In [24]:

```
class Vertex:
    def __init__(self,value):
        self.value=value
        self.dist=float("inf")
        self.pred=value
        self.visited=False
class PriorityQueue:
    def __init__(self):
        self.queue=[]
    def Enqueue(self,value):
        self.queue.append(value)
    def Dequeue(self):
        self.queue.remove(self.queue[0])
    def IsEmpty(self):
        if len(self.queue) == 0:
            return True
        else:
            return False
    def ExtractMin(self):
        lst=[]
        count=0
        while count != len(self.queue):
            lst.append(self.queue[count].dist)
            count+=1
        count=0
        for i in self.queue:
            if min(lst) == i.dist:
                self.queue.pop(count)
                return i
            count+=1
class DGraph:
    def __init__(self,vertex):
        self.vertex=vertex
        self.adj=[[0 for i in range(vertex)]for j in range(vertex)]
    def Addedge(self,src,dest,weight):
        if src==dest:
            print("source and destination are same")
        else:
            self.adj[src][dest]=weight
    def GetNeighbour(self,source):
        a=[]
        for i in range(self.vertex):
            if self.adj[source][i]>0:
                a.append(i)
        return a
    def Dijkstrashortestpath(self,source):
        cost=[]
        q=PriorityQueue()
        for i in range(self.vertex):
            cost.append(Vertex(i))
        for i in range(self.vertex):
            cost[i].dist=float("inf")
        cost[source].dist=0
        for i in range(self.vertex):
            q.Enqueue(cost[i])
        while not q.IsEmpty():
            z=q.ExtractMin()
            z.visited=True
            print("Visited:{}".format(z.value))
            neighbours=self.GetNeighbour(z.value)
            for i in neighbours:
                if cost[i].visited==False and cost[i].dist>z.dist+self.adj[i][z.value]:
                    cost[i].dist=z.dist+self.adj[z.value][i]
                    cost[i].pred=z.value
        for j in cost:
            print(j.value,j.dist,j.pred)
d=DGraph(5)
d.Addedge(0,1,5)
d.Addedge(0,2,10)
d.Addedge(1,2,20)
d.Addedge(1,3,25)
d.Addedge(2,3,30)
d.Addedge(3,4,50)
d.Addedge(4,0,6)
d.Dijkstrashortestpath(0)
```

```
Visited:0
Visited:1
Visited:2
Visited:3
Visited:4
0 0 0
1 5 0
2 25 1
```

```
3 55 2
4 105 3
```