Algoritmo de Dijkstra

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información

Sobre el algoritmo de Dijkstra

- Resuelve el problema de encontrar el camino de costo mínimo desde un vértice fuente fijo en un digrafo con costos no negativos.
- Se debe cumplir que todo $(u, v) \in E$, se tiene que $w(u, v) \geq 0$.
- Tiene un mejor tiempo que el de Bellman-Ford.
- Mantiene un lista S de vértices a los que se les ha alcanzado el valor del camino de costo mínimo.
- En cada iteración:
  - Se selecciona un $u \in V - S$ con el menor valor estimado del camino de costo mínimo hasta el momento (cola de prioridad).
  - Se agrega u a S.
  - Se relajan los lados adyacentes a u.
- Usa una estrategia Greedy.

Algoritmo de Dijkstra

Procedimiento Dijkstra($G = (V, E)$, s, w)

```
1 inicio
2     Inicializar-fuente-fija(G, s) ;
3     S ← ∅ ;
4     Q ← V ;    /* Cola de prioridad, implícito Insert  */
5     mientras  Q ≠ ∅ hacer
6         u ←Extract-MIN(Q) ;
7         S ← S ∪ {u} ;
8         para cada v ∈ G.adyacentes[u] hacer
9             Relajacion(u,v,w); /* Implícito Decrease-Key  */
```

Análisis del algoritmo de Dijkstra

- INICIALIZAR-FUENTE-FIJA es $\Theta(n)$.
- Línea 4 INSERT para todos los vértices.
- Línea 6 EXTRACT-MIN se ejecuta $|V|$ veces.
- Líneas 8-9 cada lado es examinado solo una vez el ciclo, entonces el **para** se ejecuta $|E|$ veces (análisis agregado).
- Línea 9 DECREASE-KEY se ejecuta $|E|$ veces.

Análisis del algoritmo de Dijkstra, cont.

- **Caso 1: cola de prioridad como un arreglo:** INSERT y DECREASE-KEY son $O(1)$ y EXTRACT-MIN es $O(|V|)$. En consecuencia, es $O(|V|^2 + |E|) = O(|V|^2)$.
- **Caso 2: cola de prioridad como un Min-Heap:** INSERT construye el heap en $O(|V|)$ y DECREASE-KEY y EXTRACT-MIN son $O(\log |V|)$. Entonces, Línea 6 es $O(|V| \log |V|)$ y línea 9 es $O(|E| \log |V|)$. En consecuencia, es $O((|V| + |E|) \log |V|) = O(|E| \log |V|)$.
- **Caso 3: cola de prioridad como Fibinacci-Heap:** INSERT construye el heap en $O(|V|)$, DECREASE-KEY es $O(1)$ y EXTRACT-MIN es $O(\log |V|)$. Entonces, Línea 6 es $O(|V| \log |V|)$ y línea 9 es $O(|E|)$. En consecuencia, es $O(|V| \log |V| + |E|)$.

Ejemplo del algoritmo de Dijkstra

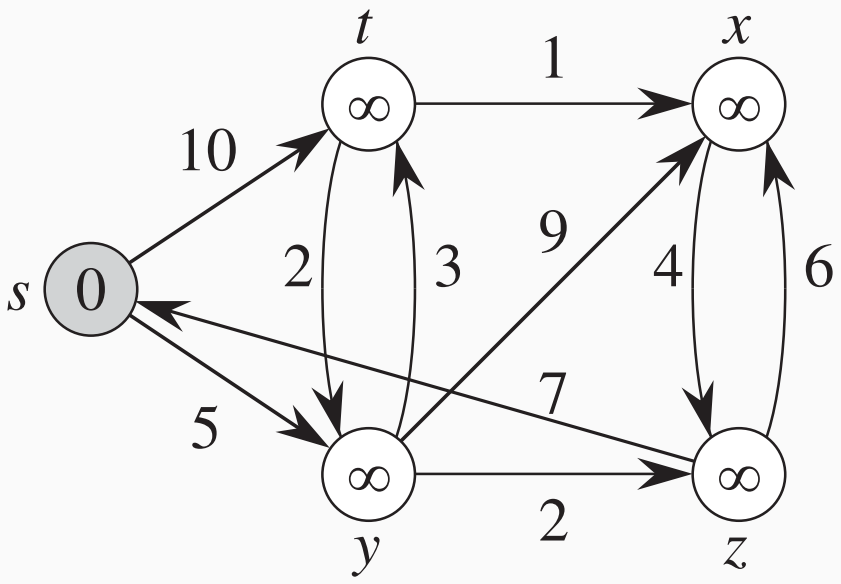


Figura 1: Estado después de INICIALIZAR-FUENTE-FIJA. En gris el vértice que es el primero en la cola de prioridad. Fuente [1].

Ejemplo del algoritmo de Dijkstra, cont.

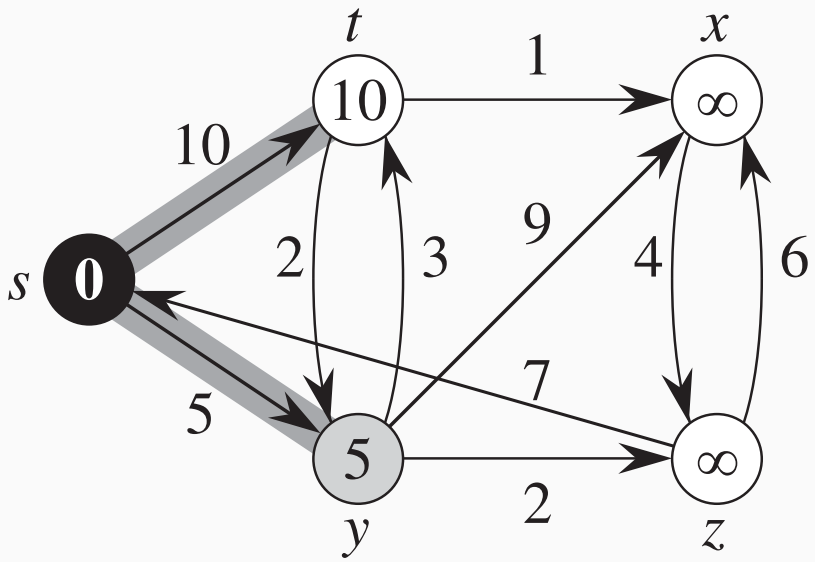


Figura 2: Estado después de la RELAJACION de los vértices adyacentes de s. En negro los vértice en el conjunto S. En gris el vértice que es el primero en la cola de prioridad. Fuente [1].

Ejemplo del algoritmo de Dijkstra, cont.

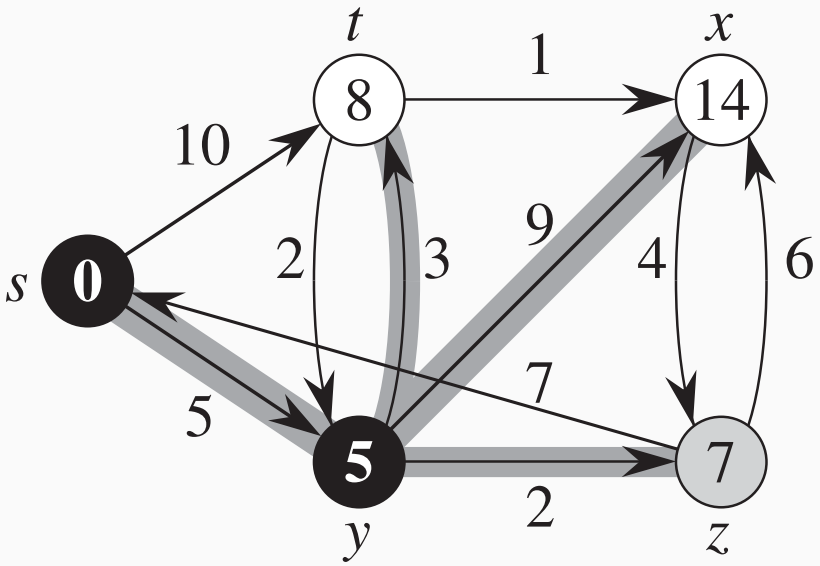


Figura 3: Estado después de la RELAJACION de los vértices adyacentes de y. En negro los vértice en el conjunto S. En gris el vértice que es el primero en la cola de prioridad. Fuente [1].

Ejemplo del algoritmo de Dijkstra, cont.

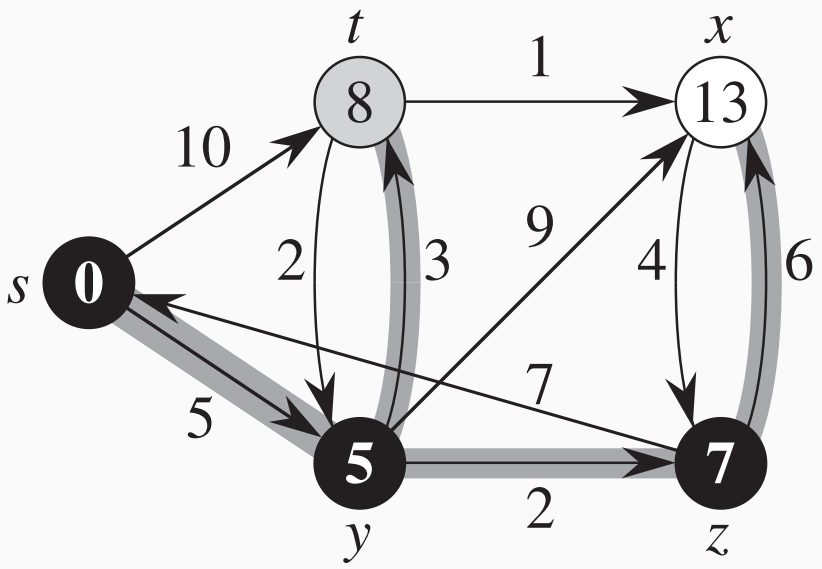


Figura 4: Estado después de la RELAJACION de los vértices adyacentes de z. En negro los vértice en el conjunto S. En gris el vértice que es el primero en la cola de prioridad. Fuente [1].

Ejemplo del algoritmo de Dijkstra, cont.

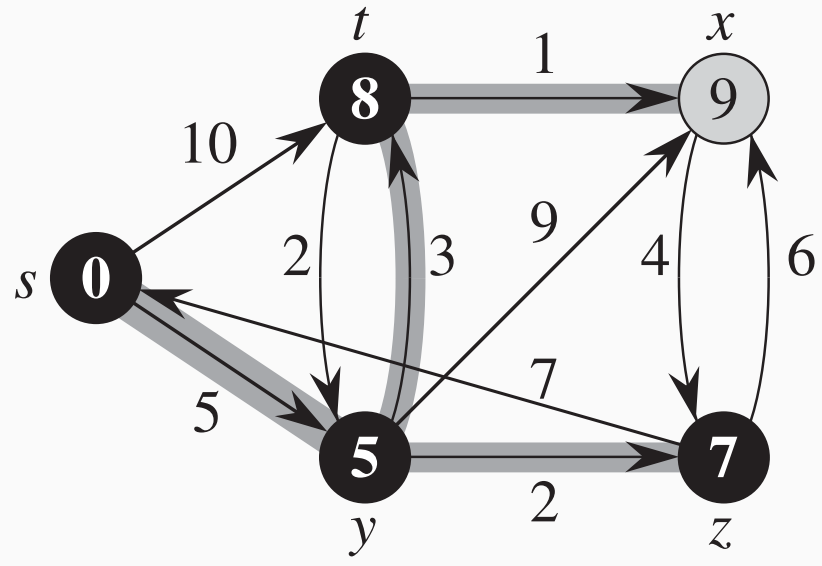


Figura 5: Estado después de la RELAJACION de los vértices adyacentes de t. En negro los vértice en el conjunto S. En gris el vértice que es el primero en la cola de prioridad. Fuente [1].

Ejemplo del algoritmo de Dijkstra, cont.

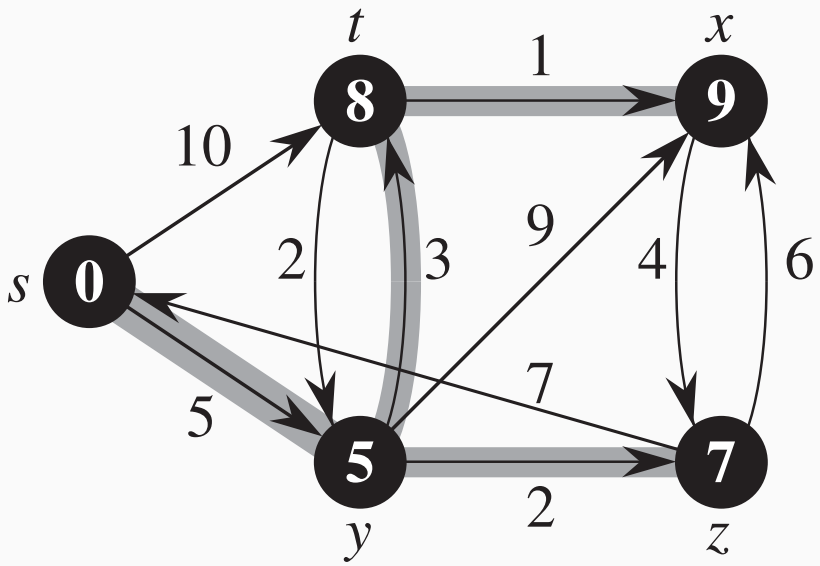


Figura 6: Estado después de la RELAJACION de los vértices adyacentes de x. En negro los vértice en el conjunto S. En gris el vértice que es el primero en la cola de prioridad. Fuente [1].

Correctitud del algoritmo de Dijkstra

Teorema 1 (Correctitud del algoritmo de Dijkstra)

Si se ejecuta el algoritmo de Dijkstra en un digrafo $G(V, E)$, con una función w que asigna un costo no negativo a cada uno de los lados del grafo, y dado un vértice fuente fijo $s \in V$, se tiene que el algoritmo termina con $v.d = \delta(s, v)$, para todos los vértices $v \in V$ del grafo.

Correctitud del algoritmo de Dijkstra, cont.

Prueba

Invariante de ciclo: Al comienzo del ciclo **mientras** (líneas 5-9) se cumple que $v.d = \delta(s, v)$ para todo $v \in S$

Inicialización: inicialmente $S = \emptyset$, por lo tanto es cierto.

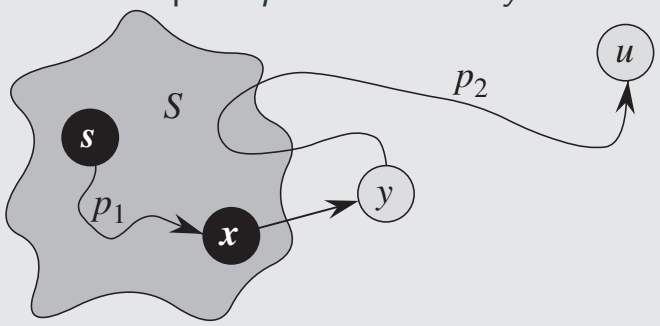
Mantenimiento: se quiere probar que $u.d = \delta(s, u)$ cuando u es agregado a S en cada iteración del ciclo **mientras**. Probamos por contradicción.

- Sea u el primer vértice agregado a S para el cual $u.d \neq \delta(s, u)$
- $u \neq s$ porque $s.d = \delta(s, s) = 0$, entonces $s \in S$ y $S \neq \emptyset$
- Debe haber un camino $s \rightsquigarrow u$, porque de otra forma $u.d = \delta(s, u) = \infty$ por *propiedad de ningún camino*
- Entonces, existe un camino de costo mínimo $s \overset{p}{\rightsquigarrow} u$

Correctitud del algoritmo de Dijkstra, cont.

Prueba, continuación de Mantenimiento

- Antes de que u sea agregado a S, hay un camino en p que conecta a un vértice en S con un vértice de $V - S$
- Sea y ese primer vértice, tal que $y \in V - S$ y sea $x \in S$ su predecesor
- Se descompone p en $s \overset{p_1}{\rightsquigarrow} x \rightarrow y \overset{p_2}{\rightsquigarrow} u$



Correctitud del algoritmo de Dijkstra, cont.

Prueba, continuación de Mantenimiento

Proposición: $y.d = \delta(s, y)$ cuando u es agregado a S

Prueba: $x \in S$ y u es el primer vértice tal que $u.d \neq \delta(s, u)$ es agregado a S, entonces $x.d = \delta(s, x)$ cuando x fue agregado a S. Debido a que hay un lado (x, y) y se aplicó $\textsc{Relajación}(x, y)$, por *propiedad de convergencia*, entonces $y.d = \delta(s, y)$. □

Correctitud del algoritmo de Dijkstra, cont.

Prueba, continuación de Mantenimiento

Mostramos la contradicción de la suposición $u.d \neq \delta(s, u)$

- Sea y un vértice en el camino $s \rightsquigarrow u$, y todos los lados tienen costo no negativo
- Entonces, $\delta(s, y) \leq \delta(s, u)$ esto es:

$$\begin{aligned} y.d &= \delta(s, y) \\ &\leq \delta(s, u) \\ &\leq u.d \text{ (\textit{propiedad de cota superior})} \end{aligned}$$

- Se tiene que y y u estaban en Q cuando se escogió u
- En consecuencia $u.d \leq y.d$, entonces $u.d = y.d$
- Por lo tanto, $y.d = \delta(s, y) = \delta(s, u) = u.d$, lo cual es una contradicción.

Correctitud del algoritmo de Dijkstra, cont.

Prueba, continuación

Terminación: Al terminar $Q = \emptyset$ y $S = V$, entonces $v.d = \delta(s, v)$, para todo $v \in V$.

En consecuencia, el algoritmo de Dijkstra es correcto. □

Referencias

[1] T. Cormen, C. Leirserson, R. Rivest, and C. Stein.
Introduction to Algorithms.
McGraw Hill, 3ra edition, 2009.