

Ruta mas corta

M. C Jorge García.

El problema de la ruta mas corta

Imaginemos que nos encontramos en una ciudad en el punto s y deseamos desplazarnos al punto x . Sabemos el costo de desplazarnos entre todos los puntos en los que es posible tomar una decisión que cambie la ruta. ¿Cual es la ruta de costo mínimo?

Definición: (La ruta mas corta)

Sea una red formada por una grafo conexo G y una función de costo $f : E(G) \rightarrow \mathbb{R}^+$ y sean $s, x \in V(G)$. Deseamos encontrar una sx -trayectoria T en G tal que el costo $\sum_{\forall e_i \in T} f(e_i)$ sea mínimo.

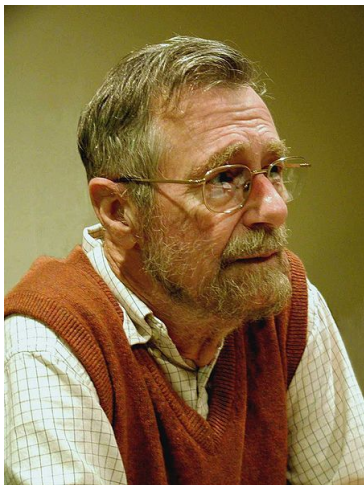


Figura: Edsger Wybe Dijkstra

- Dijkstra (1930 - 2002) fue un científico alemán famoso por ser de las primeras personas en autoconsiderarse un científico de la computación.
- Entre algunas de sus aportaciones están el algoritmo de la ruta mas corta y la notación posfija.
- Recibió el prestigioso premio al articulo mas influyente de la ACM en 2002. Después, el premio seria bautizado como el premio Dijkstra en su honor.
- Escribió todo un articulo en contra de usar el goto en la programación estructurada. ¹

¹A Case against the GO TO Statement. Edsger Dijkstra Technological University of Eindhoven. 1968

El algoritmo de Dijkstra I

- Para ejecutar el algoritmo es necesario un vértice de inicio $s \in V(G)$ y la red f, G
- El algoritmo de Dijkstra hace uso de etiquetas $L(v) = (u, c)$ donde $u \in V(G)$ y $c \in \mathbb{R}^+$.
- Las etiquetas pueden ser temporales o permanentes. El algoritmo empieza haciendo todas las etiquetas temporales y en cada iteración una de ellas se vuelve una etiqueta permanente.
- Cuando las etiquetas de todos los vértices son permanentes el algoritmo termina.
- En ese momento para cada vértice $v \neq s$, la etiqueta $L(v) = (u, c)$ tiene guardado el costo c de la mínima sv -trayectoria y el vértice u que precede a v en dicha trayectoria.

El algoritmo de Dijkstra II

Entradas: Un grafo G una función de costo f y un vértice de inicio $s \in V(G)$.

Salidas: La ruta mas corta desde u a todos los demás vértices y su costo.

- 1: $L(s) = (\emptyset, 0)$
- 2: **for all** $v \in V(G)$ con $v \neq s$ **do**
- 3: $L(v) = (\emptyset, \infty)$
- 4: **end for**
- 5: $Q = V(G)$
- 6: **while** $Q \neq \emptyset$ **do**
- 7: Seleccionar $v \in Q$ con $L(v)$ mínimo
- 8: $Q = Q - \{v\}$
- 9: **for all** $x \in Q$ adyacente a v **do**
- 10: $L(x) = \min\{L(x), L(v) + f(xv)\}$
- 11: **end for**
- 12: **end while**

El algoritmo de Dijkstra III

- Por convención en esta implementación al comparar dos etiquetas $L(u_i)$ y $L(u_j)$ comparamos el costo de las etiquetas.
- Cuando en la línea 10, actualizamos una etiqueta le ponemos el nuevo costo y en su primer miembro al vértice v . Cuando la etiqueta no se actualiza tampoco cambiamos el vértice.
- La lista Q es una cola de prioridad, y guarda todos los vértices cuyas etiquetas aun no son permanentes.
- Se sugiere implementar Q como un heap de fibonacci y a G como una lista de adyacencia.
- Cuando el algoritmo termina podemos encontrar la ruta mas corta desde s a v siguiendo la etiqueta de v , luego la etiqueta del vértice al que nos mando $L(v)$ y así sucesivamente hasta llegar a s . El costo de esa ruta es $L(v)$.

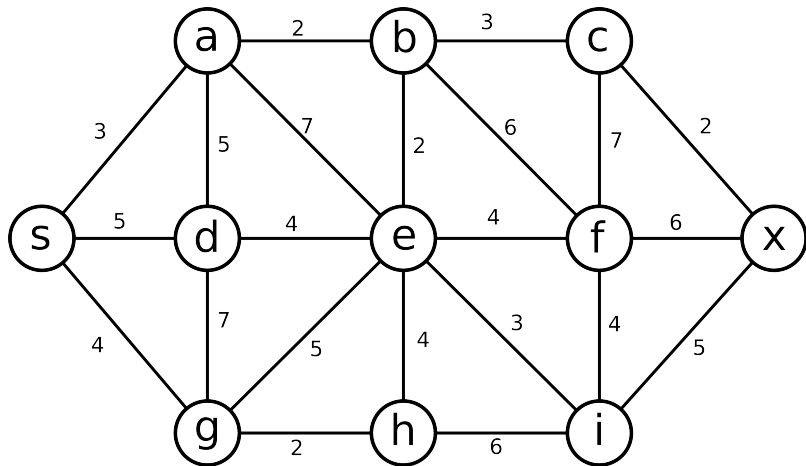


Figura: Calcular la ruta mas corta desde s a x

Teorema:

El algoritmo de Dijkstra es correcto.

Teorema:

El algoritmo de Dijkstra se ejecuta en $O(n^2)$ con $n = |V(G)|$ si se implementa Q como un arreglo.

- El algoritmo de Dijkstra es **óptimo**. Dependiendo de la implementación de Q y de G .
- Una implementación muy eficiente de Q y de G lo pueden hacer funcionar en $O(n \lg n)$