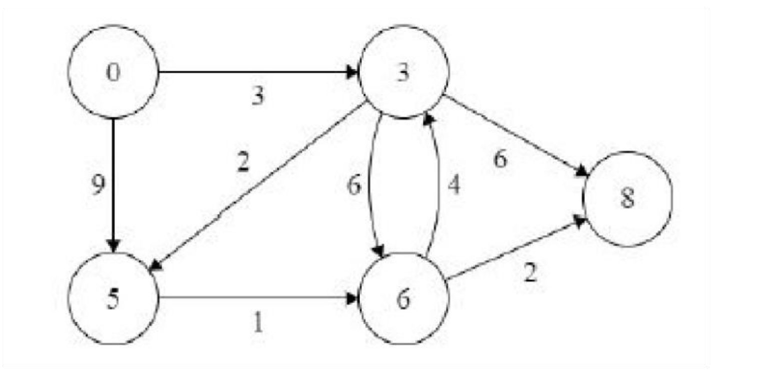
Taller 3: Caminos más cortos en grafos desde una sola fuente

Entrega: David Jose Leon Aroca

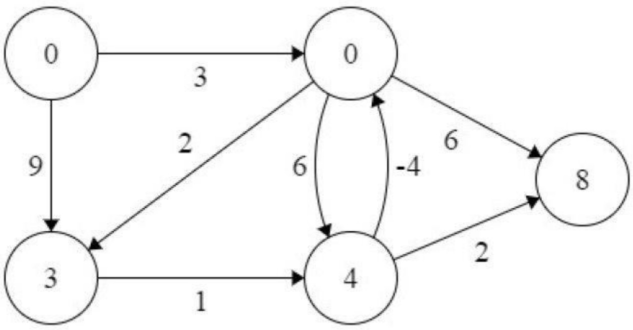
1.

Grafo a

	0	1	2	3	4
0	na	5	3	6	8
1	na	0	5	1	3
2	na	2	0	3	5
3	na	6	4	0	2
4	na	na	na	na	na



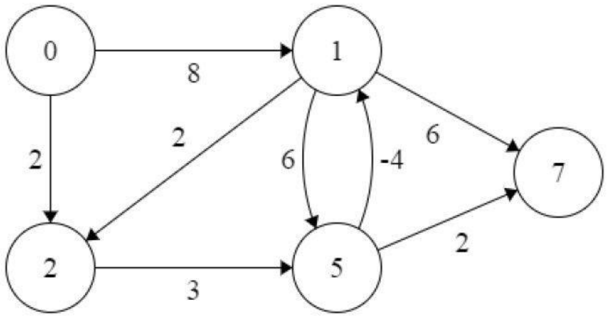
Grafo b



	0	1	2	3	4
0	na	5	3	6	8
1	na	-1	-3	1	3
2	na	2	-1	3	5
3	na	-3	-4	-1	1
4	na	na	na	na	na

No es posible encontrar un camino dado que las aristas de peso negativo generan bucles.

Grafo c



	0	1	2	3	4
0	na	2	1	5	7
1	na	1	-1	3	5
2	na	2	1	5	6
3	na	-2	-4	1	2
4	na	na	na	na	na

2.

Si, se crea un ciclo negativo infinito que constantemente reduce distancias entre los nodos, el ciclo está conformado de la siguiente manera: 0 > 2 > 1 > 3 > 2 > 1 > 3 > 2 > 1 > 3 > 2 > 1 > 3 > 2 > 1 > 3 > 2 > 1, en esta iteración (6) la distancia entre el nodo 0 y el nodo 1 es igual a 0.

3.

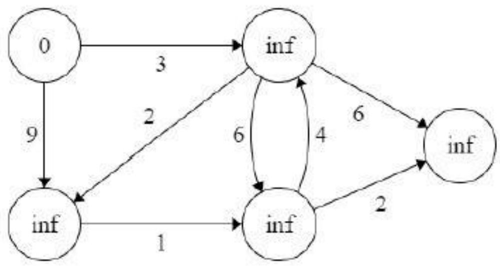
```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

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

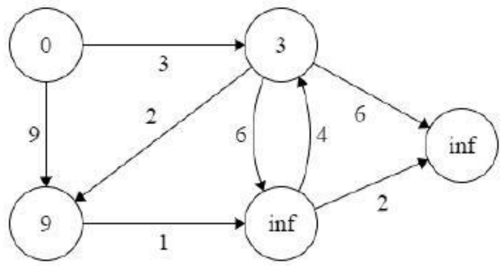
```
BELLMAN-FORD(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
for i = 1 to |G.V| - 1
    for each edge (u, v) ∈ G.E
        RELAX(u, v, w)
for each edge (u, v) ∈ G.E
    if v.d > u.d + w(u, v)
        return FALSE
return TRUE
```

Grafo a:
Orden: (1,3), (3,4), (3,2), (2,3), (2,4), (2,1), (0,1), (0,2)

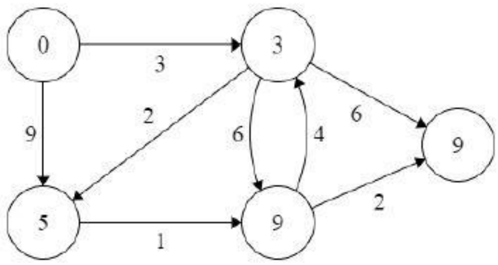
T = 0



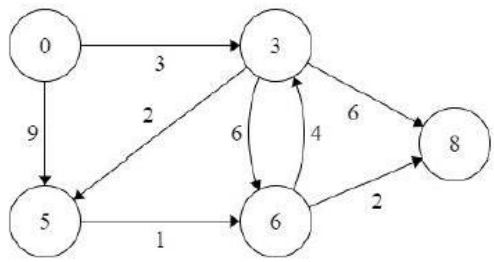
T = 1



T = 2



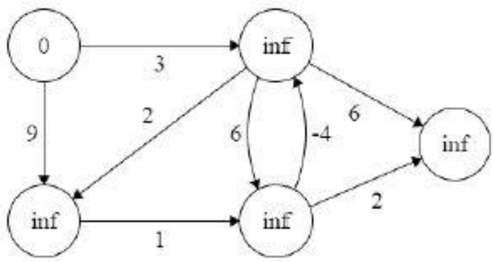
T = 3



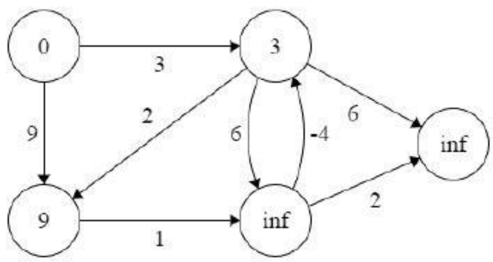
En la siguiente iteración no hay cambios.

Grafo b:
Orden: (1,3), (3,4), (3,2), (2,3), (2,4), (2,1), (0,1), (0,2)

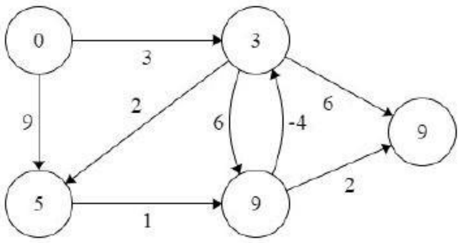
T = 0



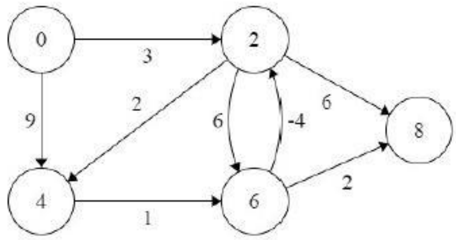
T = 1



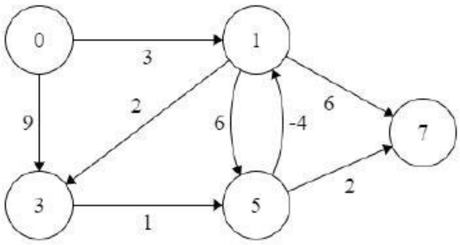
T = 2



T = 3

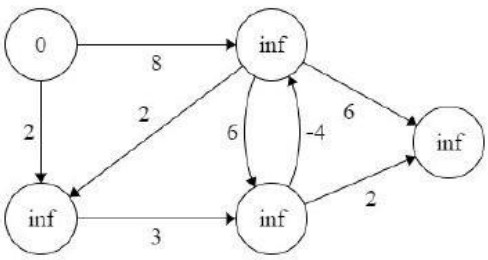


T = 4

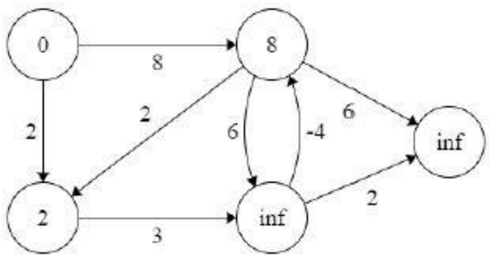


Grafo c:
Orden: (1,3), (3,4), (3,2), (2,3), (2,4), (2,1), (0,1), (0,2)

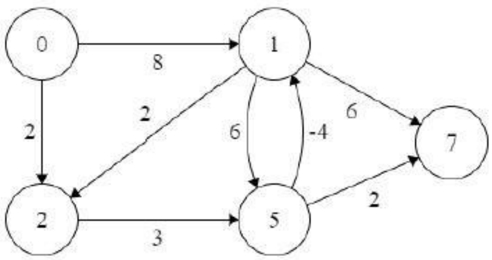
T = 0



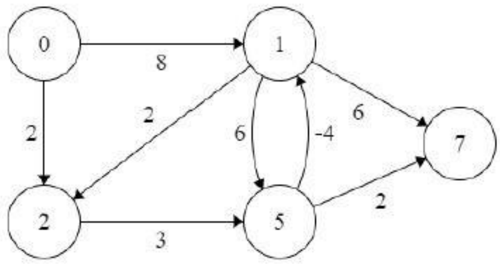
T = 1



T = 2



T = 3



Desde la 3 iteración se puede observar que el grafo no es modificado.

4.

El algoritmo de Bellman-Ford retorna un False al finalizar, esto se da por el ciclo negativo que se expuso en el segundo punto, durante la fase Relax se observa que es posible encontrar caminos más cortos de los que se obtienen al finalizar, pero como el algoritmo realiza la fase de relajación una vez por nodo, no se puede continuar con el ciclo a pesar de que es posible reducir las distancias infinitamente.

5.

En total se entran 32 veces a la función Relax dado que para cada caso se hacen 8 llamadas y en el for se realizan 4 pasos, esto es igual al número de nodos -1 * número de vértices, es posible optimizar el algoritmo si se verifica la cantidad de cambios que se hicieron al terminar cada paso del for exterior, si la cantidad es 0 se puede realizar un break y terminar el ciclo.

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.

Con el siguiente código:

```
Initialize-Single-Source( G, s ):
        For each vertex v in G.v:
        v.d =
        v.
        s.d = 0

Relax( u, v, w ):
        If v.d > u.d + w( u, v )
        v.d = u.d + w( u, v )
        v. = u
        change = True

Bellman-Ford( G, w, s ):
        Initialize-Single-Source( G, s )
        i = 1
        change = True

        while change and i < |G.v| - 1
        change = False
        for each edge ( u, v )  G.E
        Relax( u, v, w )

        For each edge ( u, v )  G.E
        If v.d > u.d + w( u, v )
        return False        return
True
```

Es posible reducir las llamadas a la función Relax, con ayuda de una bandera que verifica los cambios que se realizan en las distancias, si no se hace ningún cambio se determina que no se debe seguir revisando y se sale del ciclo asumiendo que change es global.

a)

Iteración/Nodo	0	1	2	3	4
0	0	∞	∞	∞	∞
1	0	5	3	9	9
2	0	5	3	6	8
3	0	5	3	6	8

En el grafo a se omite la última iteración y por lo tanto se evitan 8 llamadas a la función Relax

c)

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

En el grafo c son innecesarias las 2 últimas iteraciones, por lo tanto se omite la función Relax 16 veces.