Dijkstra's algorithm to compute shortest path

```
#include <bits/stdc++.h>
using namespace std;
# define V 9

int minDistance(int dist[], bool sptSet[])
{
    int min = 9999, min_index;
    for (int v=0; v<V; v++)
        if(sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    return min_index;
}
void printPath(int parent[], int j)
{
    if (parent[j] == -1)
        return;
    printPath(parent, parent[j]);
    cout << j << " ";
}
int printSolution(int dist[], int n, int parent[])
{
    int src = 0;
    cout << "Vertex \t Distance \t Path" << endl;
    for(int i=1; i<V; i++)
    {
        cout << "\n" << src << " -> " << i << " \t\t " << dist[i] << " \t\t "
            << src << endl;
        printPath(parent, i);
    }
}
```

Chirag Swamy
1BM19CS202

```
void dijkstra(int graph[V][V], int src)
{
  int dist[V];
  bool sptset[V];
  int parent[V];
  for (int i = 0; i < V; i++)
  {
     parent[0] = -1;
     dist[i] = 9999;
     sptset[i] = false;
  }
  dist[src] = 0;
  for(int count = 0; count < V-1; count++)
  {
     int u = minDistance(dist, sptset);
     sptset[u] = true;
     for(int v = 0; v < V; v++)
        if(!sptset[v] && graph[u][v] && dist[u]+graph[u][v]
              graph[u][v] < dist[v])
        {
           parent[v] = u;
           dist[v] = dist[u] + graph[u][v];
        }
  }
  printSolution(dist, v, parent);
}
```