# Dijkstra’s – Shortest Path Algorithm (SPT) – Adjacency Matrix

#include<stdio.h>

int main()

{

int cost[10][10],i,j,n,source,target,visited[10]={0},min=999,dist[10],pre[10];

int start,m,d,path[10];

printf("Enter number of nodes\n ");

scanf("%d",&n);

printf("Enter weight of all the paths in adjacency matrix form\n");

// Input graph

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

}

printf("Enter the source\n");

scanf("%d",&source);

printf("Enter the target\n");

scanf("%d",&target);

// logic for dijkstra's aglorithm

start=source;

for(i=1;i<=n;i++)

{

dist[i]=999; // here we initialize all distances with the maximum value (999) you take any other value also

pre[i]=-1; // pre for the previous node

}

visited[source]=1; // visited source node

dist[source]=0; // distance of first node from first node is 0

while(visited[target]==0)

{

min=999;

m=0;

for(i=1;i<=n;i++)

{

d=dist[start]+cost[start][i]; // calcualte the distance from the source

if(d<dist[i] && visited[i]==0)

{

dist[i]=d;

pre[i]=start;

}

if(min>dist[i] && visited[i]==0)

{

min=dist[i];

m=i;

}

}

start=m;

visited[m]=1;

}

// logic for printing path

start=target;

j=0;

while(start!=-1)

{

path[j++]=start;

start=pre[start];

}

// printing of the path

for(i=j-1;i>=0;i--)

{

if(i!=j-1)

printf(" to ");

printf("%d",path[i]);

}

printf("\n shortest path is %d",dist[target]);

return 0;

}

Output:

Enter number of nodes

5

Enter weight of all the paths in adjacency matrix form

0 10 0 30 100

10 0 50 0 0

0 50 0 20 10

30 0 20 0 60

100 0 10 60 0

0 10 0 30 100

10 0 50 0 0

0 50 0 20 10

30 0 20 0 60

100 0 10 60 0

Enter the source

1

Enter the target

5

1 to 4 to 3 to 5

shortest path is 60# Dijkstra’s – Shortest Path Algorithm (SPT) – Adjacency Matrix

#include<stdio.h>

int main()

{

int cost[10][10],i,j,n,source,target,visited[10]={0},min=999,dist[10],pre[10];

int start,m,d,path[10];

printf("Enter number of nodes\n ");

scanf("%d",&n);

printf("Enter weight of all the paths in adjacency matrix form\n");

// Input graph

for(i=1;i<=n;i++)

{

for(j=1;j<=n;j++)

{

scanf("%d",&cost[i][j]);

if(cost[i][j]==0)

cost[i][j]=999;

}

}

printf("Enter the source\n");

scanf("%d",&source);

printf("Enter the target\n");

scanf("%d",&target);

// logic for dijkstra's aglorithm

start=source;

for(i=1;i<=n;i++)

{

dist[i]=999; // here we initialize all distances with the maximum value (999) you take any other value also

pre[i]=-1; // pre for the previous node

}

visited[source]=1; // visited source node

dist[source]=0; // distance of first node from first node is 0

while(visited[target]==0)

{

min=999;

m=0;

for(i=1;i<=n;i++)

{

d=dist[start]+cost[start][i]; // calcualte the distance from the source

if(d<dist[i] && visited[i]==0)

{

dist[i]=d;

pre[i]=start;

}

if(min>dist[i] && visited[i]==0)

{

min=dist[i];

m=i;

}

}

start=m;

visited[m]=1;

}

// logic for printing path

start=target;

j=0;

while(start!=-1)

{

path[j++]=start;

start=pre[start];

}

// printing of the path

for(i=j-1;i>=0;i--)

{

if(i!=j-1)

printf(" to ");

printf("%d",path[i]);

}

printf("\n shortest path is %d",dist[target]);

return 0;

}

Output:

Enter number of nodes

5

Enter weight of all the paths in adjacency matrix form

0 10 0 30 100

10 0 50 0 0

0 50 0 20 10

30 0 20 0 60

100 0 10 60 0

0 10 0 30 100

10 0 50 0 0

0 50 0 20 10

30 0 20 0 60

100 0 10 60 0

Enter the source

1

Enter the target

5

1 to 4 to 3 to 5

shortest path is 60