19. To Implement Dijkstra's shortest path Algorithm

Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph. It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

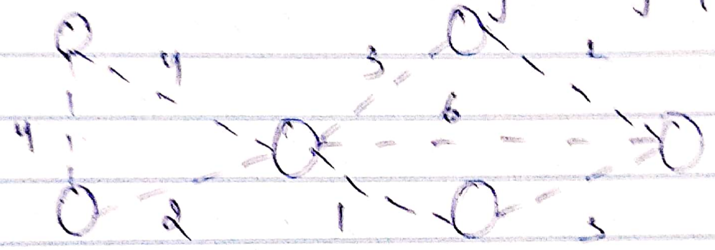
Time complexity O(Elogv)

where, E is the number of edges and v is the number of vertices

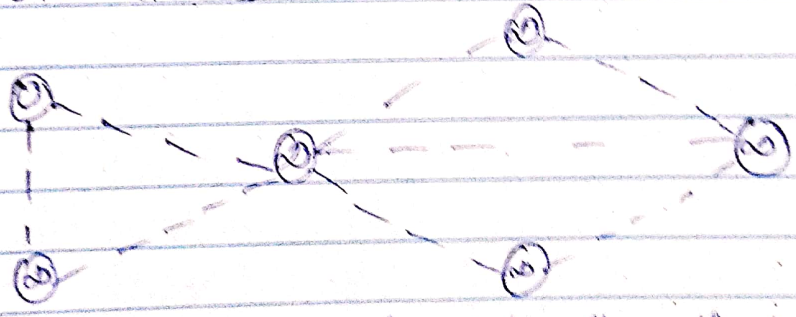
It is used for the finding shortest path and find the locations in the map.

Example:

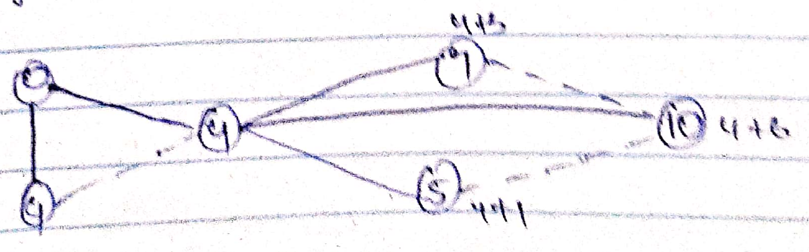
Start with a weighted graph



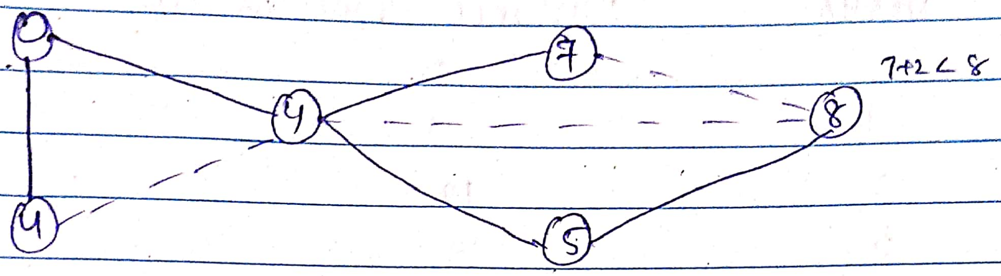
Choose a starting vertex and assign infinity path values to all other devices



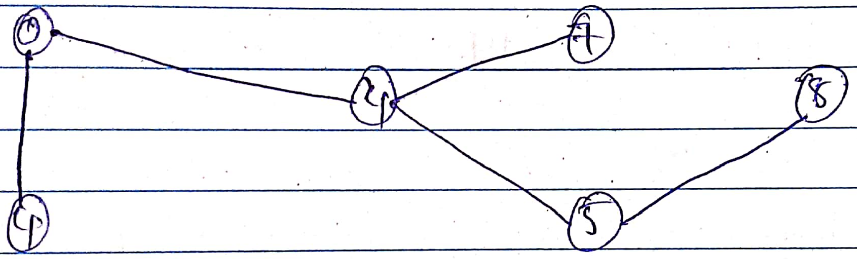
Go to each vertex and update its path length. If the path length of the adjacent vertex is lesser than new path length, don't update it.



Avoid updating path lengths of already visited vertices; After each iteration, we pick the unvisited vertex with the least path length. So we choose 5 before 7



Notice how the rightmost vertex has its path length updated twice. Repeat until all the vertices have been visited.



Code:

```
import sys
vertices=[[0,0,1,1,0,0,0],
          [0,0,1,0,0,1,0],
          [1,1,0,1,1,0,0],
          [1,0,1,0,0,0,1],
          [0,0,1,0,0,1,0],
          [0,1,0,0,1,0,1],
          [0,0,0,1,0,1,0]]
edges=[[0,0,1,2,0,0,0],
       [0,0,2,0,0,3,0],
       [1,2,0,1,3,0,0],
       [2,0,1,0,0,0,1],
            [0,0,3,0,0,2,0],
            [0,3,0,0,2,0,1],
            [0,0,0,1,0,1,0]]
def to_be_visited():
      global visited_and_distance
      V=-10
      for index in range(num_of_vertices):
          if visited_and_distance[index][0] ==0 \
            and (V<0 or visited_and_distance[index][1] <=
                visited_and_distance[V][1]):
            V=index
      return V
num_of_vertices = len (vertices[0])
visited_and_distance=[[0,0]]
for i in range(num_of_vertices-1):
     visited_and_distance.append([0,sys.maxsize])
for vertex in range(num_of_vertices):
     to_visit = to_be_visited()
     for neighbour_index in range (num_of_vertices):
i=0
for distance in visited_and_distance:
      print("Distance of", chr(ord('a')+i),
            "from source vertex:", distance[1])
      i=i+1
```

Output:-

```
Vertex        Distance from Source
  0                 0
  1                 4
  2                 12
  3                 19
  4                 21
  5                 11
  6                 9
  7                 8
  8                 14
```