

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

```
#include <iostream>
#include <limits>
using namespace std;
int a[30][30], n;
int minimum (int visited[], int dist[])
{
    int mindis = 10000, in mini;
    for (int i = 0; i < n; i++)
    {
        if (!visited[i] && dist[i] < mindis)
        {
            mindis = dist[i];
            mini = i;
        }
    }
    return mini;
}

void dijkstra [mt & src]
{
    int dist[n], visited[n];
    for (int i = 0; i < n; i++)
    {
        dist[i] = 10000;
        visited[i] = 0;
    }
}
```

dist[src] = 0;

for (int i = 0; i < n - 1; i++)

{ int u = minimum (visited, dist);

visited[u] = 1;

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

{ if (!visited[v] && a[u][v] != 10000 &&

dist[u] != 10000 && (dist[u] + a[u][v] < dist[v]))

dist[v] = dist[u] + a[u][v];

}

}

cout << "Vertices | Distance from source | n" << endl;

for (int i = 0; i < n; i++)

{ if (i == src)

cout << i << " " << dist[i] << endl;

}

}

int main()

{ cout << "Enter the no. of vertices " << endl;

cin >> n;

cout << "Enter the weighted & adjacency matrix
(enter 10000 if there is no edge.)" << endl;

```
for (int i = 0; i < n; i++)  
{  
    for (int j = 0; j < n; j++)  
        cin >> a[i][j];  
}
```

```
int src;
```

```
cout << "Enter the source vertex" << endl;
```

```
cin >> src;  
dijkstra (src);  
return 0;
```

```
}
```