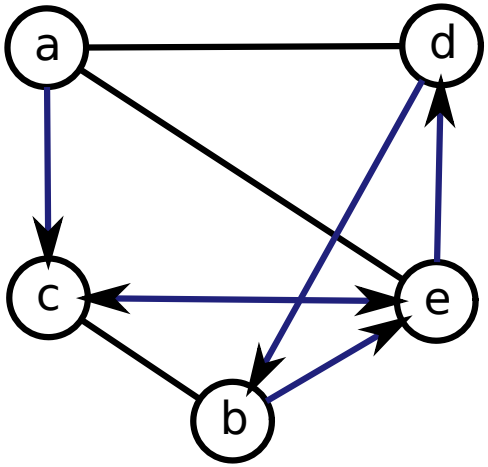
Introdución a la Teoría de Grafos

José Luis Bravo Trinidad

Las Matemáticas en la Enseñanza Secundaria
Curso 2018-2019

Caminos en un grafo

Camino, circuito y ciclo



acedbec
Camino de longitud 6

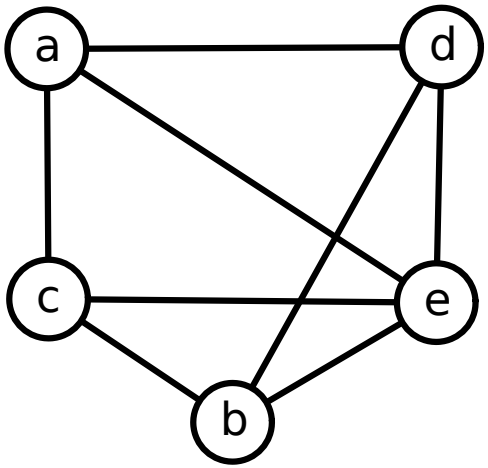
Camino: Sucesión finita de vértices de modo que cada dos consecutivos son adyacentes.

El número de aristas dentro de un camino es su **longitud**.

Se denominan **extremos** del camino al primer y último vértice.

Dos vértices u, v están **conectados** si son los extremos de algún camino.

Camino, circuito y ciclo



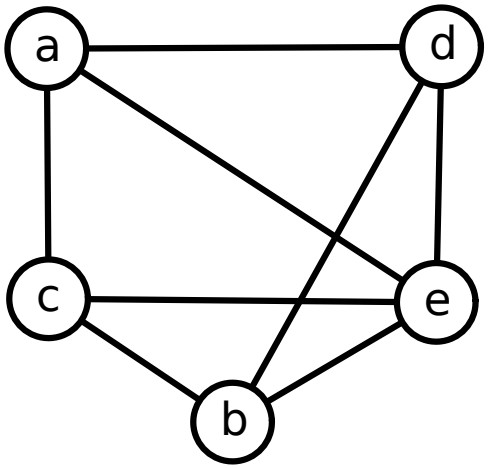
Un camino es **cerrado** cuando los extremos coinciden.

Un camino es **simple** cuando no se repite ningún vértice.

Un **ciclo** es un camino cerrado en el que no se repite ningún otro vértice.

Un **circuito** es un camino cerrado en el que no se repite ninguna arista.

Camino, circuito y ciclo

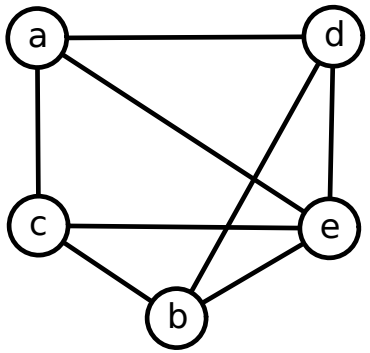


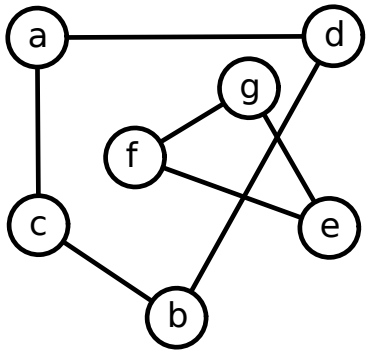
- Camino cerrado: *acbeca*
- Camino simple: *acbde*
- Ciclo: *acbea*
- Circuito: *edaebce*

Teorema

Si en un grafo G existe un camino que conecta dos vértices distintos, entonces existe un camino simple que conecta los mismos vértices.

Grafo conexo y componentes conexas





Decimos que un grafo es **conexo** si todo par de vértices están conectados.

Si un grafo no es conexo se dice **disconexo**.

"Estar conectado" es una relación de equivalencia entre los vértices del grafo. Se denomina **componente conexa** cada una de las clases de equivalencia por esta relación.

Conexión a partir de la matriz de adyacencia

Teorema

Sea G un grafo y M_G su matriz de adyacencia. La posición (i, j) de la matriz M_G^k es el número de caminos de longitud k que comienzan en el vértice i y terminan en el j.

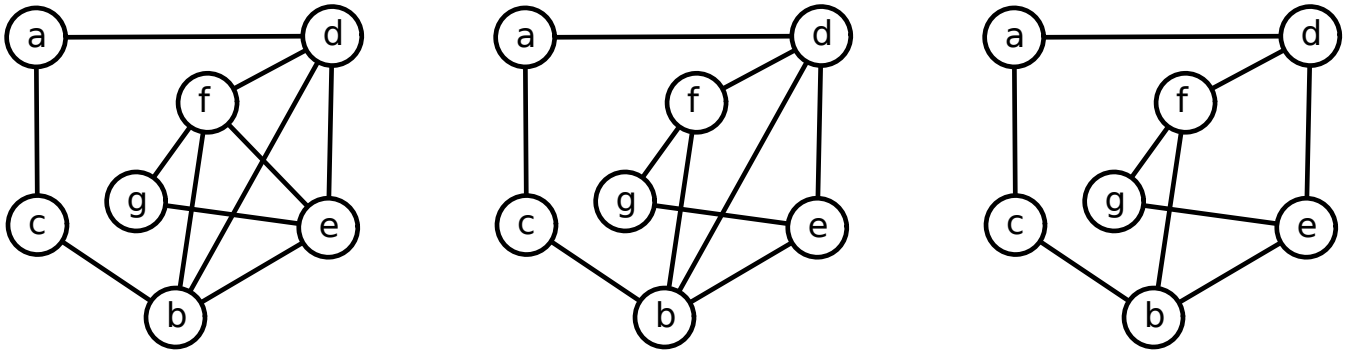
Definimos la **matriz de conexión** como

$$M_C = M_G + M_G^2 + \ldots + M_G^{n-1}$$

donde n es el número de vértices.

Entonces el grafo es conexo sí y sólo si M_C no tiene ceros.

Grafo euleriano



Un grafo es **euleriano** cuando existe un circuito que recorre todas las aristas.

Un camino es **euleriano** cuando recorre todas las aristas sin repetir ninguna.

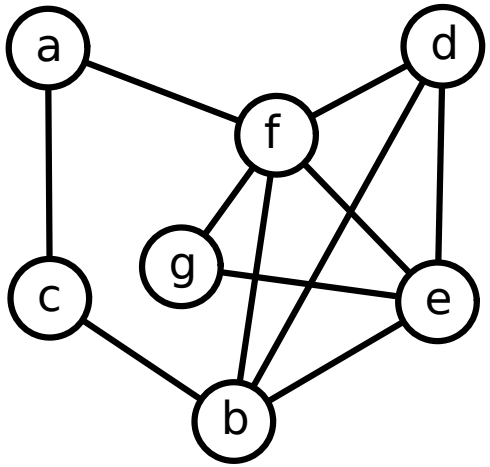
Teorema

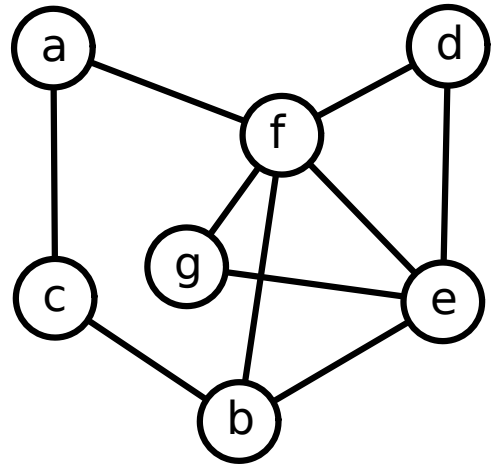
Un grafo conexo es euleriano si y solo si todo vértice tiene grado par.

Algoritmo de Hierholzer.

1. Partimos de un vértice v_0 cualquiera. Tomamos $v = v_0$.
2. Elegimos un vértice u adyacente a v y eliminamos la arista $\{u, v\}$. Tomamos $v = u$.
3. Repetimos el paso 2 hasta volver a v_0. Esto nos genera un circuito C.
4. Si no quedan aristas, hemos terminado. Si quedan aristas tomamos v un vértice de C de grado positivo.
5. Repetimos 2 y 3 y obtenemos un nuevo circuito C_1. Unimos C_1 a C.
6. Si no quedan aristas hemos terminado. En caso contrario, vamos a 4.

Grafos hamiltonianos





Un grafo es **hamiltoniano** cuando existe un ciclo que recorre todos los vértices.

Teorema (Ore 1960)

Sea G un grafo con $n \geq 3$ vértices. Si para cada par de vértices no adyacentes se verifica que $gr(v) + gr(w) \geq n$, entonces G es hamiltoniano.

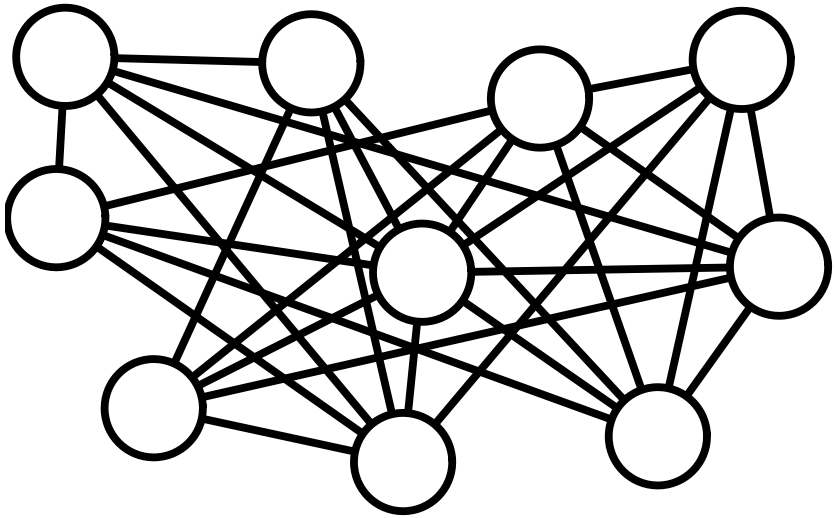
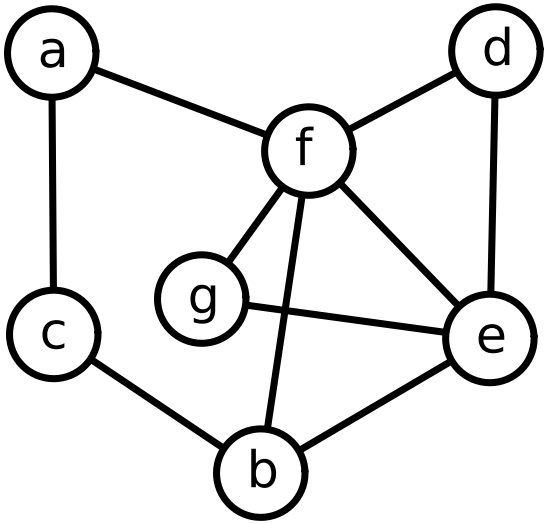
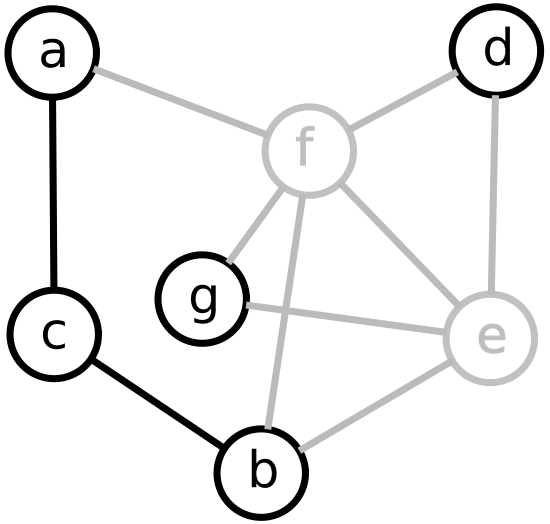


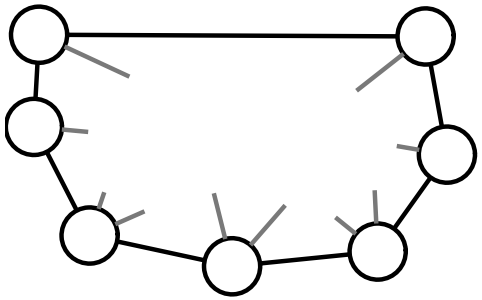
Figure: El grafo es hamiltoniano por el Teorema de Ore

Teorema

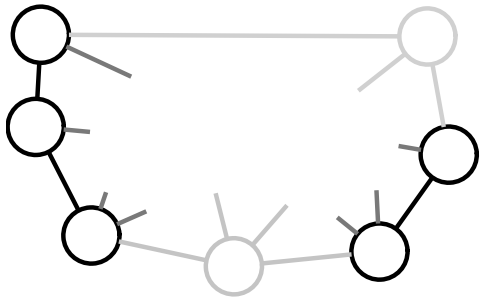
Sea $G = (V, E)$ tal que $\#V \geq 3$. Si G es hamiltoniano, entonces para cada subconjunto $U \subset V$, el subgrafo completo de G de vértices $V \setminus U$ tiene a lo sumo $\#U$ componentes.



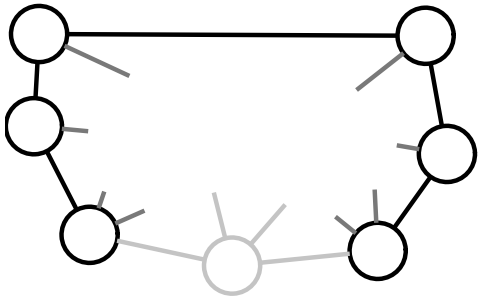




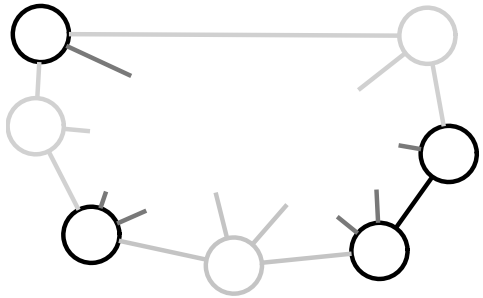
Ciclo hamiltoniano



Eliminamos 2 vértices.

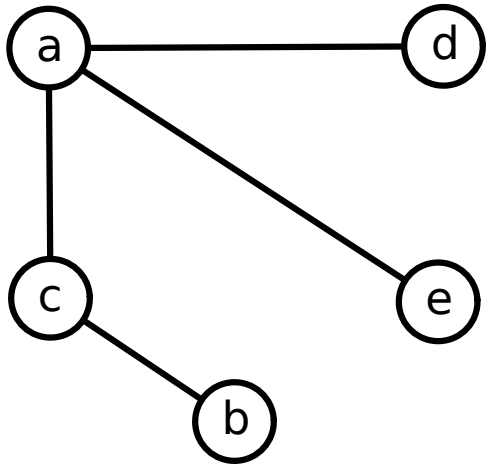


Eliminamos 1 vértice.



Eliminamos 3 vértices.

Árboles y grafos dirigidos



Decimos que un grafo es un **árbol** si es conexo y no tiene ciclos.

Si un grafo no es conexo ni tiene ciclos decimos que es un **bosque**.

Teorema

Un grafo T es un árbol si y solo si cada dos vértices distintos de T se conectan por un único camino simple.

Teorema

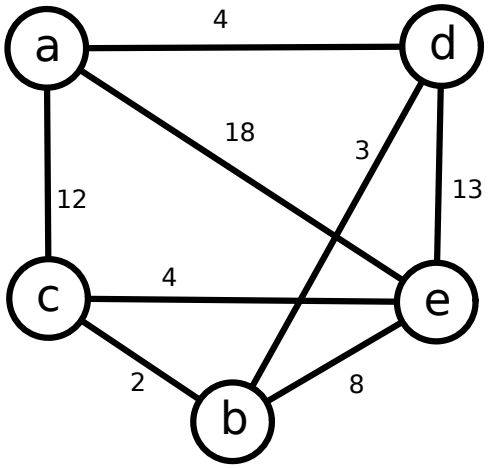
Un grafo conexo de n vértices es un árbol si y sólo si tiene $n-1$ aristas.

Decimos que un vértice v es **terminal** si $\text{gr}(v) = 1$.

Teorema

Un árbol de $n \geq 2$ vértices tiene al menos dos vértices terminales.

Grafo etiquetado



Se denomina **grafo etiquetado** a un grafo $G = (V, E)$ con una función $p\colon E \to \mathbb{R}$.

Denominamos **peso**, **etiqueta** o **distancia** de $\{u, v\} \in E$ al valor $p(\{u, v\})$.

Si $\{u, v\} \notin E$, podemos considerar $p(u, v) = +\infty$.

Definimos **longitud** de un camino como la suma de las etiquetas de las aristas.

Distancia, radio y diámetro de un grafo

Dados dos vértices u, v, se define **distancia** entre dos vértices u, v, $d(u, v)$, como el mínimo de las longitudes de los caminos que unen u y v.

Se define la **excentricidad** de un vértice v como el máximo de la distancias de v a los vértices del grafo.

$$e(v) = \max\{d(u, v) \colon u \in V(G)\}.$$

Se define el **radio** de un grafo como el mínimo de las excentricidades y el **diámetro** como el máximo.

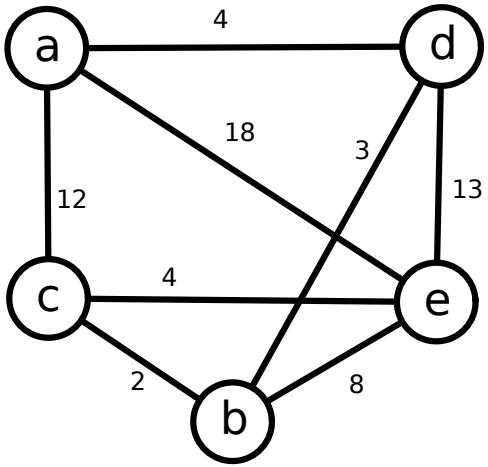
$$r(G) = \min\{e(v) \colon v \in V(G)\}, \quad d(G) = \max\{e(v) \colon v \in V(G)\}.$$

Algoritmo de Dijkstra

Teorema (Algoritmo de Dijkstra)

La distancia entre dos vértices x e y se puede obtener por el siguiente procedimiento:

1. *Generamos $T = V$ y $L(x) = 0$, $L(v) = +\infty$ para todo $v \in V$.*
2. *Calculamos $v \in T$ tal que $L(v) = \min\{L(w) : w \in T\}$.*
3. *Si $v = y$, entonces hemos terminado y la distancia es $L(y)$.*
4. *En caso contrario, hacemos $T \to T\backslash\{v\}$, $L(w) \to \min\{L(w), L(v) + d(v, w)\}$ y volvemos a 2.*



Distancia mínima de a a e.

$T = (a, b, c, d, e)$

$L = (0, \infty, \infty, \infty, \infty)$

Tomamos $v = a$:

$T = (b, c, d, e)$

$L = (\infty, 12, 4, 18)$

Tomamos $v = d$:

$T = (b, c, e)$, $L = (7, 12, 17)$

Tomamos $v = b$:

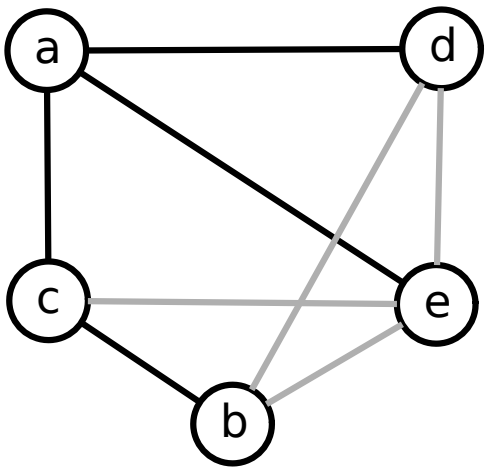
$T = (c, e)$, $L = (9, 16)$

Tomamos $v = c$:

$T = (e)$, $L = (13)$

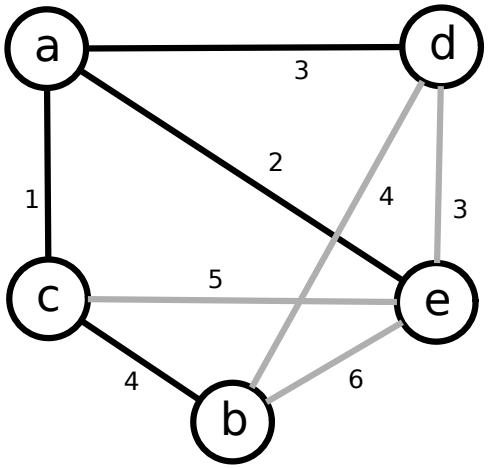
La distancia es 13.

Árbol recubridor



Dado un grafo conexo G, se denomina **árbol recubridor** o **generador** de G a cualquier árbol que sea subgrafo de G y contenga todos los vértices.

Árbol recubridor minimal



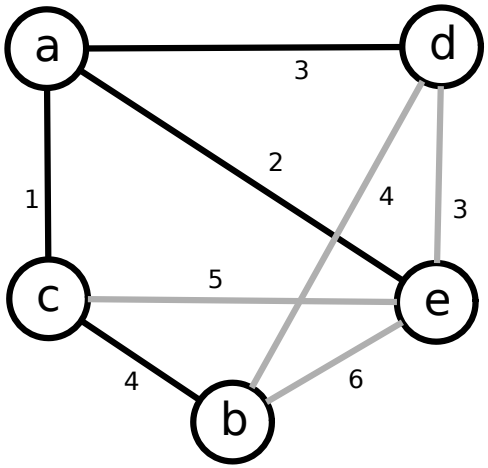
Dado un grafo etiquetado, se denomina **Árbol recubridor minimal** a cualquier árbol recubridor tal que la suma de los pesos de las aristas sea menor o igual a la de cualquier otro árbol recubridor.

Algoritmo de Kruskal

Partimos de un grafo conexo y etiquetado G.

1. Se considera un bosque B, formado en un principio por el conjunto de todos los vértices y ninguna arista.
2. Se toma la arista del grafo de menor peso.
3. Si la arista conecta dos árboles distintos del bosque B, se añade al bosque y se elimina de G.
4. Si la arista no conecta dos árboles distintos, se elimina de G.
5. Si queda alguna arista en G, se vuelve a 2

Algoritmo de Kruskal



En B sólo indicaremos las aristas en cada paso.

$B = \{\}$

$B = \{\{a, c\}\}$

$B = \{\{a, c\}, \{a, e\}\}$

$B = \{\{a, c\}, \{a, e\}, \{a, d\}\}$

Descartamos $\{d, e\}$

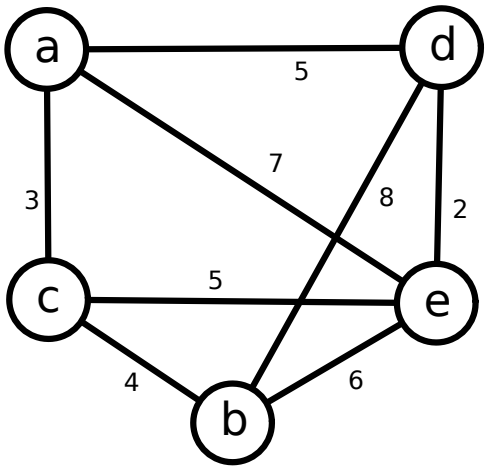
$B = \{\{a, c\}, \{a, e\}, \{a, d\}, \{c, b\}\}$

Algoritmo de Prim

Partimos de un grafo conexo $G = (V, E)$.

1. Definimos un grafo T formado incialmente por uno de los vértices.
2. De todos los vértices adyacentes a T que no estén en T, añadimos aquel cuya distancia sea mínima y la arista que lo conecta con T de dicha distancia.
3. Repetimos el paso anterior hasta agotar todos los vértices.

Algoritmo de Prim



En T indicaremos las aristas en cada paso.
Partimos del vértice a como único vértice de T.
$T = \{\{a, c\}\}$
$T = \{\{a, c\}, \{c, b\}\}$
$T = \{\{a, c\}, \{c, b\}\}$
$T = \{\{a, c\}, \{c, b\}, \{a, d\}\}$
$T = \{\{a, c\}, \{c, b\}, \{a, d\}, \{d, e\}\}$

Definición

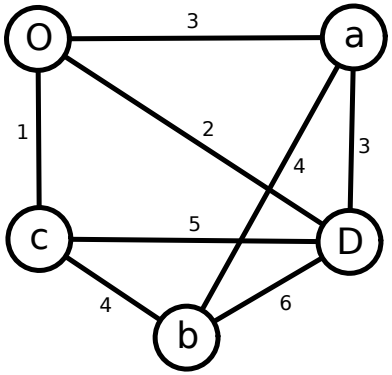
Dado un grafo etiquetado G, un **flujo entre dos vértices**, el origen O y el destivo D, es un subgrafo dirigido etiquetado F tal que:

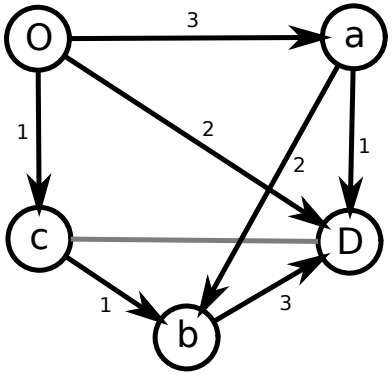
1. O, D son vértices de F.
2. Los pesos de las aristas de F son menores o iguales que los de G.
3. Para todo vértice v de F distinto de O, D la suma de los pesos de las aristas que salen es igual al de las aristas que entran.

Se define la **cantidad total de flujo** como la suma de los pesos de las aristas que salen de O menos la suma de los pesos de las aristas que llegan.

El **problema del flujo máximo** trata de obtener un flujo tal que la cantidad total de flujo sea máxima.

Ejemplo flujo





$$\begin{array}{c} \\ O \\ a \\ b \\ c \\ D \end{array} \begin{array}{c} \begin{array}{ccccc} O & a & b & c & D \end{array} \\ \begin{pmatrix} 0 & 3 & 0 & 1 & 2 \\ -3 & 0 & 2 & 0 & 1 \\ 0 & -2 & 0 & -1 & 3 \\ -1 & 0 & 1 & 0 & 0 \\ -2 & -1 & -3 & 0 & 0 \end{pmatrix} \end{array}$$

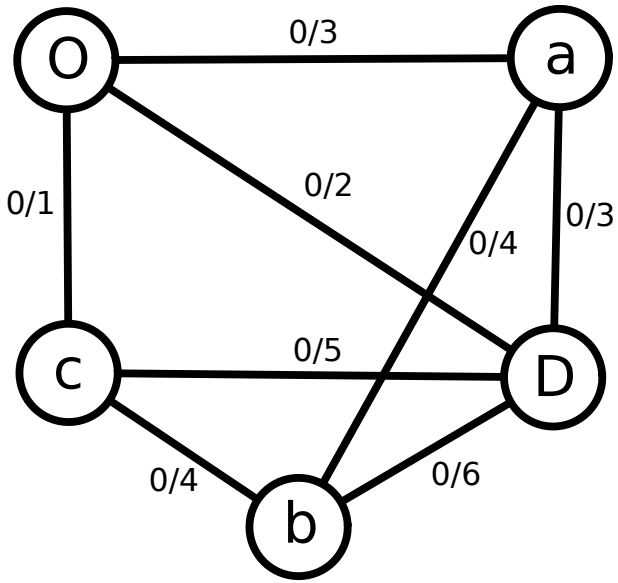
Matriz del flujo

Algoritmo de Ford-Fulkenson

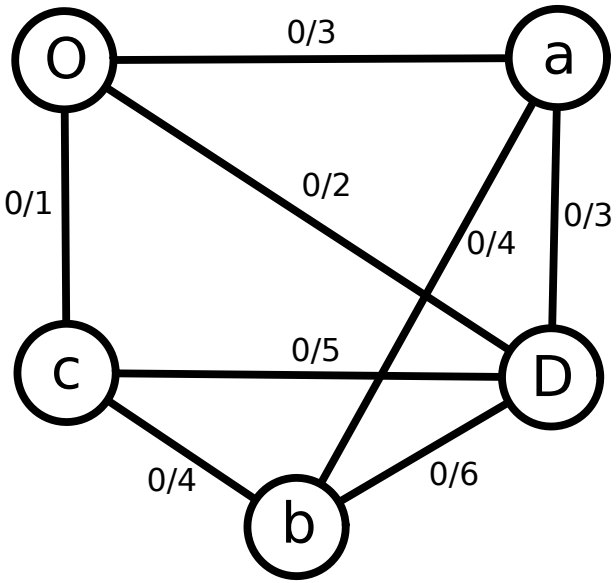
La **capacidad residual** de dos nodos $c(u, v)$ es la diferencia entre el peso de la arista $\{u, v\}$ de G y el flujo $f(u, v)$.
Una **trayectoria de aumento** como un camino simple de O a D tal que para cada dos vértices u, v del camino, tenemos $c(u, v) > 0$.

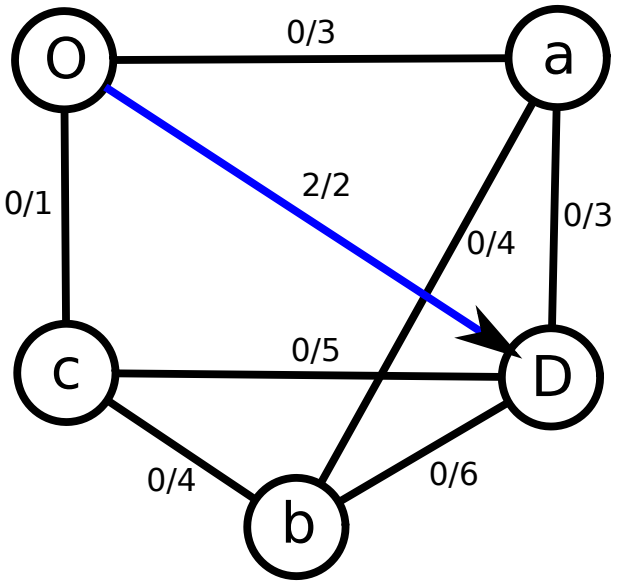
1. Comenzamos con un flujo F tal que $F(u, v) = 0$ para todo par de vértices u, v.
2. Buscamos una trayectoria de aumento. Sea $c_t > 0$ el mínimo de $c(u, v)$ donde (u, v) recorre las aristas de la trayectoria.
3. Para cada par de vértices (u, v) de la trayectoria, hacemos $F(u, v) = F(u, v) + c_t$ y $F(v, u) = F(v, u) - c_t$.
4. Volvemos a 2 hasta que no encontremos más trayectorias de aumento.

Algoritmo de Ford-Fulkenson

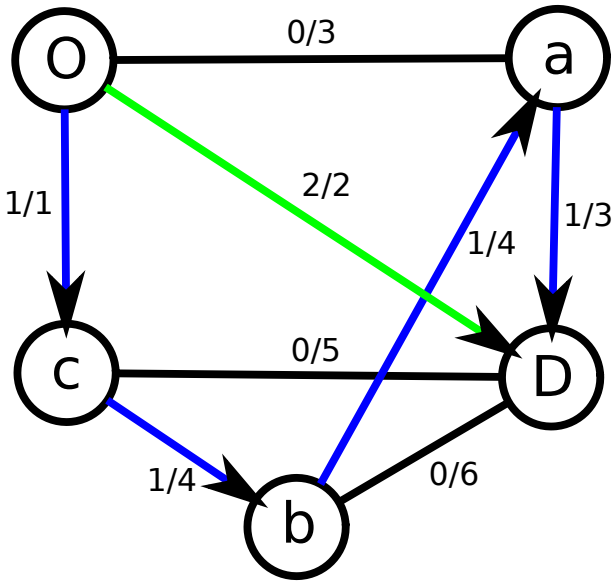


Algoritmo de Ford-Fulkenson

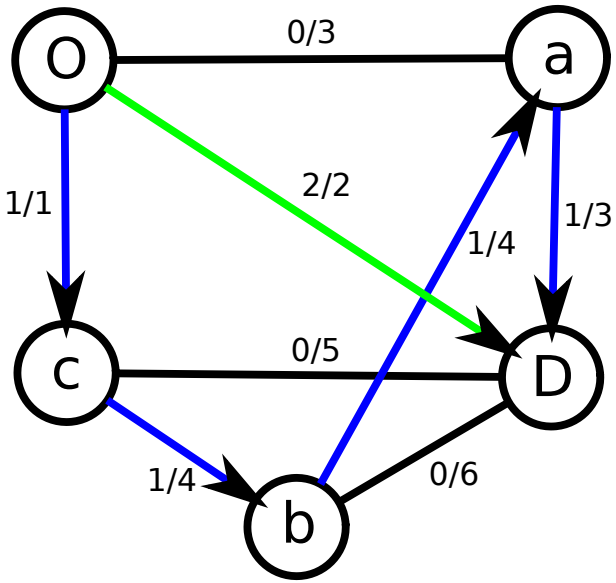


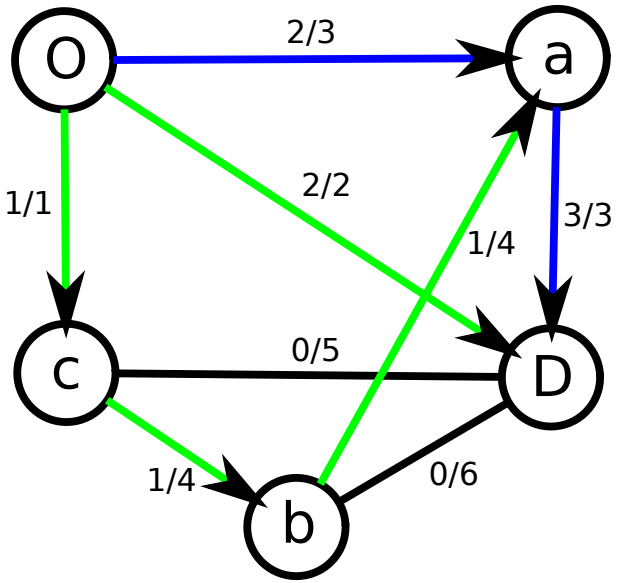


Algoritmo de Ford-Fulkenson

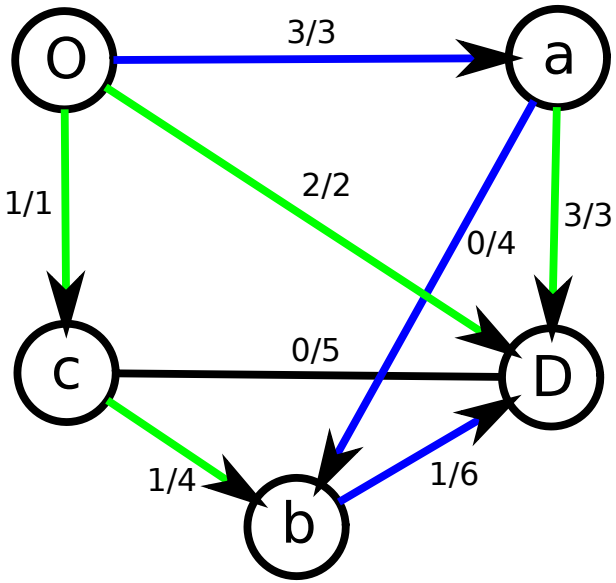


Algoritmo de Ford-Fulkenson





Algoritmo de Ford-Fulkenson



Algoritmo de Ford-Fulkenson