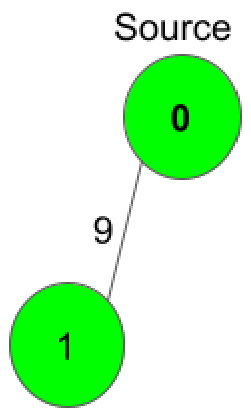
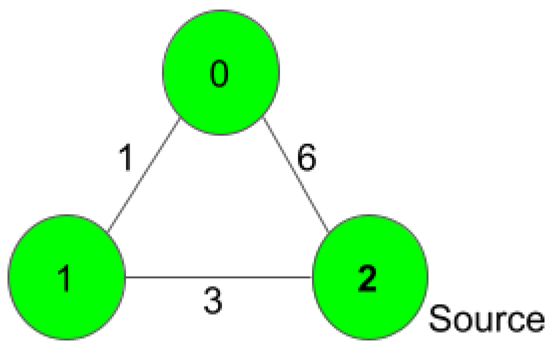
Dijkstra Algorithm

Given a weighted, undirected and connected graph of V vertices and E edges, Find the shortest distance of all the vertex's from the source vertex S.
Note: The Graph doesn't contain any negative weight cycle.



```
S = 0
Output:
0 9
Explanation:
Shortest distance of all nodes from
source is printed.
```



```
Output:
4 3 0
Explanation:
For nodes 2 to 0, we can follow the path-
2-1-0. This has a distance of 1+3 = 4,
whereas the path 2-0 has a distance of 6. So,
```

~~~~
the Shortest path from 2 to 0 is 4.
The other distances are pretty straight-forward.

``````Python
import heapq
class Solution:

    #Function to find the shortest distance of all the vertices
    #from the source vertex S.
    def dijkstra(self, V, adj, S):
        #code here
        visited = [False]*V
        heap = []
        cost = [0]*V
        heap.append((0,S))
        while len(heap):
            weigth,node = heapq.heappop(heap)
            if visited[node]==True:
                continue
            visited[node] = True
            cost[node] = weigth
            for nbr in adj[node]:
                tempNode,wt = nbr
                if visited[tempNode]==False:
                    heapq.heappush(heap,(weigth+wt,tempNode))
        return cost
~~~~