In [2]:
```
import heapq
graph={
    'A':{'B':2,'C':3},
    'B':{'D':3,'E':1},
    'C':{'F':2},
    'D':{},
    'E':{'F':1},
    'F':{}
}

def dijkstra(graph,node):
    #Assign infinity to all other nodes
    distances={node:float('inf') for node in graph}
    #The distance value of start node is zero „0“
    distances[node]=0

    print("distances::",distances)

    #Predecessor of node is stored here
    previous={node:None for node in graph}
    queue=[(0,node)] #queue stores start „node“ with edge distance value „0“.
    while queue:
        #heapq of python maintains the priority queue (min queue)
        #heappop() method of heapq will pop the minimum val of the heap
        current_distance, current_node = heapq.heappop(queue)
        # relaxation, visit all the successors of current_node and get the edge cost c

        for next_node, weight in graph[current_node].items():
            distance_temp=current_distance+weight
        #if the distance of the currently visited node is smaller than the earlier sto
        #then update the distance value of current node with smaller cost
            if distance_temp<distances[next_node]:
                distances[next_node]=distance_temp
                previous[next_node]=current_node
                heapq.heappush(queue,(distance_temp,next_node))
            print("Distances::",distances)

    return distances,previous

#Driver Code
Node_distance, Path = dijkstra(graph,'A')
print(Node_distance)
print(Path)
```

```
distances:: {'A': 0, 'B': inf, 'C': inf, 'D': inf, 'E': inf, 'F': inf}
Distances:: {'A': 0, 'B': 2, 'C': inf, 'D': inf, 'E': inf, 'F': inf}
Distances:: {'A': 0, 'B': 2, 'C': 3, 'D': inf, 'E': inf, 'F': inf}
Distances:: {'A': 0, 'B': 2, 'C': 3, 'D': 5, 'E': inf, 'F': inf}
Distances:: {'A': 0, 'B': 2, 'C': 3, 'D': 5, 'E': 3, 'F': inf}
Distances:: {'A': 0, 'B': 2, 'C': 3, 'D': 5, 'E': 3, 'F': 5}
Distances:: {'A': 0, 'B': 2, 'C': 3, 'D': 5, 'E': 3, 'F': 4}
{'A': 0, 'B': 2, 'C': 3, 'D': 5, 'E': 3, 'F': 4}
{'A': None, 'B': 'A', 'C': 'A', 'D': 'B', 'E': 'B', 'F': 'E'}
```

In [ ]: