CENTRAL UNIVERSITY

FAITH • INTEGRITY • EXCELLENCE

School Of Engineering & Technology
Department of Computer Science & Information technology

Name: Nsiah Joshua

Index number: CSC/21/01/1022

Course: COMP 416 - NETWORK PROGRAMMING & PROTOCOL

Assignment 1

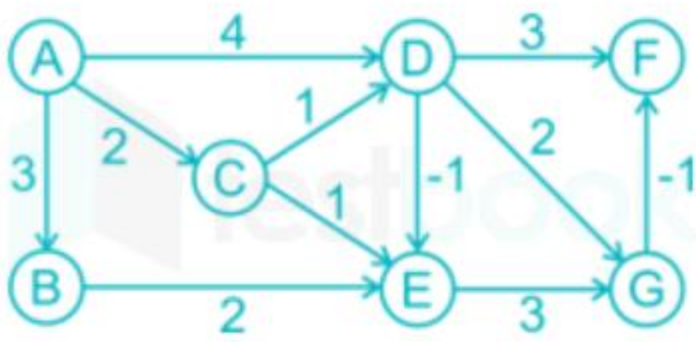
Date: 10th April, 2025

COMP 416 - NETWORK PROGRAMMING & PROTOCOL

Assignment 1

Due: 1 hour

1. The following weighted graph represents the Bellman Ford algorithm, which is implemented with source A.
   a. Find the shortest distance from the source node A to the destination node F by showing the route.
   b. Find the set of shortest paths from all nodes to the destination node.



Ans:

a.

- `dist(F) = 4`
- Path:
  `A → C → D → E → G → F`
  (0 → 2 → 3 → 2 → 5 → 4)

b.

Node	Distance from A	Path
A	0	A
B	3	A → B
C	2	A → C
D	3	A → C → D
E	2	A → C → E or A→C→D→E
G	5	A → C → D → E → G
F	4	A → C → D → E → G → F

2. Suppose you are given a directed graph that has negative values on some of the edges. The Bellman Ford algorithm will give the correct solution for shortest paths from some starting vertex s, *if there are no negative cycles*. But suppose you don't know if the given graph has negative cycles. One way to check for negative cycles is to run Bellman Ford for *n* iterations, instead of *n-1*. Explain.

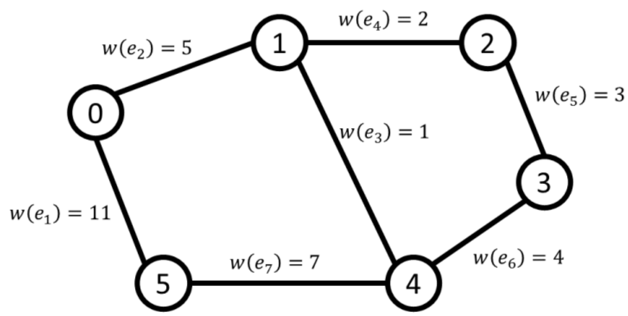
Ans:

Bellman-Ford is used to find the shortest paths from a single source vertex s to all other vertices in a weighted graph. If the graph has n vertices, then:

Any shortest path can contain at most (n - 1) edges.

So, Bellman-Ford relaxes all edges n - 1 times to ensure all shortest paths are calculated.

3. Suppose we treat the graph below as a digraph with a directed edge in both directions (so, for example, there is a directed edge of weight 7 from 4 → 5 and from 5 → 4. Suppose we run Bellman Ford from s = 0 by repeatedly relaxing the edges (e1, e2, e3, e4, e5, e6, e7) (by relaxing e7 we mean relaxing both 4 → 5 and 5 → 4 in some order). How many times will distTo[5] get updated?



Ans:

You have 6 vertices: 0 through 5.
We treat the graph as a directed graph with two directions for each edge.

Edge list with weights (as directed edges in both directions):
$0 \rightarrow 5$ and $5 \rightarrow 0$ with weight = 11

$0 \rightarrow 1$ and $1 \rightarrow 0$ with weight = 5

$1 \rightarrow 4$ and $4 \rightarrow 1$ with weight = 1

$1 \rightarrow 2$ and $2 \rightarrow 1$ with weight = 2

$2 \rightarrow 3$ and $3 \rightarrow 2$ with weight = 3

$3 \rightarrow 4$ and $4 \rightarrow 3$ with weight = 4

$4 \rightarrow 5$ and $5 \rightarrow 4$ with weight = 7

Iteration by Iteration: Track Updates to distTo[5]
Iteration 1
e1: 0 → 5 → distTo[5] = min(∞, 0+11) = 11

e2: 0 → 1 → distTo[1] = 5

e3: 1 → 4 → distTo[4] = 5 + 1 = 6

e7: 4 → 5 → distTo[5] = min(11, 6+7) = 13

But e7: 5 → 4 → distTo[4] = min(6, 11+7) = 6

Iteration 2
e2: 0 → 1 (no change)

e3: 1 → 4 (no change)

e7: 4 → 5 → distTo[5] = min(11, 6+7 = 13)

No other path gives shorter distance to 5