Sahana L
1BM18CS089

Implement Dijkstra's algorithm to compute the shortest path through a graph.

```
#include <iostream.h>
using namespace
#define INF 99
#define N 6

int dijkstra(int cost[][N], int source, int target);
int dijkstra(int cost[][N], int source, int target);
{
  int dist[N], prev[N], selected[N] = {0}, i, m, min, start,
  d, j;
  char path[N];
  for (i=1; i<N; i++)
  {
    dist[i] = INF;
    prev[i] = -1;
  }
  start = source;
  selected[start] = 1;
  dist[start] = 0;
  while (selected[target] == 0)
  {
    min = INF;
    m = 0;
    for (i=1; i<N; i++)
    {
      d = dist[start] + cost[start][i];
      if (d < dist[i] && selected[i] == 0)
      {
        dist[i] = d;
        prev[i] = start;
      }
```

```
    if (min > dist[i] && selected[i]==0)
    {
      min = dist[i];
      m=i;
    }
  }
  start=m;
  selected[start]=1;
}
start = target;
j=0;
while (start!=-1)
{
  path[j++]=start+65;
  start=prev[start];
}
path[j]='\0';
strrev(path);
cout<<path;
return dist[target];
}

int main()
{
 int cost[N][N], i,j, w,ch, co;
 int source, target, x,y;
 cout<<"Shortest Path Algorithm DIJKSTRA's ALGORITHM\n";
 for(i=1; i<N; i++)
 for(j=1; j<N; j++)
  cost[i][j]=INF;
 for(x=1; x<N; x++)
 {
   for(y=x+1; y<N; y++)
   {
```

```
cout << " Enter the weight of the path between
node";
cout << x;
cout << "and";
cout << "y";
cin >> w;
cost[x][y] = cost[y][x] = w;
}
cout << "\n";
}
cout << "\n Enter the Source");
cin >> source;
cout << "\n Enter the target";
cin >> target;
co = dijkstra (.cost, source, target);
cout << "\n Shortest Path;
cout << co;
}
```