Diseño y Análsis de Algoritmos INF-413

Sergio Hernández
PhD computer science

Universidad Católica del Maule.
shernandez@ucm.cl

Introducción

- Sea $G = (V, E, w)$ un grafo ponderado mediante un conjunto de vértices V y un conjunto de aristas E , tal que $e = (u, v)$ es un elemento de E , $u, v \in V$ y $w_{u,v} \in \mathbb{R}$
- Si hacemos que $T = u, ..., v$ sea una trayectoria que une u con v, entonces podemos calcular el largo de la trayectoria como:

$$D(u, v) = w_{u,\cdot} + \cdots + w_{\cdot,\cdot} + \cdots + w_{\cdot,v} \tag{1}$$

Introducción

- Existen múltiples rutas $T \in \mathcal{T}$ entre los vértices y por lo tanto nos interesa encontrar la menor ruta (camino más corto).
- No todos los vértices pueden ser alcanzados (grafo no conexo).
- Pueden existir múltiples aristas y loops (multi-grafos).

Programación Dinámica

- Si k es un vértice intermedio en la ruta T que une u y v,entonces la distancia $D(u, v)$ satisface la siguiente ecuación de recurrencia:

$$D_j(u, v) = \text{MIN}\left(D_{j-1}(u, v), D_{j-1}(u, k) + w_{k,v}\right) \tag{2}$$

Programación Dinámica

- Si k es un vértice intermedio en la ruta T que une u y v,entonces la distancia $D(u,v)$ satisface la siguiente ecuación de recurrencia:

$$D_j(u,v) = \text{MIN}\left(D_{j-1}(u,v), D_{j-1}(u,k) + w_{k,v}\right) \qquad (2)$$

- El algoritmo Dijsktra recorre los vértices de manera voraz y determina la distancia minina entre un vertice y todos los demas.
- El algoritmo Bellman-Ford recorre las aristas usando el principio de optimalidad.
- El algoritmo Floyd-Warshall utiliza el principio de optimalidad para determinar las distancias minimas entre todos los pares de vértices.

Algoritmo Dijsktra

Require: $G = (V, E, w), u \in V, w : E \mapsto \mathbb{R}^+$

$D(u, u) \leftarrow 0$
$D(u, v) \leftarrow w(u, v)$ para todos los vecinos v de u
$D(u, v) \leftarrow \infty$ para todos los NO vecinos de u
$U \leftarrow u$
while $U \neq V$ **do**
  $k \leftarrow$ vertice menor valor $D(u, k)$ tal que $k \in V \setminus U$
  $D(u, v) \leftarrow \mathrm{MIN}(D(u, v), D(u, k) + w_{k,v})$ para todos los vecinos de k
  $U = U \cup k$
end while
return D

Ejemplo

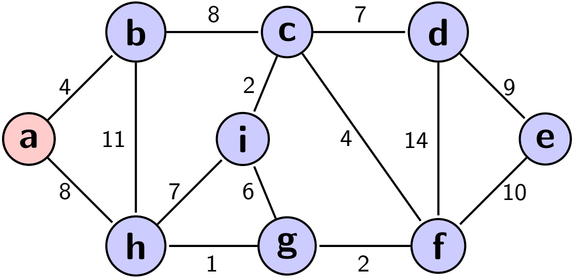

Figure: Ejecución del algoritmo Dijkstra en un grafo ponderado

Ejemplo

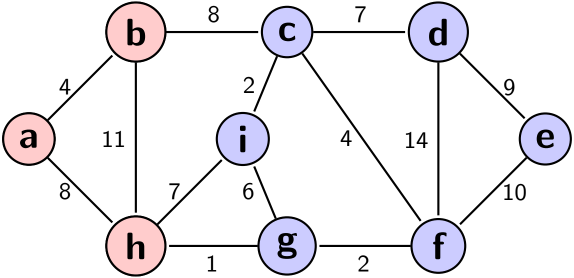

Figure: Ejecución del algoritmo Dijkstra en un grafo ponderado

Ejemplo

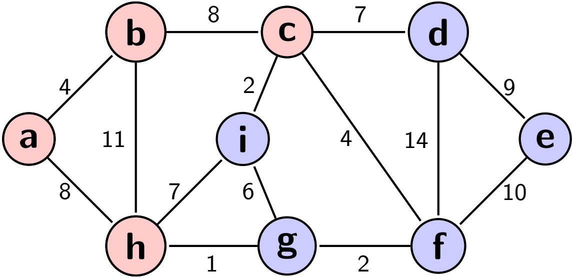

Figure: Ejecución del algoritmo Dijkstra en un grafo ponderado

Ejemplo

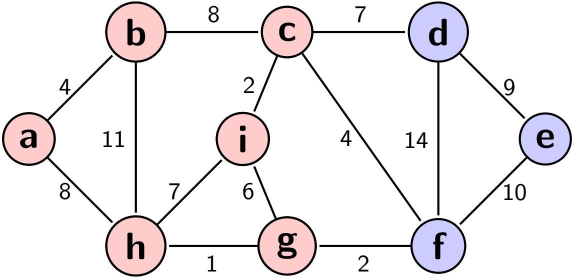

Figure: Ejecución del algoritmo Dijkstra en un grafo ponderado

Ejemplo

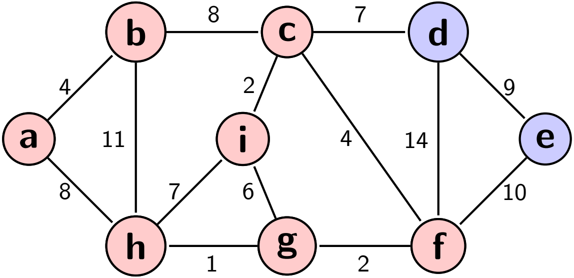

Figure: Ejecución del algoritmo Dijkstra en un grafo ponderado

Ejemplo

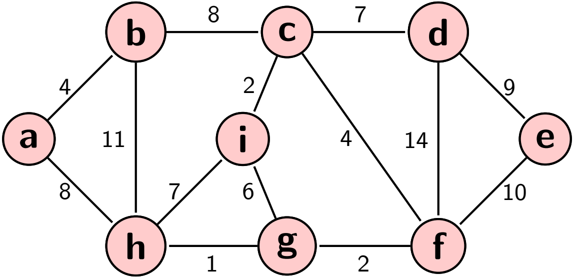

Figure: Ejecución del algoritmo Dijkstra en un grafo ponderado

Algoritmo Bellman Ford

Require: $G = (V, E, w), u \in V, w : E \mapsto \mathbb{R}^+$
  $D(u, u) \leftarrow 0$
  $D(u, v) \leftarrow \infty$ para todos los vertices $v! = u$
  for $k = 1$ **to** $|V| - 1$ **do**
    for all $(k, v) \in E$ **do**
      if $D(u, v) > D(u, k) + w_{k,v}$ **then**
        $D(u, v) = D(u, k) + w_{k,v}$
      end if
    end for
  end for
  return D

Ejemplo

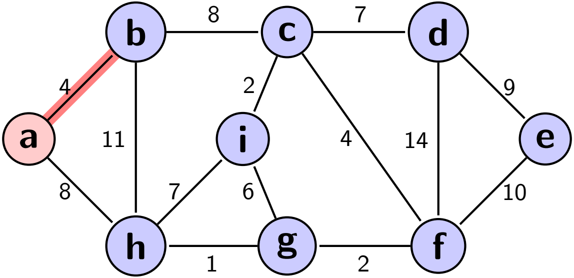

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

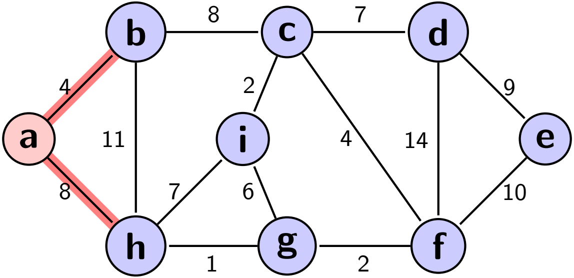

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

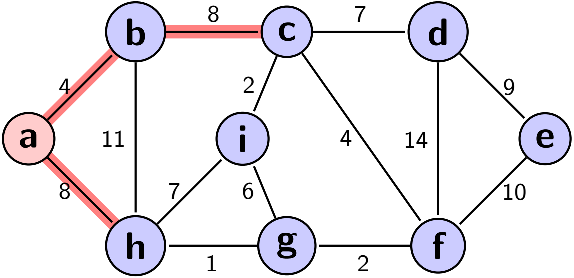

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

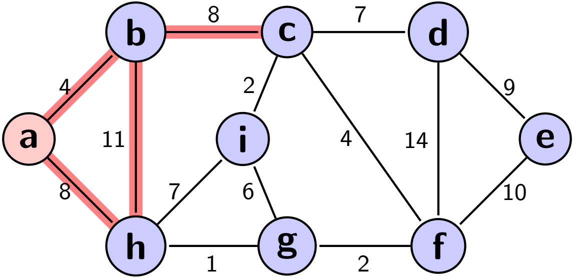

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

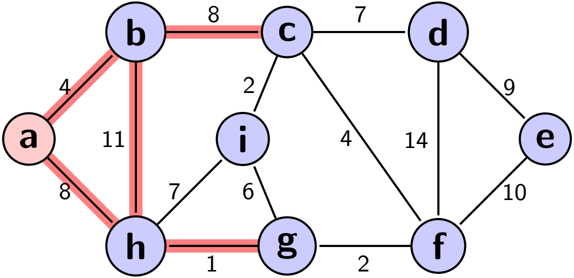

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

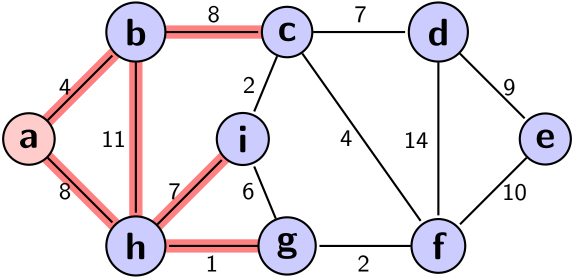

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

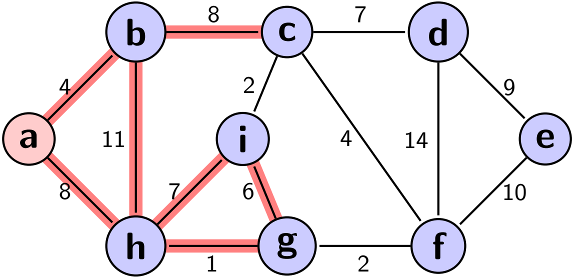

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

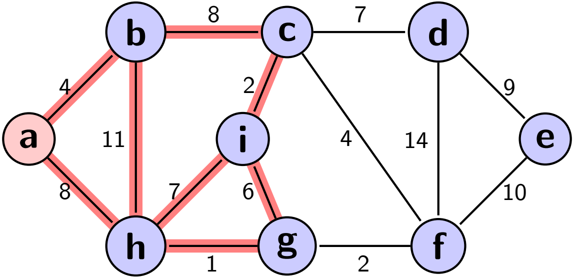

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

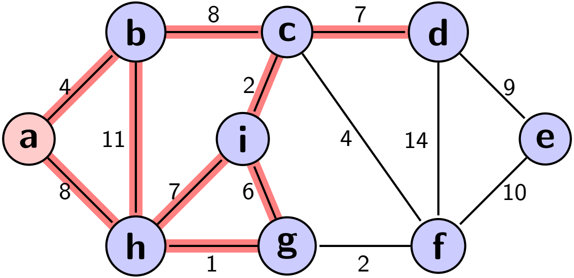

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

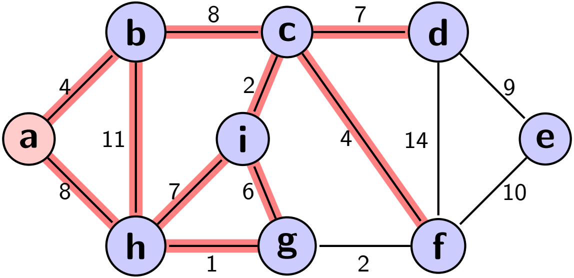

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

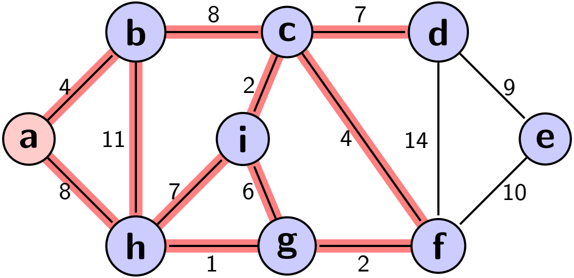

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

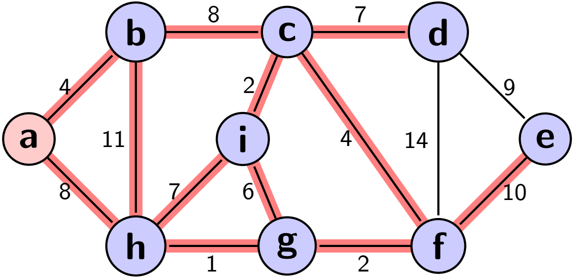

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Ejemplo

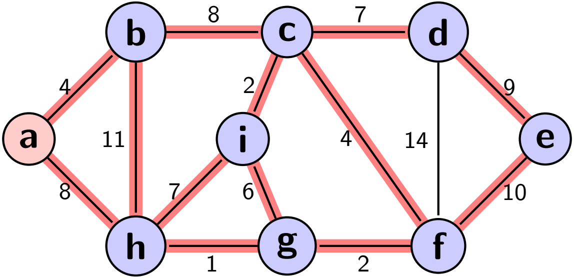

Figure: Ejecución del algoritmo Belmman Ford en un grafo ponderado

Comparación

- Ambos algoritmos encuentran la solución para el problema de caminos mas cortos desde un vertice hacia todos los demás vertices del grafo.
- El algoritmo Bellman-Ford permite trabajar con grafos con pesos negativos.
- La complejidad del algoritmo Dijkstra es $\mathcal{O}(V \log V)$
- La complejidad del algoritmo Bellman-Ford es $\mathcal{O}(VE)$