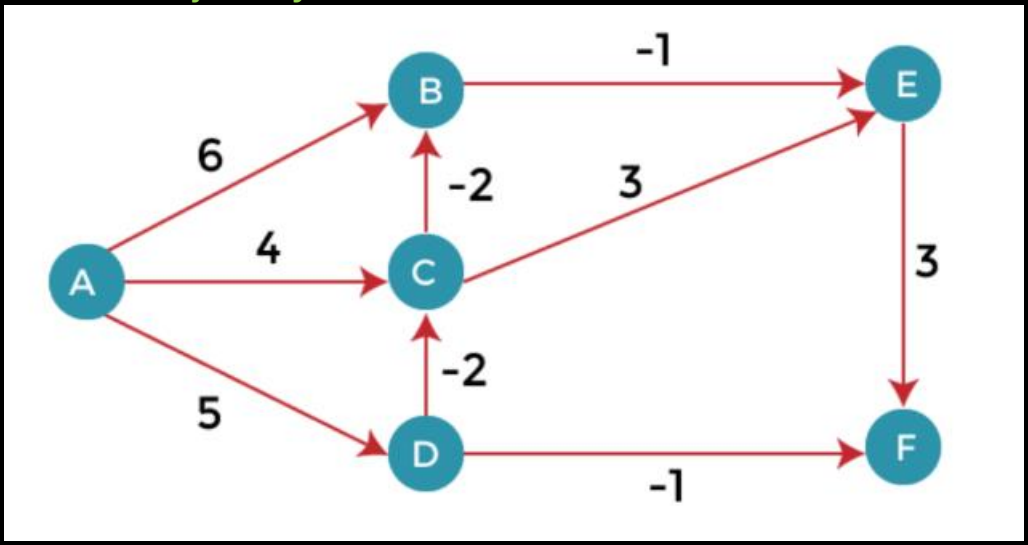
Name: Girish S Roll No.: AM.EN.U4AIE22044

-22AIE203-
DATA STRUCTURES AND ALGORITHMS -2

ASSIGNMENT 3 – Bellman-Ford'S ALGORITHM

1. **Perform Bellman-Ford's Algorithm on the given Graph using Adjacency matrix or Adjacency list**




```
dic = {
'A':{'B':2, 'C':2, 'D':1},
'B':{'D':2},
'C':{'E':1, 'D':3},
'D':{'E':2},
'E':{},
'S':{'A':1, 'B':5},
}

graph = Graph(dic)
```

```python
## [Graph]

class Graph:
    def __init__(self, adj_dic):
        self.adj_dic = adj_dic

    def child(self, parent):
        return list(self.adj_dic[parent])

    def vertices(self):
        return list(self.adj_dic)

    def Edges(self):
        edges = []
        for i in self.vertices():
            for j in self.child(i):
                edges.append((i, j))
        return edges

    def PathWeight(self, u, v):
        return self.adj_dic[u][v]
```

```python
## [ Bellman-Ford ]

def Bellman(Graph, source, _help_=False):
    dist={vertex: float('inf') for vertex in Graph.vertices()}
    dist[source] = 0

    iter_ = len(Graph.vertices())
    if _help_:
        print(0, dist)
    for i in range(iter_-1):
        dist_dup = dist.copy()
        for path in Graph.Edges():
            if dist[path[0]] + Graph.PathWeight(path[0], path[1]) < dist[path[1]]:
                dist[path[1]] = dist[path[0]] + Graph.PathWeight(path[0], path[1])
        if _help_:
            print(i+1, dist)
        if dist_dup == dist:
            print("Break: No Updation!")
            return dist
    return dist
Bellman(graph, "A", True)
```

```
>>>
 RESTART: C:\Users\giri0\OneDrive\Desktop\  \pdf\semester-3\DSA
0 {'A': 0, 'B': inf, 'C': inf, 'D': inf, 'E': inf, 'F': inf}
1 {'A': 0, 'B': 2, 'C': 3, 'D': 5, 'E': 5, 'F': 4}
2 {'A': 0, 'B': 1, 'C': 3, 'D': 5, 'E': 1, 'F': 4}
3 {'A': 0, 'B': 1, 'C': 3, 'D': 5, 'E': 0, 'F': 3}
4 {'A': 0, 'B': 1, 'C': 3, 'D': 5, 'E': 0, 'F': 3}
Break: No Updation!
>>>
```