Número de circuitos eulerianos en digrafos

Demostración (cont.)

Sean dados un árbol generador entrante T de D con raíz v y, para cada $u \in V(D)$, un ordenamiento lineal de las aristas que salen de u tal que: (i) si $u \neq v$ entonces última arista en el ordenamiento de las aristas que dejan u es una arista de T; y (ii) si $u = v$ entonces la primera arista en el ordenamiento de las que dejan v es e. Como acabamos de ver, a partir de e, T y los ordenamientos lineales de las aristas que dejan u para cada $u \in V(D)$, se puede construir un recorrido euleriano cerrado que comienza atravesando la arista e y tal que las aristas que dejan cada vértice $u \in V(D)$ se atraviesan en el orden dado. Como esta construcción es claramente inyectiva, el número de recorridos eulerianos cerrados que comienzan atravesando e es al menos $c \prod_{i=1}^{n} (d_i - 1)!$ donde c es la cantidad de árboles generadores entrantes T de D con raíz en v; en efecto, fijado T entre los c posibles, hay $\prod_{i=1}^{n} (d_i - 1)!$ combinaciones de ordenamientos de las aristas que dejan cada uno de los vértices que cumplen (i) y (ii). (...)

Número de circuitos eulerianos en digrafos

Demostración (cont.)

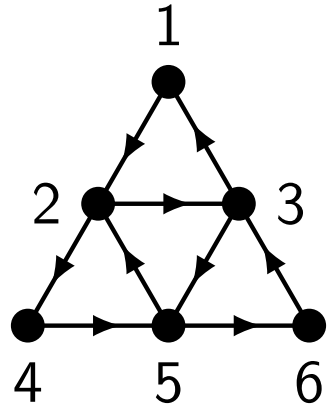
Recíprocamente, sea W un recorrido euleriano cerrado de D que comienza atravesando la arista e. Para cada $u \in V(D) - \{v\}$, sea e_u la arista que sale de u que aparece última en W. Como cada e_u se extiende a un camino de u a v por aristas en $F = \{e_u : u \in V(D) - \{v\}\}$, entonces F es el conjunto de aristas de un árbol generador entrante T de D con raíz v. Además W determina, por restricción, un ordenamiento lineal sobre el conjunto de las aristas que dejan u, para cada $u \in V(D)$. Como es posible reconstruir C aplicando el Paso 2 del algoritmo para circuitos eulerianos en digrafos a e, T y los ordenamientos lineales sobre los conjuntos de aristas que dejan cada vértice, entonces el número de recorridos eulerianos cerrados de D que comienzan atravesando la arista e es a lo sumo $c\prod_{i=1}^{n}(d_i - 1)!$. □

Corolario

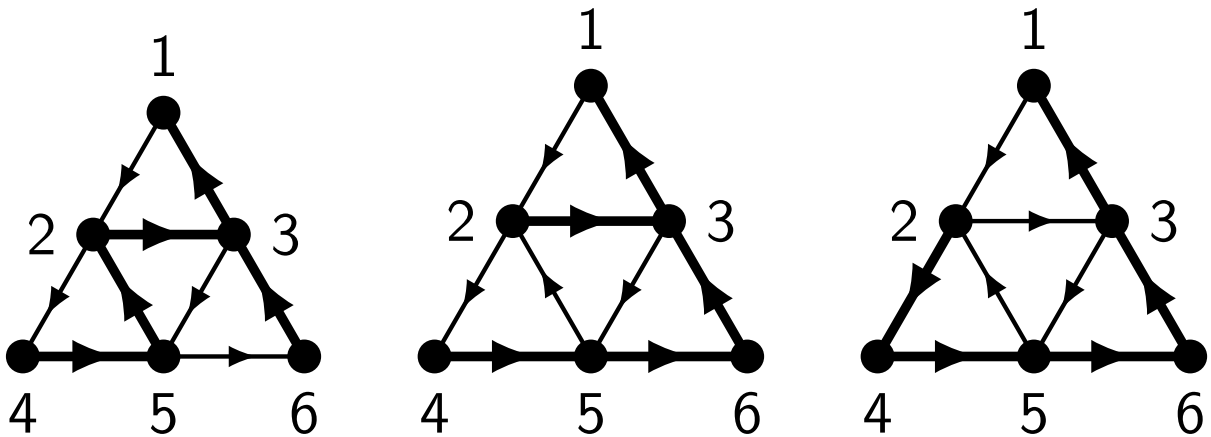
Si D es un digrafo eulariano entonces el número c de árboles entrantes en un vértice v es el mismo para todo $v \in V(D)$.

Número de circuitos eulerianos en digrafos

Ejemplo: ¿Cuántos circuitos eulerianos tiene el siguiente digrafo?



Para ello determinamos los árboles generadores entrantes con raíz 1. Todos ellos tienen las aristas 45, 63 y 31. Por lo tanto, son 3:



Luego, el número de circuitos eulerianos es

$$c\prod_{i=1}^{6}(d_i-1)! = 3\times 0!^3 1!^3 = 3.$$

Descomposición y etiquetados elegantes

Conjetura (Graham y Häggkvist, 1989)

Si T es un árbol con m aristas entonces todo grafo $2m$-regular se descompone en copias de T.

Una conjetura más débil que aún sigue irresuelta.

Conjetura (Ringel, 1964)

Si T es un árbol con m aristas entonces K_{2m+1} se descompone en $2m+1$ copias de T.

Descomposición y etiquetados elegantes

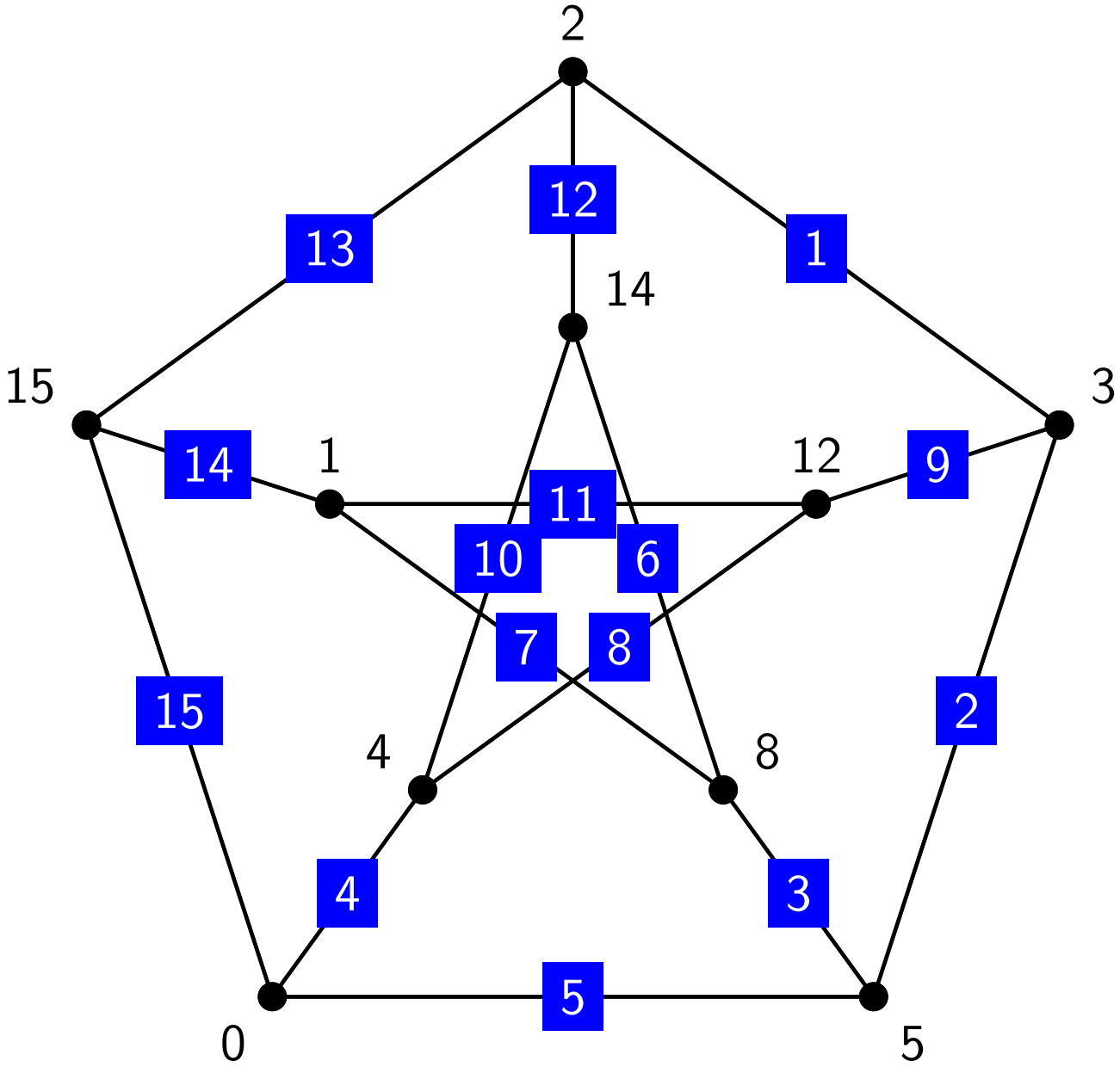
Muchos intentos de probar la conjetura de Ringel se enfocaron en el estudio de etiquetados elegantes.

Etiquetado elegante (graceful labeling)

Un etiquetado elegante de un grafo G con m aristas es una función $f : V(G) \to \{0, \ldots, m\}$ tal que distintos vértices reciben números distintos y $\{|f(u) - f(v)|\colon\ uv \in E(G)\} = \{1, \ldots, m\}$.
Un grafo es elegante si tiene un etiquetado elegante.

Descomposición y etiquetados elegantes

Ejemplo (Gardner, 1983):

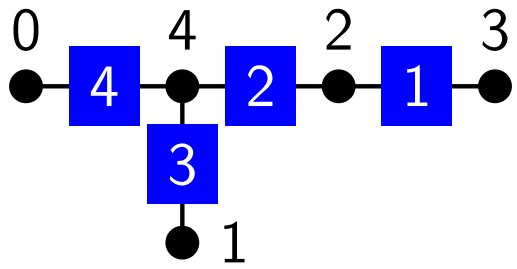


Descomposición y etiquetados elegantes

Se conjetura que todos los árboles admiten etiquetados elegantes.

Conjetura de los árboles elegantes (Ringel y Kotzig, 1964)

Todo árbol tiene un etiquetado elegante.



Descomposición y etiquetados elegantes

El siguiente resultado prueba que la conjetura de los árboles elegantes es más fuerte que la conjetura de Ringel.

Teorema (Rosa, 1967)

Si T es un árbol con m aristas que tiene un etiquetado elegante entonces K_{2m+1} tiene una descomposición en $2m+1$ copias de T.

Demostración.

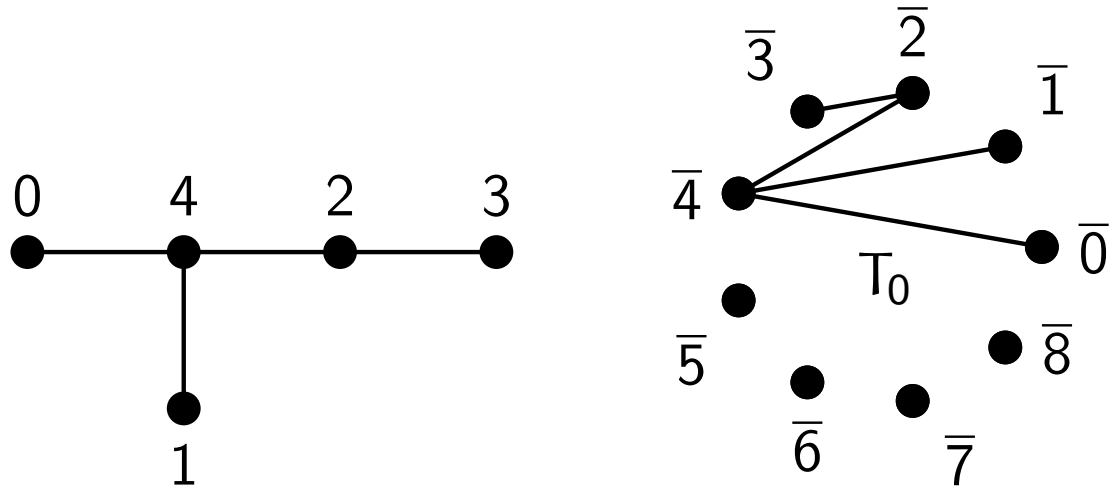
Pensamos los vértices de K_{2m+1} como las clases de equivalencia de la congruencia módulo $2m+1$ que denotamos $\overline{0}, \overline{1}, \ldots, \overline{2m}$. Por diferencia entre dos clases entendemos la distancia entre ambas en el orden cíclico $\overline{0}, \overline{1}, \ldots, \overline{2m}$. Las distancias entre dos clases toman los valores: $0, 1, \ldots, m$. Agrupamos las aristas de K_{2m+1} de acuerdo a la diferencia entre sus extremos. Notemos que la cantidad de aristas con diferencia j entre sus extremos es siempre $2m+1$ para todo $j \in \{1, \ldots, m\}$. $(\ldots)$

Descomposición y etiquetados elegantes

Demostración (cont.)

Sea T un árbol con m aristas con un etiquetado elegante de sus vértices (que por lo tanto son $0, 1, \ldots, m$). Para cada $k \in \{0, \ldots, 2m\}$, sea T_k el grafo con vértices $\overline{k}, \ldots, \overline{m+k}$ tal que

$$E(T_k) = \{(\overline{i+k})(\overline{j+k}) \colon ij \in E(T)\}.$$



Entonces T_0 es isomorfo a T y, en la secuencia $T_0, \ldots, T_{2m}$, cada árbol se obtiene rotando el anterior un lugar en sentido antihorario. Luego, $T_0, \ldots, T_{2m}$ son copias de T que descomponen K_{2m+1}. □

Etiquetados elegantes para árboles

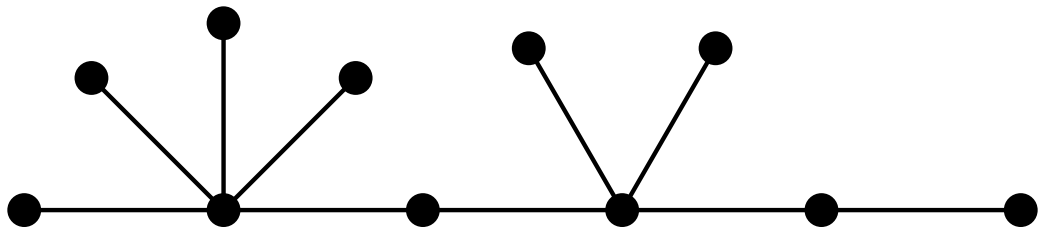
Pregunta

¿Podemos obtener etiquetados elegantes para las estrellas y los caminos?

Vamos a mostrar que una clase de árboles que generalizan estrellas y caminos admiten etiquetados elegantes.

Oruga (Harary and Schwenk, 1973)

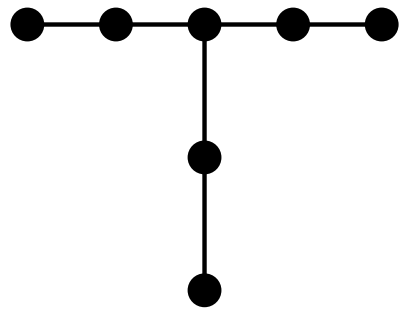
Una oruga es un árbol que contiene un camino P que es arista-dominante, es decir, cada una de las aristas del grafo es incidente en al menos un vértice de P.



Caracterización de las orugas

Teorema

Un árbol es una oruga si y sólo si no contiene T_2 como subgrafo.



T_2

Demostración.

Sea T' el grafo que se obtiene de un árbol T borrando todas sus hojas. Claramente, T es una oruga si y sólo si T' es un camino. A su vez, como T' es un árbol, T' es un camino si y sólo si $\Delta(T') \leqslant 2$. Por último, $\Delta(T') \leqslant 2$ si y sólo si T no contiene T_2 como subgrafo. De todas estas equivalencias se sigue el resultado del teorema. $\square$

Etiquetados elegantes para orugas

Teorema (Rosa, 1973)

Todo oruga tiene un etiquetado elegante.

Bosquejo de la demostración.

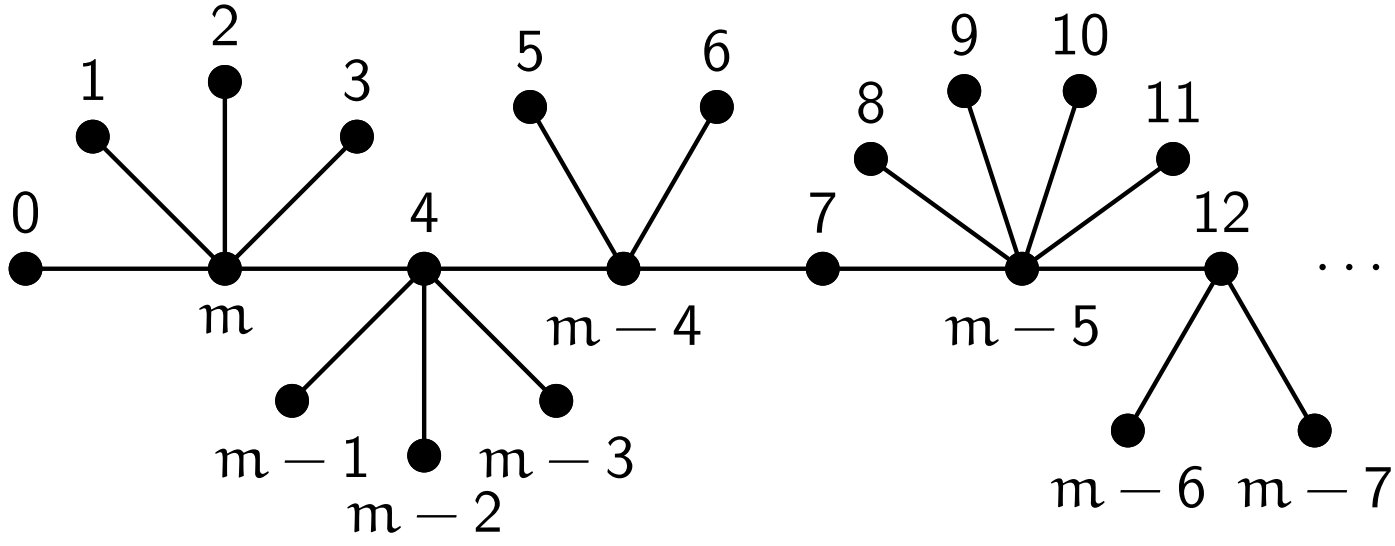
Sea T una oruga y sea m su número de aristas. Sea $P = v_1, \ldots, v_k$ un camino arista-dominante que es maximal (es decir, sus extremos son hojas).
Proponemos el siguiente método para etiquetar todos sus vértices:

- Etiquetamos con etiquetas consecutivas crecientes comenzando desde 0 los siguientes vértices: v_1, los vecinos de v_2 fuera de P, v_3, los vecinos de v_4 fuera de P, etc.
- Etiquetamos con etiquetas consecutivas decrecientes comenzando desde m los siguientes vértices: v_2, los vecinos de v_3 fuera de P, v_4, los vecinos de v_5 fuera de P, etc. $(\ldots)$

Etiquetados elegantes para orugas

Bosquejo de la demostración (cont.)

Por ejemplo:



Se observa y puede probarse que este procedimiento resulta en un etiquetado elegante para toda oruga porque las aristas incidentes en cada vértice de P tienen diferencias consecutivas.

Ejercicio: Formalizar la demostración por inducción. □

Descomposición y etiquetados elegantes

Kotzig llegó a considerar la búsqueda de una demostración de la conjetura de los árboles elegantes una "enfermedad" (Huang, Kotzig, and Rosa, 1982).

Algunos otros resultados parciales son los siguientes:

- Todos los árboles con a lo sumo cuatro hojas son elegantes (Huang, Kotzig y Rosa, 1982).
- Todos los árboles de diámetro a lo sumo 5 son elegantes (Hrnčiar and Haviar, 2001).
- Todos los árboles con a lo sumo 27 vértices son elegantes (Aldred y McKay, 2007).

Árbol generador de peso mínimo

Dado un grafo conexo en el que las aristas representan posibles conexiones con pesos asociado a cada una de ellas, ¿cómo podemos hallar un árbol generador que minimice el peso total?

Algoritmo de Kruskal

Entrada: Un grafo conexo G con un peso $w(e)$ en cada arista e.
Idea: Mantener un bosque generador H e ir extendiéndolo con aristas de peso bajo hasta un árbol generador. Las aristas se consideran en orden no decreciente de pesos, desempatando entre aristas de igual peso de forma arbitraria.
Inicialización: Sea $V(H) = V(G)$ y $E(H) = \emptyset$.
Iteración: Si la próxima arista de G en orden no decreciente de peso une dos componentes de H, agregarla a H; si no, descartarla. Repetir hasta que H sea conexo.
Salida: Devolver H.

Algoritmo de Kruskal

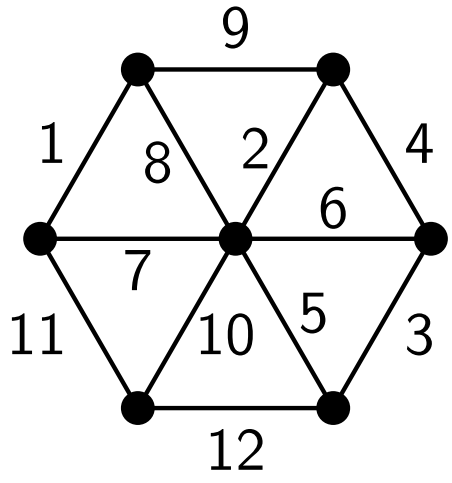
Observaciones

- El orden en que se consideran las aristas depende solamente del orden relativo de los pesos (no de sus valores en absoluto).
- En caso de haber aristas con el mismo peso, el orden en que se consideran entre ellas es arbitrario.
- Los algoritmos que construyen una solución mediante elecciones localmente óptimas en cada paso se llaman algoritmos golosos (greedy algorithms).
- Los algoritmos golosos usualmente no garantizan soluciones óptimas.
- Sin embargo, el algoritmo de Kruskal es un ejemplo de algoritmo goloso cuyo resultado es óptimo.

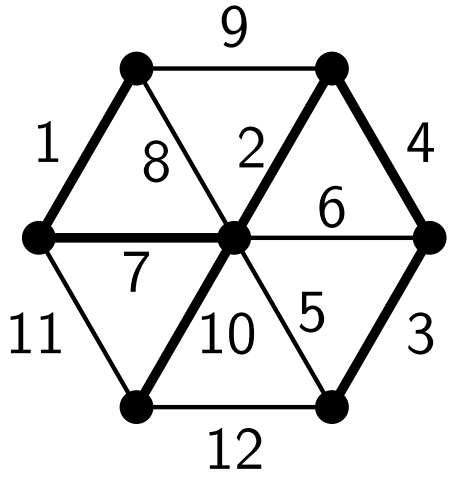
Algoritmo de Kruskal

Ejemplo

Si aplicamos el algoritmo de Kruskal al siguiente grafo:



obtenemos el siguiente árbol generador:



Algoritmo de Kruskal

Teorema (Kruskal, 1956)

Para un grafo G conexo con pesos en las aristas, el algoritmo de Kruskal construye un árbol generador de peso mínimo.

Demostración.

Veamos primero que el algoritmo se detiene. Supongamos, por el absurdo que, luego de considerar todas las aristas, H es disconexo. Como G es conexo, admite al menos una arista e que une dos componentes distintas de H. Luego, al momento de ser considerada, e debió haber sido agregada a H (por unir dos componentes distintas de H también en ese momento). Esta contradicción muestra que en alguna iteración H se vuelve conexo y el algoritmo se detiene. Además, H es acíclico puesto que la adición de una arista que une componentes distintas no introduce un ciclo. Concluimos que la salida H del algoritmo es un grafo conexo y acíclico; es decir, un árbol. $(\ldots)$

Algoritmo de Kruskal

Demostración (cont.)

Sea T el árbol resultante y sea T^* un árbol generador de mínimo peso que comparte con T las primeras k aristas escogidas por el algoritmo con k máximo posible. Supongamos, por el absurdo, que $T \neq T^*$ y sea e la primera arista elegida para T que no está presente en T^*. Agregando e a T^* se crea un ciclo C. Como T no tiene ciclos, C tiene una arista $e' \notin E(T)$. Consideremos el árbol generador $T^* + e - e'$. Afirmamos que la arista e' es considerada después que la arista e por el algoritmo. En efecto, si e' hubiese sido considerada antes que e por el algoritmo entonces hubiera sido aceptada ya que e' y todas las aristas elegidas por el algoritmo antes que e pertenecen a T^* (y T^* no contiene ciclos), lo cual contradiría $e' \notin E(T)$. Por lo tanto, $w(e) \leqslant w(e')$. Luego, $T^* + e - e'$ es un árbol generador con peso a lo sumo el peso de T^* y que coincide con T en una arista inicial más que T^*, lo que contradice la elección de T^*. Luego, $T = T^*$ y, en particular, T es un árbol generador de peso mínimo. □

Notación O

El tiempo de ejecución de un algoritmo es el número de pasos que requiere su ejecución. Esta cantidad de pasos depende de la entrada (es decir, los datos que ingresan al algoritmo).
Como el tiempo en concreto que requiere cada paso en particular varía de una computadora a otra, las constantes no se toman en cuenta.

Notación O

Si f es una función real definida sobre los naturales, el conjunto $O(f)$ es el conjunto de funciones reales g definidas sobre los naturales para las cuales existe una constante $c > 0$ (que depende de g) tal que $g(n) \leqslant c \cdot f(n)$ para todo n suficientemente grande.
Diremos que el tiempo de ejecución de un algoritmo es $O(f)$ si su tiempo de ejecución como función del tamaño de la entrada pertenece a $O(f)$.
Un algoritmo es polinomial o lineal si su tiempo de ejecución es $O(n^k)$ para algún k natural o para $k = 1$, respectivamente, donde n es el tamaño de la entrada.

Algoritmo de Kruskal

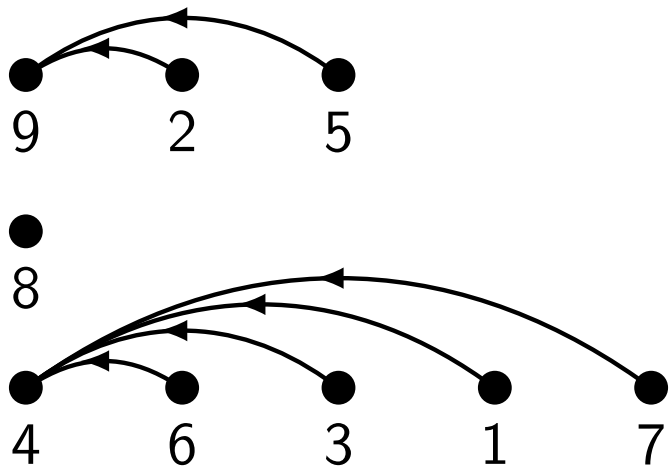
- Para implementar el algoritmo de Kruskal, comenzamos por ordenar las $\mathfrak{m}$ aristas por pesos.
- Además, mantenemos los conjuntos de vértices que forman cada componente.
- Al inicio, cada vértice es el único en su propia componente.
- Luego, al considerar cada arista, o bien aceptamos la arista porque une vértices que pertenecen a distintas componentes o bien rechazamos la arista porque une vértices de una misma componente.
- Cuando aceptamos una arista, unificamos las dos componentes que une.

Algoritmo de Kruskal

- Para ordenar eficientemente las m aristas existen algoritmos de tiempo $O(m \log m)$ y no se requieren estructuras de datos sofisticadas (con usar arreglos alcanza).
- Pero para representar los conjuntos de vértices de las distintas componentes de H utilizaremos una estructura de datos de conjuntos disjuntos que permita tres operaciones:
  - `crear`(X) que crea la familia de conjuntos unitarios $\{\{x\}\colon\ x \in X\}$;
  - `unir`(x, y) que dados dos elementos x e y, une los conjuntos a los que dichos elementos pertenecen; y
  - `encontrar`(x) que dado un elemento x devuelve un representante del conjunto al que pertenece x, de manera que para dos elementos del mismo conjunto devuelve el mismo representante (mientras no se aplique la operación `unir`).

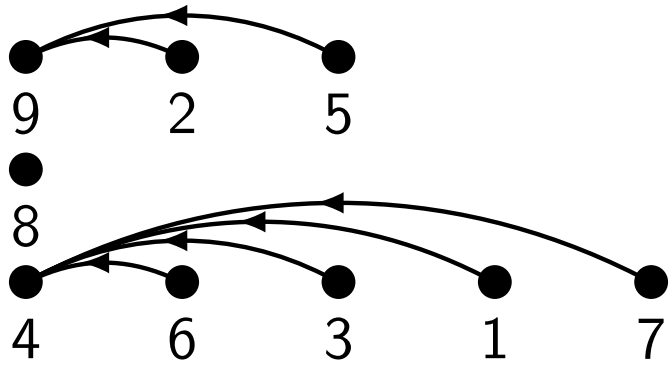
Estructura de datos para conjuntos disjuntos

- Podemos usar una representación en forma de bosque de estrellas entrantes en el que cada representante de un conjunto es el centro y la raíz de una estrella y cada elemento que no es un representante de conjunto tiene una arista que sale de él hasta el representante del conjunto al que pertenece.
- Por ejemplo, podemos representar la familia de conjuntos disjuntos $\{\{1, 3, 4, 6, 7\}, \{2, 5, 9\}, \{8\}\}$ mediante el siguiente bosque de estrellas entrantes:

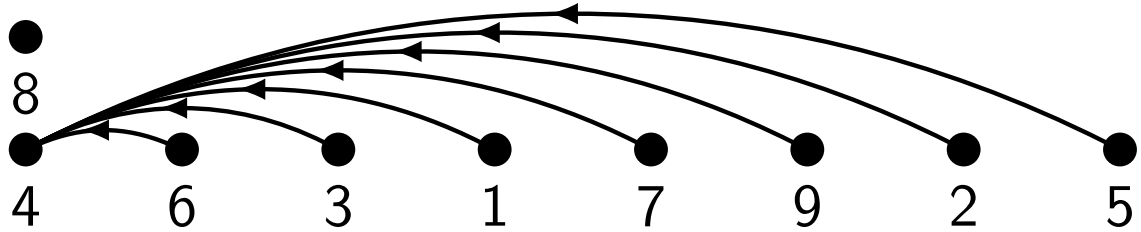


Estructura de datos para conjuntos disjuntos

- Al invocar `crear`(X), se crea un bosque con los elementos de X como vértices y sin aristas.
- Al invocar `encontrar`(x), devolvemos la cabeza de la arista con cola en x o, cuando no haya tal arista, devolvemos x.
- Al invocar `unir`(x, y), redirigimos las aristas con cabeza en el representante r_y de y (resp. r_x de x) hacia r_x (resp. r_y) y agregamos una arista de r_y a r_x (o viceversa). Por ejemplo, si sobre la estructura



se invoca `unir`$(1, 2)$ entonces podríamos obtener

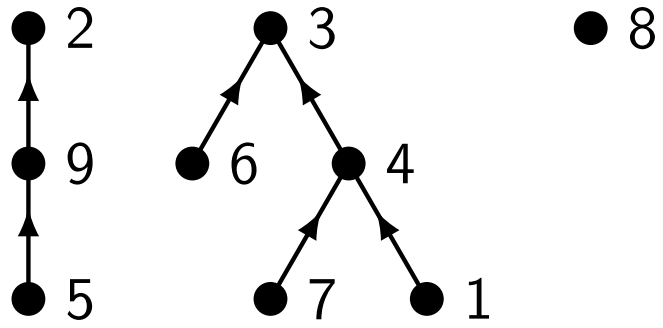


Estructura de datos para conjuntos disjuntos

- Si al invocar `unir`(x, y) se elige arbitrariamente el representante del conjunto unión, la operación puede involucrar modificar las aristas que salen de un número lineal de elementos, por lo que ejecutar $n - 1$ invocaciones de `unir` podría tomar tiempo $O(n^2)$.
- Para evitar esto, en cada operación `unir`(x, y) elegiremos como representante del conjunto unión al representante del conjunto que tenía más elementos.
- Para ello vamos a ampliar nuestra representación para almacenar en cada representante de un conjunto, la cantidad de elementos del mismo.
- Cada vez que se cambia la arista que sale de un elemento, el tamaño del conjunto al que pertenece al menos se duplica.
- Por lo tanto, la cantidad de veces que hay que actualizar la arista que sale de un un elemento es a lo sumo $\log_2 n$.
- Concluimos que, con esta representación, la ejecución de m operaciones de `encontrar` y `unir` sobre un universo X de n elementos puede implementarse en tiempo $O(m + n \log n)$.

Estructura de datos para conjuntos disjuntos

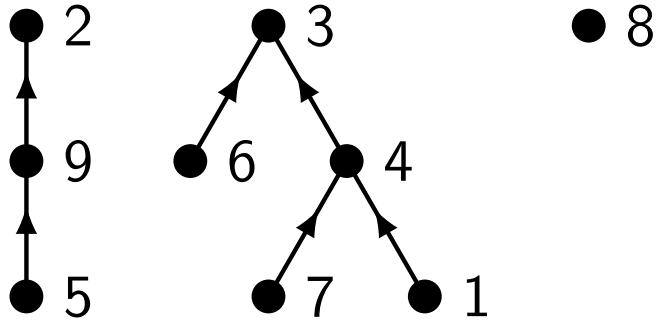
- Para representar una familia de conjuntos disjuntos se puede utilizar en su lugar un bosque de árboles entrantes.
- Por ejemplo, podemos representar la familia de conjuntos disjuntos $\{\{1, 3, 4, 6, 7\}, \{2, 5, 9\}, \{8\}\}$ mediante el bosque de árboles entrantes:



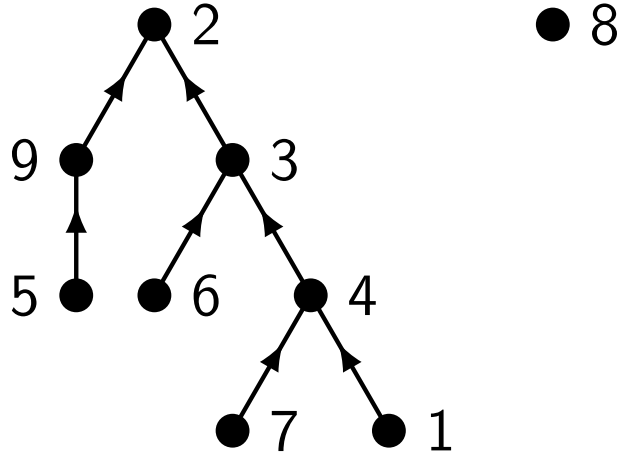
- Con esta representación, la operación `encontrar` devuelve la raíz del árbol entrante al que pertenece al elemento.
- Por ejemplo, $\texttt{encontrar}(1) = 3$, $\texttt{encontrar}(2) = 2$, $\texttt{encontrar}(5) = 2$, etc.

Estructura de datos para conjuntos disjuntos

La operación `unir`(x, y) agrega una arista de `encontrar`(x) a `encontrar`(y).
Por ejemplo, si sobre la representación



aplicamos la operación `unir`$(6, 9)$ obtenemos

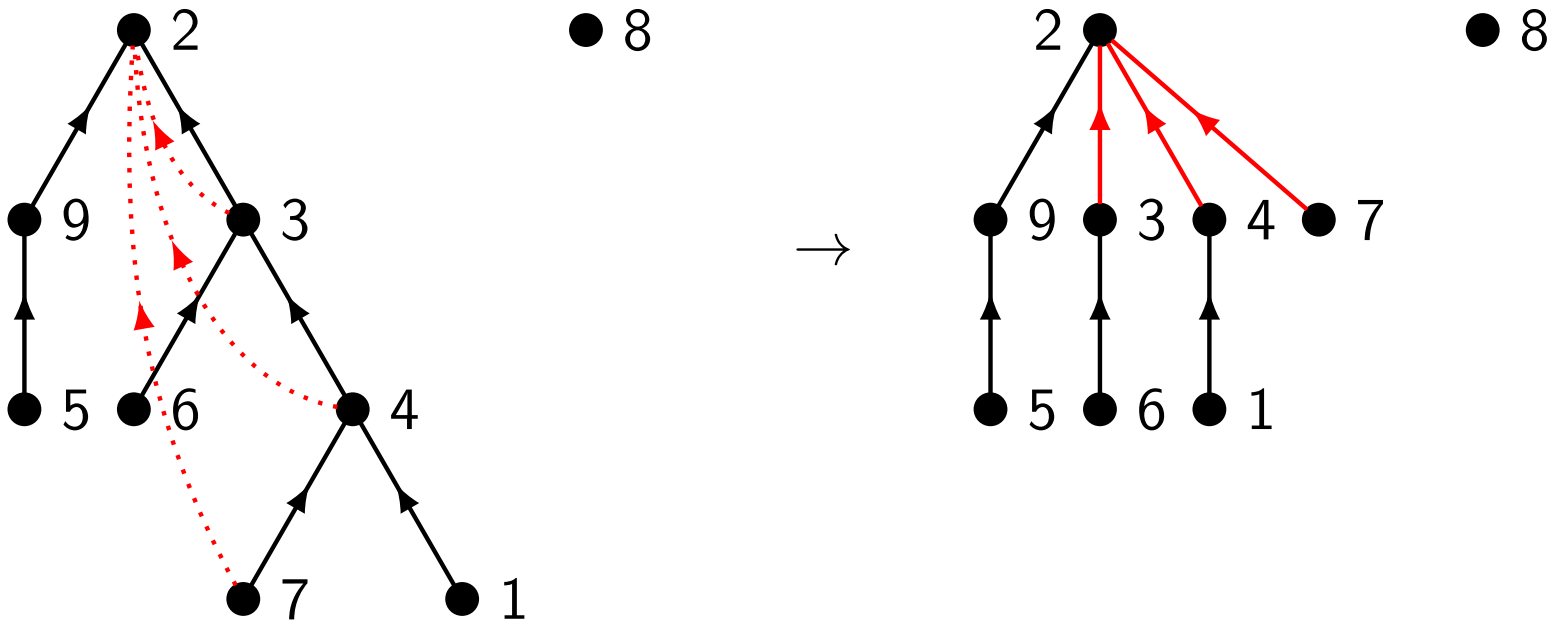


Compresión de caminos

Sobre esta idea básica es posible aplicar dos mejoras.

Una de ellas es la compresión de caminos. La compresión de caminos consiste en que cada vez que se invoca `encontrar`(x) se modifiquen las aristas que dejan los vértices visitados en el camino de x a la raíz de manera que en todas ellas la cabeza pase a ser dicha raíz.

Por ejemplo, si se invoca `encontrar`(7) con compresión de caminos sobre la estructura de la izquierda se introducen las aristas punteadas obteniéndose la estructura de la derecha:



Unión por rango

La segunda mejora que podemos introducir es la unión por rango. La unión por rango consiste en que al invocar $\texttt{unir}(x, y)$ se agrega una arista entre $\texttt{encontrar}(x)$ y $\texttt{encontrar}(y)$ de manera que el rango de la cola sea menor o igual al de la cabeza (desempatando arbitrariamente). El rango es una estimación de la altura del árbol que se determina de la siguiente forma:

- al invocar $\texttt{crear}(X)$ todos los elementos $x \in X$ tienen rango 0;
- invocar $\texttt{encontrar}(x)$, no modifica los rangos; y
- si al invocar $\texttt{unir}(x, y)$, si agrega una arista con cola a y cabeza b entonces el nuevo rango de b es el máximo entre su rango actual y el resultado de sumar 1 al rango de a.

Estructura de datos para conjuntos disjuntos

Teorema (Tarjan, 1975)

Una secuencia de m operaciones `unir` y `encontrar` sobre un universo X de n elementos se pueden completar en tiempo $O(m\,\alpha(n) + n)$ si se utilizan compresión de caminos y unión por rangos, donde $\alpha(n)$ es la función inversa de Ackermann, la cual crece tan lentamente que $0 \leqslant \alpha(n) \leqslant 4$ para todo $n \leqslant 10^{80}$.

Para todos los fines prácticos, el algoritmo puede considerarse lineal, es decir, con tiempo de ejecución $O(m + n)$ y además la constante es muy pequeña.

Algoritmo de Kruskal

- Para el análisis de complejidad, asumamos que G es simple.
- Ya mencionamos $O(m \log m)$ tiempo es suficiente para ordenar por peso las m aristas.
- Si n es la cantidad de vértices entonces $m < n^2$ y podemos decir también que el costo temporal de ordenar las m aristas es $O(m \log n)$.
- Con la representación con estrellas entrantes, el costo temporal del resto del algoritmo es $O(m + n \log n)$.
- Con la representación con árboles entrantes, el costo del algoritmo luego de ordenar las aristas es $O(m\,\alpha(n) + n)$.
- Para ambas representaciones, obtenemos un tiempo de ejecución $O(m \log n)$ para el algoritmo de Kruskal completo.
- Pero cuando las aristas se saben ordenadas previamente, tenemos una complejidad temporal de $O(m\,\alpha(n) + n)$ que, en la práctica, puede considerarse $O(m + n)$, es decir, lineal.
- Hay casos especiales en los cuales, aún cuando las aristas no están previamente ordenadas, pueden ordenarse en tiempo $O(m)$.

Algoritmo de Prim

Presentaremos otro algoritmo goloso para la construcción de árboles generadores de peso mínimo.

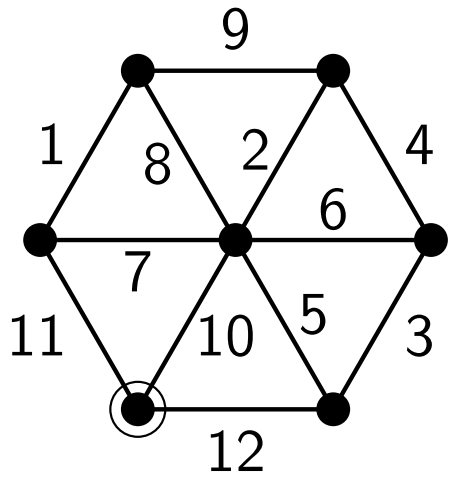
Algoritmo de Prim

Entrada: Un grafo conexo G con un peso $w(e)$ en cada arista e.
Idea: Comenzar con el árbol H con un único vértice v y sin aristas. En cada paso extender H agregando una arista de peso mínimo que une un vértice de H con uno fuera de H. Repetir hasta que $V(H) = V(G)$.
Inicialización: Sea v un vértice cualquiera de G, sea $V(H) = \{v\}$ y $E(H) = \emptyset$.
Iteración: Agregar a H una arista entre $V(H)$ y $V(G) - V(H)$ de peso mínimo. Repetir hasta que $V(H) = V(G)$.
Salida: Devolver H.

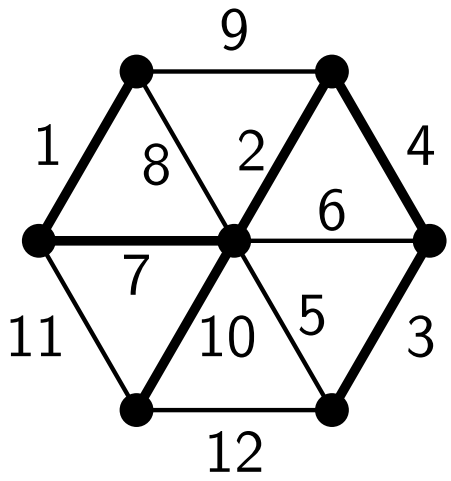
Algoritmo de Prim

Ejemplo

Al aplicar el algoritmo de Prim desde el vértice resaltado



se eligen sucesivamente las aristas de pesos: 10, 2, 4, 3, 7 y 1, obteniéndose el árbol generador:



Algoritmo de Prim

Teorema (Jarník, 1930; Prim, 1957; Dijkstra, 1959)

Para un grafo G conexo con pesos en las aristas, el algoritmo de Prim construye un árbol generador de peso mínimo.

Demostración.

Recordemos que la condición para la detención del algoritmo es que $V(G) = V(H)$. Es claro entonces que el algoritmo se detiene ya que mientras $V(H) \neq V(G)$ debe existir al menos una arista entre $V(H)$ y $V(G) - V(H)$ (por ser G conexo).
Probemos ahora que la salida del algoritmo es un árbol generador. Como el grafo H inicial es un árbol y en cada iteración se le agrega una hoja, en todo momento H es un árbol (por definición). Como al detenerse $V(H) = V(G)$, la salida es un árbol generador. $(\ldots)$

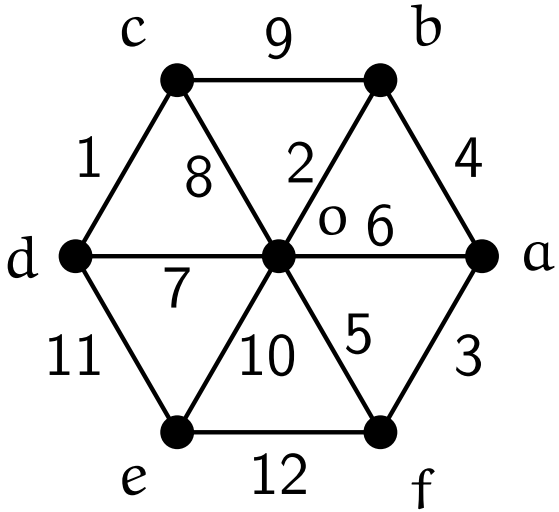
Algoritmo de Prim

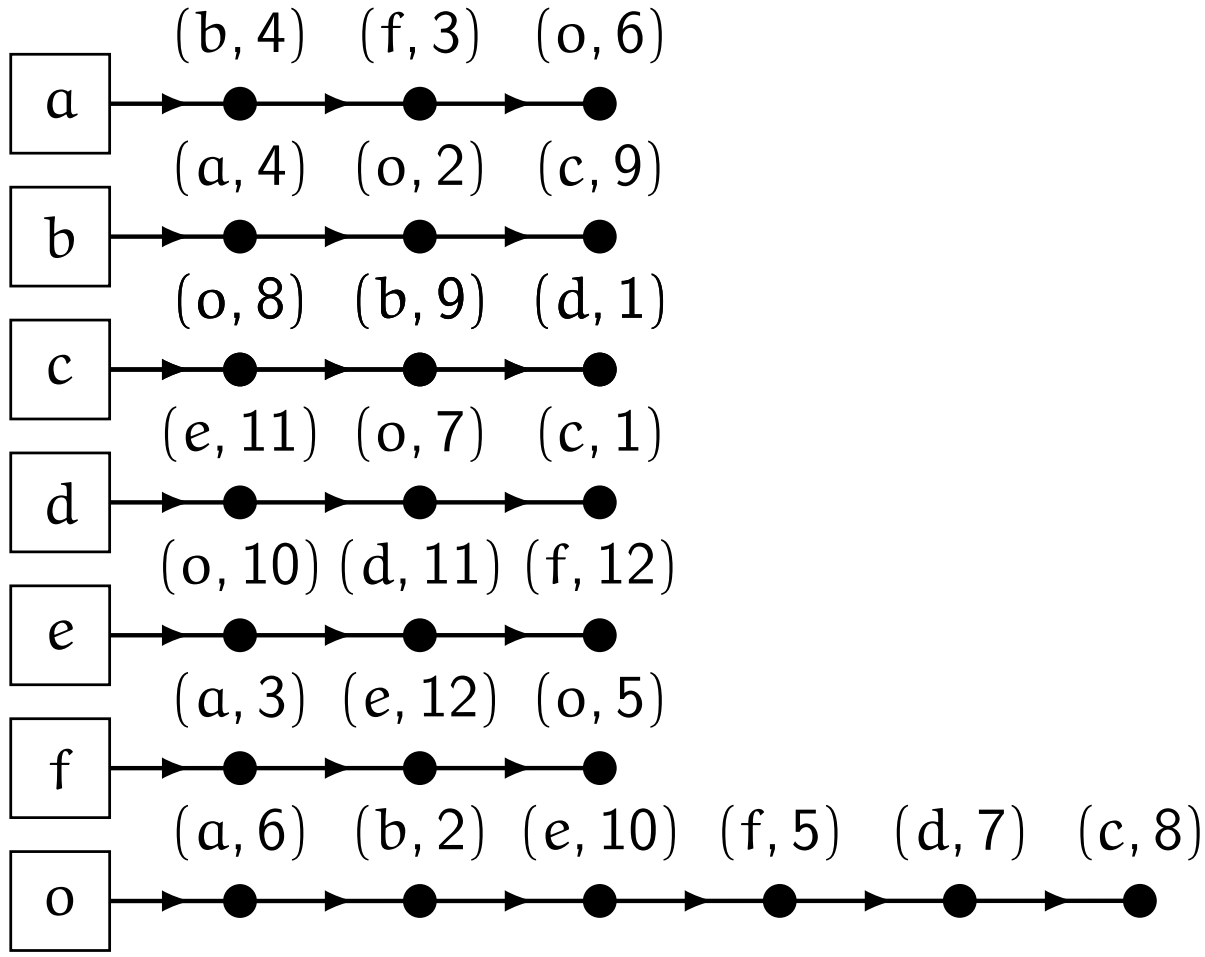
Demostración (cont.)

Sea T el árbol que devuelve el algoritmo. Sea T^* un árbol generador de mínimo peso que comparte con T las primeras k aristas escogidas por el algoritmo con k máximo posible. Supongamos, por el absurdo, que $T \neq T^*$ y sea e la primera arista elegida para T que no está presente en T^*. Sea S el conjunto de vértices de H justo antes de agregar la arista e. Entonces e tiene un extremo $u \in S$ y otro extremo $v \in V(G) - S$. El único camino que une u con v en T^* tiene al menos una arista e' con un extremo en S y el otro en $V(G) - S$. Como $T^* - e'$ tiene dos componentes, una a la que pertenece u y otra a la que pertenece v, entonces $T^* - e' + e$ es un árbol generador de G. Además, como e es una arista de peso mínimo uniendo S con $V(G) - S$, $w(e) \leqslant w(e')$. Luego $T^* - e' + e$ es un árbol generador con peso a lo sumo el peso de T^* y que coincide con T en una arista inicial más que T^*, lo que contradice la elección de T^*. Luego, $T = T^*$ y, en particular, T es un árbol generador de peso mínimo. $\square$

Algoritmo de Prim

- Consideramos que el grafo G está representado mediante listas de adyacencia; es decir, cada vértice cuenta con una lista enlazada cuyos elementos indican sus adyacencias.





- Dadas las aristas del grafo, es posible construir tal representación en tiempo lineal. (Bajo hipótesis razonables.)

Algoritmo de Prim

- Para el análisis de complejidad asumamos nuevamente que G es simple y sea n la cantidad de vértices de G.
- La implementación más simple del algoritmo de Prim consiste en almacenar en cada vértice fuera de H una de las aristas de peso mínimo que lo une con un vértice de H (en caso que exista una arista así).
- En cada una de las $n - 1$ iteraciones, debemos:
  1. Recorrer los vértices fuera de H para determinar cuál es el vértice v fuera de H que está unido a H por una arista e de peso mínimo.
  2. Agregar la arista e a H y actualizar la información en cada vértice fuera de H.

  De modo que cada iteración toma tiempo $O(n)$.
- Con esta implementación, el algoritmo se ejecuta en tiempo $O(n^2)$.

Algoritmo de Prim

- Existen otras formas de implementar este algoritmo.
- Se obtienen mediante representaciones eficientes de una estructura de datos conocida como cola de prioridad, que permite extraer el elemento de menor prioridad y decrementar la prioridad de uno cualquiera de los elementos.
- Una opción simple es utilizar una parva binaria (binary heap). Con esta representación, resulta que las operaciones de hallar la arista a agregar y actualizar la información en un vértice toman $O(\log n)$ tiempo cada una.
- Luego, el tiempo necesario para hallar todas las aristas del árbol generador es $O(n \log n)$ y el tiempo total para actualizar la información en los vértices es $O(m \log n)$.
- Esto nos da un tiempo total de ejecución $O(m \log n)$, asintóticamente igual que para Kruskal (sin las aristas previamente ordenadas).
- Usando una representación más sofisticada, llamada parva de Fibonacci, se puede implementar en tiempo $O(m + n \log n)$.

Caminos mínimos

En un mapa de rutas, ¿cómo podemos encontrar los caminos más cortos de una cierta localidad a un cierto conjunto de otras localidades? O, en término de teoría de grafos, como podemos encontrar caminos de longitud mínima de un vértice a otros vértices.

El algoritmo de Dijkstra resuelve este problema gracias a la siguiente observación: Si v es un vértice intermedio de un camino mínimo P que va de u a z entonces el subcamino de P que va de u a v también tiene que ser mínimo.

El algoritmo de Dijkstra avanza incrementalmente encontrando las distancias mínimas de un vértice u a todos los vértices z en orden creciente de distancia a u.

Algoritmo de Dijkstra

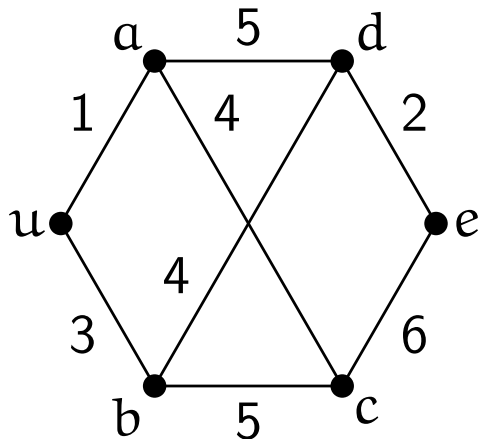
Algoritmo de Dijkstra

Entrada: Un grafo G simple con pesos no negativos en las aristas y un vértice inicial u. Sea $w(xy)$ el peso de la arista xy y $w(xy) = \infty$ si xy no es una arista del grafo.
Idea: Mantenemos el conjunto S de vértice para los que conocemos un camino mínimo desde u, incrementando S hasta que incluya todos los vértices. Para hacer esto, mantenemos una distancia tentativa $t(z)$ desde u para cada $z \notin S$ que es igual a la longitud del camino más corto de u a z hallado hasta el momento.
Inicialización: $S = \{u\}$, $t(u) = 0$ y $t(z) = w(uz)$ para todo $z \neq u$.
Iteración: Elegir un vértice v fuera de S tal que $t(v) = \text{mín}_{z \notin S}\, t(z)$ (desempatando arbitrariamente). Agregar v a S. Explorar las aristas desde v para actualizar las distancias tentativas: para cada arista vz con $z \notin S$, actualizar $t(z)$ para que se vuelva $\text{mín}\{t(z), t(v) + w(vz)\}$. La iteración continúa hasta que $S = V(G)$ o hasta que $t(z) = \infty$ para todo $z \notin S$.
Salida: $d_G(u, v) = t(v)$ para todos los vértices v de G.

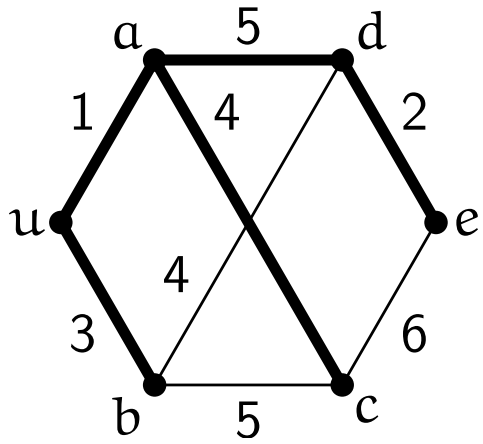
Algoritmo de Dijkstra

Ejemplo

Aplicamos el algoritmo de Dijkstra al siguiente grafo con vértice inicial u:



y obtenemos el siguiente árbol generador de caminos mínimos:



Algoritmo de Dijkstra

Teorema de Dijkstra (Dijkstra, 1959)

Dado un grafo G y un vértice $u \in V(G)$, el algoritmo de Dijkstra computa correctamente $d_G(u, z)$ para todo $z \in V(G)$.

Demostración.

Vamos a probar que en cada iteración valen las siguientes condiciones:

1. para todo $z \in S$, $t(z) = d_G(u, z)$; y
2. para todo $z \notin S$, $t(z)$ es la longitud de un camino más corto de u a z que llega a z directamente desde S o $t(z) = \infty$ si no existe tal camino.

Usamos inducción en el número de iteraciones.

En la inicialización, $S = \{u\}$ y se cumplen: 1 $d_G(u, u) = t(u) = 0$ y 2 el camino más corto que une u con z y llega a z directamente desde S mide $t(z) = w(uz)$ (que es infinito cuando no existe la arista uz). (...)

Algoritmo de Dijkstra

Demostración (cont.)

Para el paso inductivo, supongamos que 1 y 2 valen para S. Sea v el vértice que elige el algoritmo para generar el nuevo conjunto $S' = S \cup \{v\}$ (es decir, v es un vértice fuera de S que minimiza $t(v)$) y sea t' la nueva función t al finalizar esta iteración. Debemos probar que:

1' *Para todo* $z \in S' = S \cup \{v\}$, $t'(z) = d_G(u, z)$.
Como 1 vale para S, alcanza con probar que $t'(v)$ es igual a $d_G(u, v)$. Por el algoritmo sabemos que $t'(v) = t(v)$.
Claramente todo camino de u a v deja en algún momento S para poder llegar a v. Por una parte, como 2 vale para S, $t(v)$ es la longitud mínima entre los caminos de u a v que llegan a v directamente desde S. Y también porque 2 vale para S y por la elección de v se puede garantizar que cualquier camino que sale de S y pasa por otro vértice antes de pasar por v tiene longitud al menos $t(v)$. Concluimos que $d_G(u, v) = t(v) = t'(v)$. (...)

Algoritmo de Dijkstra

Demostración (cont.)

2' *Para todo* $z \notin S' = S \cup \{v\}$, $t'(z)$ *es el camino más corto de* u *a* z *que llega a* z *directamente desde* S' *o* $t'(z) = \infty$ *si no existe tal camino.*
Sea $z \notin S' = S \cup \{v\}$. Por la hipótesis inductiva, la longitud de un camino más corto de u a z que llega a z directamente desde S es $t(z)$ o $t(z) = \infty$ si no existe tal camino. Cuando agregamos v a S, debemos considerar también los caminos que llegan a z directamente desde v. Como ya comprobamos que $d_G(u, v) = t(v)$ entonces estos caminos tienen longitud $t(v) + w(vz)$. Justamente el algoritmo compara $t(z)$ con $t(v) + w(vz)$ para computar $t'(z)$ que, por lo tanto, coincide con la longitud del camino más corto que llega a z directamente desde S' o ∞ si no existe tal camino. □

Algoritmo de Dijkstra

- Desde el punto de vista del tiempo de ejecución, el análisis es similar al del algoritmo de Prim y depende de la representación utilizada de cola de prioridad.
- Manteniendo en cada vértice la distancia mínima hasta el momento, se consigue una implementación de tiempo $O(n^2)$.
- Utilizando una representación con parvas binarias se obtiene una implementación alternativa de complejidad temporal $O(m \log n)$.
- Utilizando parvas de Fibonacci es posible implementar el algoritmo para que su ejecución tome tiempo $O(m + n \log n)$.

Breadth-first search (BFS)

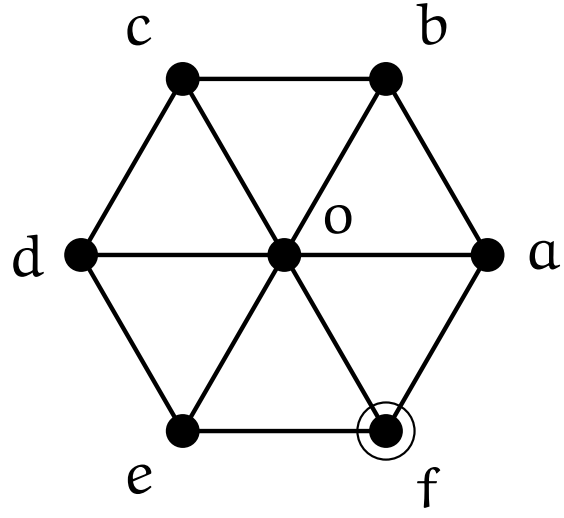
Es un caso especial del algoritmo de Dijkstra para grafos sin pesos.

Breadth-first search (BFS)

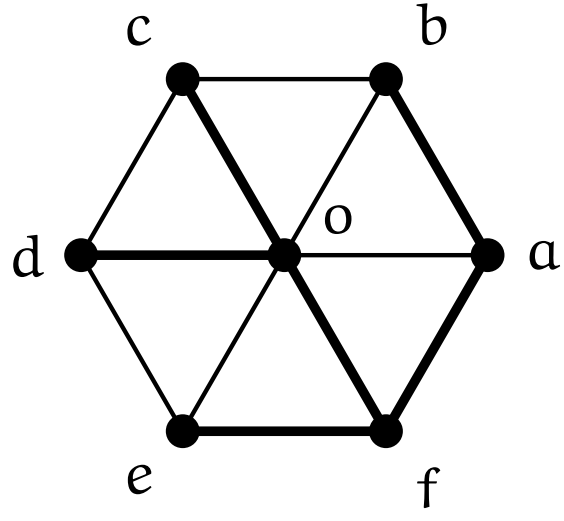
Entrada: Un grafo (no pesado) y un vértice inicial u.
Idea: Mantenemos un conjunto R de vértices que fueron alcanzados pero no explorados y un conjunto S de vértices que fueron explorados. El conjunto R se comporta como una cola (el primer elemento que entra es el primero en salir) de manera que los primeros vértices que son alcanzados son los primeros explorados.
Inicialización: $R = \{u\}$, $S = \emptyset$, $d_G(u, u) = 0$.
Iteración: Mientras que $R \neq \emptyset$, explorar el primer vértice v de la cola R: es decir, encolar en R los vecinos de v que no están en $S \cup R$ asignándoles la distancia $d_G(u, v) + 1$, y luego quitar v de R e insertarlo en S.
Salida: Al final de la ejecución se calcularon las distancias de u a todos los demás vértices de G.

Breadth-first search

Si aplicamos BFS sobre el siguiente grafo desde el vértice f:



podemos obtener el siguiente árbol de distancias mínimas desde f:



Breadth-first search

- Como cada vértice se encola y desencola a lo sumo una vez, estas operaciones llevan en total tiempo $O(n)$.
- Y como la vecindad de cada vértice se recorre solo una vez, estas operaciones se pueden completar en tiempo $O(m + n)$.
- Esto nos da una complejidad temporal total para el BFS de $O(m + n)$, es decir, lineal.
- En particular, es posible hallar la excentricidad $\epsilon(u)$ de un vértice u dado en tiempo lineal.
- Aplicándolo en cada vértice, permite hallar el diámetro y el radio del grafo en tiempo $O(n(m + n))$.