# CI2613: Algoritmos y Estructuras III

Blai Bonet

Universidad Simón Bolívar, Caracas, Venezuela

**Enero-Marzo 2015**

# Caminos de costo mínimo en grafos

# Algoritmo de Dijkstra



## Algoritmo de Dijkstra

El algoritmo de Dijkstra computa todas las distancias mínimas a todos los vértices de un grafo $G = (V, E)$ desde un vértice fuente $s$:

– Sólo funciona para **pesos no-negativos** $w : E \to \mathbb{R}^{\geq 0}$

– El algoritmo de Dijkstra es más eficiente que el algoritmo de Bellman-Ford

Al igual que el algoritmo de Bellman-Ford, el algoritmo de Dijkstra ejecuta una secuencia $\sigma$ de relajaciones de aristas del grafo



## Algoritmo de Dijkstra

El algoritmo de Dijkstra mantiene un conjunto $S$ de vértices para los cuales **la distancia más corta desde $s$ ha sido determinada**

El algoritmo iterativamente realiza:

– Selecciona un vértice $u$ en $V \setminus S$ que tenga un valor mínimo $d[u]$

– Agrega $u$ al conjunto $S$

– Relaja todas las aristas que salen de $u$

Dijkstra se implementa con una **cola de prioridad** para mantener el conjunto de vértices $V \setminus S$, y poder seleccionar un vértice $u$ con menor estimado $d[u]$

# Dijkstra: Pseudocódigo

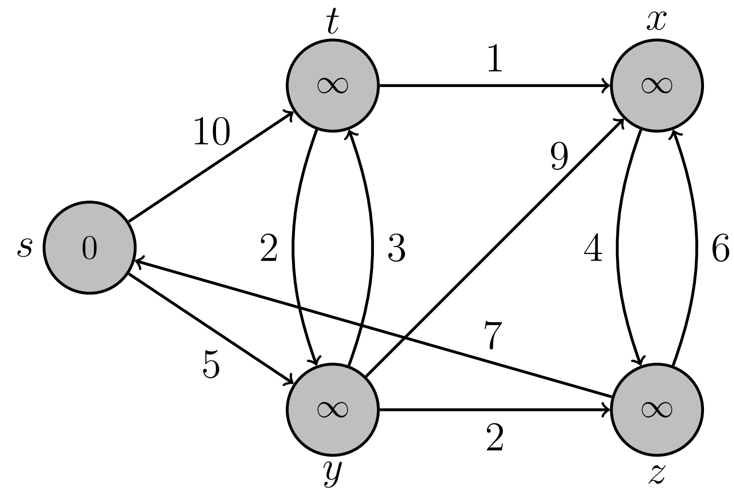
```
void Dijkstra(G, w, s):
    Inicializar-vertice-fuente(G, s)
    S = ∅

    PriorityQueue q
    foreach Vertice v
        q.insert(v)

    while !q.empty()
        u = q.extract-min()
        S = S ∪ { u }
        foreach Vertice v ∈ adyacentes[u]
            Relajar(u, v, w)
```



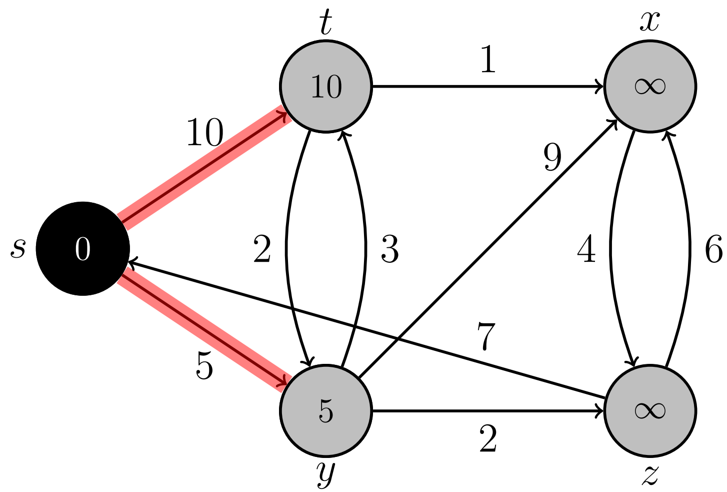

# Dijkstra: Ejemplo







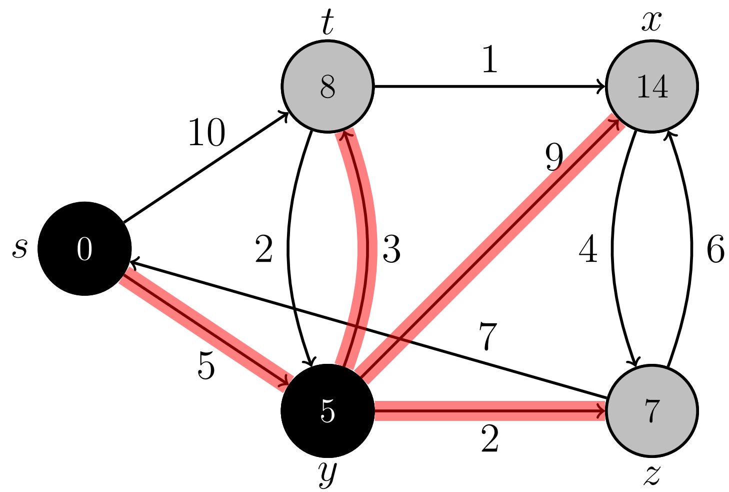
# Dijkstra: Ejemplo



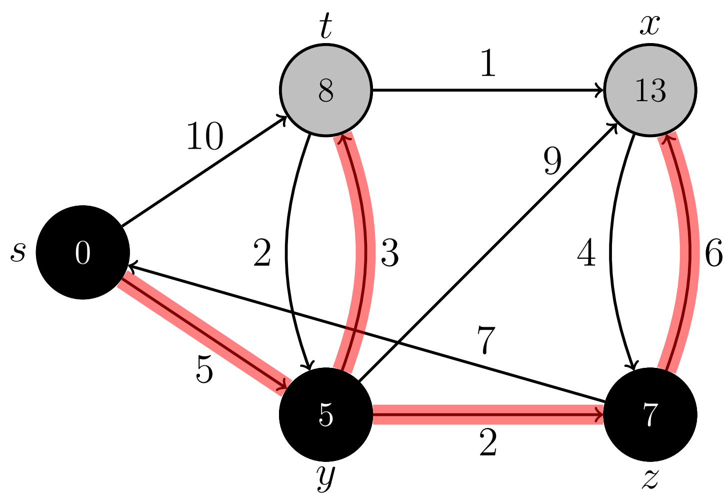




# Dijkstra: Ejemplo

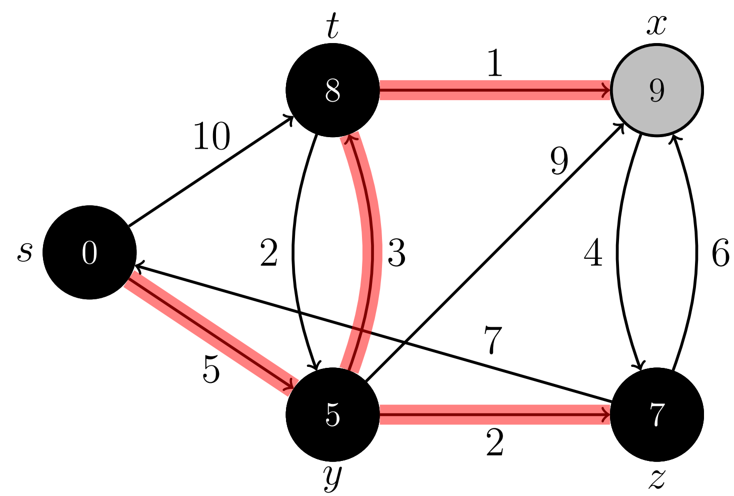


## Dijkstra: Ejemplo



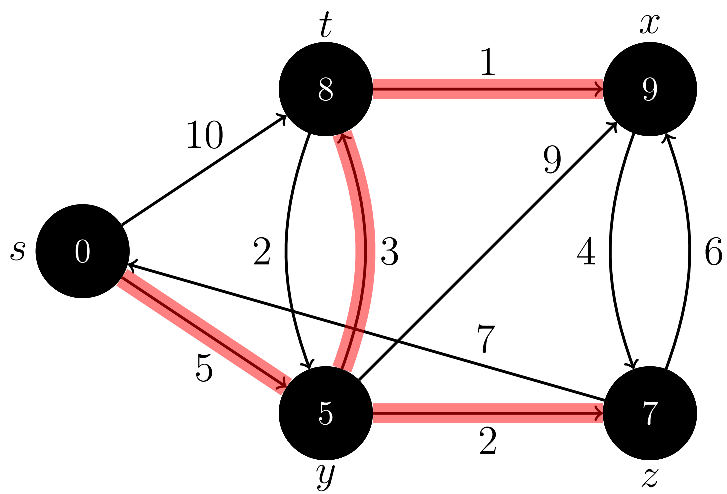

 

## Dijkstra: Ejemplo



## Dijkstra: Ejemplo



## Dijkstra: Pseudocódigo

```
void Dijkstra(G, w, s):
    Inicializar-vertice-fuente(G, s)
    S = ∅

    PriorityQueue q
    foreach Vertice v
        q.insert(v)

    while !q.empty()
        u = q.extract-min()
        S = S ∪ { u }
        foreach Vertice v ∈ adyacentes[u]
            Relajar(u, v, w)
```

# Dijkstra: Análisis

1. La inicialización toma tiempo $\Theta(V)$
2. Las $|V|$ inserciones en la cola toman tiempo $O(V \log V)$
3. Se realizan $|E|$ relajaciones en tiempo $\Theta(E)$
4. Cada relajación involucra un `decrease-key` sobre la cola $q$
5. Se realizan $|V|$ `extract-min` sobre la cola $q$

**Heap binario:** todas las operaciones en 4 y 5 toman tiempo $O(E \log V)$ y $O(V \log V)$ para un tiempo total de $O((E+V) \log V)$

**Heap de Fibonacci:** todas las operaciones en 4 y 5 toman tiempo (amortizado) $O(E)$ y $O(V)$ para un tiemp total de $O(E + V \log V)$




# Dijkstra: Pseudocódigo

```
1  void Dijkstra(G, w, s):
2      Inicializar-vertice-fuente(G, s)
3      S = ∅
4
5      PriorityQueue q
6      foreach Vertice v
7          q.insert(v)
8
9      while !q.empty()
10         u = q.extract-min()
11         S = S ∪ { u }
12         foreach Vertice v ∈ adyacentes[u]
13             Relajar(u, v, w)
```




# Dijkstra: Correctitud

**Teorema**

*Sea $G = (V, E)$ un digrafo con vértice fuente $s$ y pesos no-negativos $w : E \to \mathbb{R}^{\geq 0}$. Al correr el algoritmo de Dijkstra sobre $G$ con fuente $s$, al terminar, $d[v] = \delta(s, v)$ para todo vértice $v$.*

**Prueba:** demostraremos el siguiente invariante:

> *Al inicio de cada iteración del lazo 9–13, $d[v] = \delta(s, v)$ para cada vértice $v \in S$*

Es suficiente mostrar $d[u] = \delta(s, u)$ cada vez que $u$ se inserta en $S$ (línea 11)

Esto porque por el Invariante 1:

> *Una vez que $d[u] = \delta(s, u)$ se hace cierto, la igualdad se mantiene posteriormente*

Utilizaremos inducción en el número de inserciones en $S$




# Dijkstra: Correctitud

**Al inicio (antes de cualquier inserción):** $S$ es vacío y la propieded es cierta trivialmente

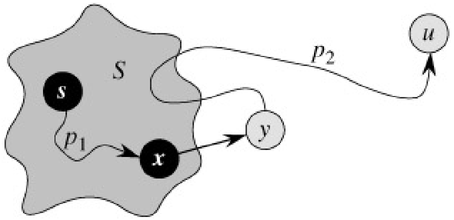
**Paso inductivo**: suponga que la propiedad es cierta despues de $k - 1$ inserciones y considere la $k$-ésima inserción. Sea $u$ el vértice que se inserta

Queremos ver $d[u] = \delta(s, u)$. Para una demostración por contradicción, suponga $d[u] \neq \delta(s, u)$. Consideramos dos casos:

## Dijkstra: Correctitud

1. $\delta(s,u)=\infty$: por el Invariante 1, $d[u]=\infty$ y este caso no puede ser

2. $\delta(s,u)<\infty$: como $d[s]=\delta(s,s)=0$, $u$ no puede ser $s$. Considere un camino de costo mínimo $(s,\ldots,x,y,\ldots,u)$ donde $x\in S$ y $y\in V\setminus S$

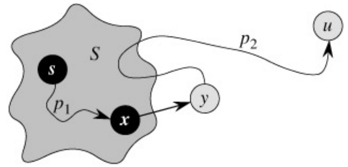



Como $x\in S$, por hipótesis inductiva, $d[x]=\delta(s,x)$ al insertar $x$ en $S$

En ese momento, se relaja la arista $(x,y)$ y por el Invariante 3, $d[y]=\delta(s,y)$ al momento de decolar $u$




## Dijkstra: Correctitud




- Un camino óptimo $(s,\ldots,x,y,\ldots,u)$
- Por optimalidad de subcaminos y costos no-negativos, $\delta(s,y)\leq\delta(s,u)$
- $d[x]=\delta(s,x)$ y $d[y]=\delta(s,y)$
- Como $u$ se decola antes que $y$, $d[u]\leq d[y]$

$$d[y] \;=\; \delta(s,y) \;\leq\; \delta(s,u) \;\leq\; d[u] \;\leq\; d[y]$$

$$d[y] \;=\; \delta(s,y) \;=\; \delta(s,u) \;=\; d[u] \;=\; d[y]$$

En particular, $\delta(s,u)=d[u]$ $\square$




## Dijkstra: Correctitud

**Corolario**

*Al finalizar el algoritmo de Dijkstra, el grafo de predecesores es un árbol de caminos óptimos.*

**Prueba:** aplicar el Invariante 5 ya que por el Teorema, $d[v]=\delta(s,v)$ para todo vértice $v$ al finalizar el algoritmo de Dijkstra $\square$