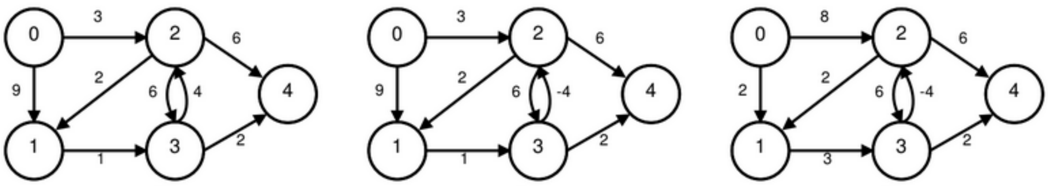
Jhonatan Rivera Saumeth

Taller Algoritmos
Caminos más cortos en grafos desde una sola fuente

Universidad Nacional de Colombia, Sede Bogotá

Considere los siguientes grafos:



(a) (b) (c)

Figura 1

1. Para los 3 grafos, calcule (por inspección) la longitud los caminos más cortos desde el nodo 0 hacia el resto de nodos.

2. Para el grafo b, ¿puede encontrar un camino desde 0 a 1 de costo 0? Explique.

3. El siguiente código (Cormen et al, 2009) implementa el algoritmo de Bellman-Ford para encontrar caminos más cortos desde una sola fuente:

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

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

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| - 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```

Aplique el algoritmo a cada uno de los grafos a, b y c. En cada caso muestre el estado de los grafos (valores de $v.d$ para cada nodo, puede mostrarlos dentro de cada nodo omitiendo la etiqueta del nodo) después de cada iteración del `for` en la línea 2.

4. ¿Qué pasó con el grafo b? Explique.

5. En cada caso, ¿cuántas llamadas se hicieron a la función `Relax`? ¿puede hacerse de manera más eficiente?

6. Para los grafos a y c, muestre una secuencia de llamadas a `Relax` que le permita calcular los caminos más cortos de una manera más eficiente.

1) Para el grafo a)

Node/S.Path	1	2	3	4
0	5	3	6	8

Para el grafo b)

Node/S.Path	1	2	3	4
0	-INF	-INF	-INF	-INF

Recorriendo el ciclo 1-3-2-1 podemos restar cuanto deseemos y acceder a todos los nodos de ahí que el costo es tan bajo como deseemos.

Para el grafo c)

Node/S.Path	1	2	3	4
0	2	1	5	7

2) En el grafo b se puede encontrar un camino a cualquier nodo excepto al nodo 0 con costo tan bajo como se desee la explicación esta en el literal b del primer punto, particularmente se puede encontrar un camino de costo 0 al nodo 1.

3) Aplicamos el algoritmo de Bellman-Ford

Para el grafo (a), x-y quiere decir pasando por x a una distancia de y se llega al nodo

Nodo\iteración	Iteración 1	Iteración 2	Iteracion 3	Iteración 4
0	0-0	0-0	0-0	0-0
1	0-9	2-5	2-5	2-5
2	0-3	0-3	0-3	0-3
3	INF	2-9	1-6	1-6
4	INF	2-9	2-9	3-8

Para el grafo (b), x-y quiere decir pasando por x a una distancia de y se llega al nodo

Nodo\iteración	Iteración 1	Iteración 2	Iteracion 3	Iteración 4
0	0-0	0-0	0-0	-
1	0-9	2-5	2-5	-
2	0-3	0-3	0-3	-
3	INF	2-9	1-6	-
4	INF	2-9	2-9	-

Para el grafo (c), x-y quiere decir pasando por x a una distancia de y se llega al nodo

Nodo\iteración	Iteración 1	Iteración 2	Iteracion 3
0	0-0	0-0	0-0
1	0-2	0-2	0-2
2	0-8	0-8	3-1
3	INF	1-5	1-5
4	INF	2-14	3-7

4) En el grafo b el algoritmo retornó falso al encontrar que existia un ciclo cuyo peso era negativo.
5) En cada caso la función Relax se llamó el número de vértices -1 * número de aristas ((|V|-1)*|E|).
   Si, se puede reducir el numero de llamadas a la función Relax
6) Motivo: Si un vértice v tiene una distancia que no ha cambiado desde la última vez que los edges fuera de v fueron relajados, entonces no hay necesidad de relajar los edges fuera de v una segunda vez.

   El pseudo-código del algoritmo de Bellman Ford Mejorado :

```
bool BellmanFord_Optimizado(Grafo G, nodo_origen s)
  // inicializamos el grafo. Ponemos distancias a INFINITO menos el nodo origen que
  // tiene distancia 0. Para ello lo hacemos recorriéndonos todos los vértices del
grafo
  for v ∈ V[G] do
    distancia[v]=INFINITO
    padre[v]=NULL
  distancia[s]=0
  encolar(s, Q)
  en_cola[s]=TRUE
  while Q!=0 then
    u = extraer(Q)
    en_cola[u]=FALSE
    // relajamos las aristas
    for v ∈ ady[u] do
      if distancia[v]>distancia[u] + peso(u, v) then
        distancia[v] = distancia[u] + peso (u, v)
        padre[v] = u
        if en_cola[v]==FALSE then
          encolar(v, Q)
          en_cola[v]=TRUE
```

   Tomado de Wikipedia: https://es.wikipedia.org/wiki/Algoritmo_de_Bellman-Ford