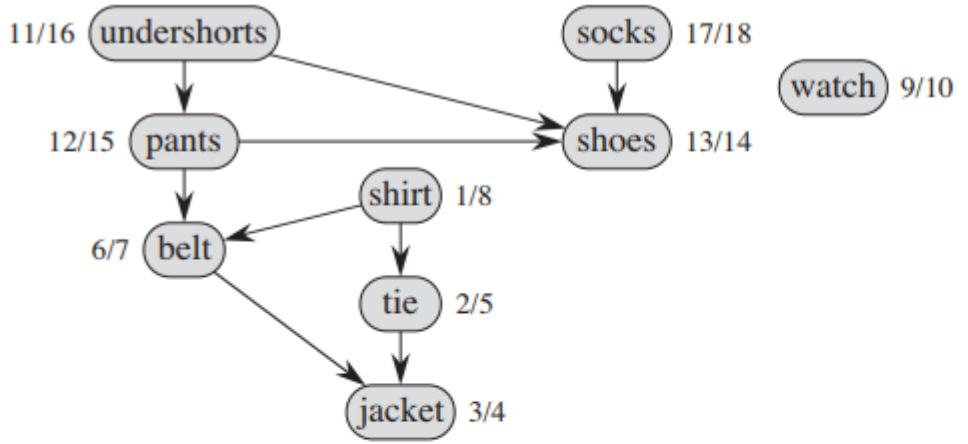
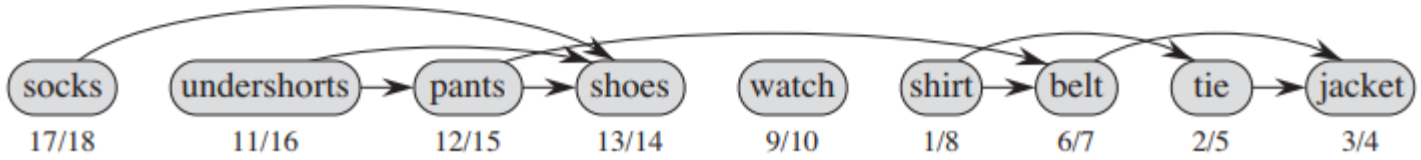
Algoritmos aplicados a Grafos

Ordenamiento Topológico

- El ordenamiento topológico consiste en crear una lista lineal de nodos donde el orden depende de las aristas
- Se usa en grafos dirigidos y se basa en el proceso por profundidad (Deep First Search)
- Si hay ciclos, NO se puede hacer ordenamiento topológico
- La regla es: si existe la arista (u,v) entonces u va en la lista antes de v.





Ordenamiento Topológico

- Aplique la DFS(G)
- Cuando “termina” con cada vértice, métalo en una lista enlazada
- Retorne la lista de vértices

Ejemplo de código

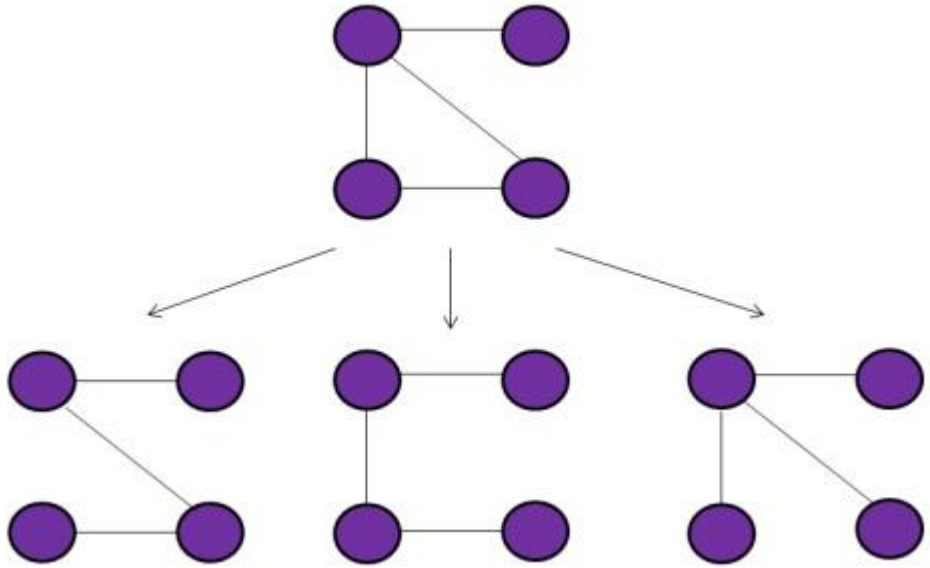
Árbol de mínima expansión

- Se aplica a grafos no dirigidos totalmente conexos

- Dado un grafo G, un árbol de minima expansión (Minimun Spanning Tree) es un árbol que tiene:

  - Los mismos nodos de G
  - Hay rutas entre todos los nodos
  - NO hay ciclos.

Árbol de Mínima Expansión



A partir de un grafo no dirigido G, se pueden obtener varios árboles de mínima expansión.

Este es un algoritmo muy usado en ramas como electrónica o ingeniería vial

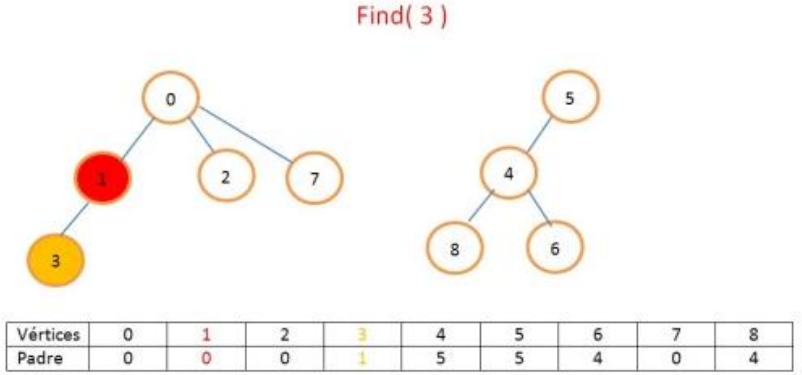
Arbol de mínima expansión - Kruskal

- El algoritmo de Kruskal para determinar un árbol de mínima expansión es un algoritmo “greedy”: trata de buscar a cada paso la solución correcta asumiendo que optimizar en pasos locales optimizará el resultado total.

- Para su funcionamiento se requiere que cualquier arista (u,v) tenga un peso (valor) que llamaremos w

Arbol de Mínima Expansión - Kruskal

- Se necesitarán 2 operaciones:
  - Find(u) que encuentra el padre de un nodo



Vértices	0	1	2	3	4	5	6	7	8
Padre	0	0	0	1	5	5	4	0	4

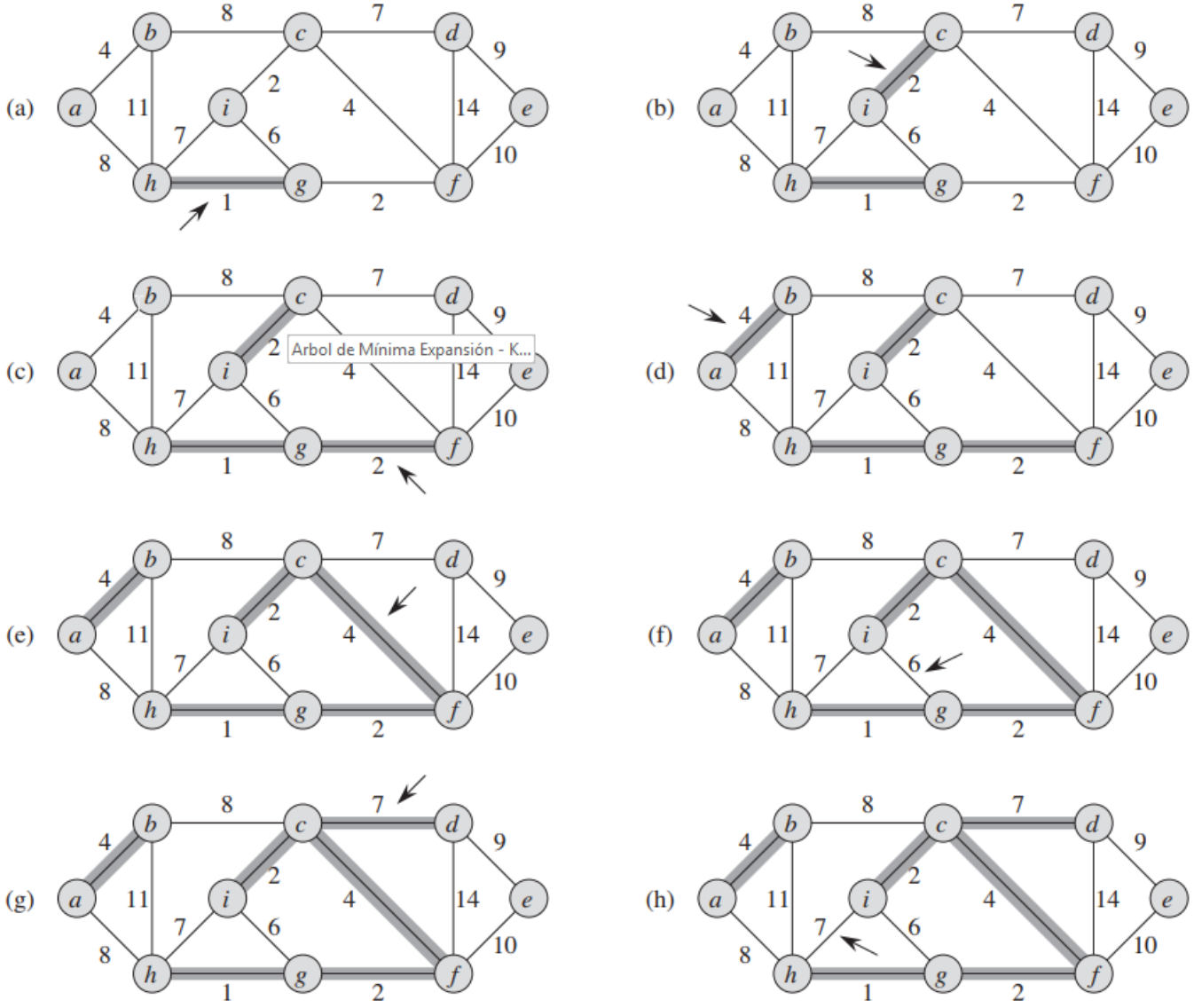
  - Union(x,y) dados 2 nodos distintos no

Conectados, se conectan ambos si existe enlace

Arbol de Mínima Expansión - Kruskal

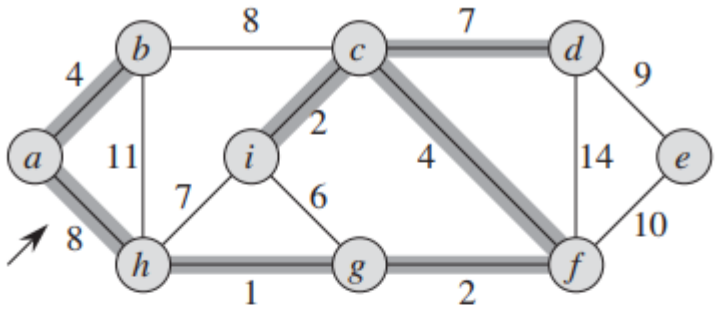
```
MST-KRUSKAL(G, w)
1  A = ∅
2  for each vertex v ∈ G.V
3       MAKE-SET(v)
4  sort the edges of G.E into nondecreasing order by weight w
5  for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
6       if FIND-SET(u) ≠ FIND-SET(v)
7            A = A ∪ {(u, v)}
8            UNION(u, v)
9  return A
```

Arbol de Mínima Expansión - Kruskal

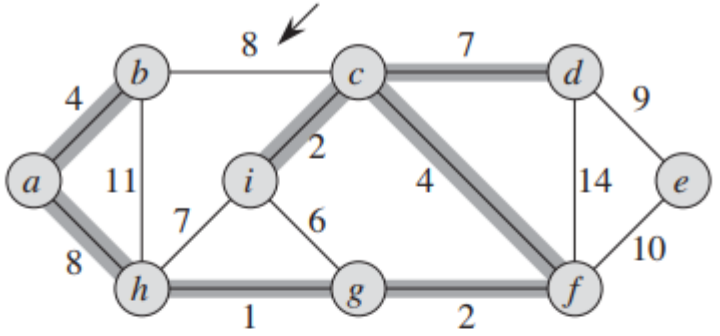


Arbol de Mínima Expansión - Kruskal

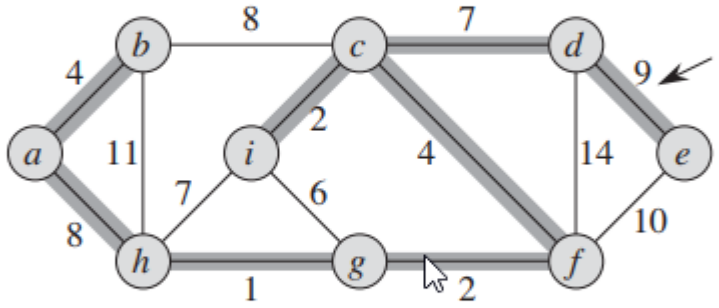
(i)



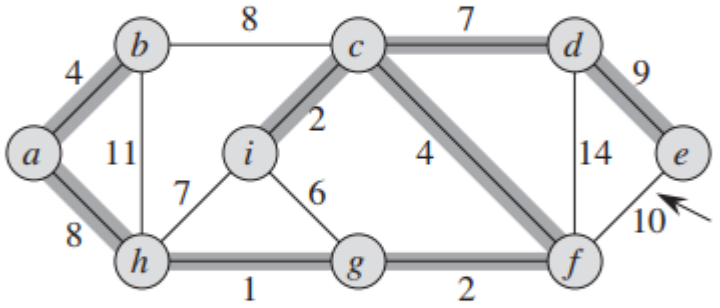
(j)



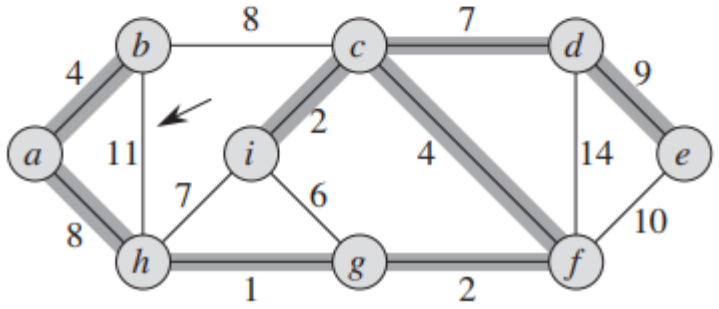
(k)



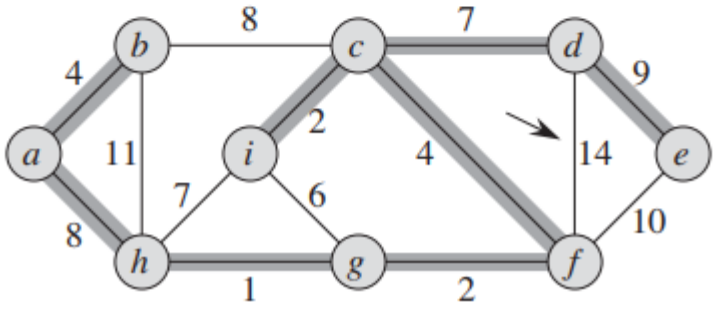
(l)



(m)



(n)



El camino más corto

- Dado un grafo DIRIGIDO, con pesos en las aristas, se dice que buscamos una ruta R entre (u,v) tal que la suma total de los pesos en las aristas de ella sea el mínimo posible entre todas las rutas (u,v)

- Existen 3 variantes:
  - Encuente la ruta minima (camino más corto) a un nodo v desde cualquier otro vértice u
  - Encuentre la ruta minima entre los nodos u y v
  - Encuentre todas las rutas mínimas entre todos los nodos

Algoritmo de Dijkstra

- El algoritmo de Dijkstra se basa en optimizaciones parciales, donde a partir de un nodo inicial va creando subconjuntos de nodos y progresivamente avanza a través de sub-optimizaciones.

- Este algoritmo encuentra las rutas minimas a cualquier nodo desde un nodo dado.

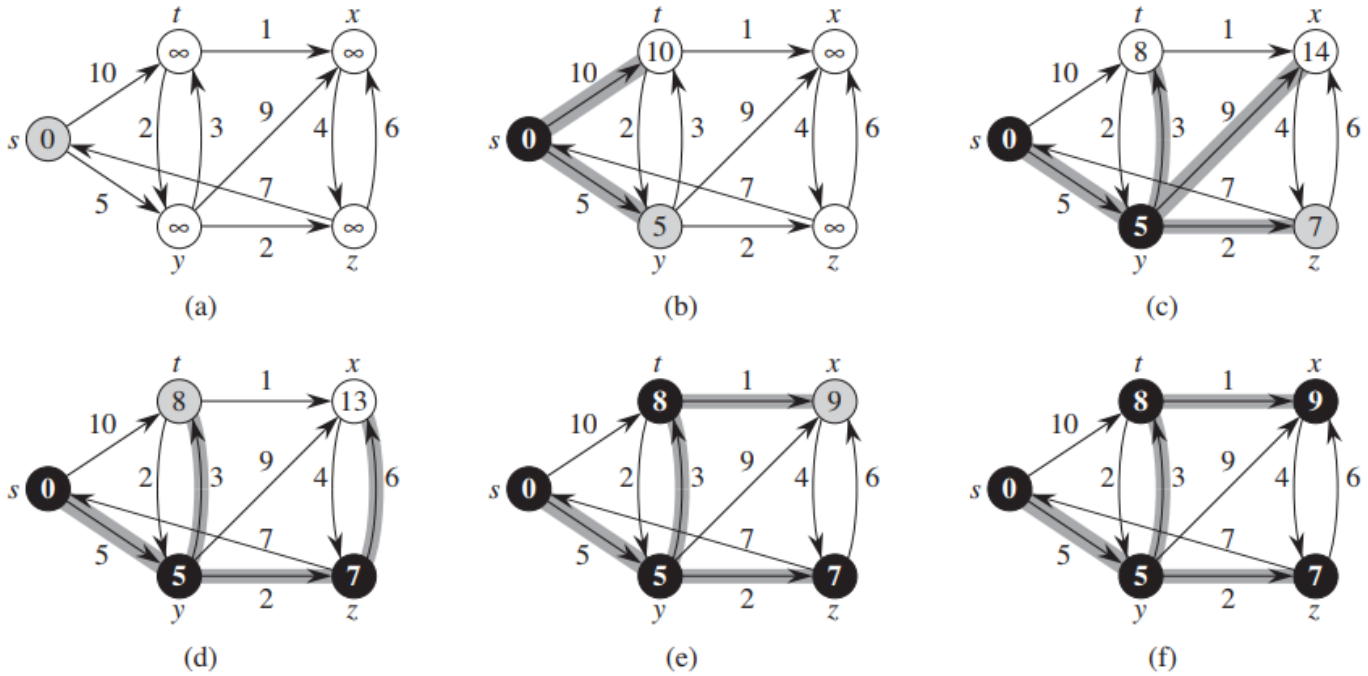
- Es un algoritmo greedy (voraz) muy utilizado.

- Usa el concepto de COLA DE MINIMA PRIORIDAD: que es una cola donde la prioridad no se da por el valor máximo sino por el minimo.

Relajación de Nodos

- Es una operación que consiste en optimizar una ruta entre 2 nodos:

- Relax (u,v) significa en probar que existe otro part (u,x), (x,v) que me permite llegar de u a v a un costo menor.



Algoritmo de Dijkstra

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S = ∅
3  Q = G.V
4  while Q ≠ ∅
5       u = EXTRACT-MIN(Q)
6       S = S ∪ {u}
7       for each vertex v ∈ G.Adj[u]
8            RELAX(u, v, w)
```