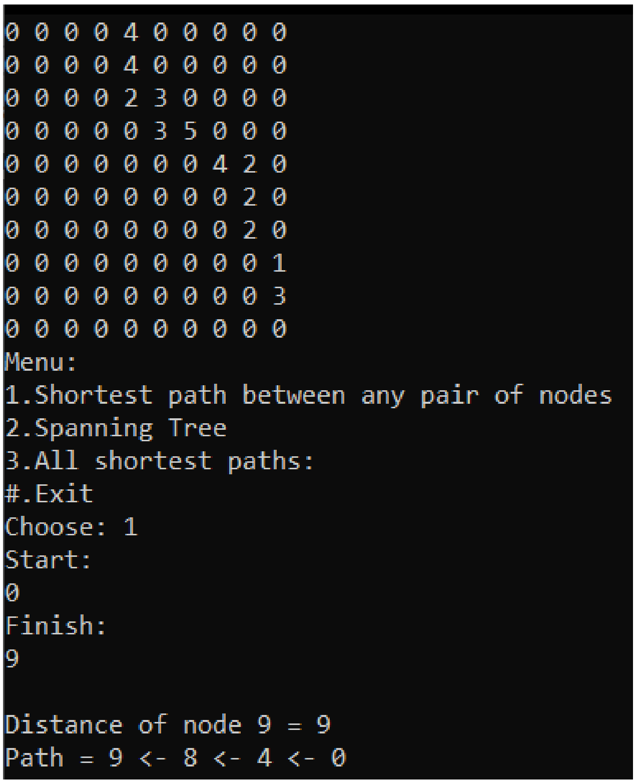
Πανεπιστήμιο Κύπρου

Τμήμα Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

ECE 325 - Iterative Methods

Practical Assignment 11



```
0 0 0 0 4 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0
0 0 0 0 2 3 0 0 0 0
0 0 0 0 0 3 5 0 0 0
0 0 0 0 0 0 0 4 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0
Menu:
1.Shortest path between any pair of nodes
2.Spanning Tree
3.All shortest paths:
#.Exit
Choose: 1
Start:
0
Finish:
9

Distance of node 9 = 9
Path = 9 <- 8 <- 4 <- 0
```

Παναγιώτης Ρούσου – 932935

Λουκιανός Κατωμονιάτη – 923279

Κωνσταντίνος Χατζηκώστα – 918088

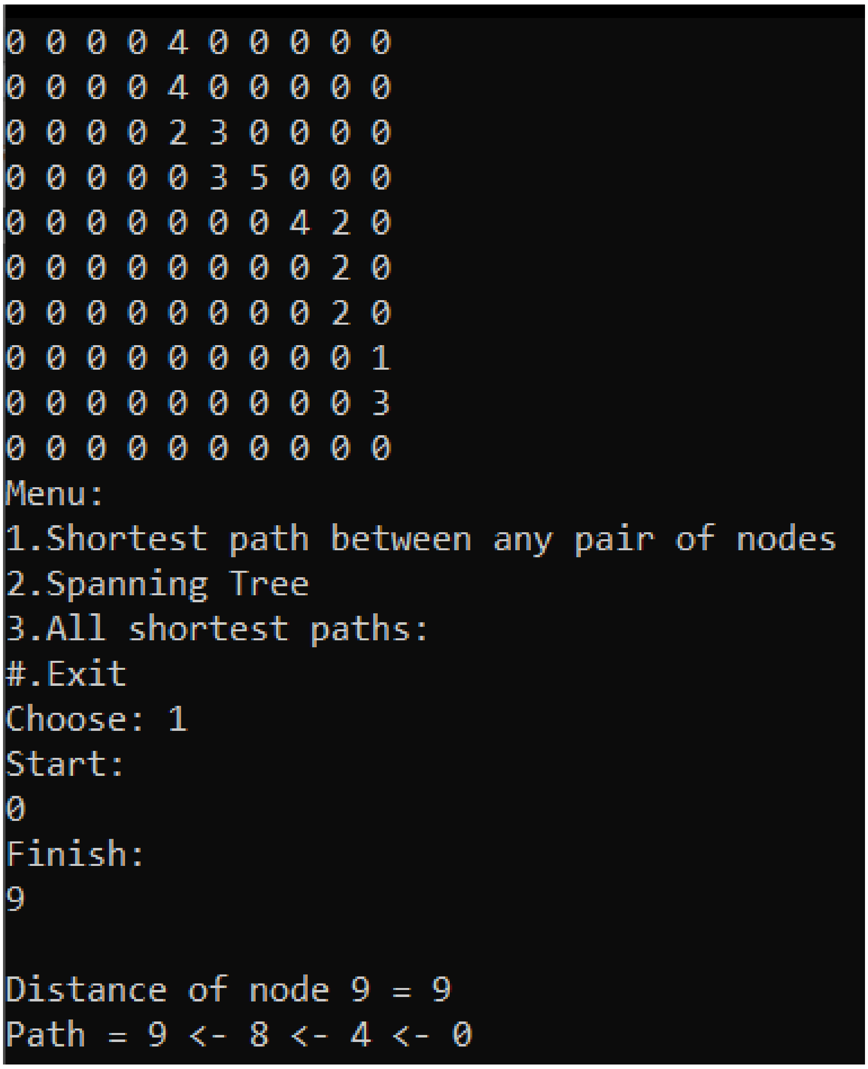
ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ ΑΣΚΗΣΗΣ: 25/11/2019

Πρόγραμμα 1:

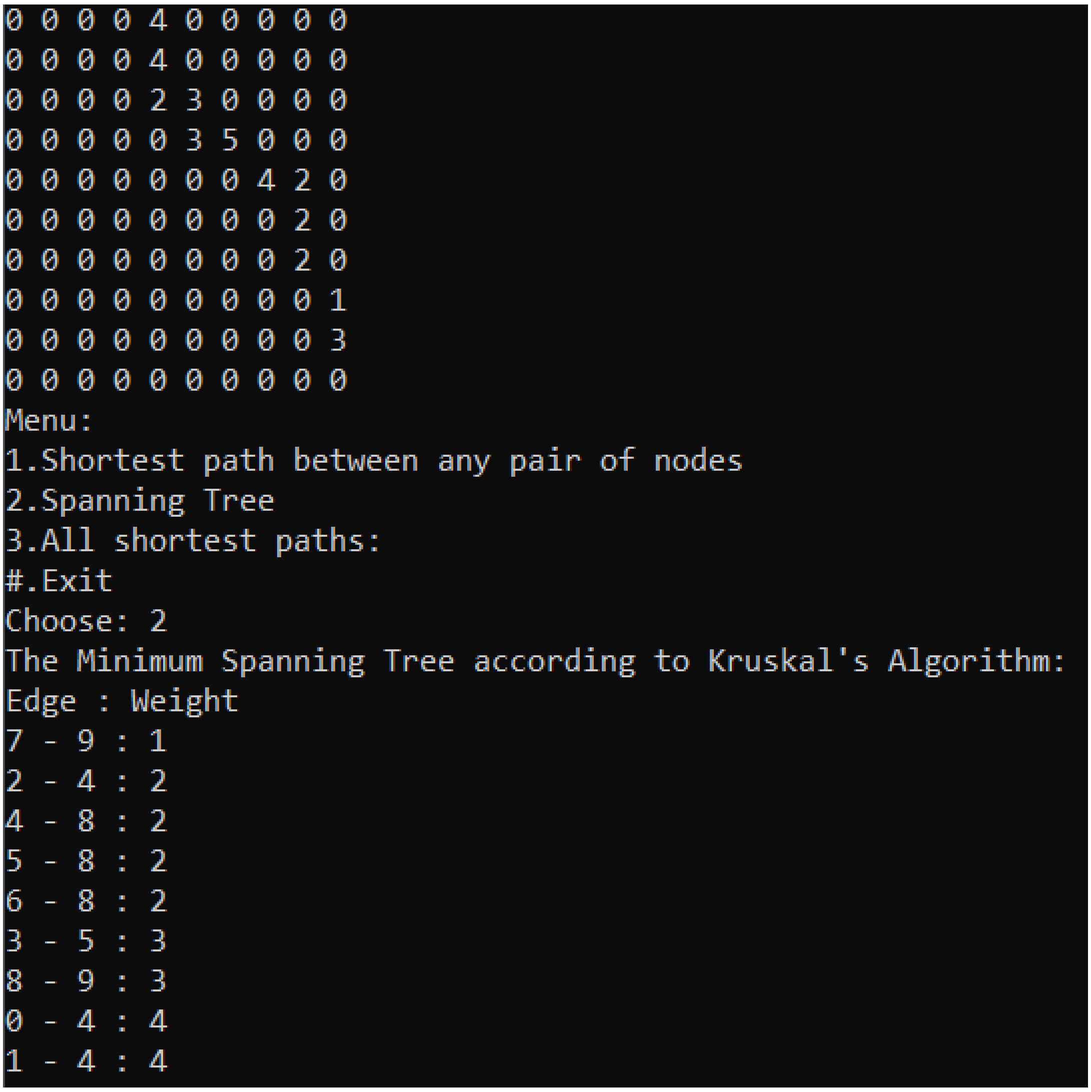
```
//choice 1
pairdijkstra(n, startnode, finish)
{
        for i=0 to n step 1
                for j=0 to n step 1
                        if( G[i][j] equal to zero )
                                cost[i][j]=INFINITY;   // constant
                         else
                                cost[i][j]=G[i][j];
        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);
        }
    }
}
```

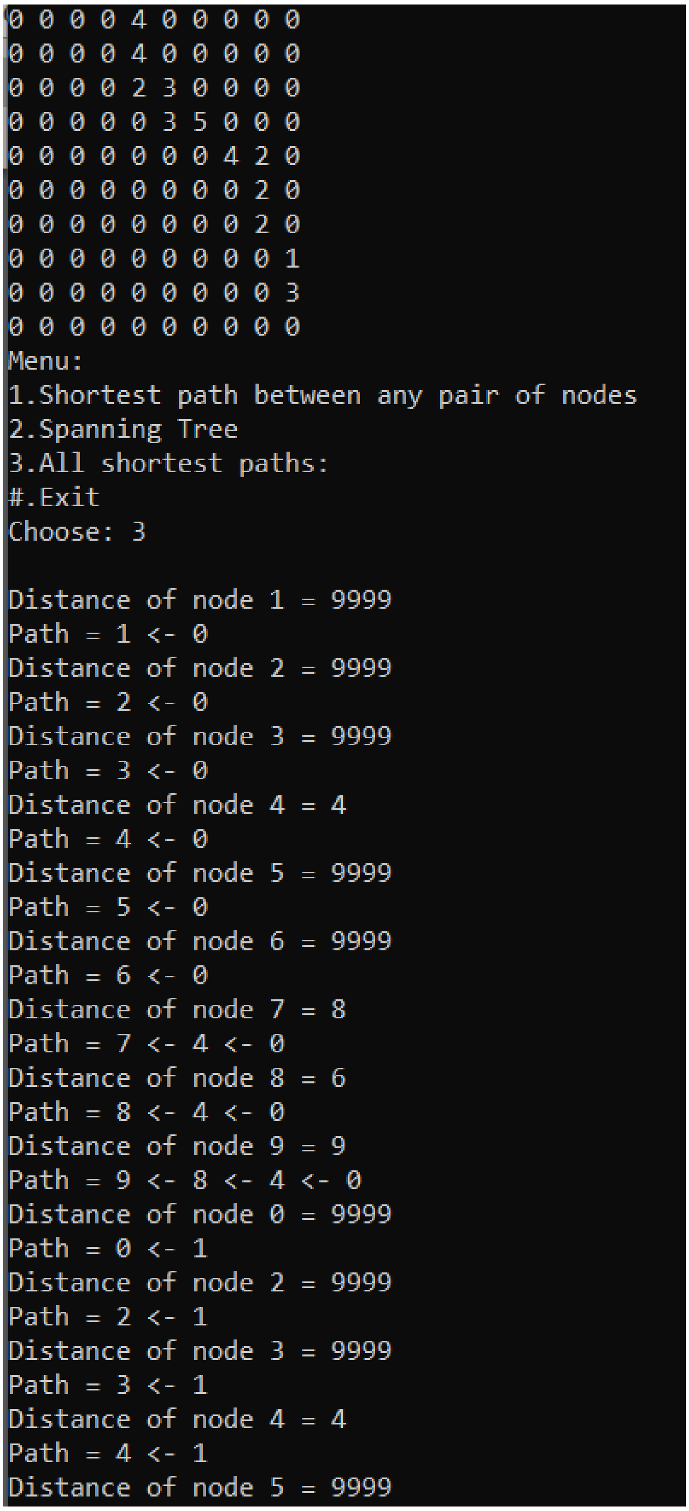
<u>Παραδείγματα εκτέλεσης :</u>



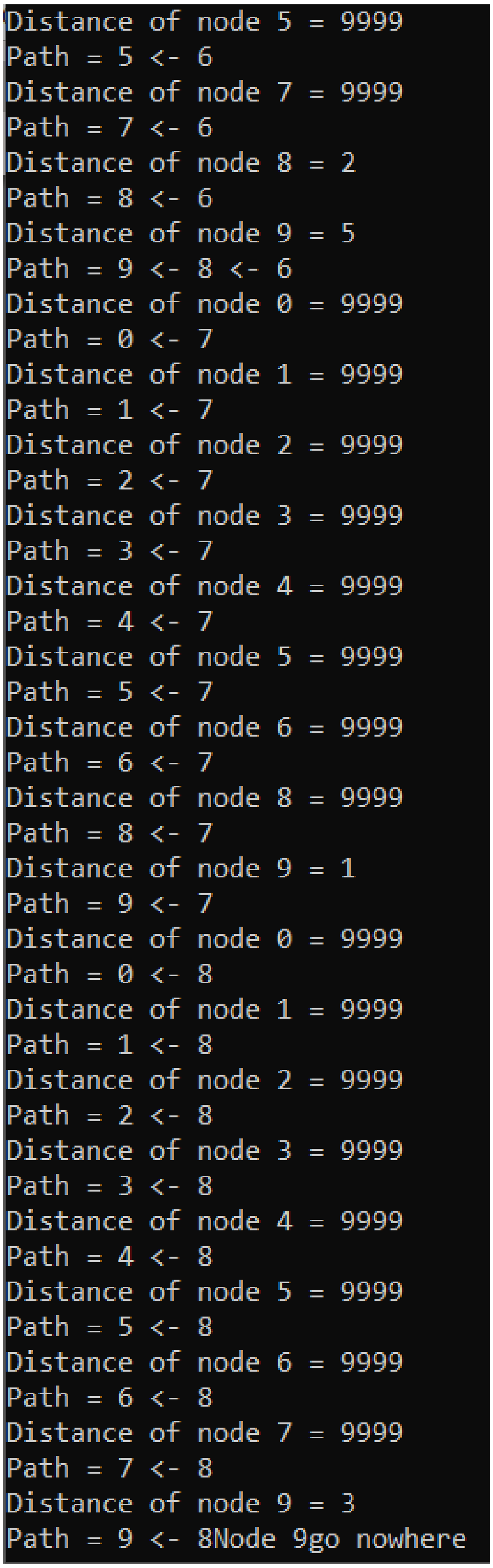
```
0 0 0 0 4 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0
0 0 0 0 2 3 0 0 0 0
0 0 0 0 0 3 5 0 0 0
0 0 0 0 0 0 0 4 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0
Menu:
1.Shortest path between any pair of nodes
2.Spanning Tree
3.All shortest paths:
#.Exit
Choose: 1
Start:
0
Finish:
9

Distance of node 9 = 9
Path = 9 <- 8 <- 4 <- 0
```



```
0 0 0 0 4 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0
0 0 0 0 2 3 0 0 0 0
0 0 0 0 0 3 5 0 0 0
0 0 0 0 0 0 0 4 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0
Menu:
1.Shortest path between any pair of nodes
2.Spanning Tree
3.All shortest paths:
#.Exit
Choose: 2
The Minimum Spanning Tree according to Kruskal's Algorithm:
Edge : Weight
7 - 9 : 1
2 - 4 : 2
4 - 8 : 2
5 - 8 : 2
6 - 8 : 2
3 - 5 : 3
8 - 9 : 3
0 - 4 : 4
1 - 4 : 4
```



```
0 0 0 0 4 0 0 0 0 0
0 0 0 0 4 0 0 0 0 0
0 0 0 0 2 3 0 0 0 0
0 0 0 0 0 3 5 0 0 0
0 0 0 0 0 0 0 4 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 2 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 3
0 0 0 0 0 0 0 0 0 0
Menu:
1.Shortest path between any pair of nodes
2.Spanning Tree
3.All shortest paths:
#.Exit
Choose: 3

Distance of node 1 = 9999
Path = 1 <- 0
Distance of node 2 = 9999
Path = 2 <- 0
Distance of node 3 = 9999
Path = 3 <- 0
Distance of node 4 = 4
Path = 4 <- 0
Distance of node 5 = 9999
Path = 5 <- 0
Distance of node 6 = 9999
Path = 6 <- 0
Distance of node 7 = 8
Path = 7 <- 4 <- 0
Distance of node 8 = 6
Path = 8 <- 4 <- 0
Distance of node 9 = 9
Path = 9 <- 8 <- 4 <- 0
Distance of node 0 = 9999
Path = 0 <- 1
Distance of node 2 = 9999
Path = 2 <- 1
Distance of node 3 = 9999
Path = 3 <- 1
Distance of node 4 = 4
Path = 4 <- 1
Distance of node 5 = 9999
```



```
Distance of node 5 = 9999
Path = 5 <- 6
Distance of node 7 = 9999
Path = 7 <- 6
Distance of node 8 = 2
Path = 8 <- 6
Distance of node 9 = 5
Path = 9 <- 8 <- 6
Distance of node 0 = 9999
Path = 0 <- 7
Distance of node 1 = 9999
Path = 1 <- 7
Distance of node 2 = 9999
Path = 2 <- 7
Distance of node 3 = 9999
Path = 3 <- 7
Distance of node 4 = 9999
Path = 4 <- 7
Distance of node 5 = 9999
Path = 5 <- 7
Distance of node 6 = 9999
Path = 6 <- 7
Distance of node 8 = 9999
Path = 8 <- 7
Distance of node 9 = 1
Path = 9 <- 7
Distance of node 0 = 9999
Path = 0 <- 8
Distance of node 1 = 9999
Path = 1 <- 8
Distance of node 2 = 9999
Path = 2 <- 8
Distance of node 3 = 9999
Path = 3 <- 8
Distance of node 4 = 9999
Path = 4 <- 8
Distance of node 5 = 9999
Path = 5 <- 8
Distance of node 6 = 9999
Path = 6 <- 8
Distance of node 7 = 9999
Path = 7 <- 8
Distance of node 9 = 3
Path = 9 <- 8Node 9go nowhere
```