Dijkstra's algorithm to compute shortest path

```
void dijkstra(int G[max][max], int n, int startnode)
{
    int cost[max][max], distance[max], pred[max];
    int visited[max], count, mindistance, nextnode, i, j;
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            if(G[i][j]==0)
                cost[i][j] = INFINITY;
            else
                cost[i][j] = G[i][j];
        for(i=0; i<n; i++)
        {
            distance[i] = cost[startnode][i];
            pred[i] = startnode;
            visited[i] = 0;
        }
        distance[startnode] = 0;
        visited[startnode] = 1;
        count = 1;
        while(count<n-1)
        {
            mindistance = INFINITY;
            for(i=0; i<n; i++)
                if(distance[i]<mindistance && !visited[i])
                {
                    mindistance = distance[i];
                    nextnode = i;
                }
            visited[nextnode] = 1;
```

```
for (i = 0 ; i<n ; i++)
    if(!visited[i])
        if (min distance + cost [next node] [i] < distance [i])
        {
            distance[i] = mindistance + cost [next node] [i];
            pred[i] = next node;
        }
        count++;
    }
for (i = 0; i<n ; i++)
if (i != startnode)
{
    cout << "\nDistance of node " << i << " = " << distance[i];
    cout << "\npath = " << i;
    j = i;
    do {
        j = pred[j];
        cout << " <- " << j;
    }
    while (j != startnode);
}
}
```