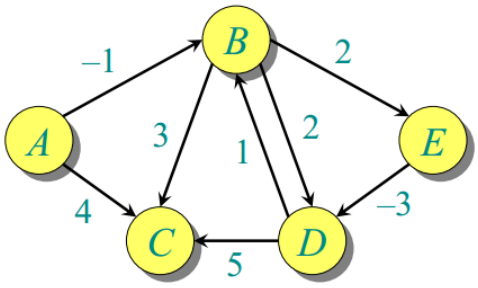
190031187 **NERELLA VENKATA RADHAKRISHNA**

Tutorial-4

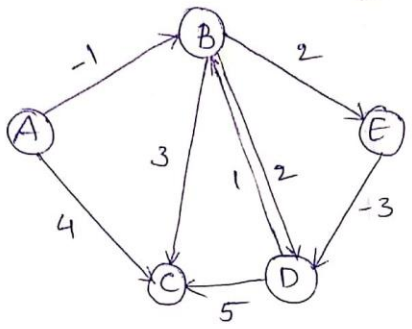
Question: Single-Source Shortest Paths – Bellman Ford Algorithm

Given a source vertex from set of vertex V in a weighted graph when its edge weights w(u,v) can be negative, find the shortest-path weights d(s,v) from given s from all vertices v present in the graph. If the graph contains negative-weight cycle, report it. Consider the below graph.



190031187 MP-2 Tutorial-4
Radhakrishna

1.



No. of vertices = 5

let source vertex = A

Iteration to be done (atmost) = vertices - 1
= 4

Initially

A	B	C	D	E
0	∞	∞	∞	∞

Edges - (A,B) (A,C) (B,C) (B,D) (D,C) (D,B) (B,E) (E,D)

Note - Need not be in same order

Iteration - 1

After (A,B)

A	B	C	D	E
0	-1	∞	∞	∞

After (A,C)

A	B	C	D	E
0	-1	4	∞	∞

After (B,C)

A	B	C	D	E
0	-1	2	∞	∞

After (B,D)

A	B	C	D	E
0	-1	2	1	∞

After (D,C)

A	B	C	D	E
0	-1	2	1	∞

pass (D, B)

A	B	C	D	E
0	-1	2	1	∞

pass (B, E)

A	B	C	D	E
0	-1	2	1	1

pass (E, D)

A	B	C	D	E
0	-1	2	-2	1

Iteration - 2

pass (A, B)

A	B	C	D	E
0	-1	2	-2	1

pass (A, C)

A	B	C	D	E
0	-1	2	-2	1

pass (B, C)

A	B	C	D	E
0	-1	2	-2	1

pass (B, D)

A	B	C	D	E
0	-1	2	-2	1

pass (D, C)

A	B	C	D	E
0	-1	2	-2	1

pass (D, B)

A	B	C	D	E
0	-1	2	-2	1

pass (B, E)

A	B	C	D	E
0	-1	2	-2	1

pass (E, D)

A	B	C	D	E
0	-1	2	-2	1

we can observe that there is no change in this whole iteration. so we can stop here

Note:- While coding we cannot find out these things so we will do atmost of (Vertices - 1) iterations

So Final values of vertices are

A - 0

B - (-1)

C - 2

D - (-2)

E - 1

In [5]:

```
class BellManFordAlgorithm:
    def __init__(self,vertices):
        self.vertices=vertices
        self.graph=[]

    def AddEdge(self,u,v,w):
        self.graph.append([u,v,w])

    def printSol(self,sol):
        print("Vertex\t\tDistanceFromSource")
        for i in range(self.vertices):
            print(i,'\t','\t',sol[i])

    def Algorithm(self,src):

        dis=[float("Inf")]*self.vertices
        dis[src]=0

        for i in range(self.vertices-1):
            for u,v,w in self.graph:
                if dis[u]!=float("Inf") and dis[u]+w<dis[v]:
                    dis[v]=dis[u]+w

        for u,v,w in self.graph:
            if dis[u]!=float("Inf") and dis[u]+w<dis[v]:
                print("Graph contains negative weight cycle")
                return

        self.printSol(dis)
```

In [6]:

```
g = BellManFordAlgorithm(5)
g.AddEdge(0,1,-1)
g.AddEdge(0,2,4)
g.AddEdge(1,2,3)
g.AddEdge(1,3,2)
g.AddEdge(3,2,5)
g.AddEdge(3,1,1)
g.AddEdge(1,4,2)
g.AddEdge(4,3,-3)

g.Algorithm(0)
```

```
Vertex          DistanceFromSource
0                0
1                -1
2                2
3                -2
4                1
```