Optimización Combinatoria y en Redes

- Problema de flujo de coste mínimo.
- Problema de flujo máximo.
- Problema de la mochila.
- Problema de emparejamiento.
- Árbol de expansión de coste mínimo.
- Problema del viajante.

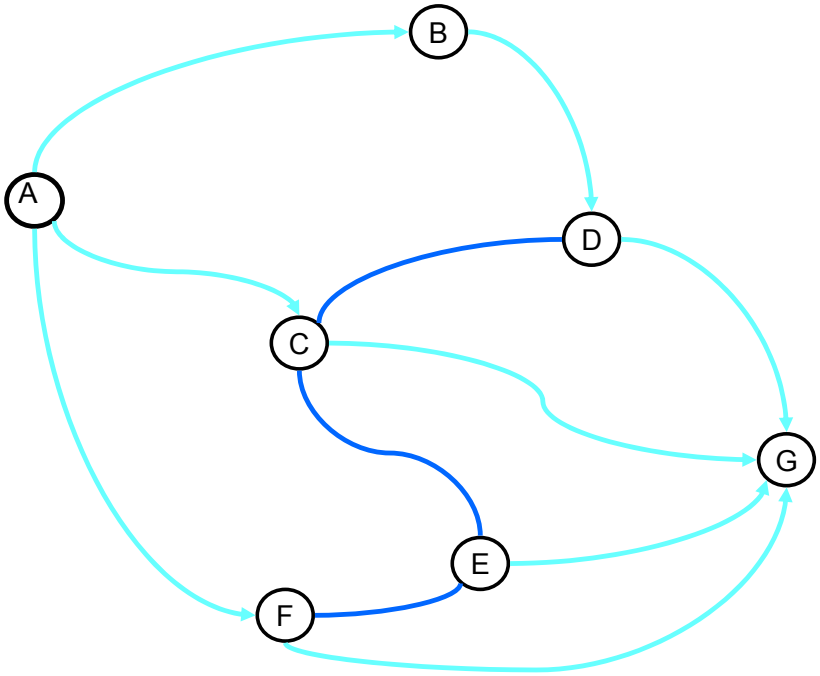
Problema de flujo de coste mínimo

Hay muchos problemas que, a pesar de admitir modelos de **programación entera**, tienen estructuras especiales $\Rightarrow$ recomendable no usar herramientas estándar.

- Problemas de flujos en redes.
  Variables $\leftrightarrow$ flujos en arcos de una red. Las herramientas de PL nos proporcionarán soluciones enteras.

- Problemas combinatorios.
  Modelos enormes pero muy estructurados.

Optimización en Redes: Terminología



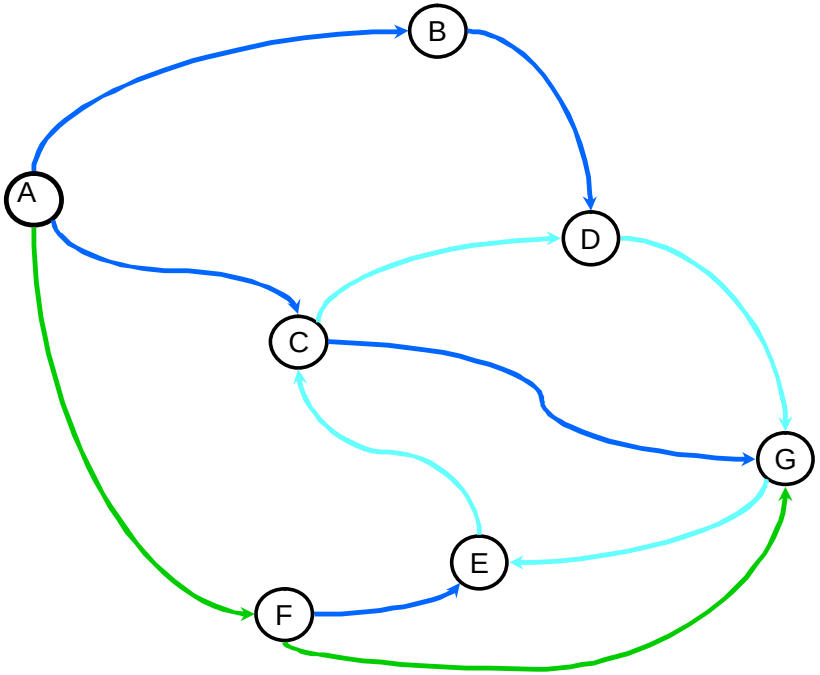
→ Arcos:
Grafo oriantado o red

— Aristas:
Grafo no orientado

— Grafos mixtos

$$G = (N, A) \quad o \quad G = (V, A)$$

Optimización en Redes: Terminología



Un camino es una sucesión de arcos adyacentes que parten de un nodo (A) y llegan a otro (G), y un ciclo es un camino de un nodo a él mismo. Un grafo **conexo** y sin ciclos se denomina **árbol**.

Optimización en Redes: Elementos básicos

- Variables x_{ij} asociadas a los arcos: $x_{ij} =$ flujo en (i, j).
- Costes $c_{ij} =$ coste por unidad de flujo en (i, j).
- Capacidades $l_{ij}/u_{ij} =$ flujo mínimo/máximo permitido en (i, j).
- Restriciones de conservación de flujo; para cada nodo i:

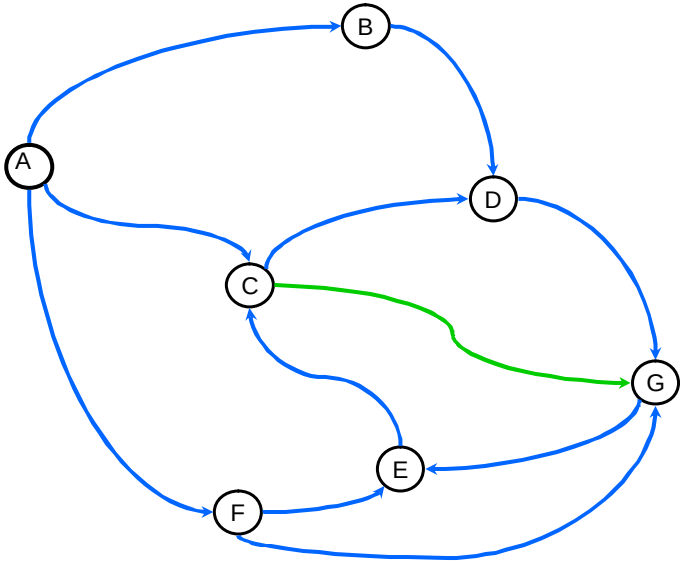
$$\overbrace{\sum_{k \in N} x_{ik}}^{\text{flujo saliente}} - \overbrace{\sum_{j \in N} x_{ji}}^{\text{flujo entrante}} = b_i = \begin{cases} \text{producción}(i) & \text{si } i \text{ genera flujo,} \\ -\text{demanda}(i) & \text{si } i \text{ absorve flujo,} \\ 0 & \text{en otro caso.} \end{cases}$$

Flujos en redes: Programación Matemática

Flujo factible: viene dado por vector x tal que:

$$Ax = b$$
$$l \leq x \leq u$$

donde A es la **matriz de incidencia nodos-arcos**:



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

Flujos en redes: Programación Matemática

Factibilidad

Para que un problema de flujos en redes tenga buenas propiedades es necesario que sea **equilibrado**:

$$\textbf{oferta} = \sum_{i:b_i>0} b_i = -\sum_{i:b_i<0} b_i = \textbf{demanda}.$$

Si no, el problema puede ser infactible. Si no lo es, podemos equilibrar el problema añadiendo un **nodo artificial**.

Flujos en redes: Programación Matemática

Propiedad de integridad: Si todos los elementos de b, l y u son enteros, los vértices de $\begin{cases} Ax = b \\ l \leqslant x \leqslant u \end{cases}$ tienen componentes enteras.

$$\Downarrow$$

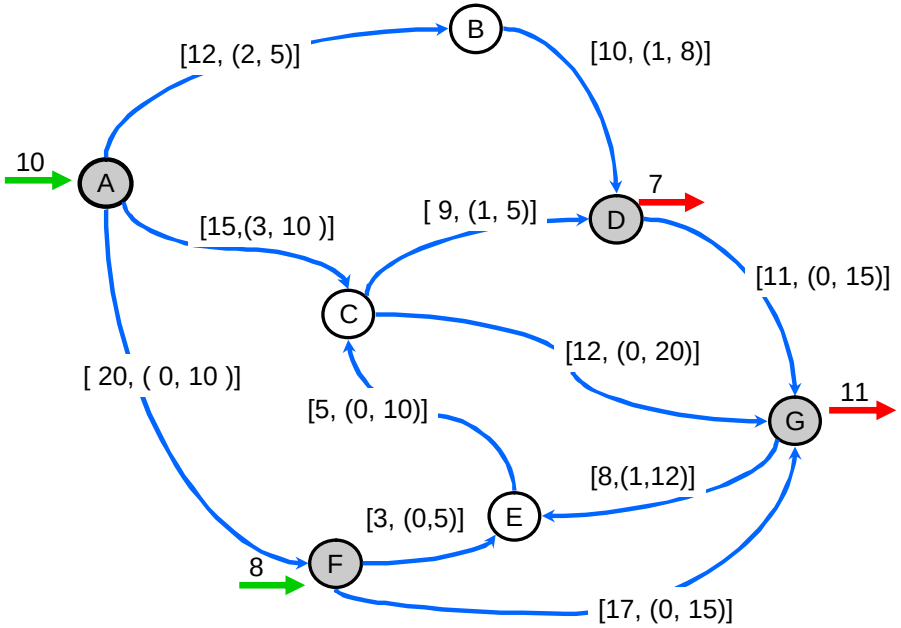
Podemos encontrar flujos enteros utilizando el método del Símplex!

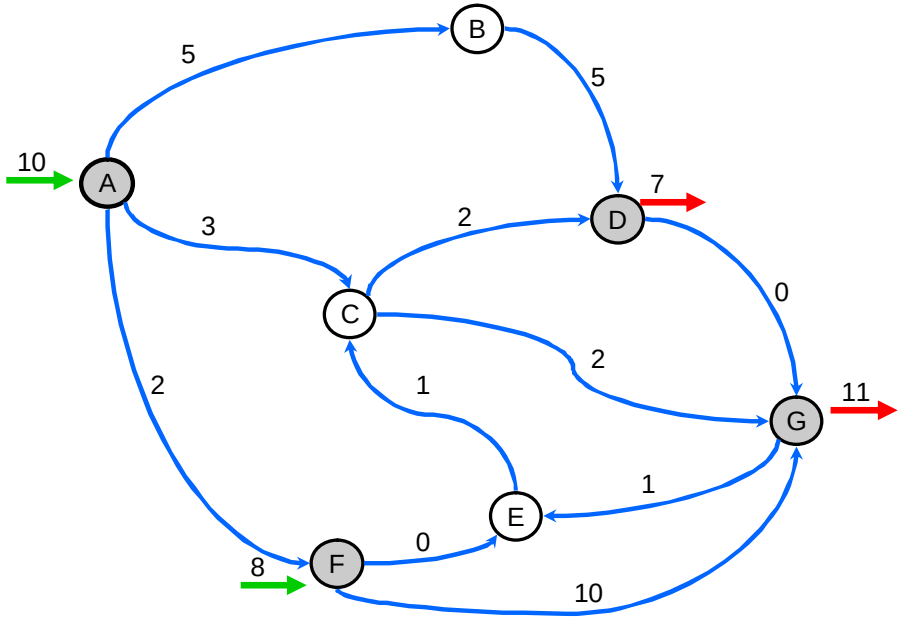
$$\Downarrow$$

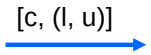
Adaptaciones del Símplex a problemas en redes.

Problema de flujo de coste mínimo

$$\min\{c'x | Ax = b; l \leqslant x \leqslant u\}$$





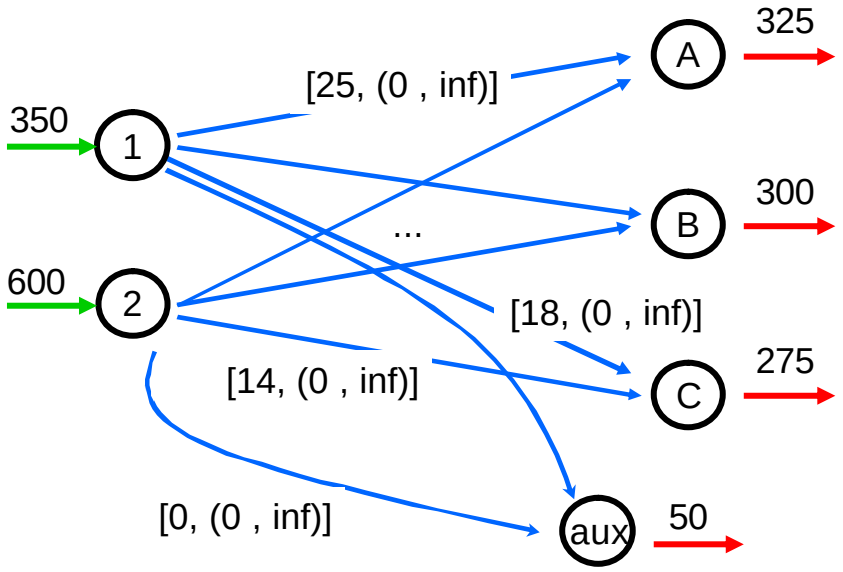


coste $= 420$

Flujo de coste mínimo: casos particulares

Podemos ver el **problema de transporte** como un problema de flujo de coste mínimo si consideramos la red adecuada:

25	17	18	350
25	18	14	600
325	300	275	



El problema de transporte

El Símplex permite resolver el problema de transporte, incluso con unidades indivisibles.

Si queremos que se requieran pocas iteraciones, necesitaremos una buena **solución inicial**. Para hallarla, podemos usar **métodos heurísticos**

Para inicializar el método del Símplex no nos sirve cualquier heurística. ¿Por qué?
Porque necesitamos una heurística que proporcione **siempre** soluciones **básicas**.

El problema de transporte

En el caso del problema de transporte, tenemos varias heurísticas con esa propiedad.
Veremos la más simple: **La esquina noroeste**

La veremos sobre el ejemplo:

		Destinos			
		A	B	C	producción
Orígenes	1	25	17	18	350
	2	25	18	14	600
demanda		325	300	275	

Problema de transporte: N-W corner

25 ?	17	18	0	350
25	18	14	0	600
325	300	275	50	

25 **325**	17 ?	18	0	~~350~~ 25
25	18	14	0	600
~~325~~	300	275	50	

Problema de transporte: N-W corner

25 325	17 **25**	18	0	~~25~~
25	18 ?	14	0	600
0	~~300~~ 275	275	50	

25 325	17 25	18	0	0
25	18 **275**	14 ?	0	~~600~~ 325
0	~~275~~	275	50	

Problema de transporte: N-W corner

25 325	17 25	18	0	0
25	18 275	14 **275**	0 ?	~~325~~ 50
0	0	~~275~~	50	

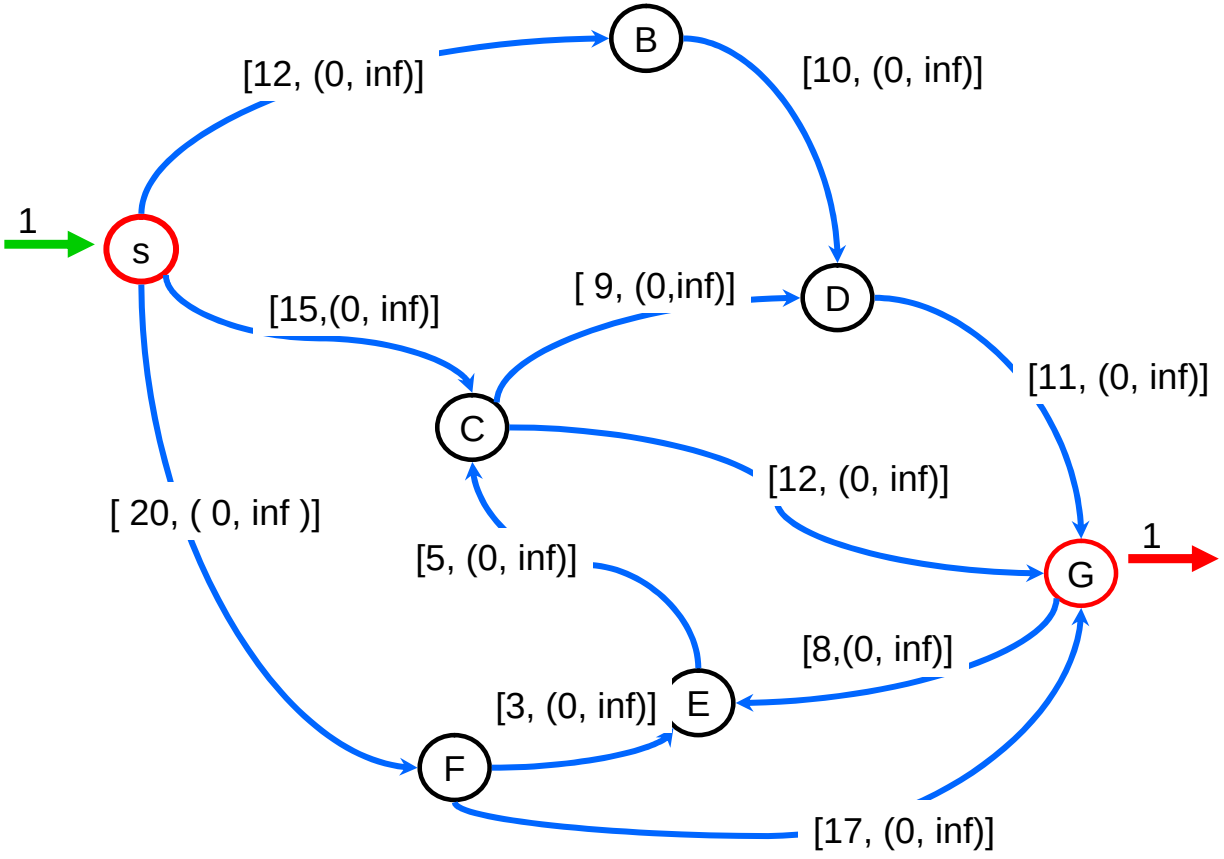
25 325	17 25	18	0	0
25	18 275	14 275	0 **50**	~~50~~
0	~~275~~	275	~~50~~	

Flujo de coste mínimo: casos particulares

Dada una red con costes en los arcos y con dos nodos distinguidos s y t el **problema de caminos mínimos** consiste en encontrar el camino de coste total mínimo entre todos los que unen s y t.

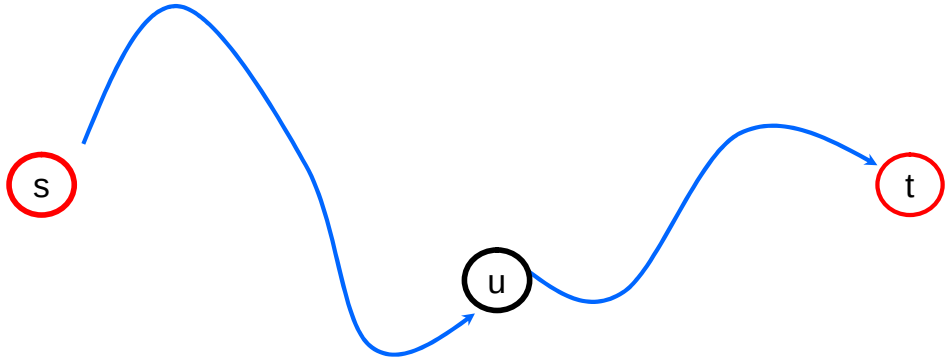
Esto es equivalente a resolver el problema de flujo de coste mínimo en la red con producción 1 en el nodo s, demanda 1 en el nodo t, capacidades mínimas igual a 0 y capacidades máximas infinitas.

Problema de caminos mínimos



Problema de caminos mínimos

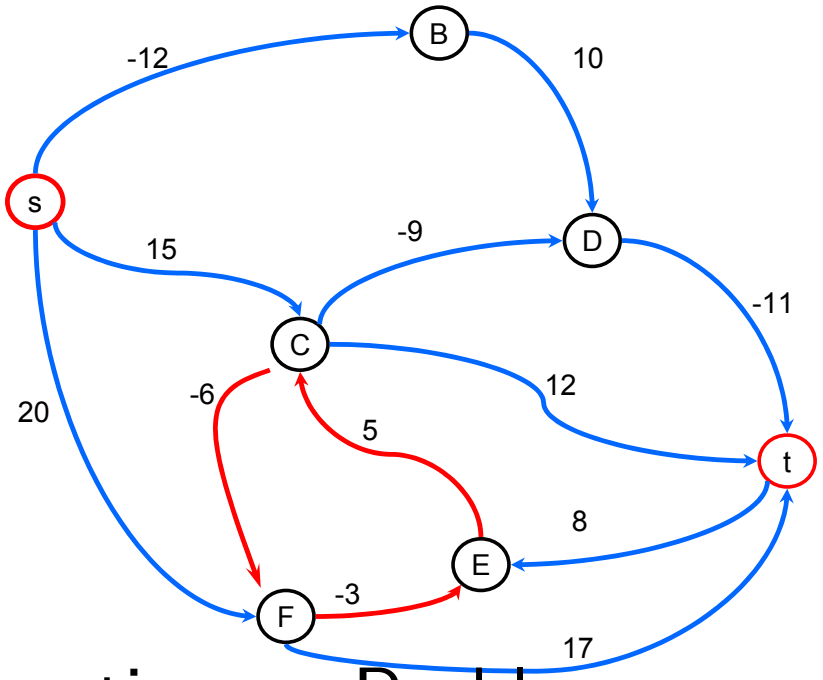
Un propiedad crucial de este problema es que las secciones de un camino mínimo son a su vez caminos mínimos:



Esta propiedad es la base de la mayoria de los algoritmos que se han diseñado para resolver este problema.

Problema de caminos mínimos: costes

Cuando una red tiene algunos arcos con costes negativos, debemos tener mucho cuidado:



Ciclo de coste negativo $\Rightarrow$ Problema no acotado

Problema de caminos mínimos

Árbol de caminos desde s al resto de nodos

- Algoritmo de Dijkstra (Greedy).
  Edsger W. Dijkstra (1930-2002), publicado en el 59
  Sólo para grafos con pesos/costes positivos.

  Coincide con el símplex para redes.

- Algoritmo de Bellman-Ford (Programación Dinámica)
  Si hay ciclos de coste negativo, los detecta.

Problema de caminos mínimos

Conjunto de caminos entre todos los pares de nodos

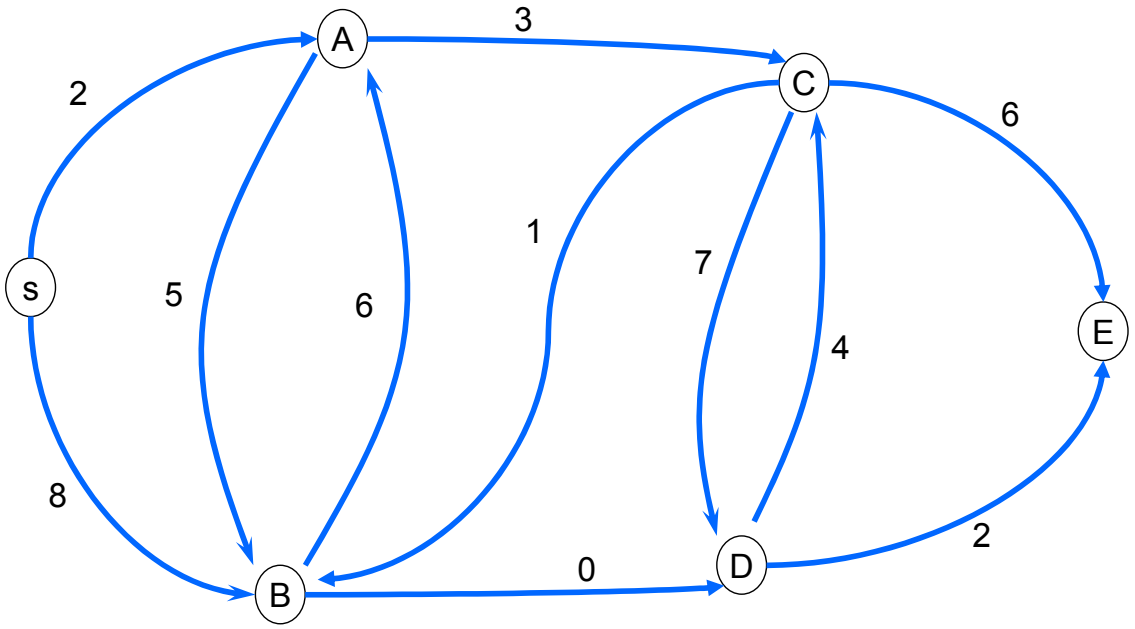
- Algoritmo de Floyd-Warshall (1962)
  (Programación Dinámica)

  Algoritmo muy parecido al de Bellman-Ford.

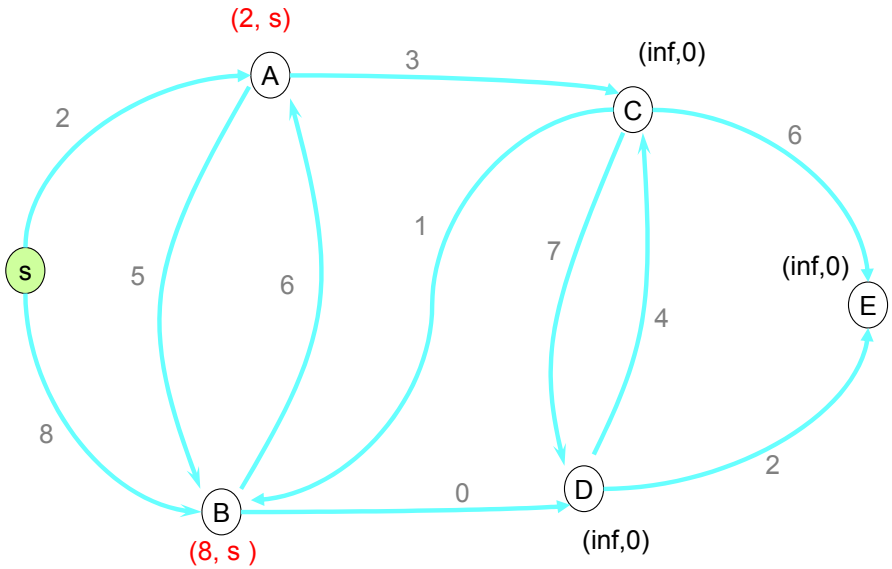
  Se basan en el hecho que, dados 3 nodos cualesquiera i, j, k, el coste del camino mínimo desde i hasta j no puede superar a la suma de los costes del camino desde i hasta k más el del camino desde k hasta j.

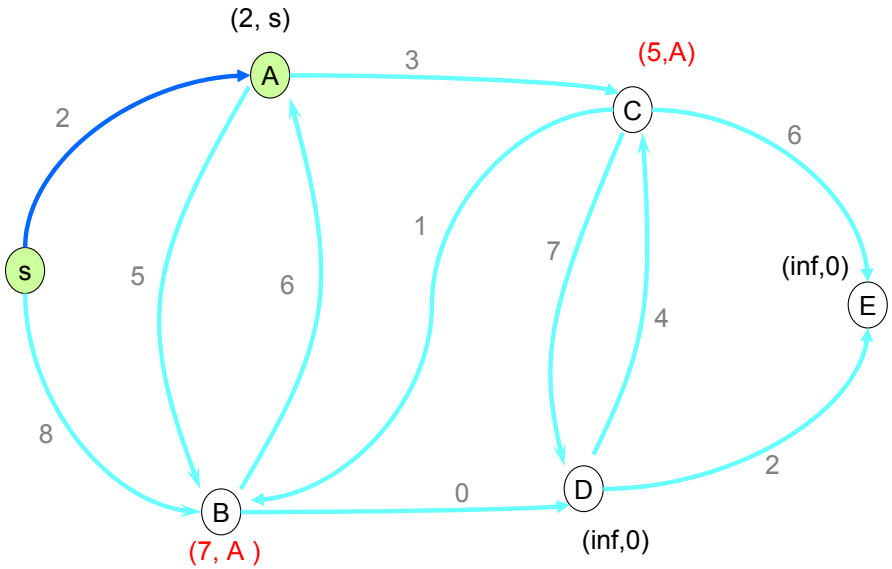
Caminos mínimos: Algoritmo de Dijkstra

Lo veremos sobre un ejemplo:

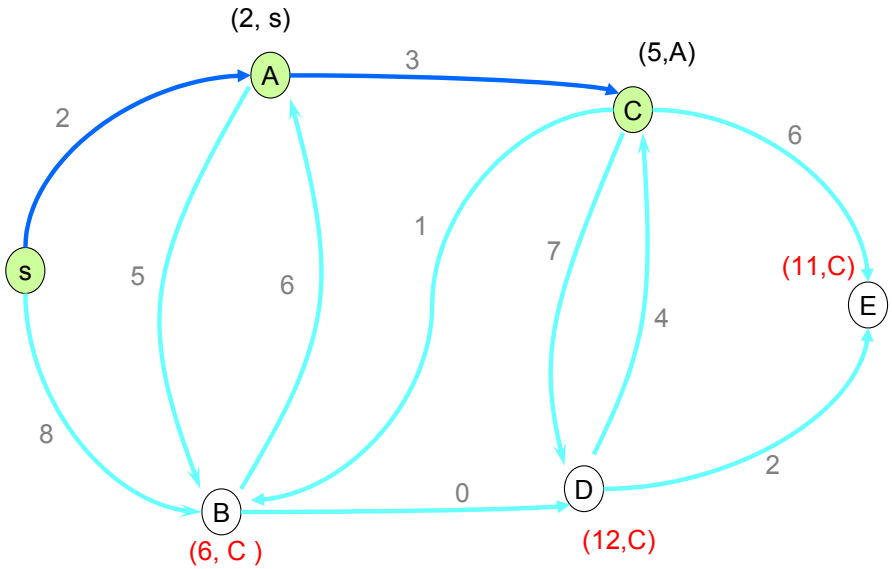


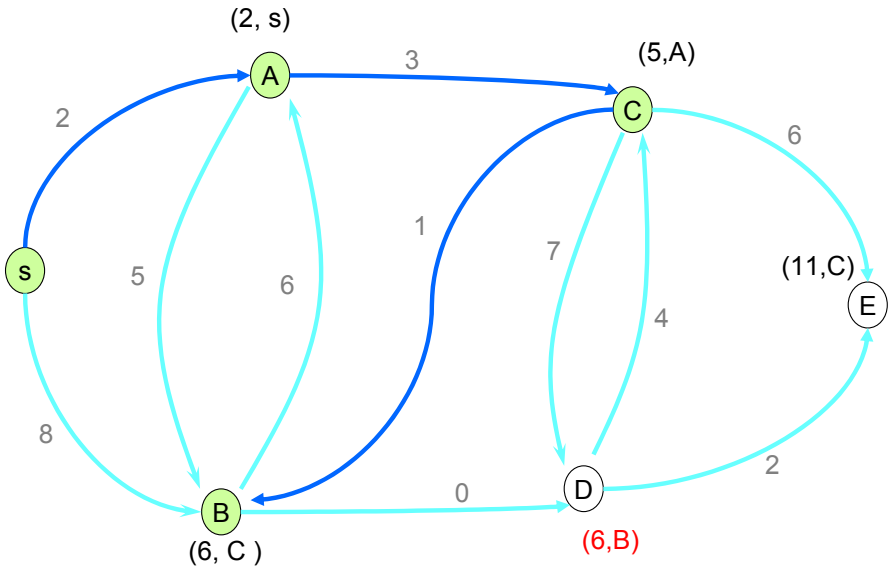
Algoritmo de Dijkstra: Ejemplo



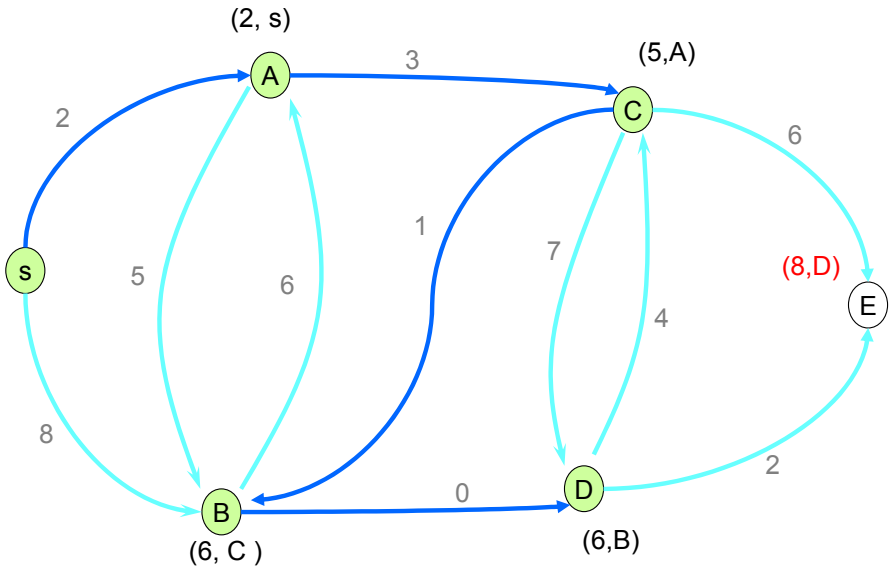


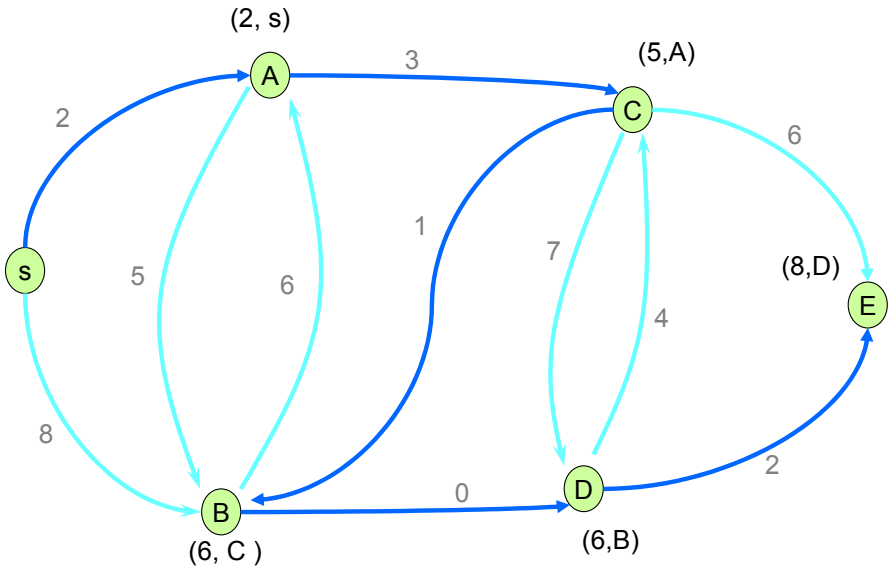
Algoritmo de Dijkstra: Ejemplo





Algoritmo de Dijkstra: Ejemplo





El algoritmo de Dijkstra: Resumen

Paso 0 Inicializar $S = \emptyset, \bar{S} = N$. Para $i \in N$, inicializamos etiquetas $d(i) = \infty$, salvo la del nodo origen, d(s)=0. Predecesor p(s)=0.

Paso 1 **Si** $\bar{S} \neq \emptyset$, seleccionar $i \in \bar{S}$ con $d(i) = \text{mín}\{d(j) : j \in \bar{S}\}$. Mover i de $\bar{S}$ a S.
en caso contrario, FIN. Para cada nodo $i, d(i)$ es la longitud del camino mínimo de s a i.

Paso 2 Actualizar las etiquetas de los nodos adyacentes a i. Para cada arco (i, j), si $d(j) > d(i) + c_{ij}$, entonces $d(j) = d(i) + c_{ij}$ y $p(j) = i$. Ir al **Paso 1**.

Problema de flujo máximo

El **problema de flujo máximo** se define sobre una red con capacidades en los arcos, y dos nodos selectos; el nodo fuente (s) y el nodo terminal (t).

El problema consiste en encontrar la mayor cantidad de flujo que se puede mandar desde s hasta t respetando las capacidades de los arcos.

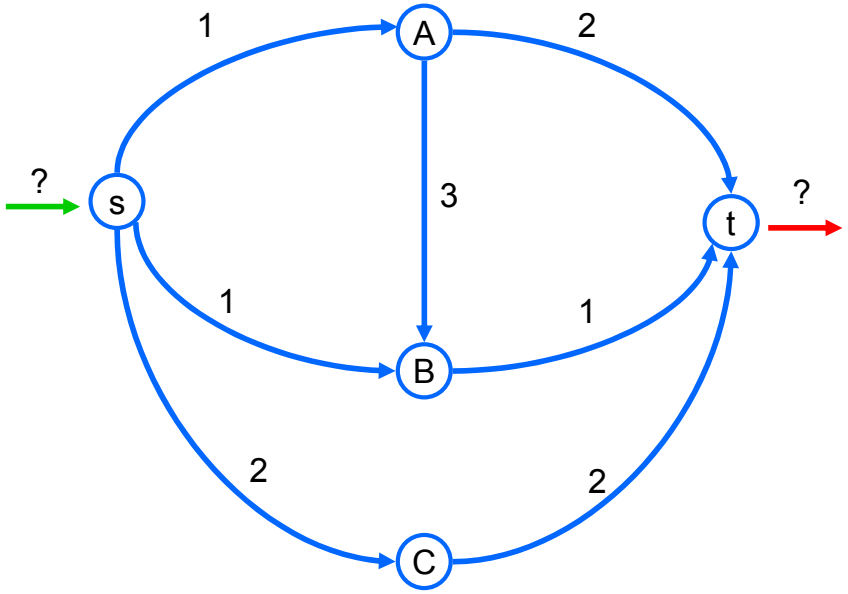
El problema de flujo máximo también tiene la propiedad de integridad.

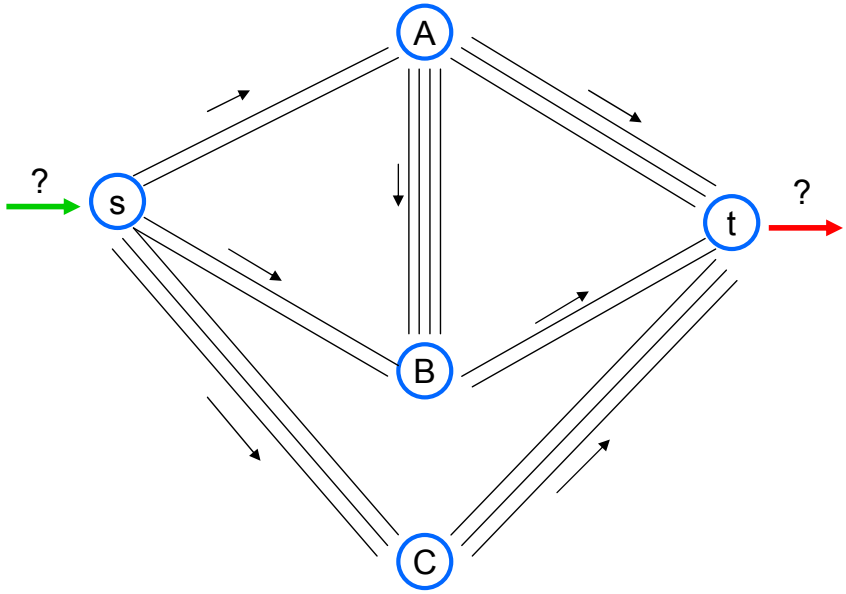
El problema de flujo máximo

Existen distintos algoritmos para resolver este problema. La mayoria se basan en la idea de **camino aumentativo.** Un camino aumentativo es un camino en la red, que permite aumentar la cantidad de flujo desde s hasta t:

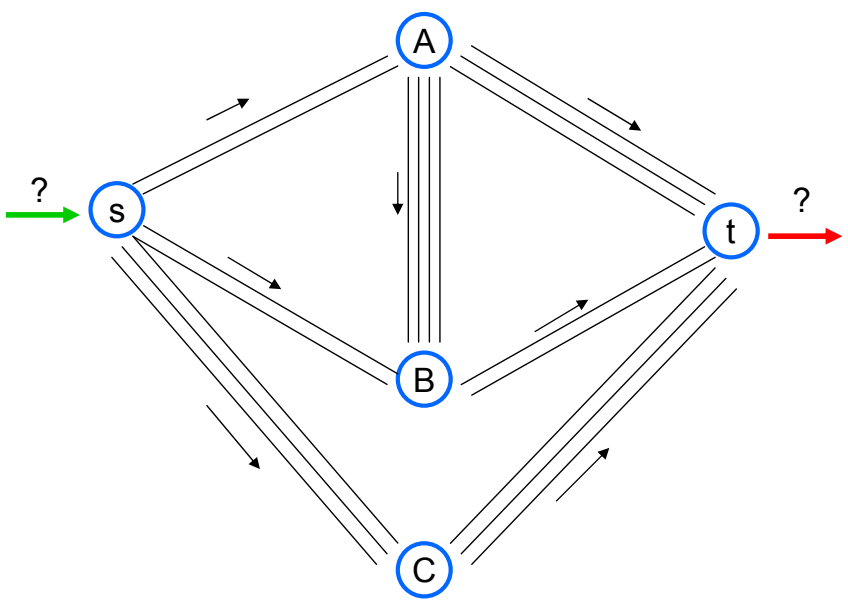
- mandándolo directamente
- o redirigiendo el flujo existente

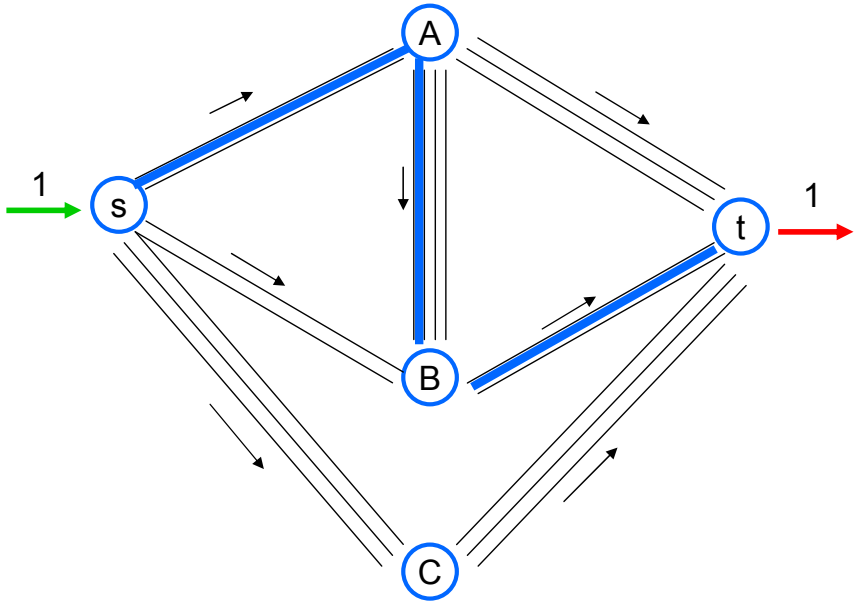
Problema de flujo máximo



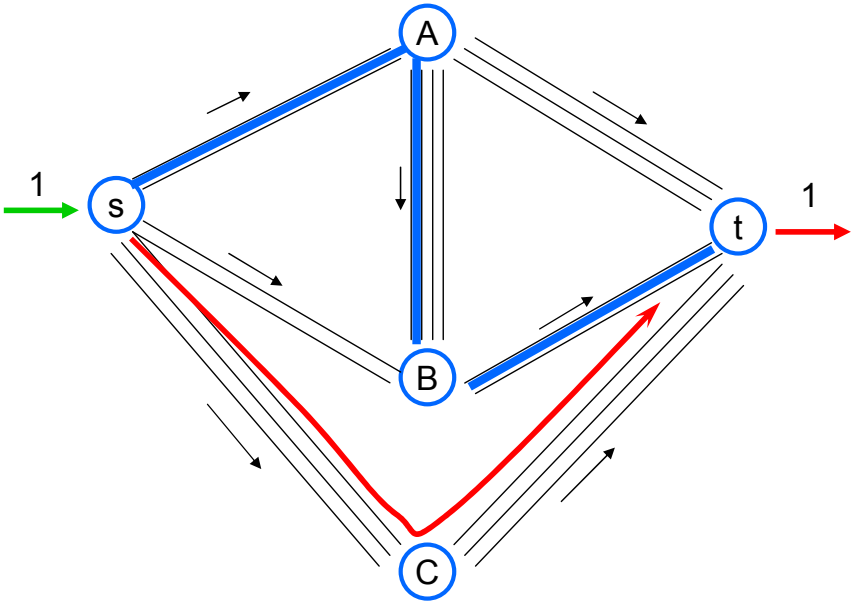


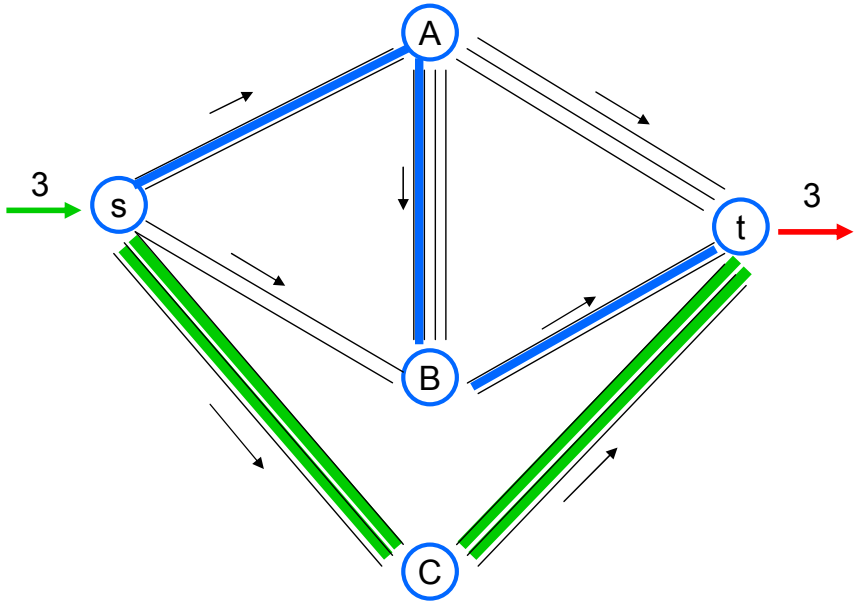
Problema de flujo máximo



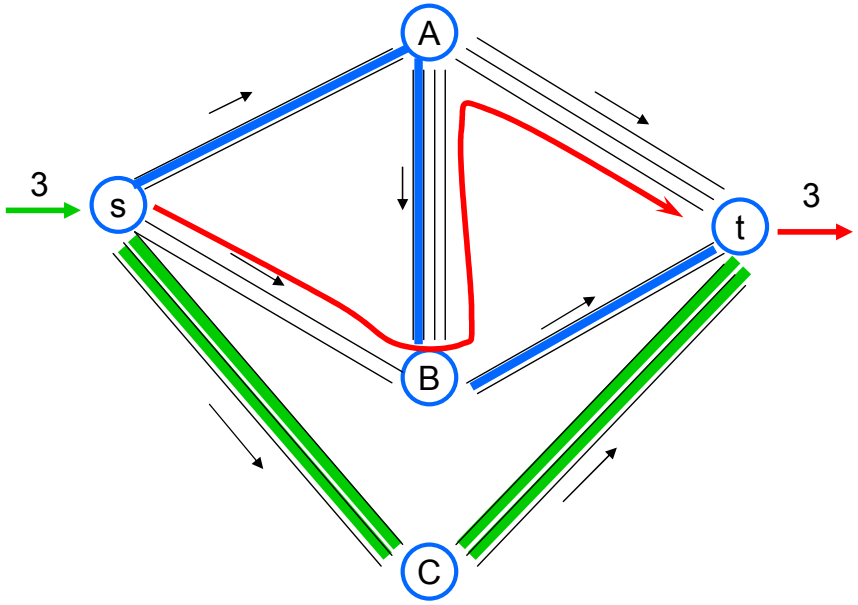


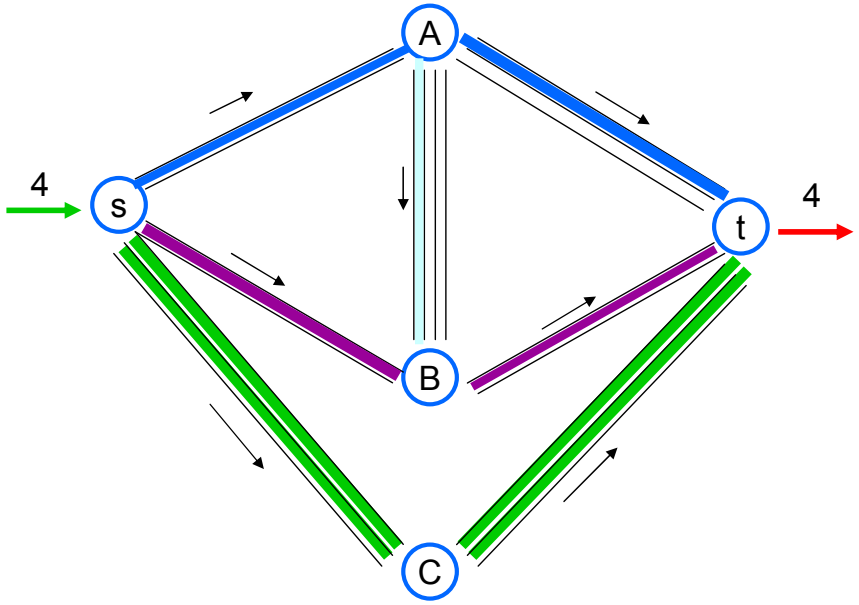
Problema de flujo máximo





Problema de flujo máximo





Problema de la mochila

Los problemas combinatorios consisten en identificar subconjuntos de un conjunto dado que, satisfaciendo algunas propiedades preestablecidas, minimicen/maximicen su coste asociado. Formalmente:

Dado un conjunto S y una familia de subconjuntos de S, $\mathcal{I} \subseteq \mathcal{P}(S)$, encontrar el conjunto $I \in \mathcal{I}$ con el menor coste (resp. mayor beneficio) asociado.

En general,

$$\text{coste}(I) = \sum\nolimits_{e \in I} c_e$$

El problema de la mochila (*knapsack problem*)

Dado un conjunto de objetos, cada uno con un volumen y un beneficio asociados, determinar cuál es el conjunto de objetos más provechoso que cabe en una mochila de una capacidad determinada.

Programación matemática	**Optimización Combinatoria**
x_i : tomo el objeto i	S = conjunto de objetos
restricción: $\sum_i v_i x_i \leqslant$ cap.	$\mathcal{I}$: conjuntos de obj. que $caben$
objetivo: max $\sum_i c_i x_i$	objetivo:arg $\max\{c(I) : I \in \mathcal{I}\}$

Problema de la mochila: Ejemplos

- Escoger asignaturas de libre configuración para alcanzar los 14 créditos con el menor esfuerzo posible.
- Decidir entre las inversiones más prometedoras con un presupuesto limitado.
- También se conoce como problema de la mochila la versión con una restricción de igualdad. (*Min Set Sum*). Éste es la base de toda una família de sistemas de criptografia de clave pública. Entre ellos el de **Merkle y Hellman (1978)** que fue uno de los primeros.

Problema de la mochila: Resolución

Habitualmente, mediante métodos enumerativos: programación dinámica, ramificación y acotación.

Algunos autores recurren a métodos heurísticos. Por ejemplo, tomar preferentemente artículos con valores altos de c_j/v_j (greedy).

Muchos trabajos estudian la estructura de la envolvente convexa de las soluciones factibles.

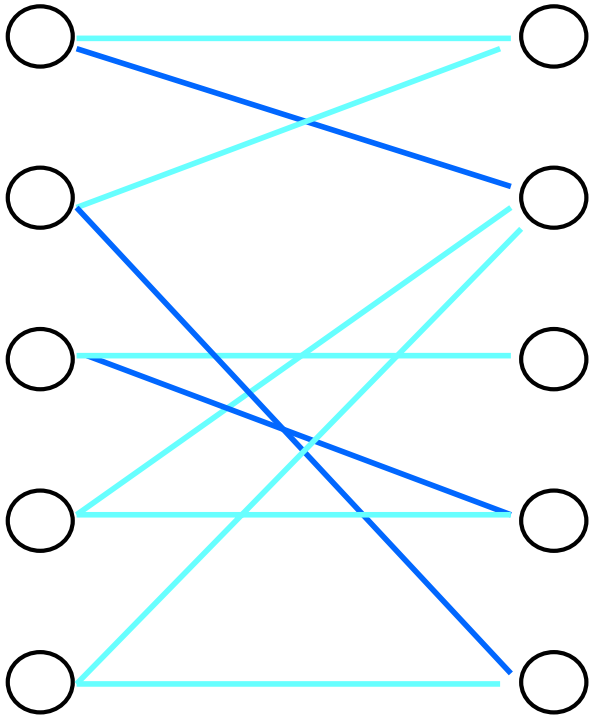
Problema de emparejamiento

Dado un grafo **G=(N,A)**, $B \subseteq A$ es un **emparejamiento (perfecto)**, si cada nodo de N es incidente, como mucho (exactamente), con una arista de B.

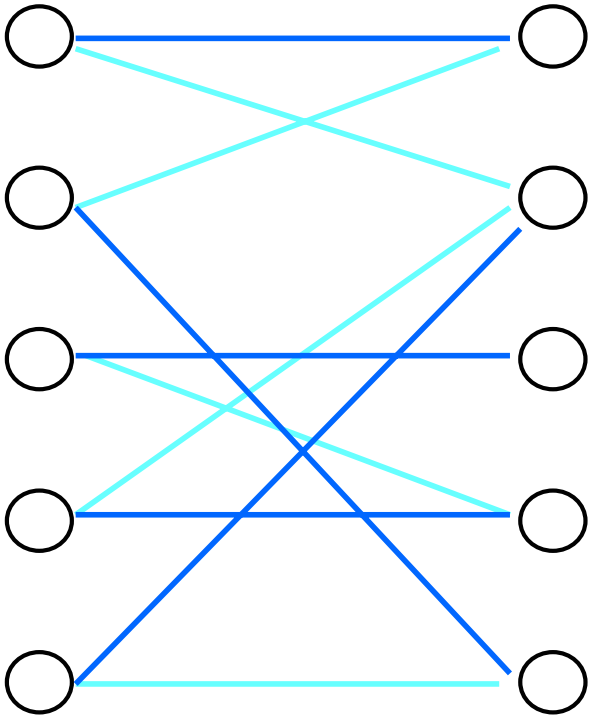
Objetivo: Encontrar el emparejamiento

$$\begin{cases} \text{de cardinal máximo (EM)} \\ \text{máximo/perfecto de peso mínimo (EPM)} \end{cases}$$

Problema de emparejamiento: ejemplos

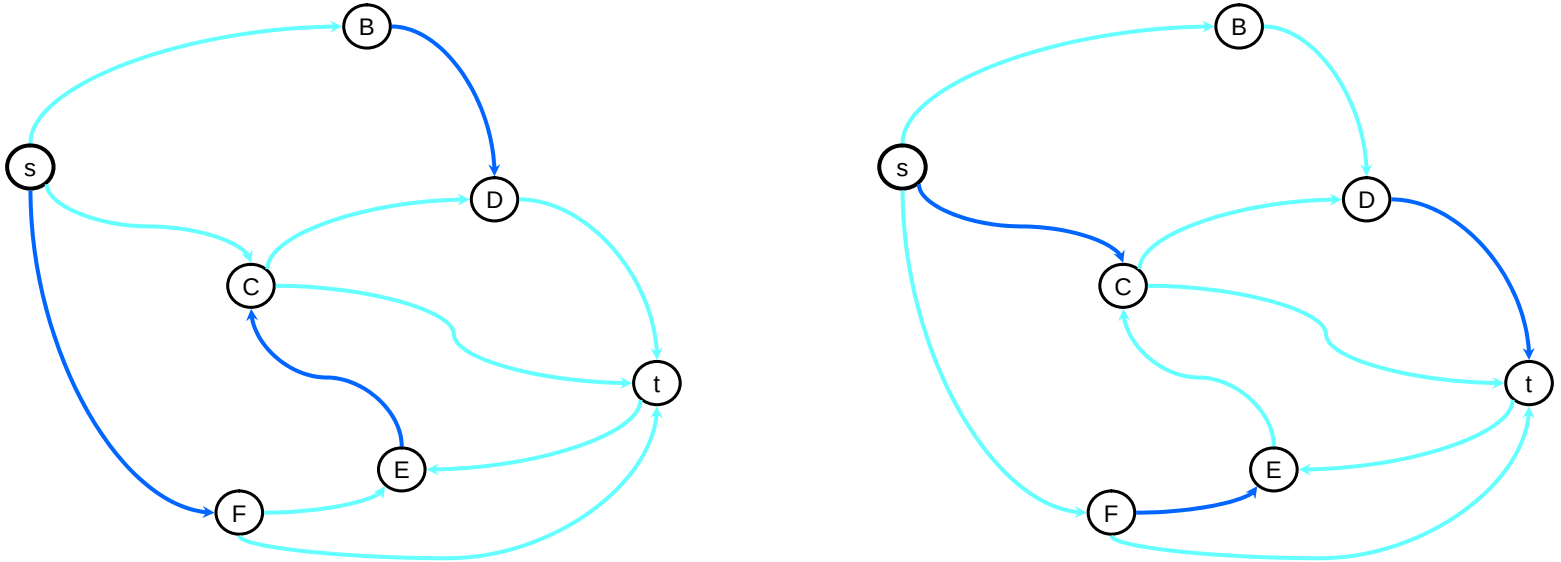


Emparejamiento no perfecto



Emparejamiento perfecto

Problema de emparejamiento: ejemplos



Existen grafos sin un emparejamiento perfecto.

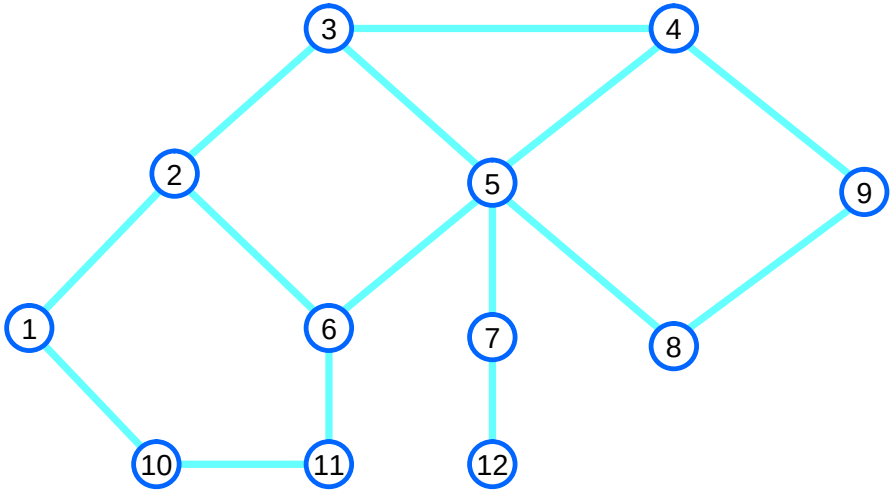
El emparejamiento máximo no tiene por qué ser único.

Problema de emparejamiento: ejemplos

En la segunda guerra mundial la Royal Air Force (RAF) inglesa estaba compuesta por varios pilotos procedentes de distintos paises, que hablaban diferentes idiomas y tenian distintos niveles de entrenamiento. La RAF tenia que asignar dos pilotos a cada avión, siempre asignando pilotos con idiomas y niveles de entrenamiento compatibles al mismo avión. A la RAF le interesaba hacer volar el máximo número de aviones posible.

Problema de emparejamiento: ejemplos

El grafo siguiente representa las calles de un barrio de cierta ciudad. Un policía ubicado en el centro de una calle puede vigilar los cruces de los dos extremos. ¿Es posible vigilar todos los cruces del barrio con 6 policías?



Emparejamiento de cardinal máximo: enfoque OC

- Conjuto de elementos S: Aristas del grafo
- Subconjuntos válidos $I \in \mathcal{I}$:

  conj. de aristas disjuntas 2 a 2
- Costes:

  maximizar cardinal (coste 1 en cada arista)

Emparejamiento de cardinal máximo: enfoque PM

- Variables: $\forall a \in A$,

$$x_a = \begin{cases} 1 & \text{si } a \text{ está en el emparejamiento} \\ 0 & \text{si no} \end{cases}$$

- Restricciones: $\forall v \in N$,

  $\sum_{a \in \delta(v)} x_a \leqslant 1$

- Objetivo

  $\text{máx} \sum_{a \in A} x_a$

Problema de emparejamiento: resolución

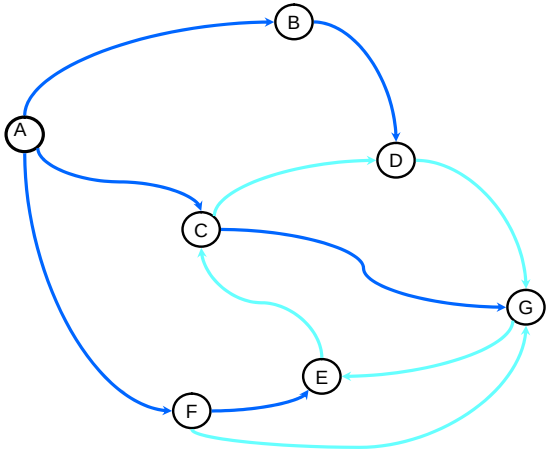
La mayoría de los algoritmos específicos para encontrar el EM son algoritmos iterativos que aumentan el cardinal de un emparejamiento factible a cada iteración. Las especializaciones de estos algoritmos para grafos bipartidos son bastante más eficientes.

Cuando se resuelven problemas de emparejamiento desde la PM, se usan algoritmos de tipo branch and cut, ya que se conoce la descripción completa de la envolvente convexa de las soluciones factibles.

Árbol de expansión de coste mínimo

Minimum spanning tree (MST)

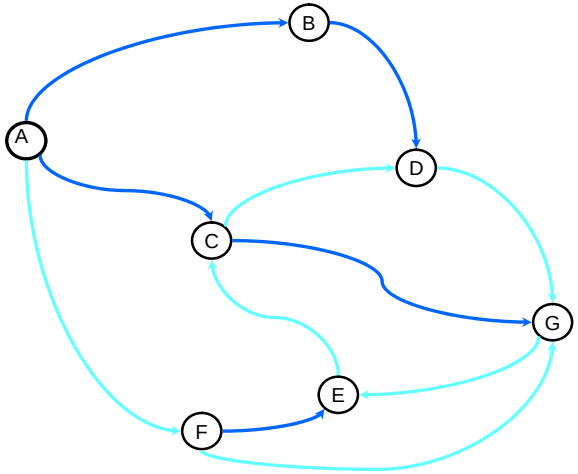
Dado un grafo **G=(N,A)**, $T \subseteq A$ es un **árbol de expansión** si G'=(N,T) es un árbol;es decir:

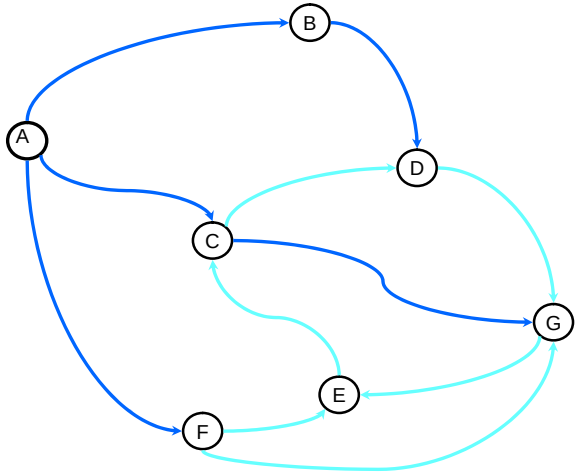


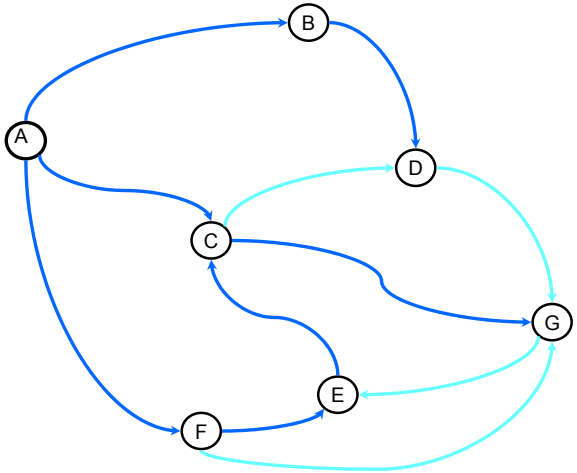
- no contiene subcircuitos
- es conexo

En grafos dirigidos ignoraremos el sentido de los arcos.

Árboles de expansión: (contra)ejemplos







Árboles de expansión

- ¿Qué pasa si a un árbol de expansión le quitamos un arco?
- ¿Qué pasa si a un árbol de expansión le añadimos un arco?
- ¿Cuántos arcos tiene un árbol de expansión?

Árbol de expansión de coste mínimo

Dado un grafo **G=(N,A)**, con costes c asociados a los arcos, el **problema del árbol de expansión de coste mínimo** consiste en encontrar el árbol de expansión de G con el menor coste total, donde

$$\text{coste}(T) = \sum_{a \in T} c_a$$

MST: Ejemplos

El primer trabajo sobre el MST es debido a Dijkstra, que desarrolló un algoritmo para este problema, para minimizar la cantidad de cobre que necesitaba para las conexiones en el panel su ordenador X1.

En realidad, este algoritmo fue publicado en el mismo artículo donde Dijkstra presentaba su algoritmo para el problema de caminos mínimos.

MST: Ejemplos

Compañía de reforestación quiere sembrar árboles en ocho zonas en la misma área. Debe desarrollar un sistema de caminos de tierra para tener acceso a cualquier zona desde cualquier otra. Las distancias entre cada par de zonas son:

	1	2	3	4	5	6	7	8
1	-	13	21	9	7	18	20	15
2		-	9	18	12	26	23	11
3			-	26	17	25	19	10
4				-	7	16	15	9
5					-	9	11	8
6						-	6	10
7							-	5
8								-

¿Entre qué pares de zonas construiríais caminos?

2-3, 2-4, 4-5, 1-5, 5-8, 6-7, 7-8

MST: Resolución

Uno de los algoritmos más conocidos para resolver este problema es el **algoritmo de Kruskal**.

Se trata de un algoritmo tipo Greedy. A pesar de esto, la estructura de este problema garantiza que este algoritmo proporcione la solución óptima.

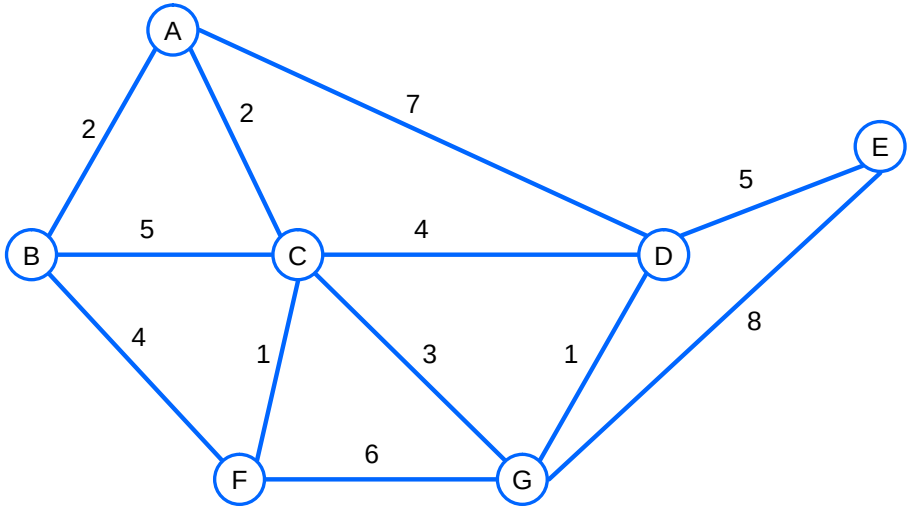
MST: El algoritmo de Kruskal

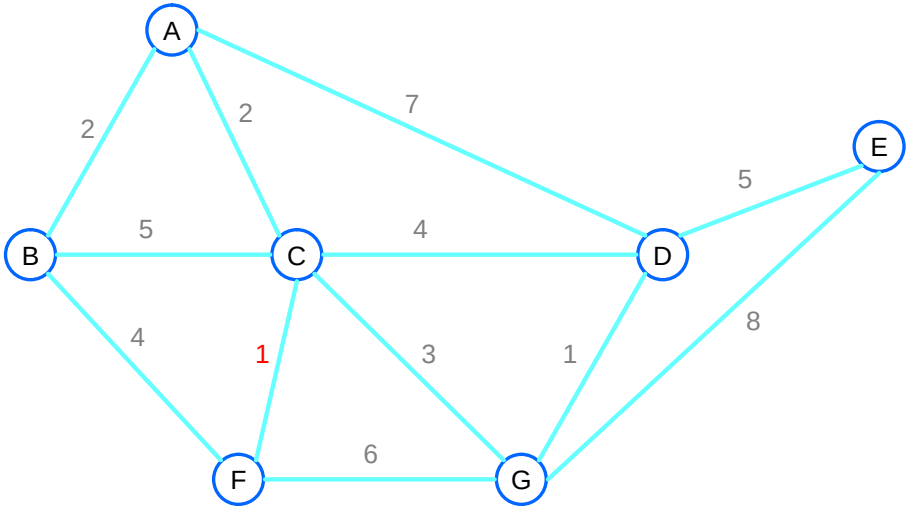
Es un algoritmo iterativo.

En cada iteración del algoritmo, se añade al conjunto de arcos seleccionados el arco disponible de coste menor que no forme circuito con los tomados anteriormente.

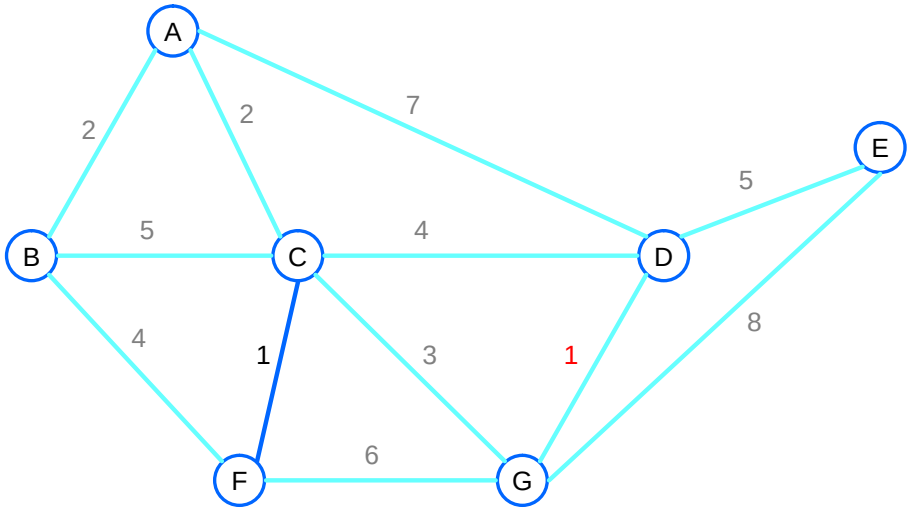
Finaliza cuando el subgrafo generado es conexo.

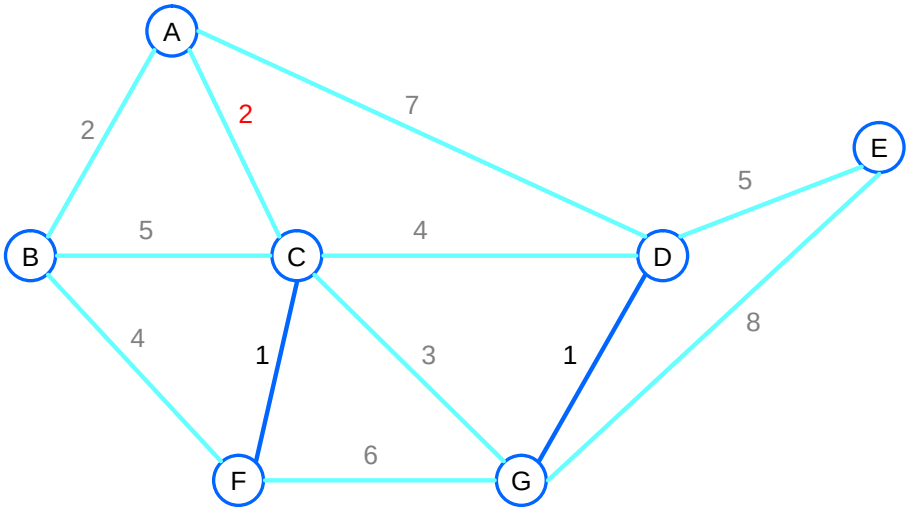
Algoritmo de Kruskal: ejemplo



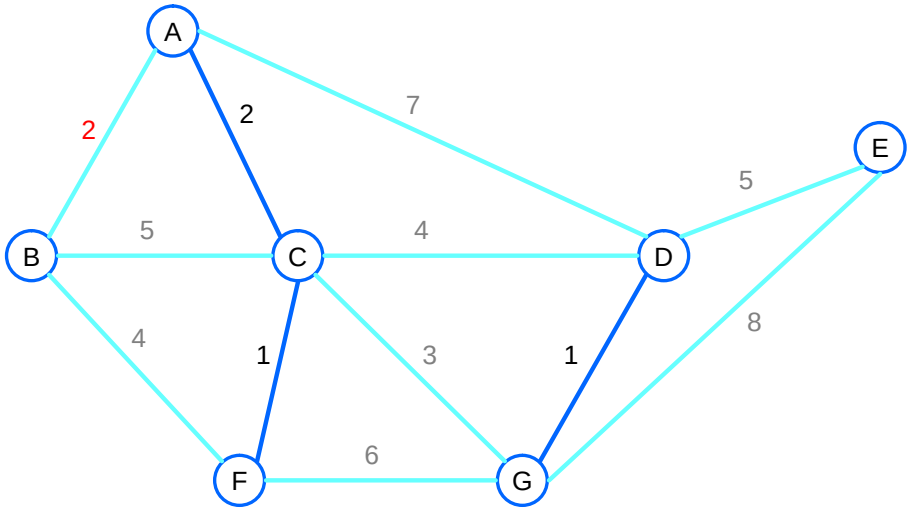


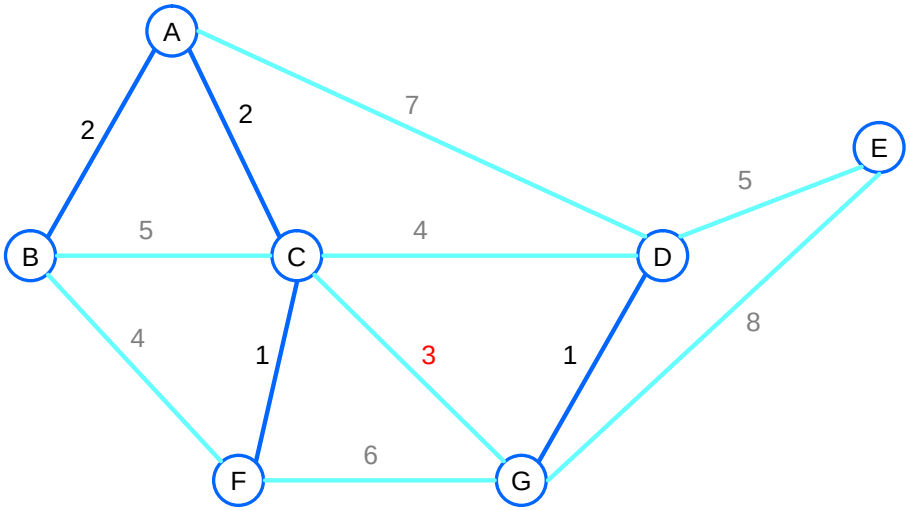
Algoritmo de Kruskal: ejemplo



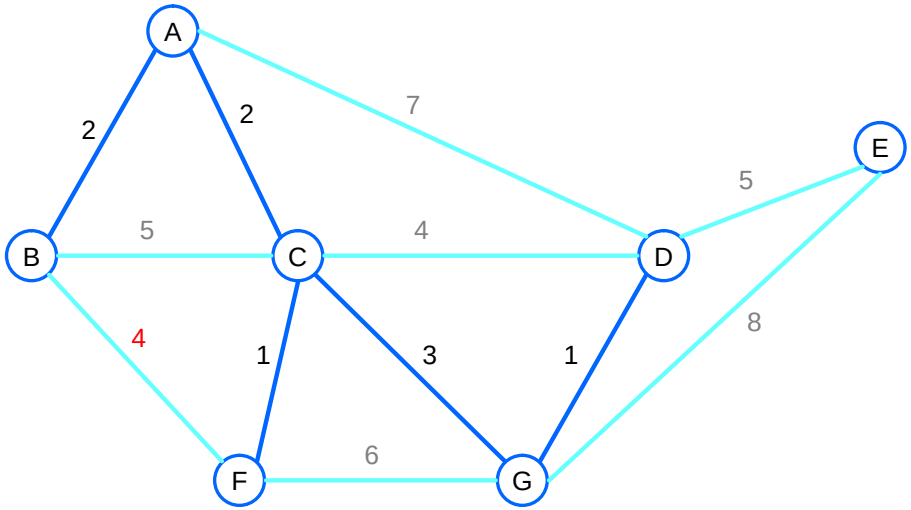


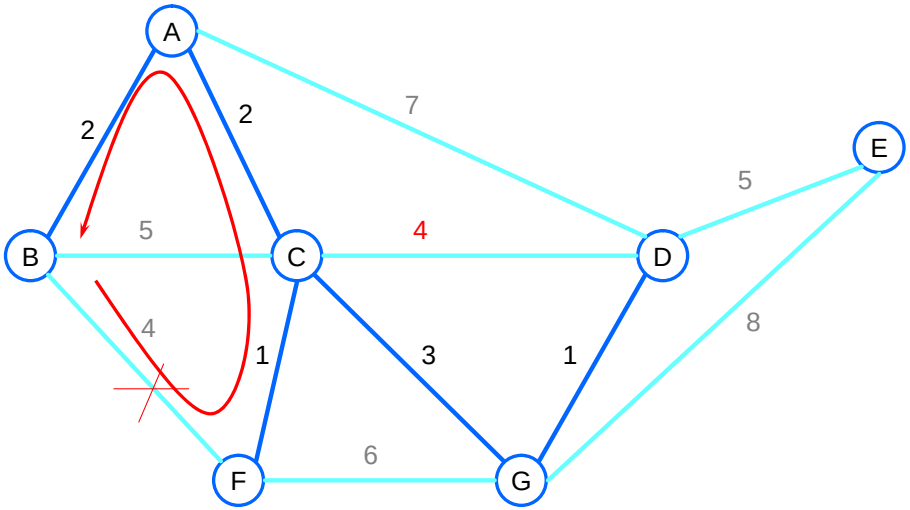
Algoritmo de Kruskal: ejemplo



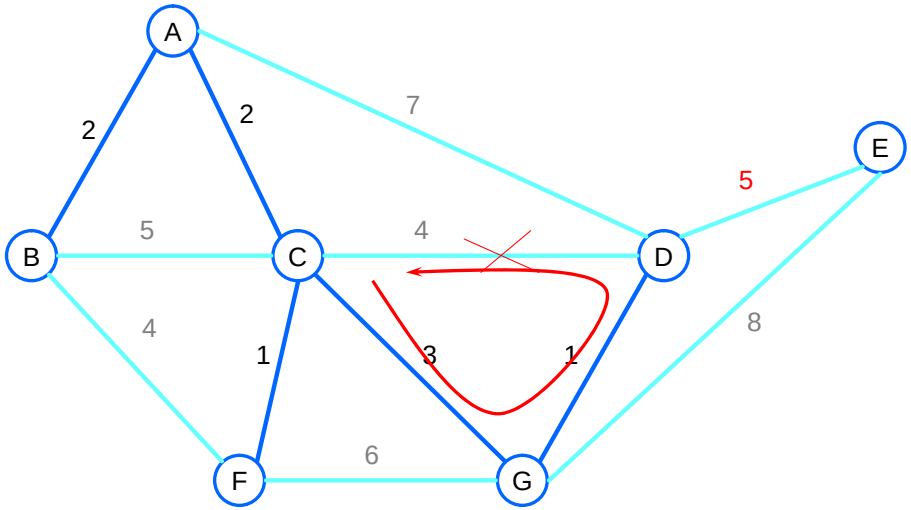


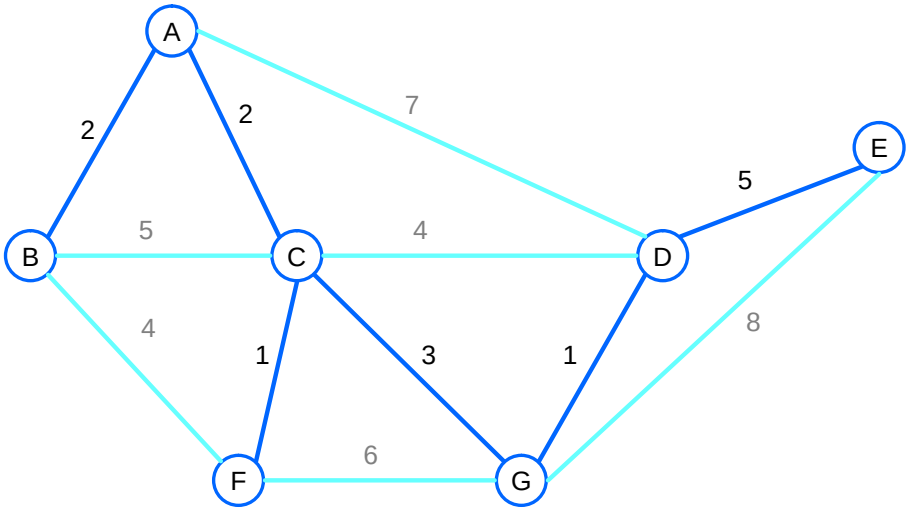
Algoritmo de Kruskal: ejemplo





Algoritmo de Kruskal: ejemplo





El algoritmo de Kruskal: Resumen

Paso 0 Inicializar $L =$, conjunto de arcos ordenados. $T = \emptyset$

Paso 1 **Si** $G = (N, T)$ es conexo, FIN. T define el árbol óptimo.
en caso contrario sacar el siguiente elemento de L, a.

Paso 2 **Si** a no forma un circuito con los arcos de T, actualizar $T = T \cup \{a\}$. Volver a **Paso 1**

MST: Extensiones

Una extensión del MST es el problema del árbol de Steiner de coste mínimo.

En este problema se pretende conectar un nodo fuente a todo un conjunto de nodos sumideros, utilizando eventualmente otros nodos auxiliares de la red a un coste mínimo.

Este problema aparece, por ejemplo, en el diseño de circuitos integrados.

Problema del viajante

Travelling Salesman Problem (TSP)

Un viajante de comercio tiene que visitar un conjunto de clientes y regresar después al punto de partida. Antes de salir, debe decidir en qué orden visitará a sus clientes para minimizar la distancia total recorrida.

Más formalmente ...

El problema del viajante

Dado un grafo $G = (N, A)$, un **circuito Hamiltoniano** en G es es un circuito que pasa exactamente una vez por cada nodo $v \in N$.

El problema del viajante de comercio se define sobre un grafo (generalmente completo) con costes asociados a los arcos y consiste en encontrar el circuito Hamiltoniano de coste mínimo.

TSP desde la programación matemática

Variables: Para cada par de nodos $i, j \in N$ definimos

$$x_{ij} = \begin{cases} 1 & \text{si se visita el nodo } j \text{ después de } i \\ 0 & \text{en otro caso} \end{cases}$$

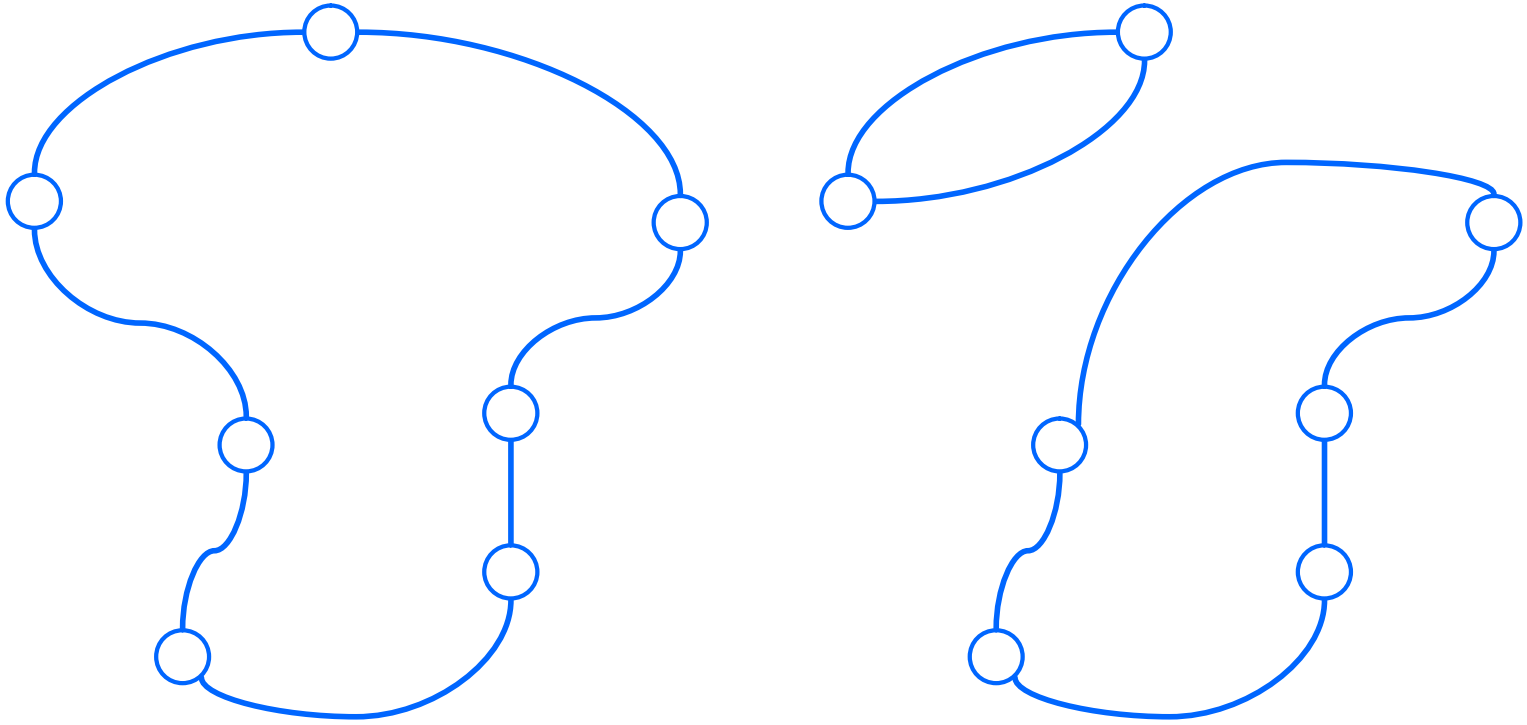
Objetivo: Minimizar $\sum_{i \neq j} c_{ij} x_{ij}$

Restricciones:

- Cada nodo debe preceder a otro: $\sum_{i} x_{ij} = 1$
- Cada nodo debe tener un predecesor: $\sum_{j} x_{ij} = 1$

TSP desde la programación matemática

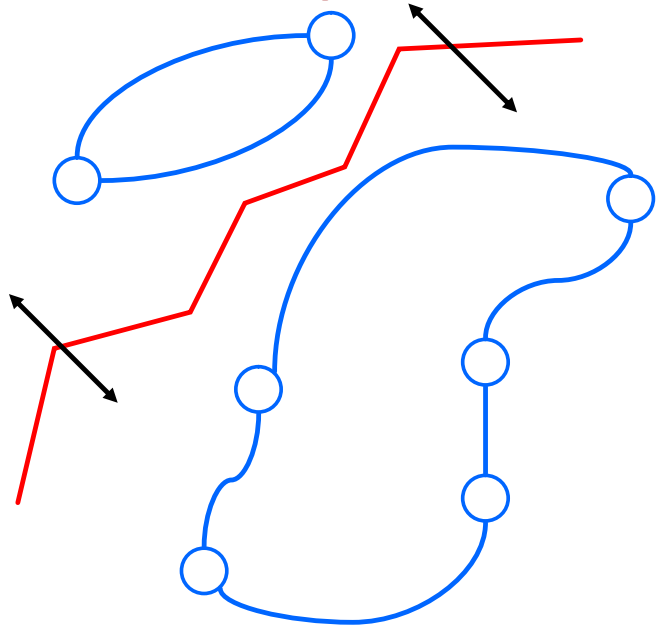
Con este modelo podemos obtener:



¿Qué falta?

El problema del viajante

Restricciones de ruptura de subcircuito



Para cada conjunto $S \subsetneq N$ con $|S| \leqslant n/2$: $\sum_{a \in \delta(S)} x_a \geqslant 1$

donde $\delta(S)$ es el conjunto de arcos con un extremo en S y otro fuera

El problema del viajante

Restricciones de ruptura de subcircuito

¿Cuántas de estas restricciones aparecen si tenemos

- 5 nodos? 16
- 10 nodos? 638
- 25 nodos? 16777216 !!

No se puede aplicar *branch and bound* al modelo completo. **Se relajan** estas restricciones y se van añadiendo a medida que hacen falta $\longrightarrow$ branch and cut

El problema del viajante

Métodos heurísticos

Resolver el problema del viajante de comercio de forma exacta es muy caro. Cuando no sea imprescindible encontrar el óptimo recurriremos a métodos heurísticos.

Se han desarrollado muchos métodos heurísticos para este problema, dadas sus extensas aplicaciones. Algunas heurísticas son **constructivas** y otras **de mejora**.

TSP: Métodos heurísticos

- **Heur. constructivas**
  - El vecino más próximo
  - Métodos de inserción
  - Heurística de Christofides
- **Heur. de mejora**
  - Búsqueda local (problema: optimos locales)
    - 2-intercambios, etc.
  - Métodos avanzados/ Metaheurísticas
    - Búsqueda tabú , Algoritmos genéticos, Sistemas de colonias de hormigas, simulated annealing. . .