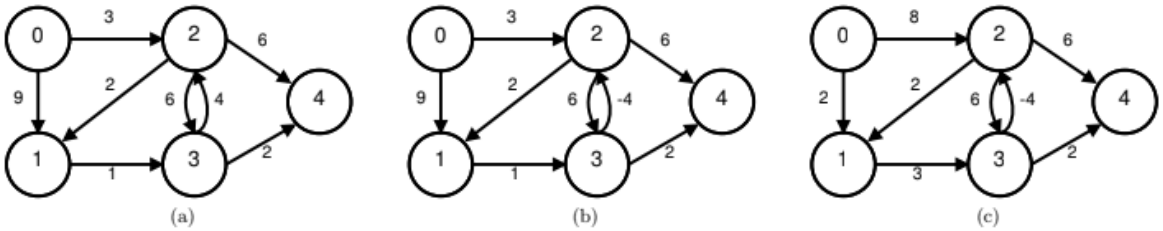
UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERIA

ALGORITMOS – Taller 3

EDWIN RICARDO MAHECHA PARRA – 1013679888

Con los siguientes grafos:



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

Nodo	1	2	3	4
Distancia	5	3	6	8

   Grafo B:

Nodo	1	2	3	4
Distancia	5	3	6	8

   Grafo C:

Nodo	1	2	3	4
Distancia	2	8	5	7

2. Para le grafo b, ¿puede encontrar un camino desde 0 a 1 de costo 0? Explique.
   Solución: Es posible siguiendo la secuencia de nodos 0,2,1,3 y se repite la parte de 2,1,3. Esto logra que por cada repetición se disminuya en 1 el peso total para el camino de 0 a 1. En otras palabras, es un ciclo negativo.
3. Aplicar Bellman-Ford para los grafos a,b,c y mostrar sus estados.
   Solución: Primero realizamos la implementación en Python.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  4 12:18:30 2018

@author: Mahecha
"""

class edge: #edge creation between nodes a,b with weight w
    def __init__(self, a,b,w):
        self.a = a
        self.b = b
        self.w = w
```

```
    def getVals(self):
        return self.a,self.b,self.w

def bellman(n_vertex,edges,s):#s is the source
    d = [10000000]*n_vertex
    parent = [-1]*n_vertex

    d[s] = 0 #source distance

    for i in range(n_vertex-1):
        for e in edges:
            a,b,w = e.getVals()
            if(d[a] + w < d[b]):
                d[b] = d[a]+w
                parent[b] = a
        print(d)

    for e in edges:
        a,b,w = e.getVals()
        if(d[a]+w<d[b]):
            print("[Graph with negative cycle, dist > ",d,parent,"]")

    print ("[",d,parent,"]")


graph = [edge(0,2,3),
          edge(0,1,9),
          edge(1,3,1),
          edge(2,1,2),
          edge(2,3,6),
          edge(2,4,6),
          edge(3,2,4),
          edge(3,4,2)]

print("Graph A")
bellman(5,graph,0)

graph = [edge(0,2,3),
          edge(0,1,9),
          edge(1,3,1),
          edge(2,1,2),
          edge(2,3,6),
          edge(2,4,6),
          edge(3,2,-4),
          edge(3,4,2)]

print("Graph B")
bellman(5,graph,0)

graph = [edge(0,2,8),
          edge(0,1,2),
          edge(1,3,3),
          edge(2,1,2),
          edge(2,3,6),
          edge(2,4,6),
          edge(3,2,-4),
```

```
            edge(3,4,2)]

print("Graph C")
bellman(5,graph,0)
```

La salida del programa es:

```
In [3]: runfile('C:/Users/Mahecha/Desktop/final algo/Talleres/Taller 3/bellı
wdir='C:/Users/Mahecha/Desktop/final algo/Talleres/Taller 3')
Graph A
[0, 5, 3, 9, 9]
[0, 5, 3, 6, 8]
[0, 5, 3, 6, 8]
[0, 5, 3, 6, 8]
[ [0, 5, 3, 6, 8] [-1, 2, 0, 1, 3] ]
Graph B
[0, 5, 3, 9, 9]
[0, 5, 2, 6, 8]
[0, 4, 2, 6, 8]
[0, 4, 1, 5, 7]
[Graph with negative cycle, dist >  [0, 4, 1, 5, 7] [-1, 2, 3, 1, 3] ]
[ [0, 4, 1, 5, 7] [-1, 2, 3, 1, 3] ]
Graph C
[0, 2, 1, 5, 7]
[0, 2, 1, 5, 7]
[0, 2, 1, 5, 7]
[0, 2, 1, 5, 7]
[ [0, 2, 1, 5, 7] [-1, 0, 3, 1, 3] ]
```

4. ¿Qué pasó con el grafo b?. Explique.
   Solución: El grafo b posee un ciclo negativo detectado por el algoritmo bellman-ford.
5. En cada caso, ¿Cuántas llamadas se realizaron a la función Relax? ¿Puede hacerse más eficiente?
   Solución: En cada uno de los grafos se llama la función relax (n_vertex-1)*#edges, es decir 32 llamados en total. El algoritmo como tal no se puede mejorar, pero se puede usar en conjunto con la idea de queue en Dijkstra manteniendo en esta los candidatos a ser relajados.
6. Para los grafos a y c, muestre una secuencia de llamadas a Relax que le permita calcular los caminos más cortos de manera más eficiente.
   Solución: Haciendo uso de la idea presentada en el punto anterior de Djkstra.
   Para el grafo a:
   - 0,2
   - 2,1
   - 1,3
   - 3,4
   - Después de esto ya se posee la respuesta óptima y no es necesario llamar más al relax.

   Para el grafo c:
   - 0,2

- 0,1
- 1,3
- 3,4
- Después de esto ya se posee la respuesta optima y no es necesario llamar más al relax.