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
```