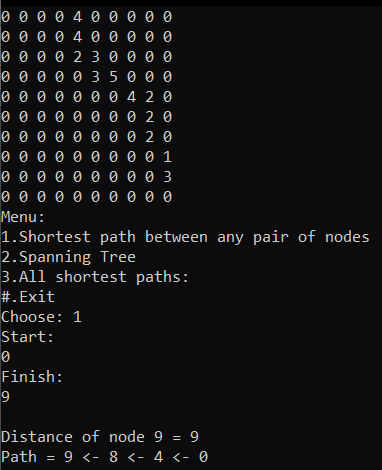


**ECE 325 - Iterative Methods**

**Practical Assignment 11**



**Παναγιώτης Ρούσου – 932935**

**Λουκιανός Κατωμονιάτη – 923279**

**Κωνσταντίνος Χατζηκώστα – 918088**

**ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ ΑΣΚΗΣΗΣ: 25/11/2019**

**Πρόγραμμα 1:**

//choice 1

pairdijkstra(n, startnode, finish)

{

for i=0 to n step 1

for j=0 to n step 1

if( G[i][j] equal to zero )

cost[i][j]=INFINITY; // constant

else

cost[i][j]=G[i][j];

for i=0 to n step 1

{

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 to n step 1

if(distance[i]<mindistanceAND not visited[i])

{

mindistance=distance[i];

nextnode=i;

}

visited[nextnode]=1;

for i=0 to n step 1

if(not visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

for i=0 to n step 1

{

if (i is not equal to startnode)

{

if (i == finish)

{

print "Distance of node " + i + " = " + distance[i];

print "Path = " + i;

j = i;

do

{

j = pred[j];

print " <- " + j;

} while (j != startnode);

}

}

}

}

//choice 2

Graph::kruskal\_algorithm()

{

sort(G.begin(), G.end());

for i=0 to G.size() step 1

{

uSt = find\_set(G[i].second.first);

vEd = find\_set(G[i].second.second);

if (uSt is not equal vEd)

{

push(G[i]) to T;

union\_set(uSt, vEd);

}

}

}

//choice 3

dijkstra(n, startnode)

{

for i=0 to n step 1

for j=0 to n step 1

if( G[i][j] equal to zero )

cost[i][j]=INFINITY; // constant

else

cost[i][j]=G[i][j];

for i=0 to n step 1

{

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 to n step 1

if(distance[i]<mindistanceAND not visited[i])

{

mindistance=distance[i];

nextnode=i;

}

visited[nextnode]=1;

for i=0 to n step 1

if(not visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

for i=0 to n step 1

{

if (i is not equal to startnode)

{

print "Distance of node " + i + " = " + distance[i];

print "Path = " + i;

j = i;

do

{

j = pred[j];

print " <- " + j;

} while (j != startnode);

}

}

}

**Παραδείγματα εκτέλεσης :**

