Análisis y Diseño de Algoritmos II

Jesús Alexander Aranda Ph.D *Robinson Duque, Ph.D*
Juan Francisco Díaz, Ph. D

Universidad del Valle

jesus.aranda@correounivalle.edu.co
robinson.duque@correounivalle.edu.co
juanfco.diaz@correounivalle.edu.co
Programa de Ingeniería de Sistemas
Escuela de Ingeniería de Sistemas y Computación

El problema de encontrar el camino más corto

How to find the shortest route between two points on a map.

Input:

- Directed graph $G = (V, E)$
- Weight function $w : E \to \mathbb{R}$

Shortest-path weight u to v:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there exists a path } u \rightsquigarrow v\ , \\ \infty & \text{otherwise}\ . \end{cases}$$

Shortest path u to v is any path p such that $w(p) = \delta(u, v)$.

Salida de un algoritmo que encuentra el camino más corto desde un único nodo fuente

For each vertex $v \in V$:

- $v.d = \delta(s, v)$.
    - Initially, $v.d = \infty$.
    - Reduces as algorithms progress. But always maintain $v.d \geq \delta(s, v)$.
    - Call $v.d$ a ***shortest-path estimate***.
- $v.\pi$ = predecessor of v on a shortest path from s.
    - If no predecessor, $v.\pi = \text{NIL}$.
    - π induces a tree—***shortest-path tree***.

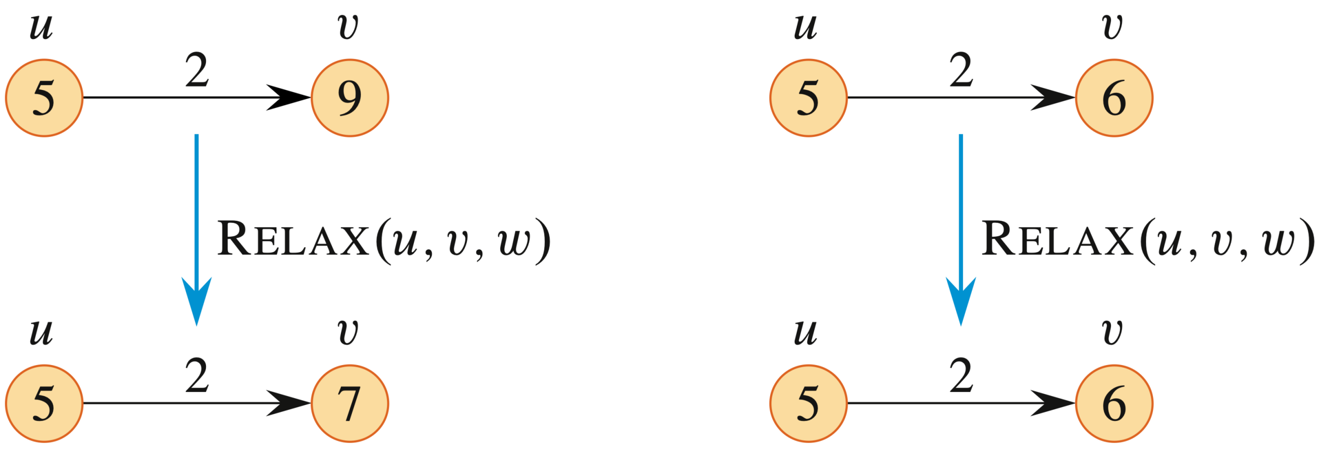
Camino más corto desde un único nodo fuente

INITIALIZE-SINGLE-SOURCE(G, s)

```
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

RELAX(u, v, w)

```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```



(a) (b)

ALGORITMO DE DIJKSTRA

Las aristas no tienen peso negativo.

Una versión con pesos de una búsqueda primero en amplitud.

-- El criterio para seleccionar un nuevo vértice, entre los no seleccionados, corresponde a la distancia mínima que lo separa del vértice fuente (similar a Prim pero no igual).

Se consideran tres conjuntos de vértices:

V = el conjunto de vértices del grafo.
S = el conjunto de vértices para los cuales la distancia mínima que lo separa del vértice fuente ya ha sido determinada.
Q = V - S

ALGORITMO DE DIJKSTRA

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

ALGORITMO DE DIJKSTRA

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

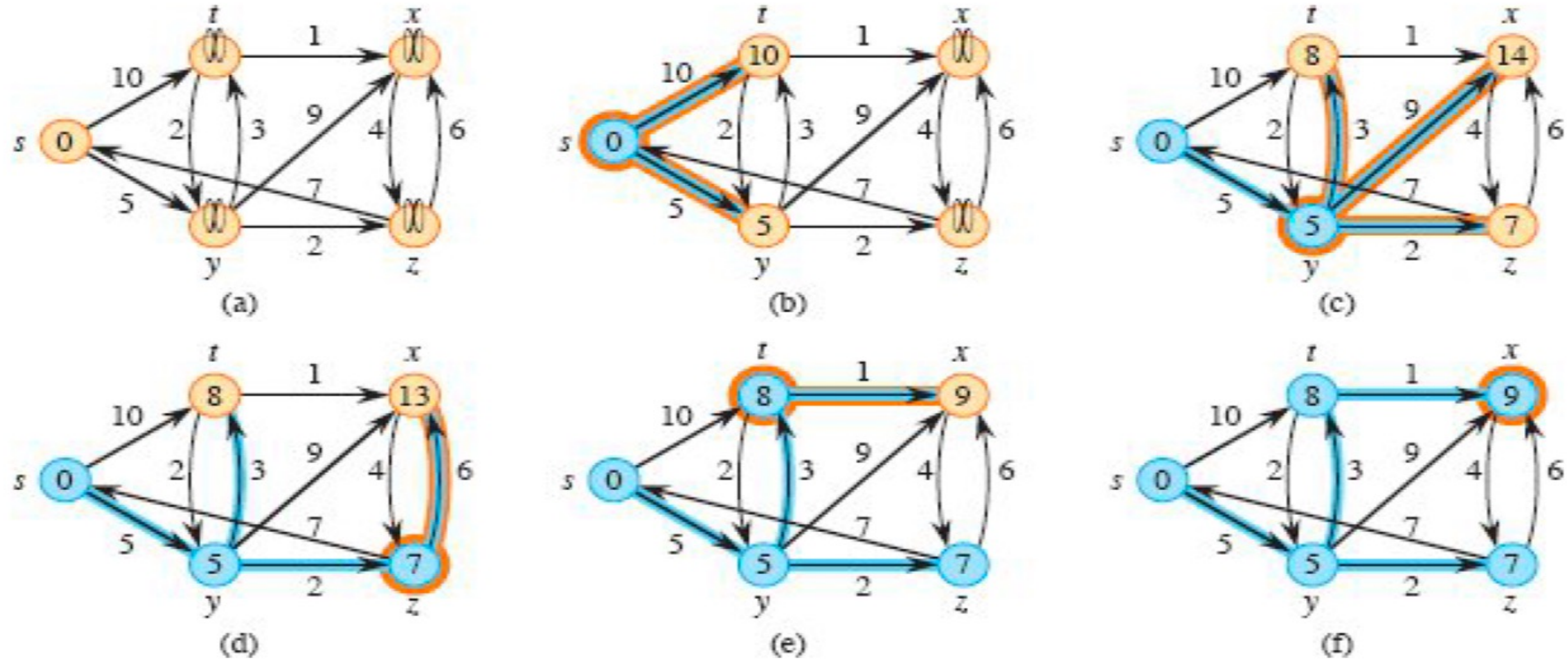
ALGORITMO DE DIJKSTRA

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

Algoritmo de Dijkstra



ALGORITMO DE DIJKSTRA

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = ∅
for each vertex u ∈ G.V
    INSERT(Q, u)
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v in G.Adj[u]
        RELAX(u, v, w)
        if the call of RELAX decreased v.d
            DECREASE-KEY(Q, v, v.d)
```

Q se puede modelar por medio de una cola de prioridad.

La cola de prioridad se puede implementar mediante un montículo-min.

ALGORITMO DE DIJKSTRA

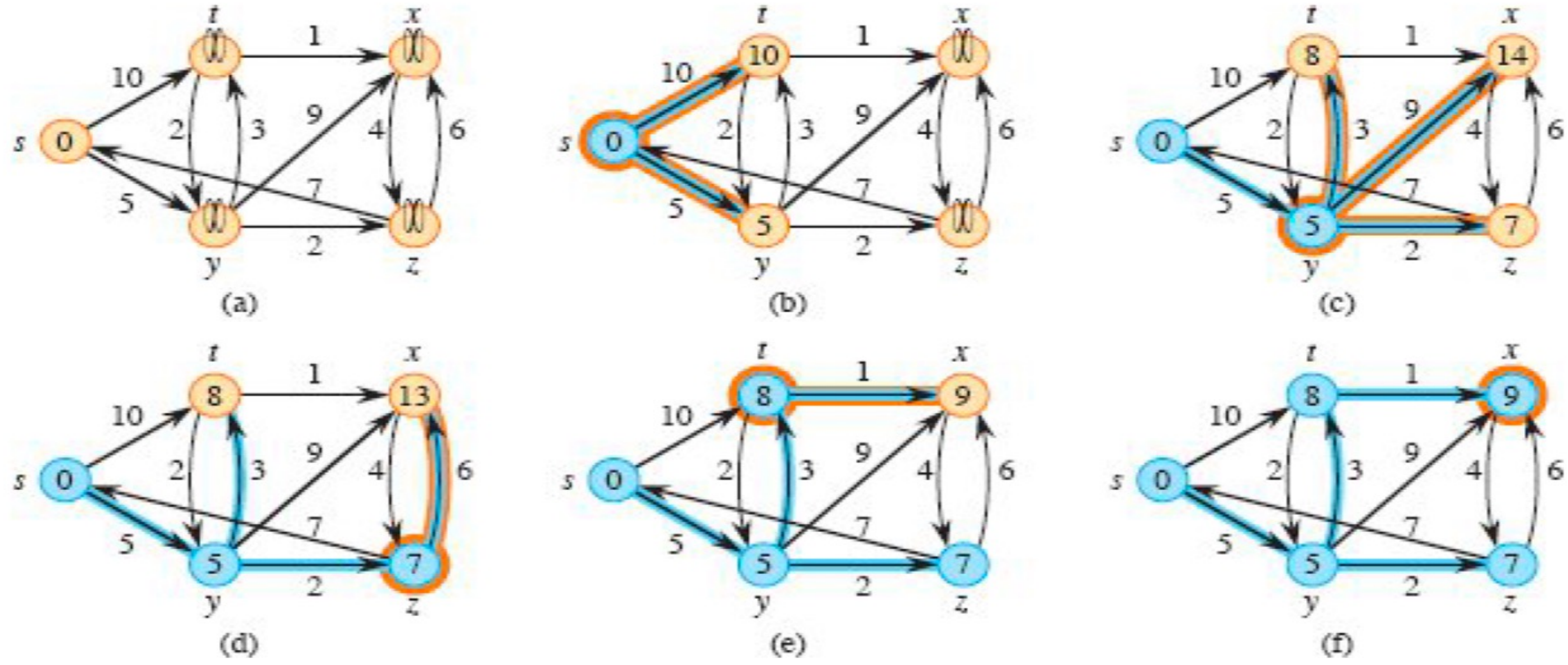
```
DIJKSTRA(G, w, s)
    INITIALIZE-SINGLE-SOURCE(G, s)
    S = Ø
    Q = Ø
    for each vertex u ∈ G.V
        INSERT(Q, u)
    while Q ≠ Ø
        u = EXTRACT-MIN(Q)
        S = S ∪ {u}
        for each vertex v in G.Adj[u]
            RELAX(u, v, w)
            if the call of RELAX decreased v.d
                DECREASE-KEY(Q, v, v.d)
```

Q *se puede modelar por medio de una cola* de prioridad.

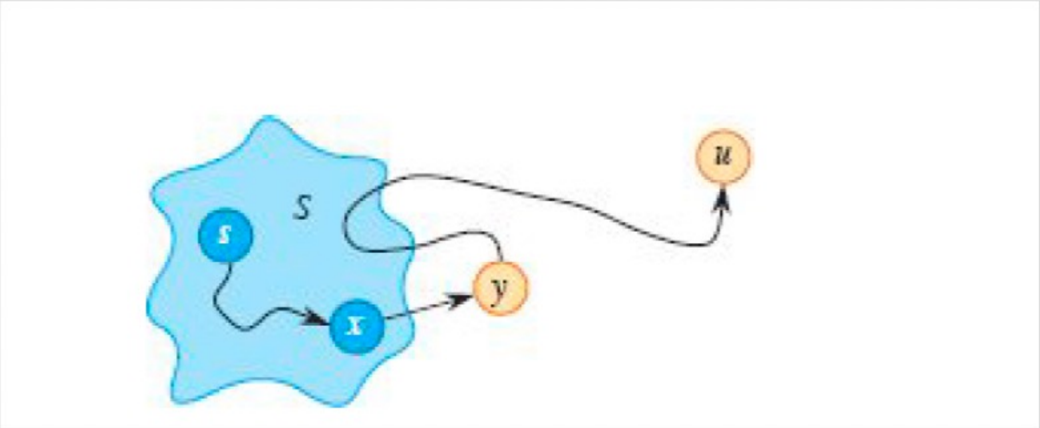
La cola de prioridad se puede implementar mediante un montículo-min.

Operación	Cantidad	Costo por Operación	Costo Total
Insert	|V|	$O(\log \lvert V \rvert)$	$O(\lvert V \rvert * \log \lvert V \rvert)$
Extract-min	|V|	$O(\log \lvert V \rvert)$	$O(\lvert V \rvert * \log \lvert V \rvert)$
Decrease-key	≤ |E|	$O(\log \lvert V \rvert)$	$O(\lvert E \rvert * \log \lvert V \rvert)$

Algoritmo de Dijkstra



Algoritmo Dijkstra



Estrategia Voraz:

Escoger aquel vértice, entre los no seleccionados, que tenga la menor distancia estimada desde la fuente

Correctitud

Se puede demostrar inductivamente que para todos los vertices que hacen parte del conjunto S todas sus distancias estimadas corresponden a las mínimas. Se puede apreciar que el vertice más cercano en V-S puede ser encontrado a partir de un arista que conecte a un vértice en S con él.