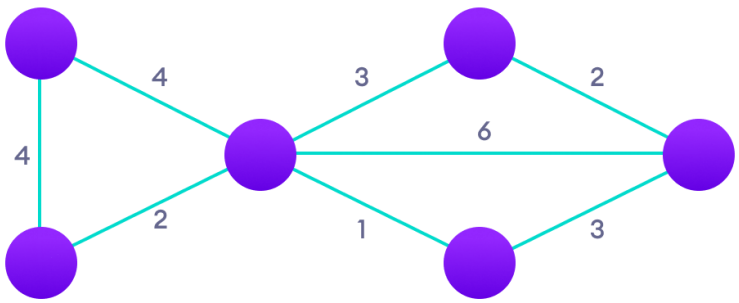
Aim: To implement Single Source Shortest Path using Greedy Approach (Dijkstra)

Theory: 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.

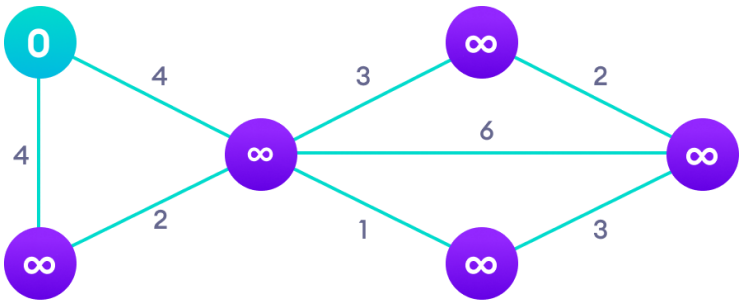
The algorithm uses a greedy approach in the sense that we find the next best solution hoping that the end result is the best solution for the whole problem.

Example: For a given graph



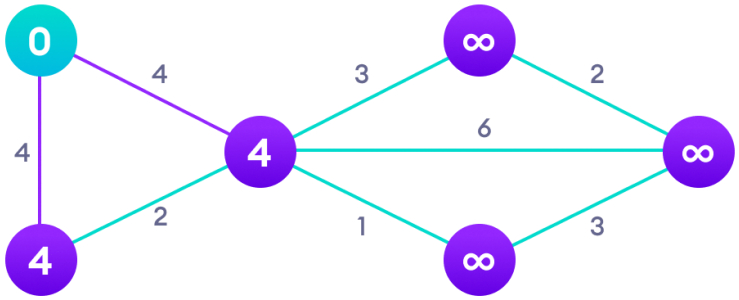
Step: 1

Start with a weighted graph



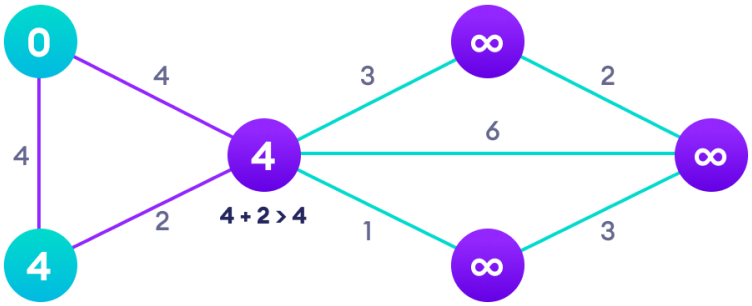
Step: 2

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



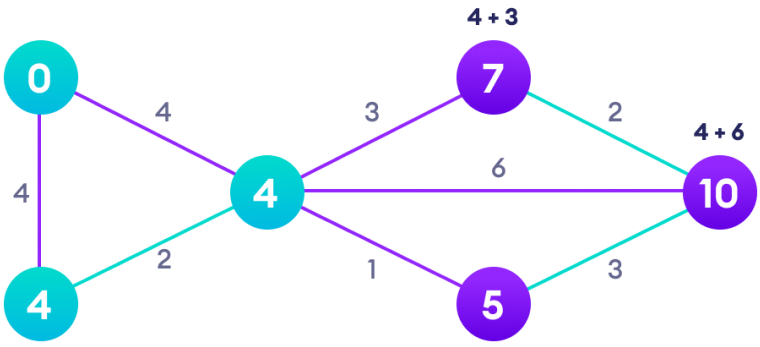
Step: 3

Go to each vertex and update its path length



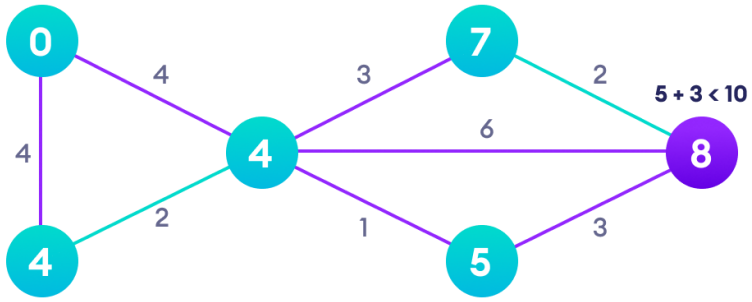
Step: 4

If the path length of the adjacent vertex is lesser than new path length, don't update it



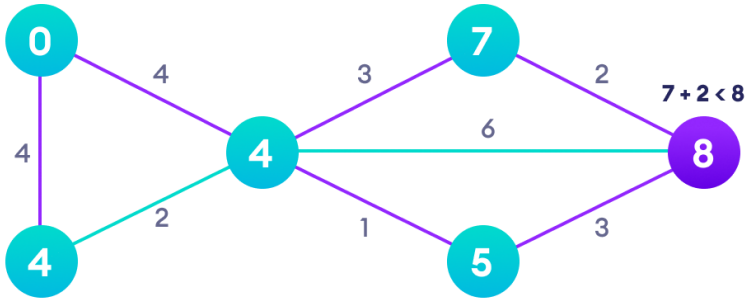
Step: 5

Avoid updating path lengths of already visited vertices



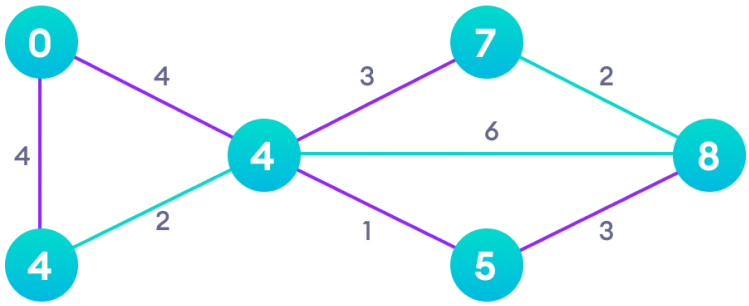
Step: 6

After each iteration, we pick the unvisited vertex with the least path length. So we choose 5 before 7



Step: 7

Notice how the rightmost vertex has its path length updated twice



Step: 8

Repeat until all the vertices have been visited

Algorithm:

```
Dijkstra(G,w,s)
        Initialize_single_source(G,s)
        S = Φ
        Q = G.V
        While(Q!= Φ)
                u = Extract_min(Q)
                S = S U {u}
                For each vertex v of G Adj[u]
                        Relax(u,v,w)
Initialize_single_source(G,s)
        For each vertex v of G.V
                v.d = ∞
                v. π = NIL
Relax(u,v,w)
        If v.d > u.d + w(u,v)
                v.d = u.d + w(u,v)
                v. π = u
```

Code: //dijkstra algo shortest distance from source

```
#include <stdio.h>
#include<stdbool.h>
#include <limits.h>
int minDistance(int dist[], bool sptSet[])
{
   int min = INT_MAX, min_index;
```

```
    for (int v = 0; v < 6; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;

    return min_index;
}
void dijkstra(int graph[6][6],int src){
    int dist[6];
    bool sptSet[6];
    for (int i = 0; i < 6; i++)
        {dist[i] = INT_MAX;
        sptSet[i] = false;}

    dist[src] = 0;

    for (int count = 0; count < 5; count++) {

        int u = minDistance(dist, sptSet);

        sptSet[u] = true;

        for (int v = 0; v < 6; v++)

            if (!sptSet[v] && graph[u][v]
                && dist[u] != INT_MAX
```

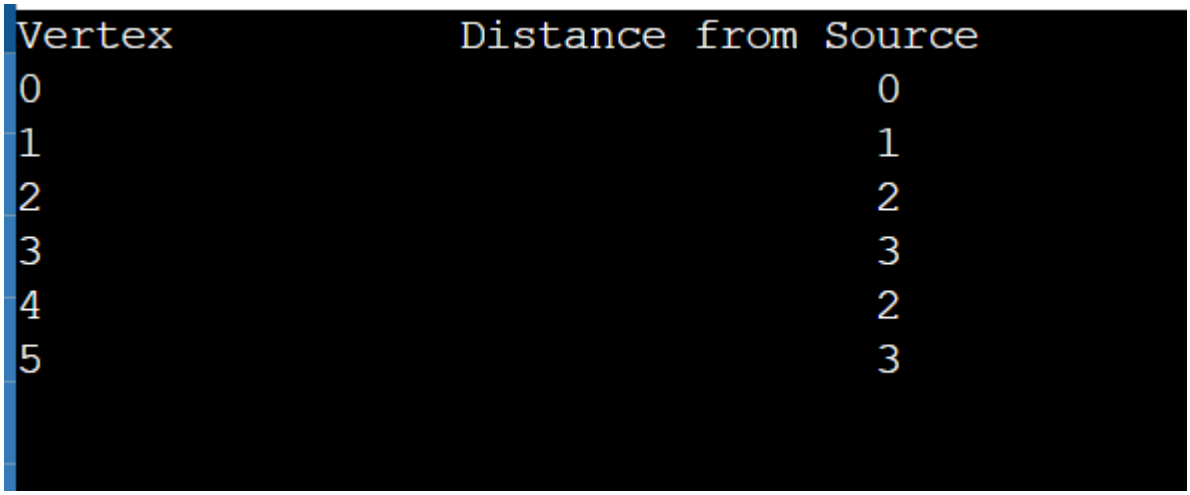
```
            && dist[u] + graph[u][v] < dist[v])
            dist[v] = dist[u] + graph[u][v];
    }

    printf("Vertex \t\t Distance from Source\n");
    for (int i = 0; i < 6; i++)
        printf("%d \t\t\t\t %d\n", i, dist[i]);
}



void main()
{
    int graph[6][6]={{0,1,4,0,0,0},
                {0,0,1,2,1,0},
                {0,0,0,0,5,0},
                {0,0,0,0,0,2},
                {0,0,0,0,0,1},
                {0,0,0,0,0,0}};
    dijkstra(graph,0);
}
```

Output:



Conclusion: Thus we have implemented shortest path of each node from the source using Greedy Approach (Dijkstra's Algorithm)