Grafos
Algoritmo de Dijkstra
Algoritmo de Prim

Estructuras de Datos

Gabriel Ávila

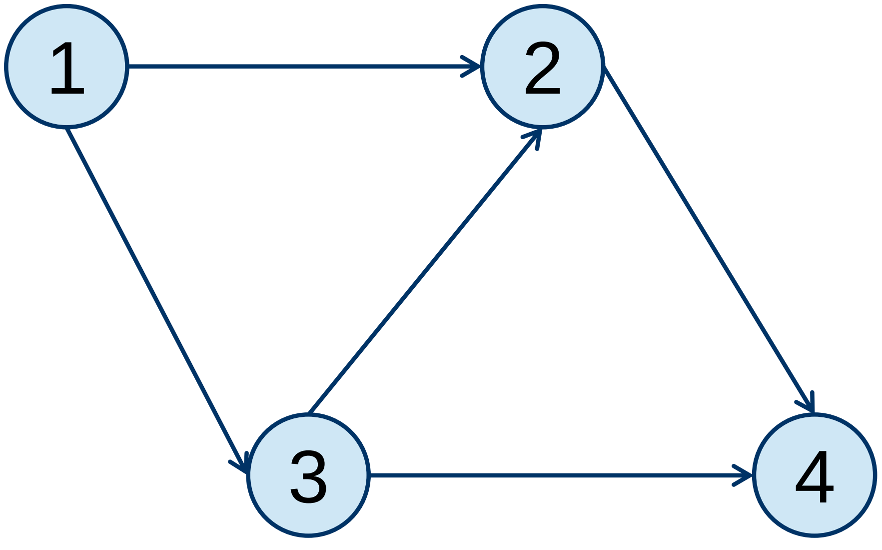
Pontificia Universidad Javeriana

Departamento de Ingeniería de Sistemas

Matriz de Caminos

Ejercicio

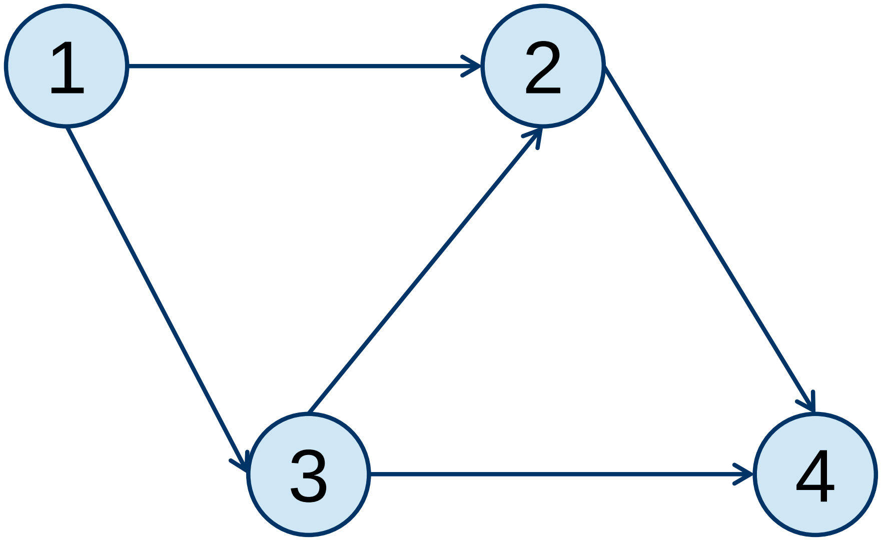
- Para el siguiente grafo:



- Escribir la matriz de adyacencia A.

Ejercicio

- Matriz de adyacencia:



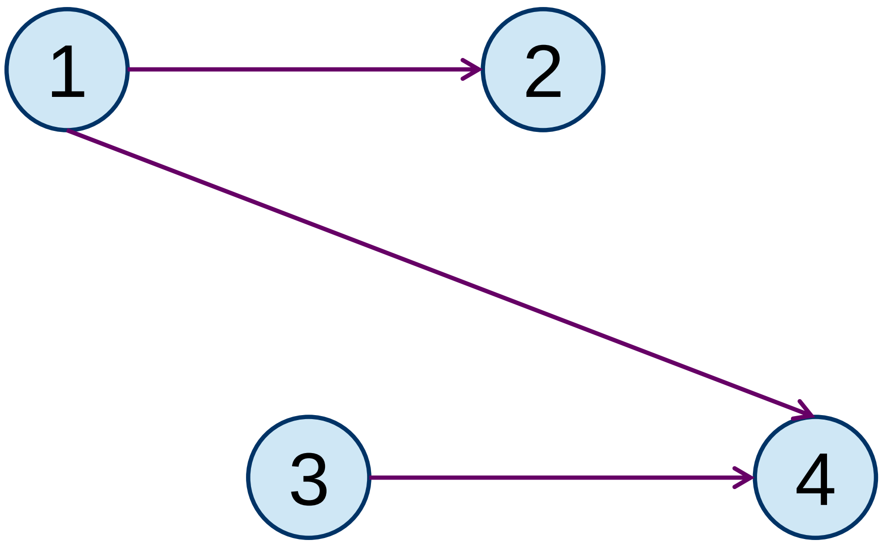
$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Ejercicio

- Calcular $A^2=A*A$.
- Dibujar el grafo representado por A^2.

Ejercicio

$$A^2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

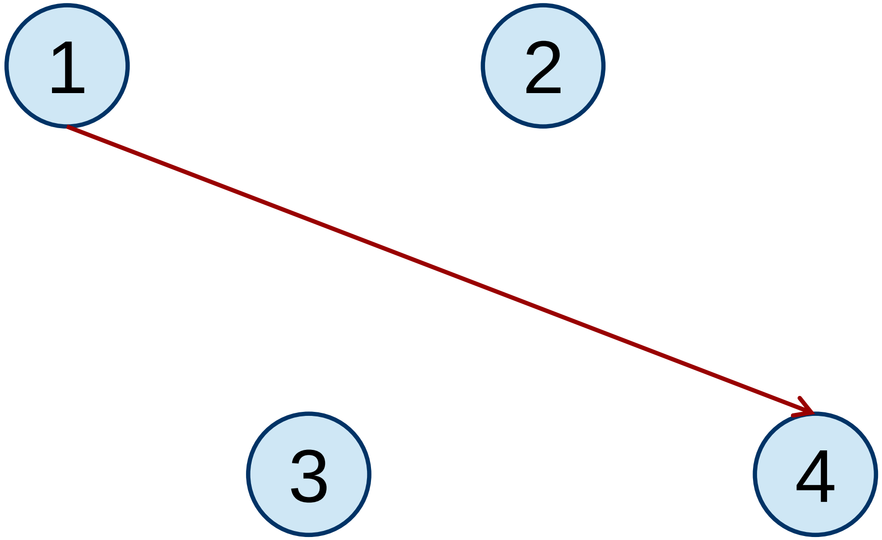


Ejercicio

- Calcular $A^3=A*A*A$.
- Dibujar el grafo representado por A^3.
- ¿Qué puede concluir?

Ejercicio

$$A^3 = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



Caminos en Grafos

- A^k corresponde a la matriz de adyacencia obtenida de multiplicar A por sí misma k veces.
- Las entradas o elementos de A^k representan la cantidad de caminos de longitud k entre los nodos.

Caminos en Grafos

- Grafo (fuertemente) conectado:
- En términos de matrices, si existe un entero positivo k de forma que la matriz $B = I + A + A^2 + A^3 + \ldots + A^k$ es positiva (todos sus elementos diferentes de cero), entonces el grafo es (fuertemente) conectado.
- Si en la matriz B se reemplazan todos los valores mayores a cero por 1, se obtiene la matriz de caminos del grafo.

Caminos en Grafos

- En el ejemplo:
- A^4 y sucesivas corresponden a matrices con sólo ceros, entonces $B = I + A + A^2 + A^3 + \ldots + A^k$ es:

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- La matriz B no es positiva, por lo que el grafo dirigido no está fuertemente conectado.

Caminos en Grafos

- En el ejemplo:
- La matriz de caminos se obtiene al reemplazar todos los valores diferentes de cero por 1:

$$B = \begin{bmatrix} 1 & 2 & 1 & 3 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \longrightarrow C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Nos indica si es posible llegar a cierto vértice desde otro. ¿Cómo será la matriz de caminos de un grafo (fuertemente) conectado?

Algoritmo de Dijkstra

Algoritmo de Dijkstra

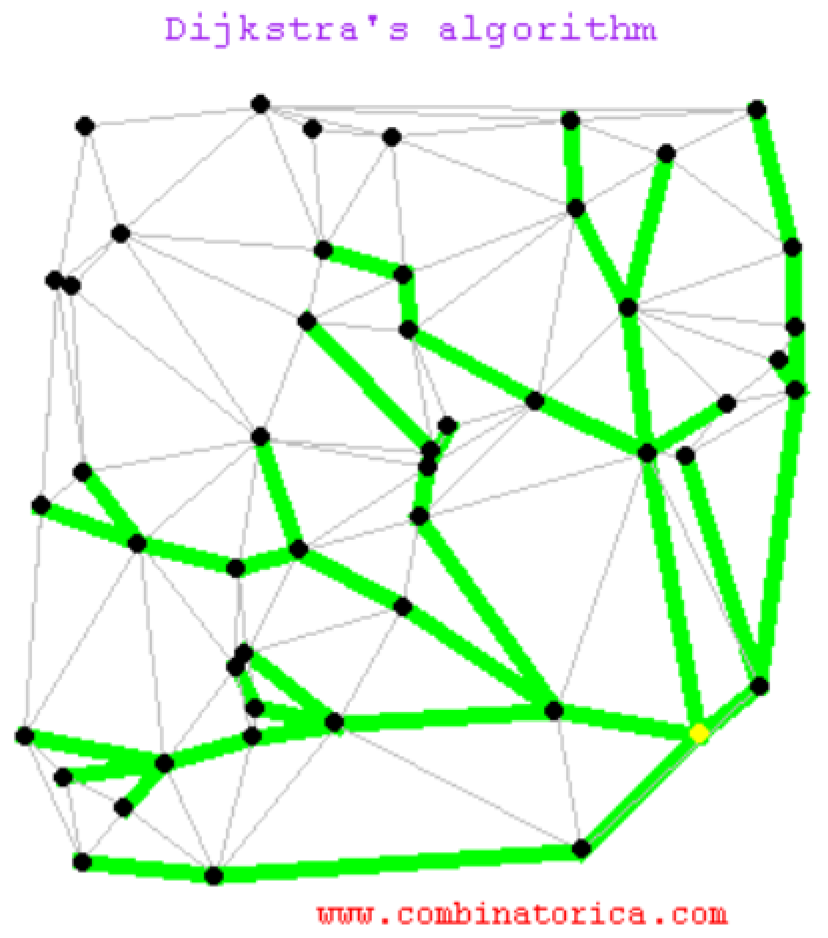
- Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs". Numerische Mathematik 1: 269–271.

- ¿Cómo convertir un grafo en un árbol?
  - Árbol de recubrimiento de “costos mínimos” (*minimal spanning tree*).

- Encontrar TODOS los caminos menos costosos entre un vértice y cualquier otro.

Algoritmo de Dijkstra

- Respuesta al problema de encontrar la ruta más corta a partir de un nodo en el grafo.
- Trabaja sobre un grafo con pesos G = (E,V).
- Desde un vértice s $\in$ V hacia todos los vértices v $\in$ V.



Algoritmo de Dijkstra

- Mantiene un conjunto S de vértices cuyos pesos de la ruta más corta desde el vértice origen han sido ya determinados.
- El algoritmo iterativamente selecciona un vértice $u \in$ V-S con el mínimo estimado de ruta más corta, lo añade a S, y relaja todas las aristas salientes de *u*.
- Relajar una arista (*u*,*v*) implica verificar si se puede mejorar el estimado de costo de la ruta más corta a *v* a través de *u*.

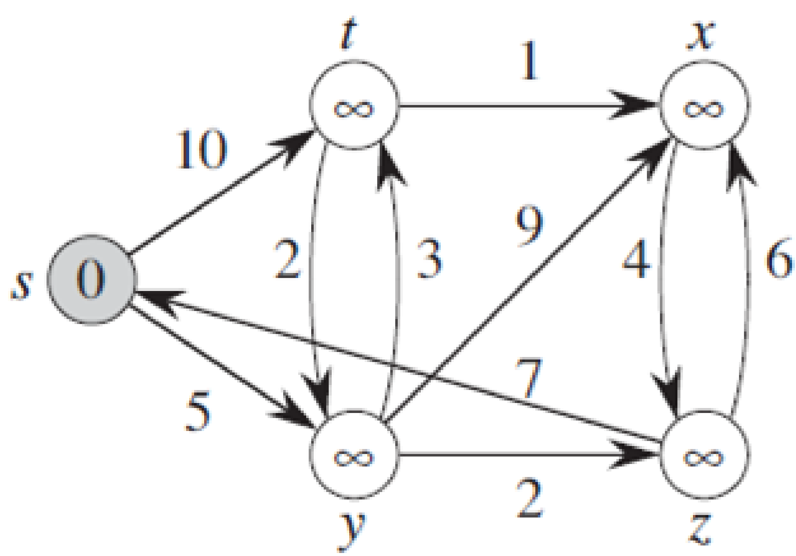
Algoritmo de Dijkstra

Pseudocode

dist[s] ←0	(distance to source vertex is zero)
for all v ∈ V–{s}	
do dist[v] ←∞	(set all other distances to infinity)
S←∅	(S, the set of visited vertices is initially empty)
Q←V	(Q, the queue initially contains all vertices)
while Q ≠∅	(while the queue is not empty)
do u ← mindistance(Q,dist)	(select the element of Q with the min. distance)
S←S ∪ {u}	(add u to list of visited vertices)
for all v ∈ neighbors[u]	
do if dist[v] > dist[u] + w(u, v)	(if new shortest path found)
then d[v] ←d[u] + w(u, v)	(set new value of shortest path)
	(if desired, add traceback code)
return dist	

http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf

Algoritmo de Dijkstra



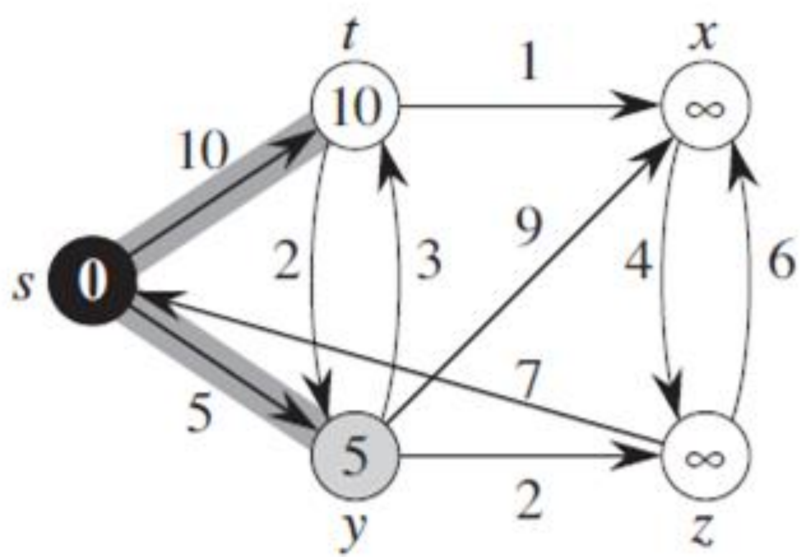
(a)

```
dist[s] ←o
for  all v ∈ V–{s}
       do  dist[v] ←∞
S←∅
Q←V
while Q ≠∅
do   u ← mindistance(Q,dist)
     S←S∪{u}
      for all v ∈ neighbors[u]
            do  if   dist[v] > dist[u] + w(u, v)
                      then      d[v] ←d[u] + w(u, v)


return dist
```

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Dijkstra



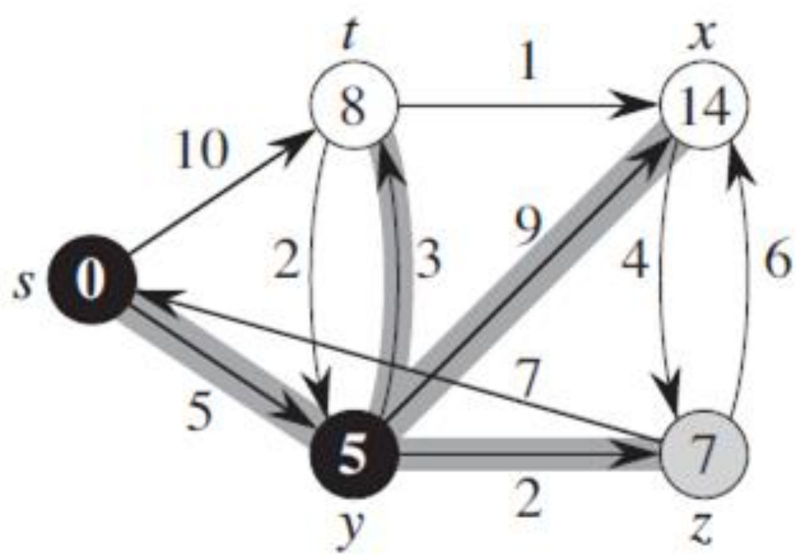
(b)

```
dist[s] ←o
for  all v ∈ V–{s}
        do  dist[v] ←∞
S←∅
Q←V
while Q ≠∅
do   u ← mindistance(Q,dist)
     S←S∪{u}
      for all v ∈ neighbors[u]
            do  if   dist[v] > dist[u] + w(u, v)
                      then      d[v] ←d[u] + w(u, v)


return dist
```

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Dijkstra



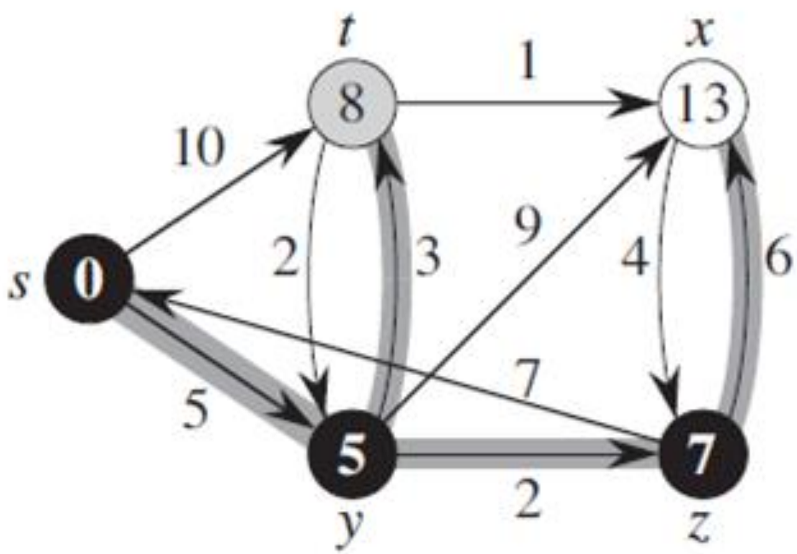
(c)

```
dist[s] ←o
for  all v ∈ V–{s}
      do  dist[v] ←∞
S←∅
Q←V
while Q ≠∅
do   u ← mindistance(Q,dist)
     S←S∪{u}
      for all v ∈ neighbors[u]
           do  if   dist[v] > dist[u] + w(u, v)
                    then      d[v] ←d[u] + w(u, v)


return dist
```

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Dijkstra



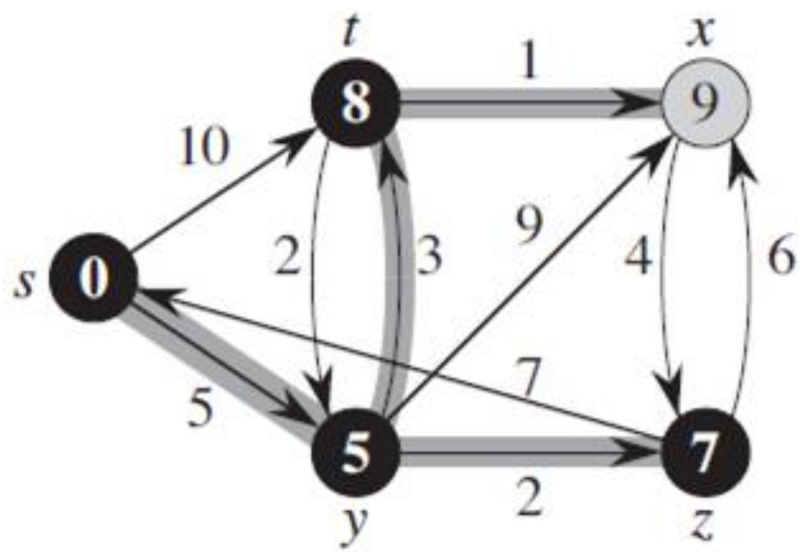
(d)

```
dist[s] ←o
for  all v ∈ V–{s}
        do  dist[v] ←∞
S←∅
Q←V
while Q ≠∅
do   u ← mindistance(Q,dist)
      S←S∪{u}
       for all v ∈ neighbors[u]
              do  if   dist[v] > dist[u] + w(u, v)
                         then      d[v] ←d[u] + w(u, v)


return dist
```

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Dijkstra



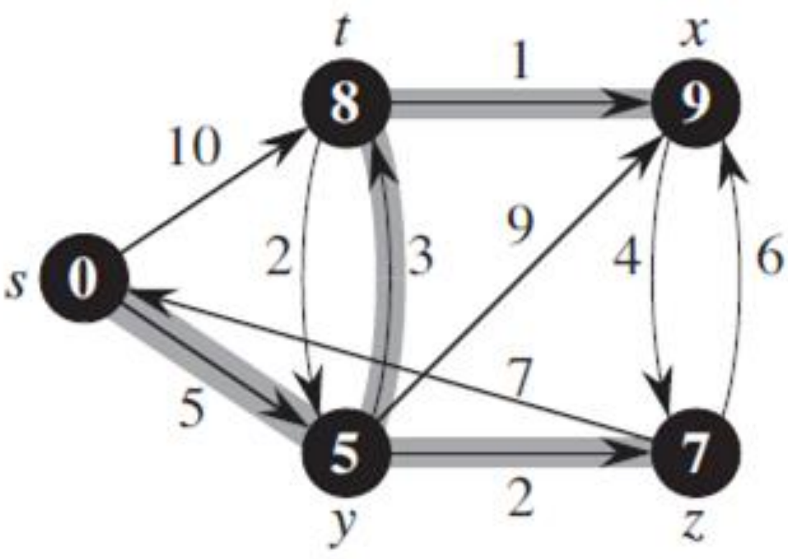
(e)

```
dist[s] ←o
for  all v ∈ V–{s}
       do  dist[v] ←∞
S←∅
Q←V
while Q ≠∅
do   u ← mindistance(Q,dist)
     S←S∪{u}
      for all v ∈ neighbors[u]
            do  if   dist[v] > dist[u] + w(u, v)
                     then      d[v] ←d[u] + w(u, v)


return dist
```

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Dijkstra



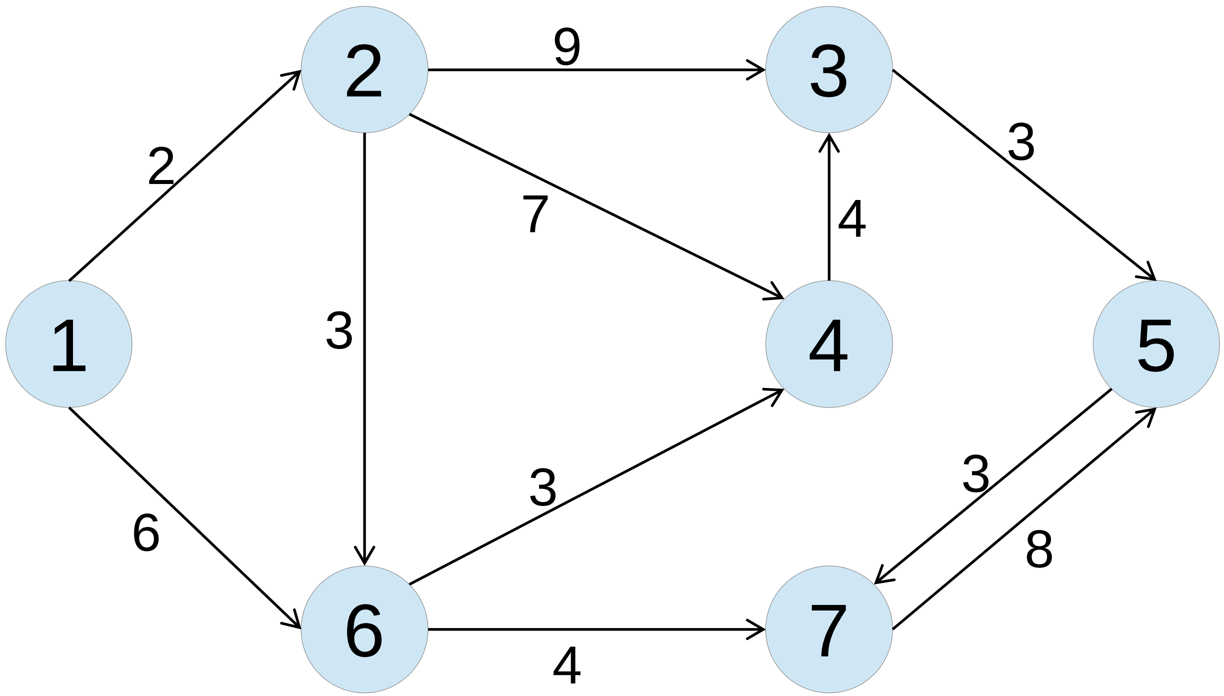
(f)

dist[s] ←o
for all v ∈ V–{s}
    do dist[v] ←∞
S←∅
Q←V
while Q ≠∅
do u ← mindistance(Q,dist)
    S←S∪{u}
    for all v ∈ neighbors[u]
        do if dist[v] > dist[u] + w(u, v)
            then d[v] ←d[u] + w(u, v)

return dist

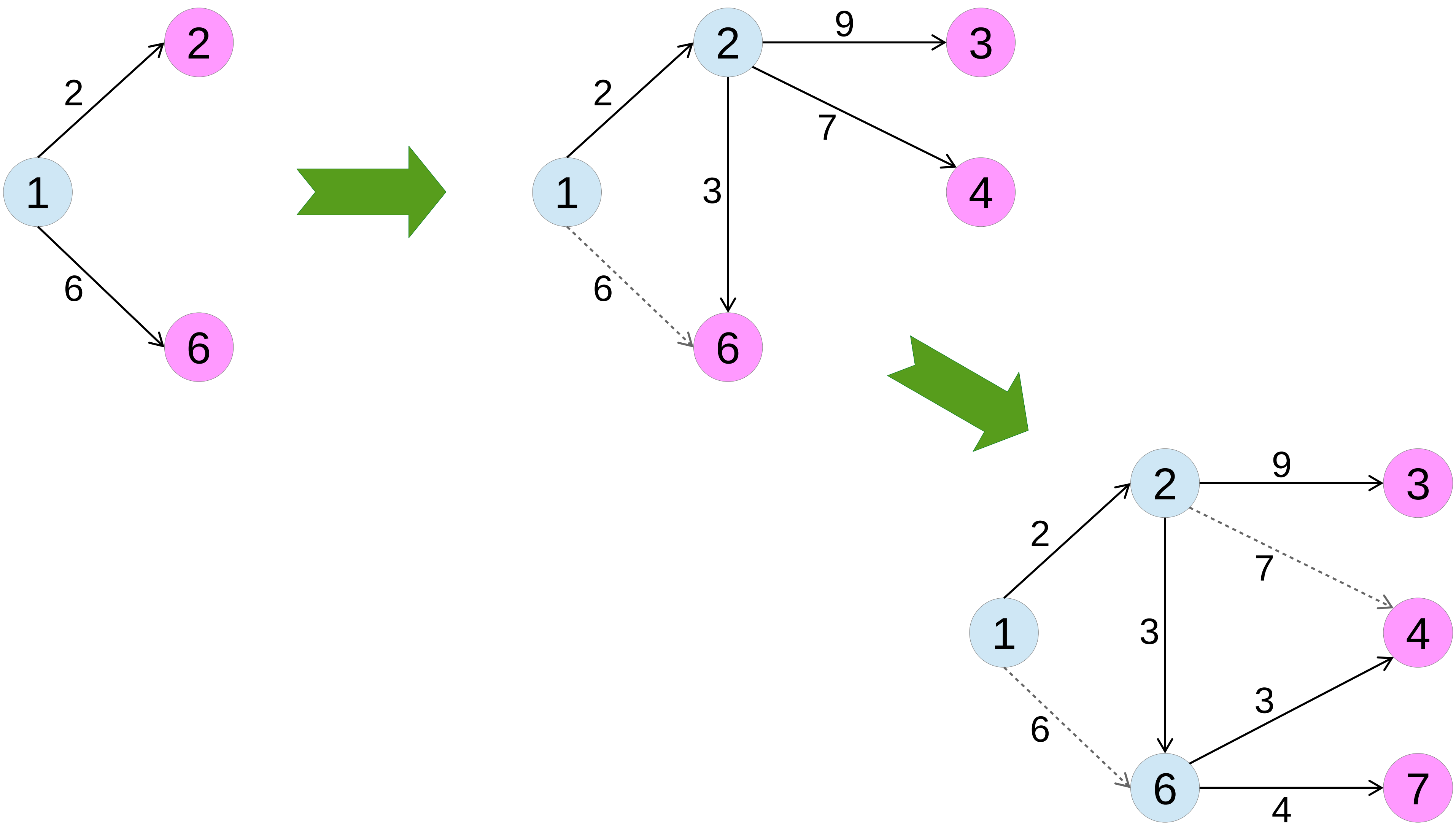
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Dijkstra



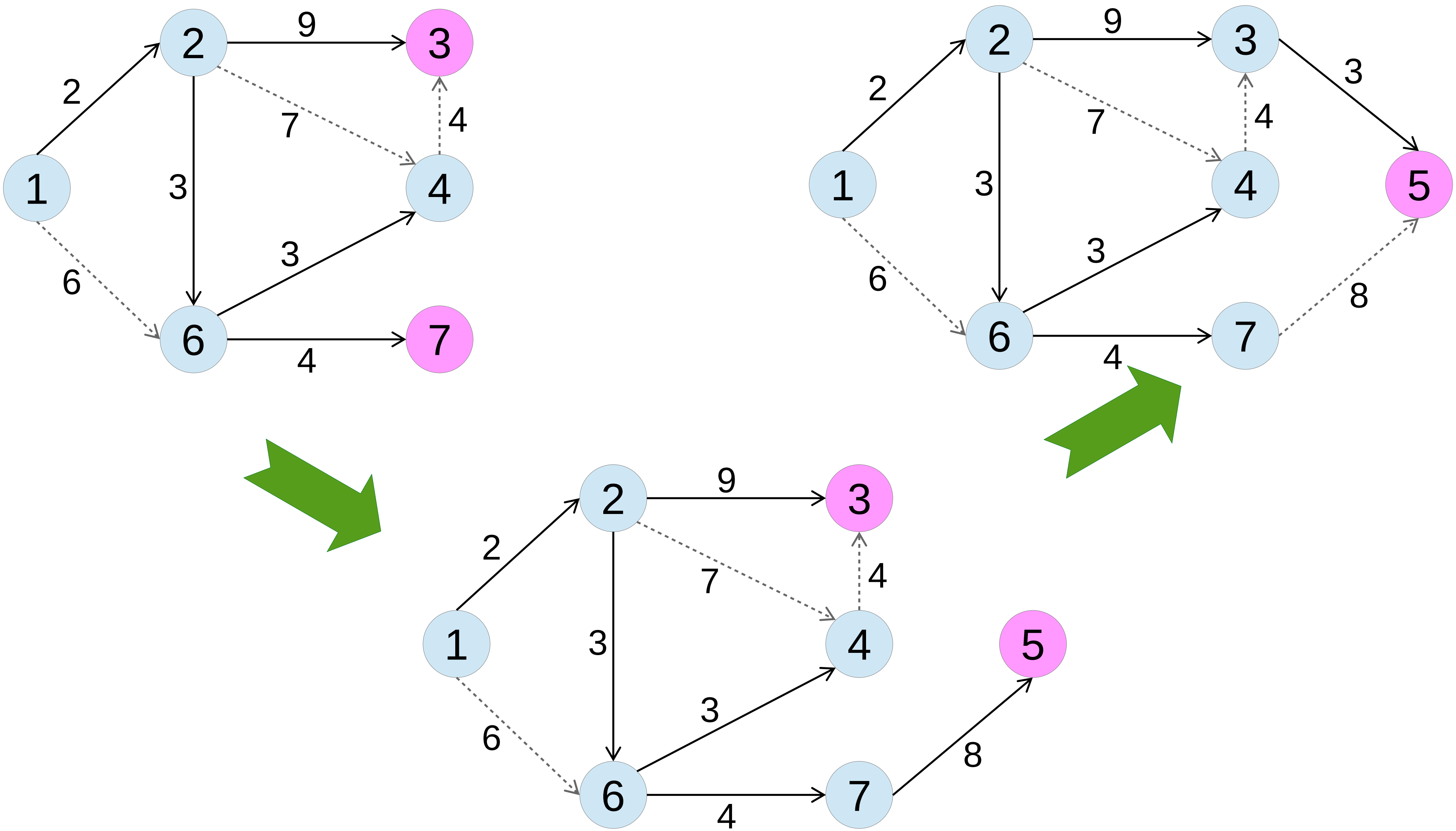
http://www.cosc.canterbury.ac.nz/tad.takaoka/alg/graphalg/graphalg.html

Algoritmo de Dijkstra



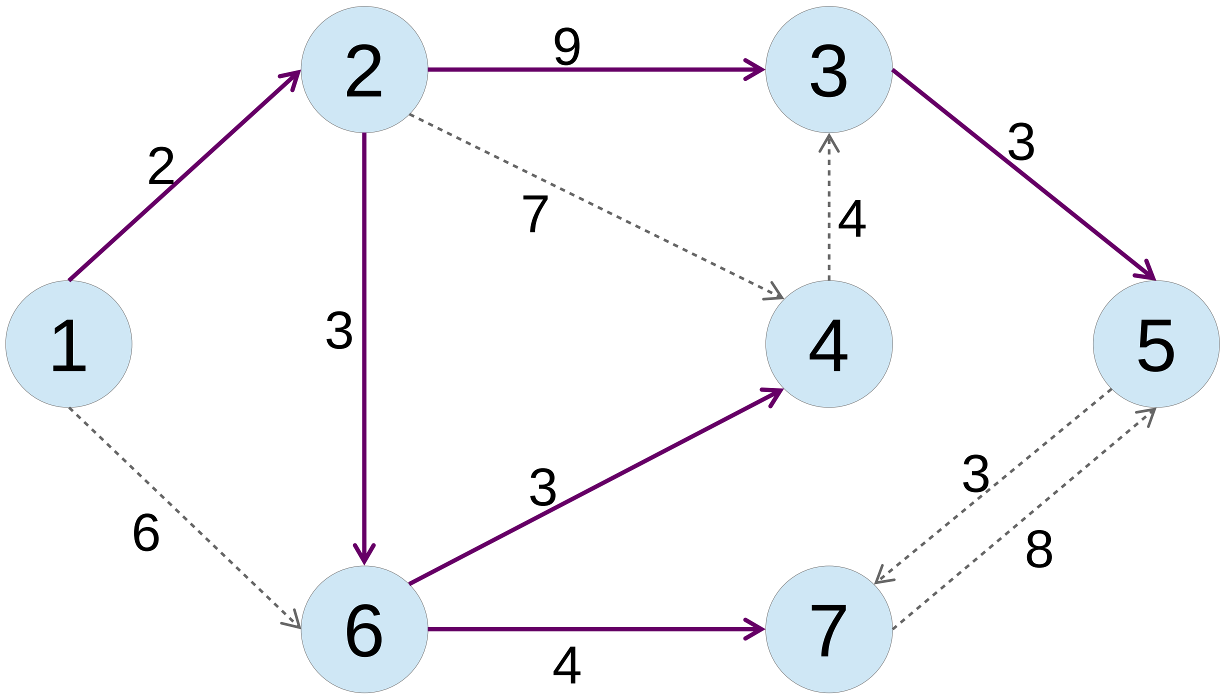
http://www.cosc.canterbury.ac.nz/tad.takaoka/alg/graphalg/graphalg.html

Algoritmo de Dijkstra



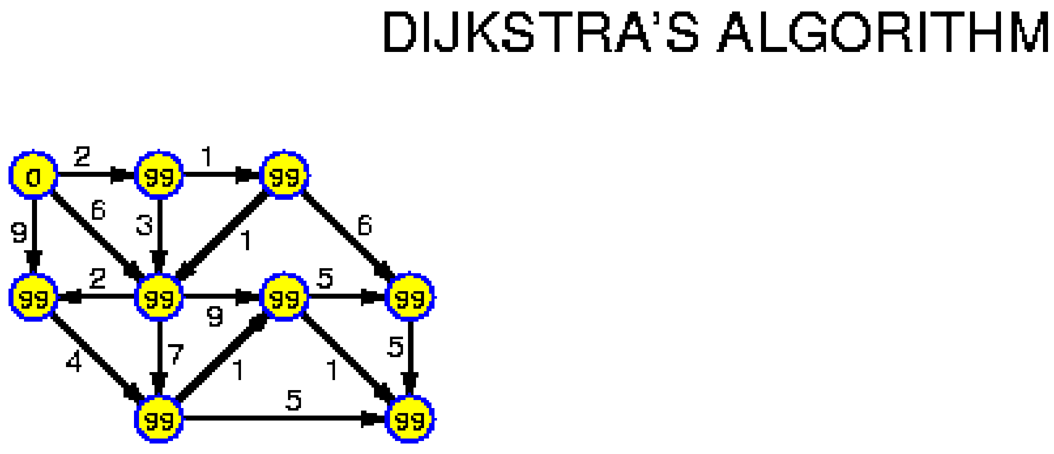
http://www.cosc.canterbury.ac.nz/tad.takaoka/alg/graphalg/graphalg.html

Algoritmo de Dijkstra

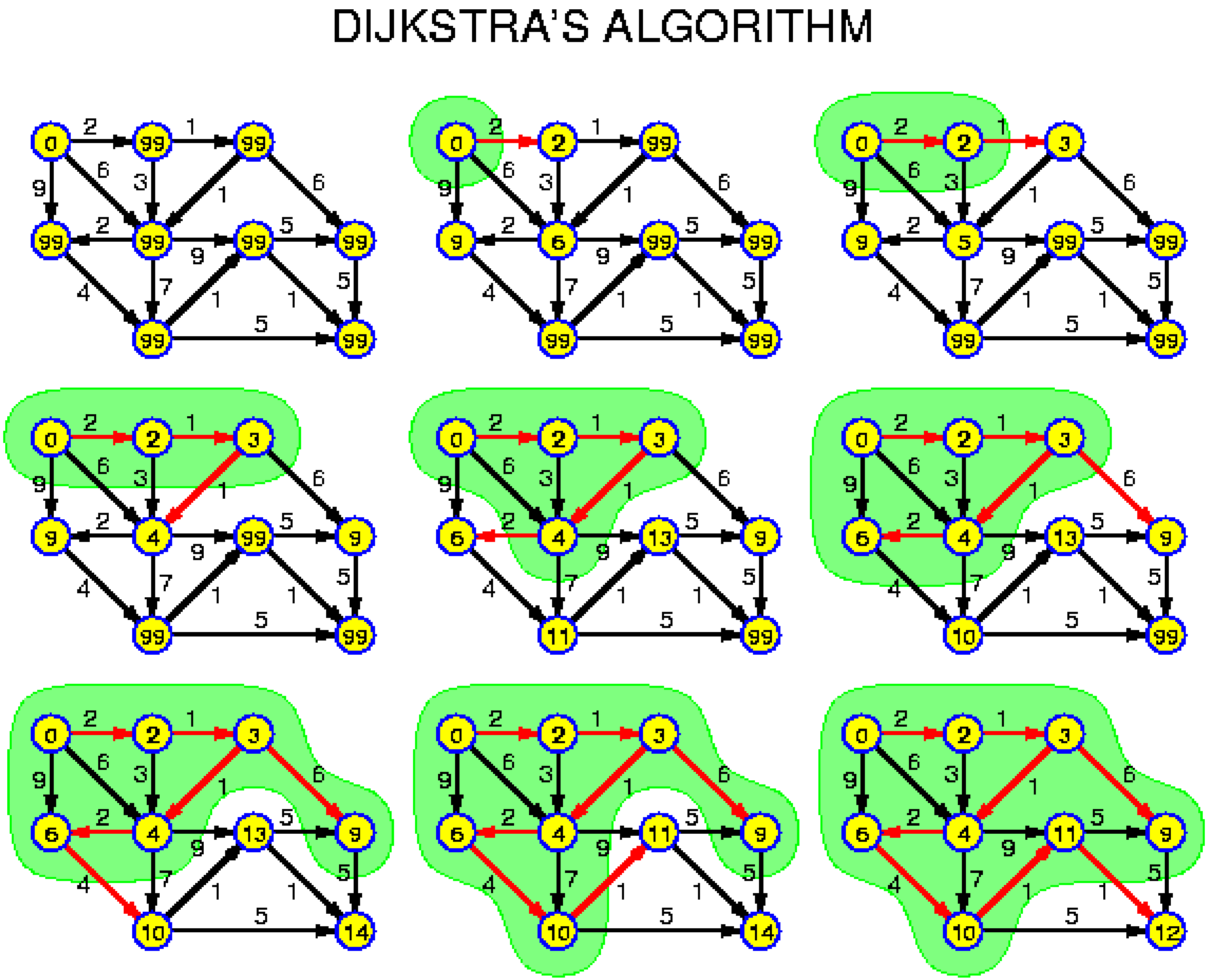


http://www.cosc.canterbury.ac.nz/tad.takaoka/alg/graphalg/graphalg.html

Algoritmo de Dijkstra



Algoritmo de Dijkstra



http://www.iekucukcay.com/?p=65

Algoritmo de Dijkstra

- Applet de demostración:
- www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html

Algoritmo de Dijkstra

Implementación:

- Uso de nodos auxiliares:
  - Vértice.
  - Padre.
  - Costo (***<u>SÓLO POSITIVOS).</u>***
- Cola de prioridades:
  - La prioridad más alta la da el costo más bajo.
  - Árboles RN, AVL, montículos (*heap*).
- Árbol “hacia el padre”.

Algoritmo de Prim

Algoritmo de Prim

- 1930: Vojtěch Jarník (~30 años antes que Dijkstra).
- 1957: Robert C. Prim.
- 1959: Edsger W. Dijkstra.
- También sirve para encontrar un árbol de recubrimiento mínimo (*minimum spanning tree*).
- A diferencia del algoritmo de Dijkstra, soporta costos negativos.

Algoritmo de Prim

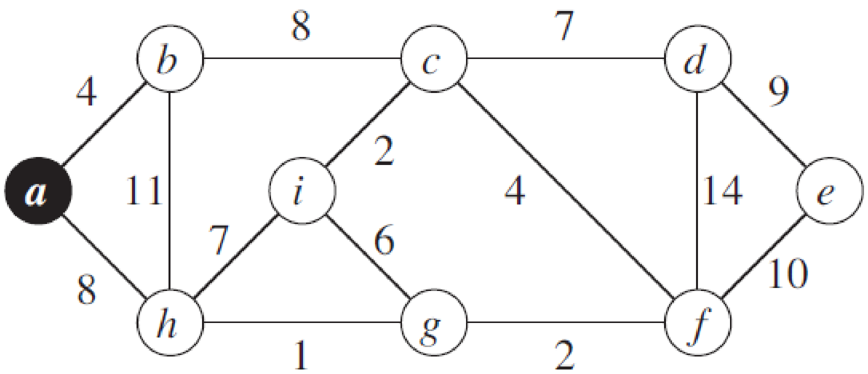
- Sigue la metodología clásica de un algoritmo voraz → hacer en cada paso lo mejor que se pueda hacer.
- Partiendo desde un vértice, va agregando en cada paso la arista con el menor peso que conecte un vértice ya visitado con otro no visitado todavía.
- Su complejidad depende de la estructura utilizada para ordenar las aristas por peso.

Algoritmo de Prim

- Seudo-algoritmo:
- Prim ($G = \{V, E\}$, x)
- $V_{new} = \{x\}$
  $E_{new} = \{\ \}$
- mientras ($V_{new} \neq V$)
- escoger arista $\{u,v\}$ con peso mínimo, u debe estar en V_{new} y v no
- añadir v a V_{new} y $\{u,v\}$ a E_{new}
- fin_mientras

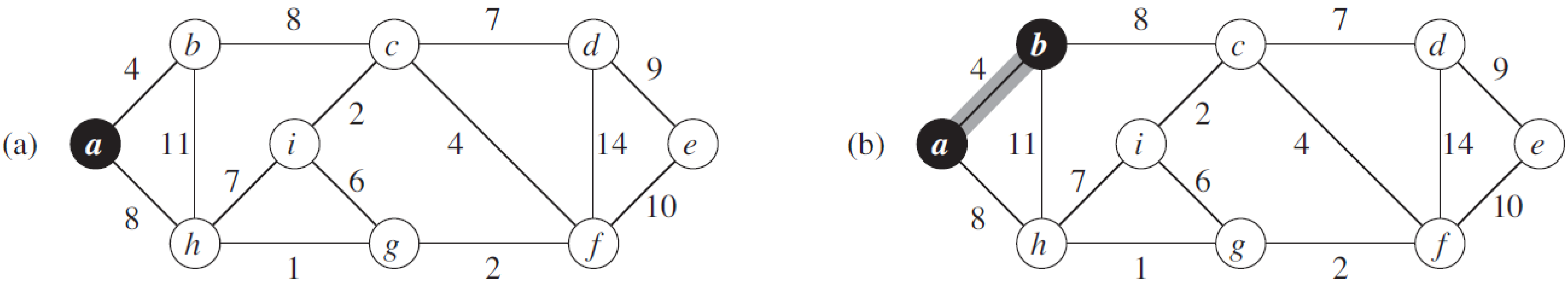
Algoritmo de Prim

(a)



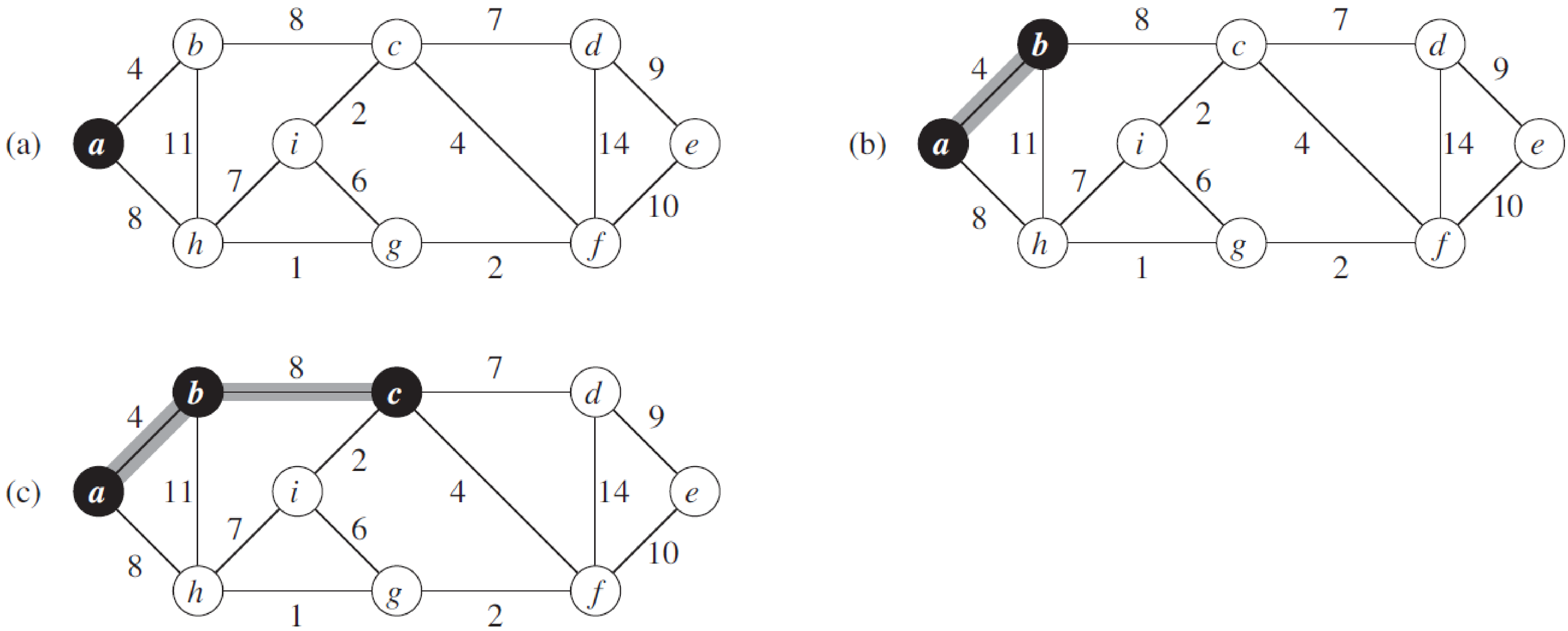
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Prim



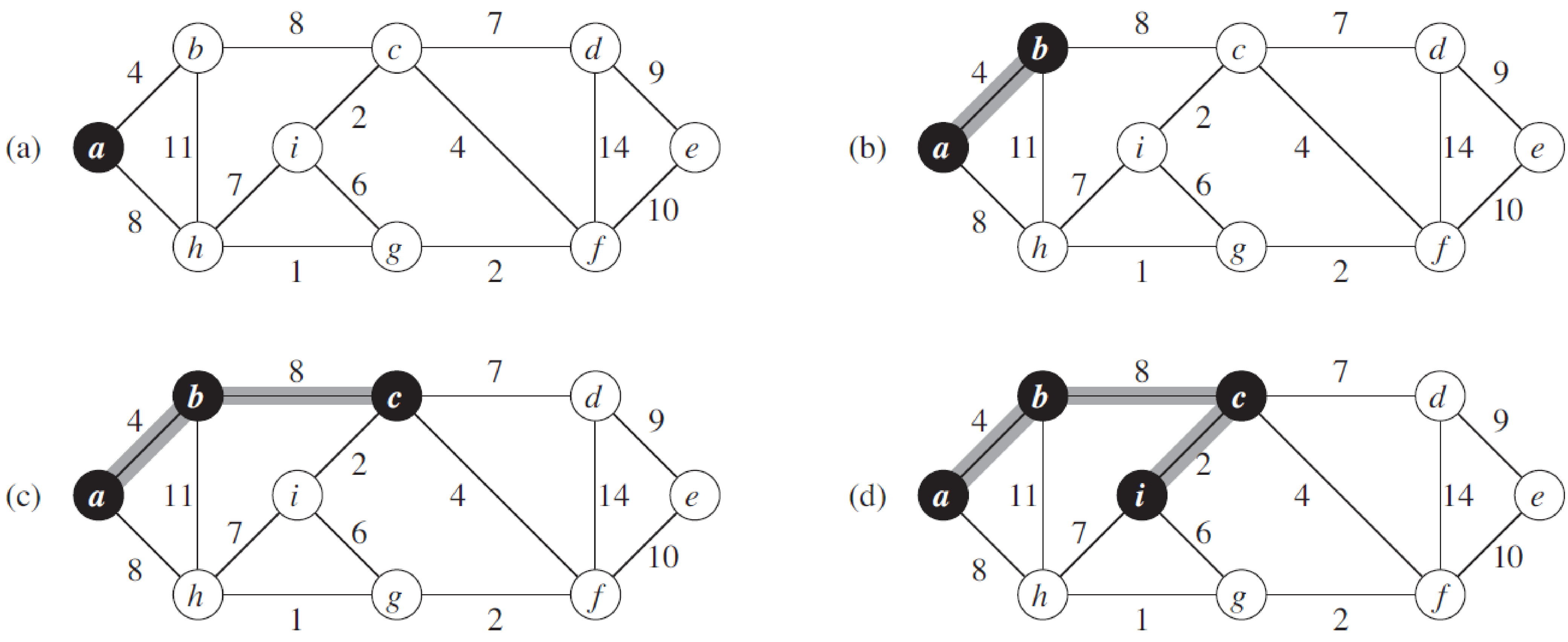
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Prim



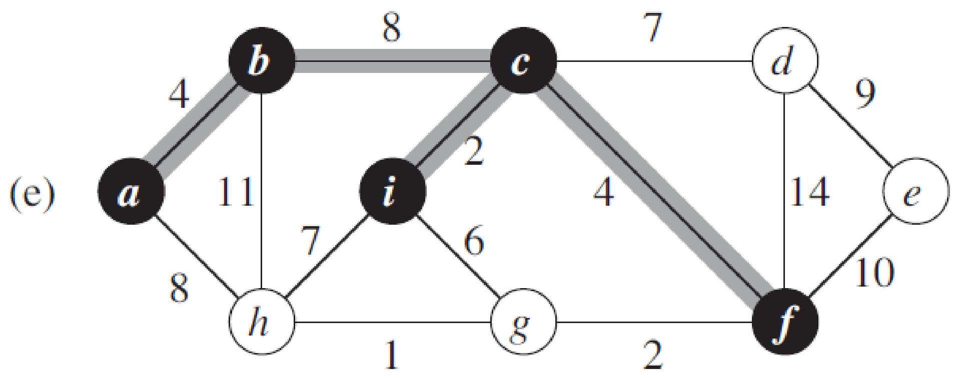
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Prim



Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

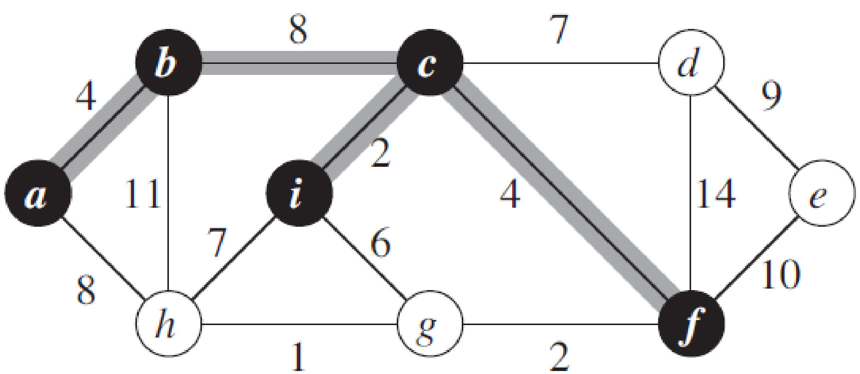
Algoritmo de Prim



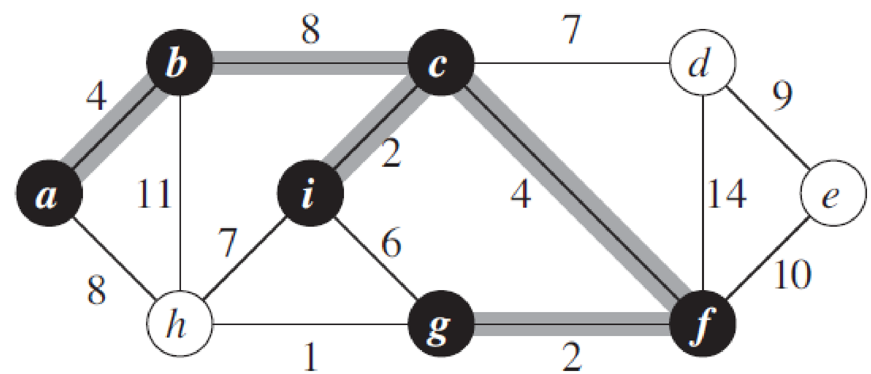
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Prim

(e)



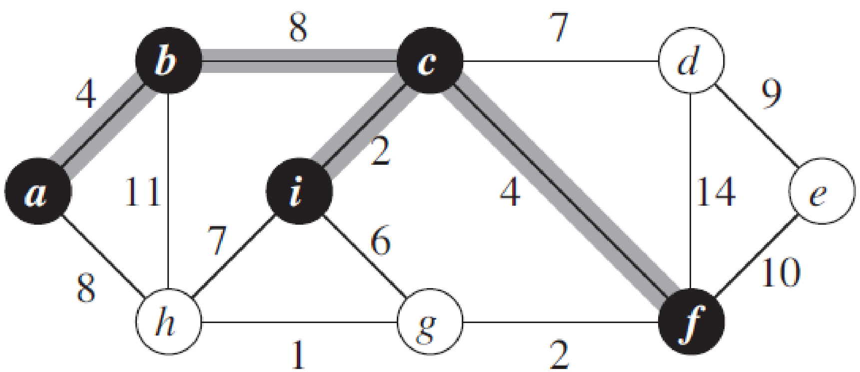
(f)



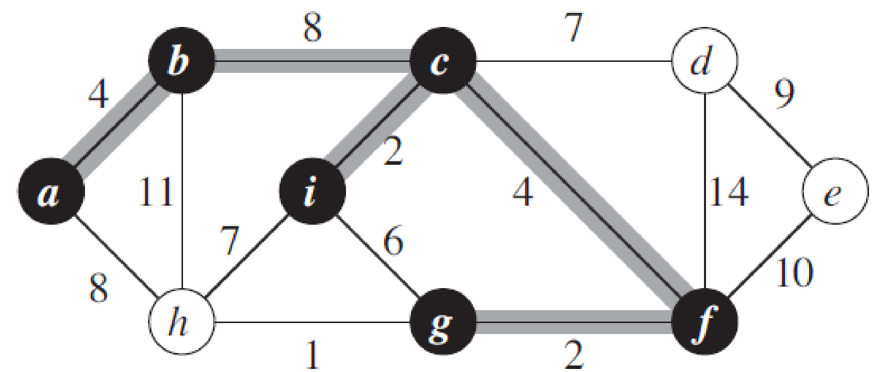
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Prim

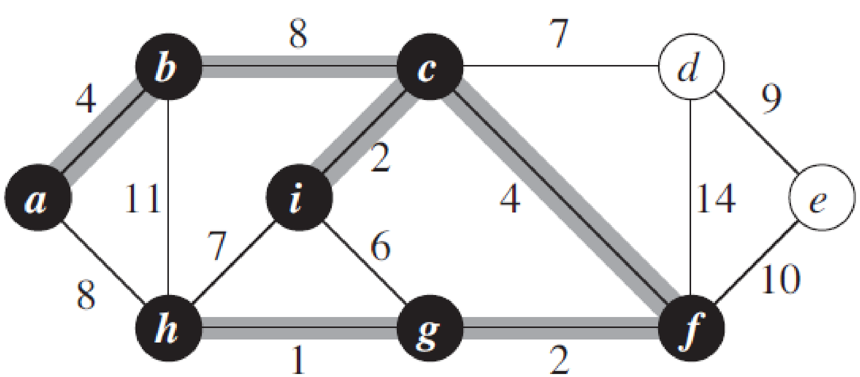
(e)



(f)

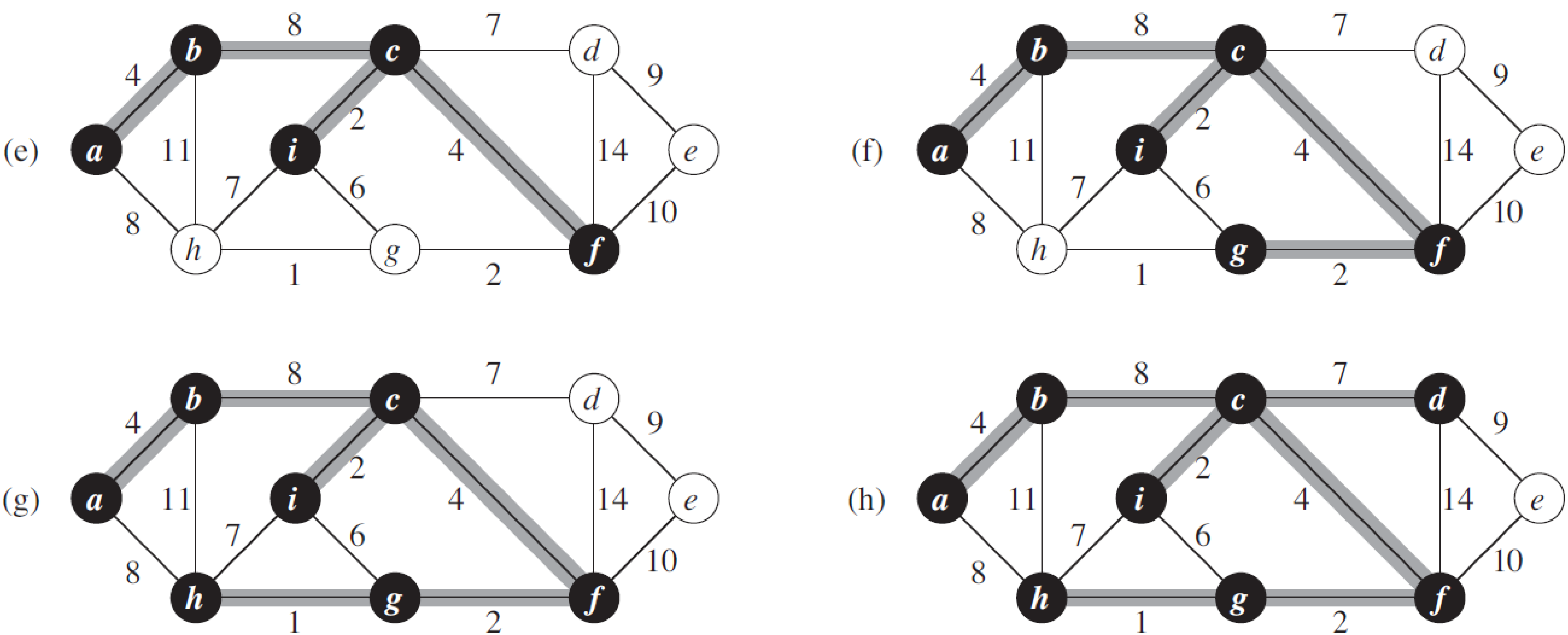


(g)



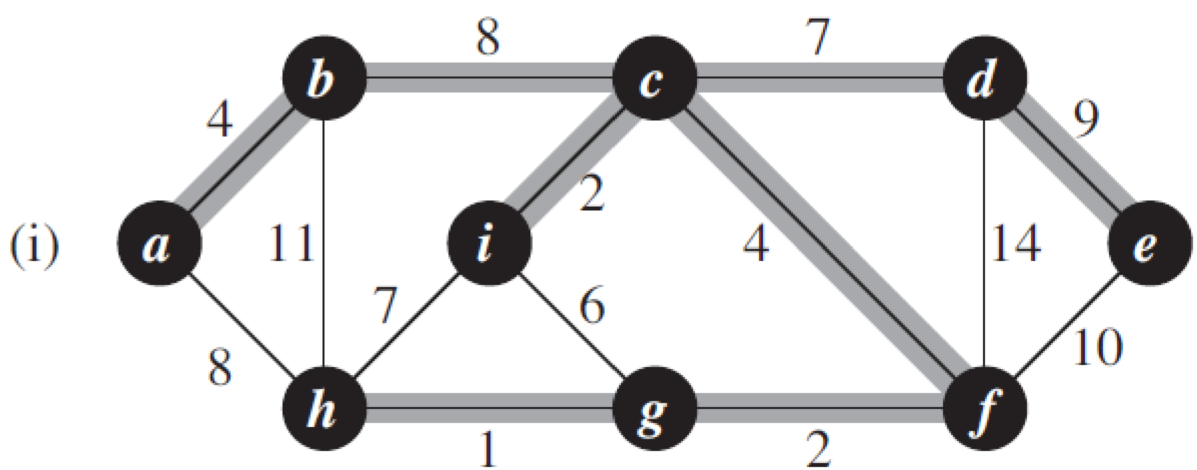
Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

Algoritmo de Prim



Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

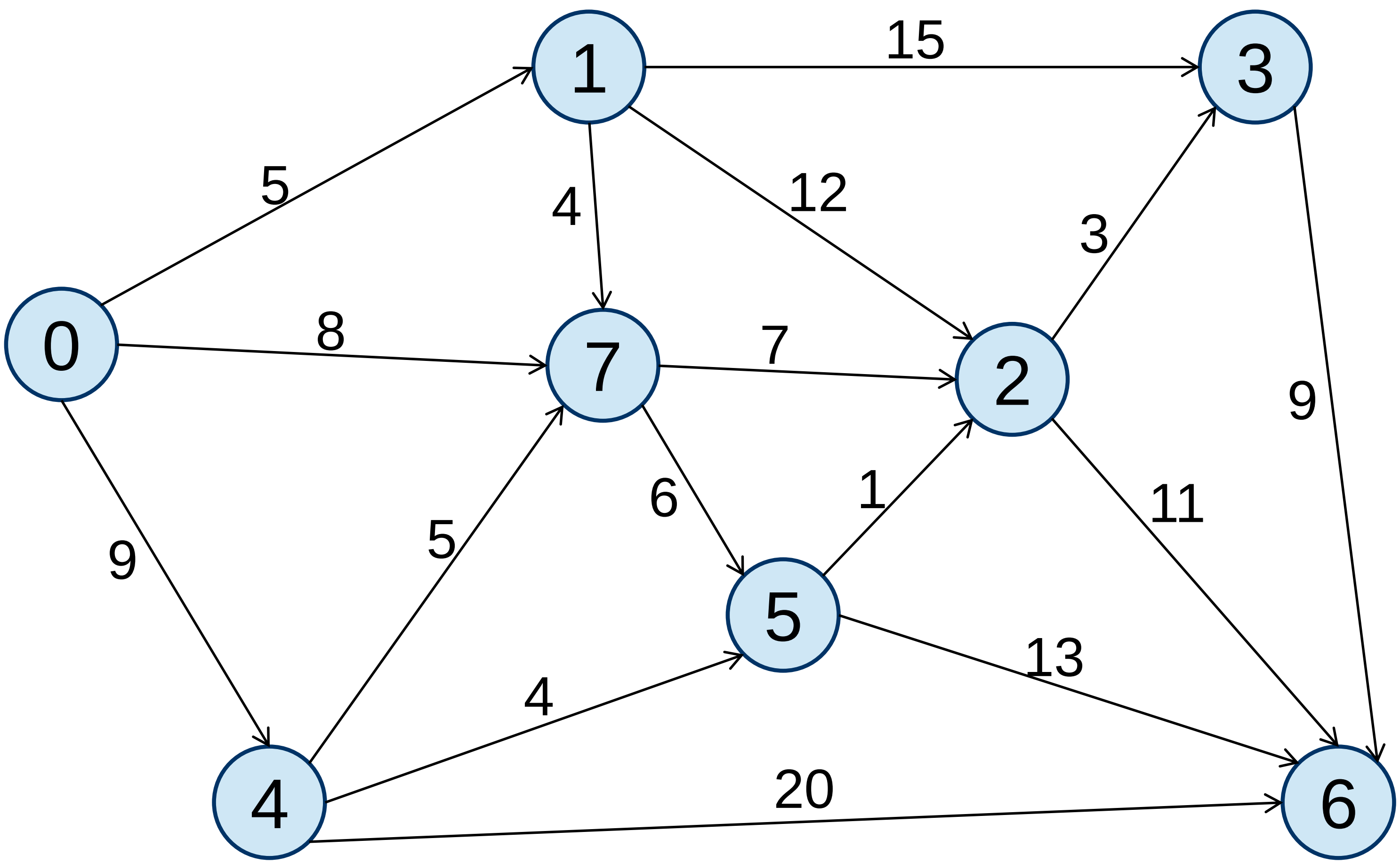
Algoritmo de Prim



Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).

¡ Quiz tarea !

- Sobre el siguiente grafo, aplique los algoritmos de Prim y Dijkstra desde el nodo 0:



Algoritmos

Ejercicios

- Implementar el algoritmo de Prim en el TAD del grafo
- Implementar el algoritmo de Dijkstra en el TAD del grafo

Referencias

- Joyanes, L., Zahonero, I. Algoritmos y estructuras de datos. Una perspectiva en C. McGraw-Hill.
- Kolman, B., Busby, R.C., Ross, S. Estructuras de matemáticas discretas para la computación. Prentice-Hall, Pearson Educación.
- Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). Introduction to Algorithms (Third Ed.).
- www.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture2/lecture2.html

Referencias

- en.wikipedia.org/wiki/Dijkstra's_algorithm
- www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf
- math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Melissa.pdf