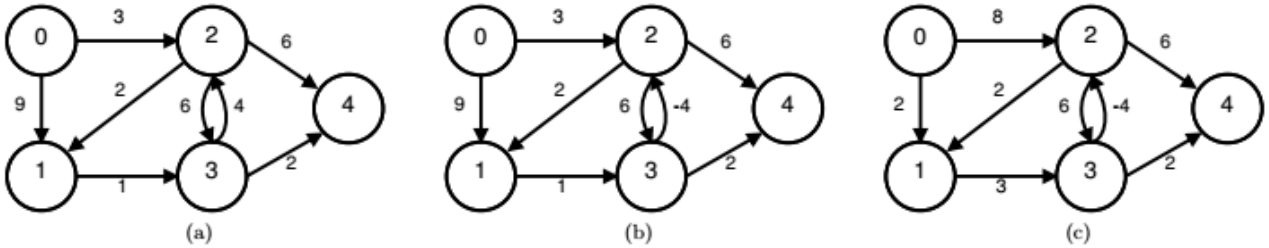
ALGORITMOS, TALLER 3

Juan Sebastián Narváez Beltrán
Cod. 2879781

Considere los siguientes grafos:



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

Grafo (a)

	Nodo 1	Nodo 2	Nodo 3	Nodo 4
Distancia desde el Nodo 0	5	3	6	8

Grafo (b)

	Nodo 1	Nodo 2	Nodo 3	Nodo 4
Distancia desde el Nodo 0	-inf	-inf	-inf	-inf

Grafo (c)

	Nodo 1	Nodo 2	Nodo 3	Nodo 4
Distancia desde el Nodo 0	2	1	5	7

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

   Si, existe un camino con peso -4, al pasar de los nodos 2 al 1 tiene un costo de 2, del nodo 1 al nodo 3 tiene un costo de 1 lo que suma un costo del recorrido de 4, al pasar del nodo 3 al nodo 2 el costo del recorrido llega a 0. Así el recorrido completo seria.

Nodo	0	2	1	3	2	1	3	2	1	3	2	1	3	2	1	3	2	1
Costo total	0	3	5	6	2	4	5	1	3	4	0	2	3	-1	1	2	-2	0

3. El siguiente código (Cormen et al, 2009) implementa el algoritmo 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.

Grafo (a)

Iteración/ Nodo	0	1	2	3	4
0	0	∞	∞	∞	∞
1	0	5	3	6	8
2	0	5	3	6	8
3	0	5	3	6	8
4	0	5	3	6	8

Grafo (b)

Iteración/ Nodo	0	1	2	3	4
0	0	∞	∞	∞	∞
1	0	5	2	6	8
2	0	4	1	5	7
3	0	3	0	4	6
4	0	2	1	3	5

Grafo(c)

Iteración/ Nodo	0	1	2	3	4
0	0	∞	∞	∞	∞
1	0	2	1	5	7
2	0	2	1	5	7
3	0	2	1	5	7
4	0	2	1	5	7

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

   El grafo b tiene un camino con costo -4, y afecta en encontrar el camino mas corto retornando como camino mas corto un peso negativo por lo tanto la función Bellman-Ford no puede calcular correctamente los caminos mas cortos, y retorna como en la ultima iteración en el nodo 2 un costo de 1 cuando en realidad es un costo de -1.

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

   Para cada iteración en el for en la linea 2 se analizan todas las aristas, 8 en total, por lo tanto se llama 8 veces la función relax por cada iteración, son 4 iteraciones entonces en total se llama 32 veces la función relax. Una manera de hacerlo más eficiente el algoritmo seria

con programación dinámica, almacenando en un arreglo los nodos que ya no reciben cambios por parte de la función relax y dejándolos por fuera de la siguiente iteración.

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

```
Relax(u,v,w)
  if (v.d > u.d + w(u,v)) AND (array[v]!="Distancia min")
    v.d = u.d + w(u,v)
    v.pi = u
  else
    array[v]= "Distancia min"
```

Grafo (a)

Iteración/ Nodo	0	1	2	3	4
0	0	∞	∞	∞	∞
1	0	5	3	6	8
2	0	5	3	6	8

Array Programación dinamica Grafo (a) en la función Relax

Iteración/ Nodo	0	1	2	3	4
0	Distancia min				
1	Distancia min		Distancia Min	Distancia Min	
2	Distancia min	Distancia Min	Distancia Min	Distancia Min	Distancia Min

Grafo (c)

Iteración/ Nodo	0	1	2	3	4
0	0	∞	∞	∞	∞
1	0	2	1	5	7
2	0	2	1	5	7

Array Programación dinámica Grafo (c) en la función relax

Iteración/ Nodo	0	1	2	3	4
0	Distancia min				
1	Distancia min	Distancia min		Distancia min	Distancia min
2	Distancia min	Distancia min	Distancia min	Distancia min	Distancia min