ECE374 Fall2020

Lab11: Bellman Ford Algorithm

Name: Zhang Yichi 3180111309

Dec. 23th 2020

1 Introduction

In this lab, we will implement a program to find all shortest-path lengths from a source $s \in V$ to all $v \in V$ or determine that a negative-weight cycle exists with Bellman Ford algorithm.

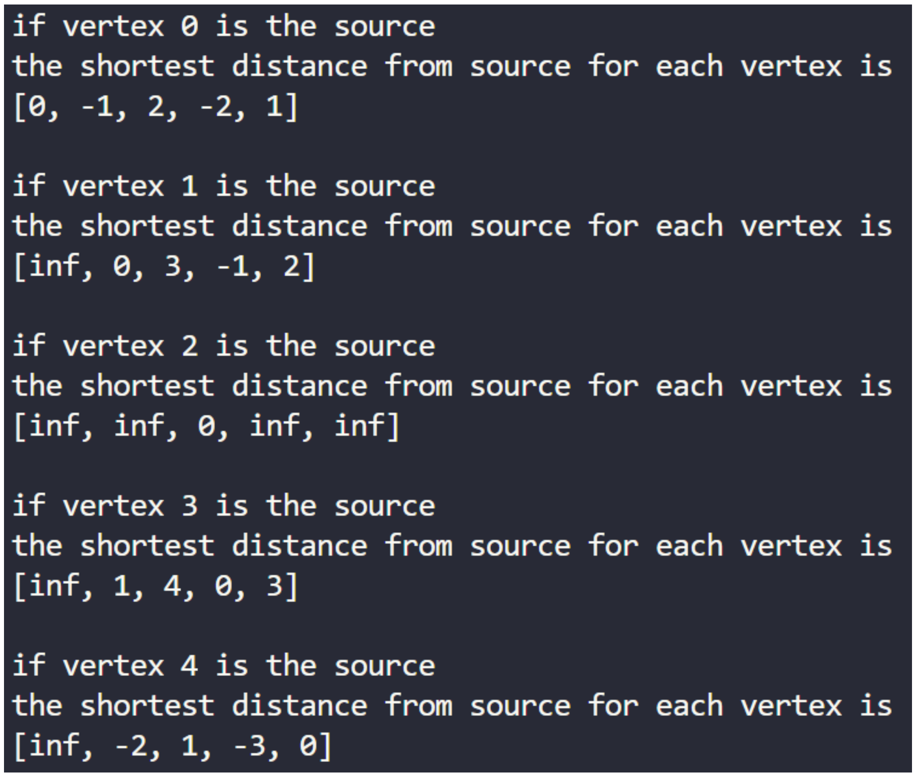
2 Python Code for the Bellman Ford Algorithm

```python
class Edge:
    def __init__(self,u,v,w):
        self.u = u
        self.v = v
        self.w = w

def BellmanFord(E,V,source):
    d = [float('inf')]*len(V)
    d[source] = 0
    for _ in range(len(V)-1):
        for e in E:
            if d[e.v] > d[e.u] + e.w:
                d[e.v] = d[e.u] + e.w
    for e in E:
        if d[e.v] > d[e.u] + e.w:
            print("negative weight cycle detected")
            return -1
    return d

def ModifiedBellmanFord(E,V,source):
    d = [float('inf')]*len(V)
    d[source] = 0
    for _ in range(len(V)-1):
        for e in E:
            if d[e.v] > d[e.u] + e.w:
                d[e.v] = d[e.u] + e.w
    for _ in range(len(V)-1):
        for e in E:
            if d[e.v] > d[e.u] + e.w:
                d[e.v] = -float('inf')
    return d
```

With the first for loop, we update all the distances. With the second for loop, we update them another time, if there is something to update, it means that negative weight cycles are detected.

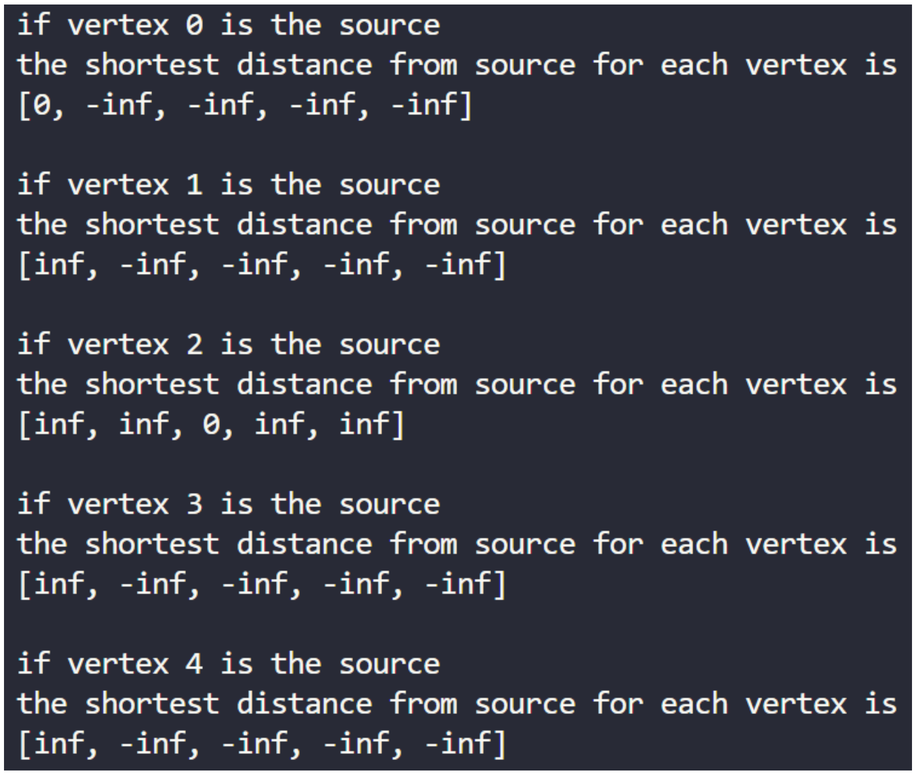
3 Test Example for the LCS Problem

```
graph = [(0,1,-1),(0,2,4),(1,2,3),(1,3,2),(1,4,2),(3,2,5),(3,1,1),(4,3,-3)]
E = []
V = []
for i in graph:
    E.append(Edge(i[0],i[1],i[2]))
V = [i for i in range(5)]
for i in V:
    print(f"if vertex {i} is the source")
    d = BellmanFord(E,V,i)
    if d != -1:
        print(f"the shortest distance from source for each vertex is")
        print(d)
    print()

print()
print("for another graph with negative weight cycle")
graph = [(0,1,-1),(0,2,4),(1,2,3),(1,3,2),(1,4,2),(3,2,5),(3,1,-1),(4,3,-3)]
E = []
V = []
for i in graph:
    E.append(Edge(i[0],i[1],i[2]))
V = [i for i in range(5)]
for i in V:
    print(f"if vertex {i} is the source")
    d = ModifiedBellmanFord(E,V,i)
    if d != -1:
        print(f"the shortest distance from source for each vertex is")
        print(d)
    print()
```



```
if vertex 0 is the source
the shortest distance from source for each vertex is
[0, -1, 2, -2, 1]

if vertex 1 is the source
the shortest distance from source for each vertex is
[inf, 0, 3, -1, 2]

if vertex 2 is the source
the shortest distance from source for each vertex is
[inf, inf, 0, inf, inf]

if vertex 3 is the source
the shortest distance from source for each vertex is
[inf, 1, 4, 0, 3]

if vertex 4 is the source
the shortest distance from source for each vertex is
[inf, -2, 1, -3, 0]
```

Figure 1: Test results for graph without negative weight cycle with Bellman Ford algorithm.



```
if vertex 0 is the source
the shortest distance from source for each vertex is
[0, -inf, -inf, -inf, -inf]

if vertex 1 is the source
the shortest distance from source for each vertex is
[inf, -inf, -inf, -inf, -inf]

if vertex 2 is the source
the shortest distance from source for each vertex is
[inf, inf, 0, inf, inf]

if vertex 3 is the source
the shortest distance from source for each vertex is
[inf, -inf, -inf, -inf, -inf]

if vertex 4 is the source
the shortest distance from source for each vertex is
[inf, -inf, -inf, -inf, -inf]
```

Figure 2: Test results for graph with negative weight cycle with modified Bellman Ford algorithm.

4 Time Complexity

```
    for _ in range(len(V)-1):
        for e in E:
            if d[e.v] > d[e.u] + e.w:
                d[e.v] = d[e.u] + e.w
```

The outer loop is O(V). The inner loop is O(E). Therefore, the total time complexity for the Bellman Ford algorithm and the modified Bellman Ford algorithm are both O(VE).