

**Project Title: MODIFY DIJKSTRA’S ALGORITHM TO FIND THE LONGEST PATH OF A GRAPH**

**Homework Assignment-II Submission Date: 10th April, 2019.**

**Term Project submitted for:**

**Dr. Hossen A Mustafa**

### Assistant Professor,

### Institute of Information and Communication Technology,BUET.

**Homework Assignment-II submitted by:**

**Md. Noor Alam**

## Student ID: 1017311002

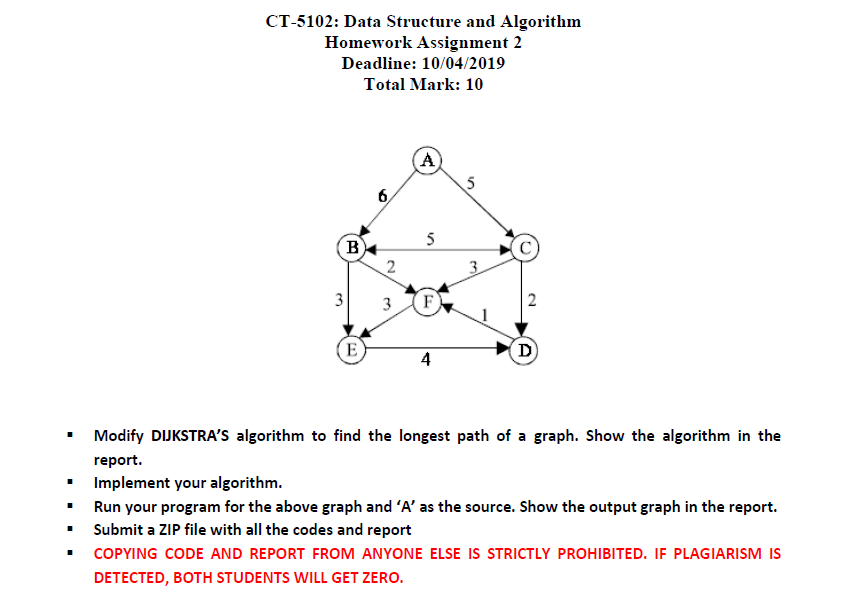
## Course Code: ICT 5102

## Course Name: Data Structure and Algorithm

**Post Graduate Diploma Program in (ICT),**

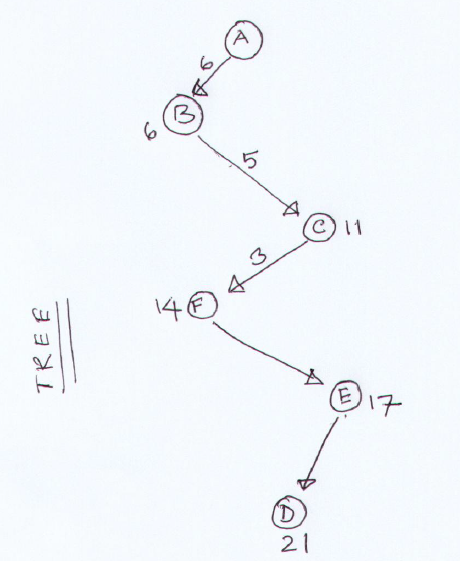
**Institute of Information and Communication Technology,BUET**

**Problem on:**

****

**SOLUTION:**

**Modifying DIJKSTRA’S algorithm to find the longest path of a graph:**

****

*Figure 1.0: Modifying DIJKSTRA’S algorithm to find the longest path of a graph (Tree)*

**Showing the algorithm in the report:**

**Pseudo Code for Dijkstra’s Algorithm:**

DIJKSTRA(G,s)

1 dist[s] = 0;

2 for each vertex v in Vertices V[G] – s

3 dist[v] ← ∞

4 S ← Ø

5 Q ← V[G]

6 while Q ≠ Ø

7 do u ← maxdistance(Q,dist)

8 S ← S U {u}

9 for each vertex v ∈ neighbors[u]

10 do if dist[v] < dist[u] + w(u,v)

11 then d[v] ←d[u] + w(u,v)

12 return dist[]

**Implementing the Algorithm in C:**

#include<stdio.h>

#include<conio.h>

#define INFINITY 9999

#define MAX 10

void dijikstra(int G[MAX][MAX], int n, int startnode);

int main(){

int G[MAX][MAX];

int i, j, n, u;

printf("\nEnter the no. of vertices:: ");

scanf("%d", &n);

printf("\nEnter the adjacency matrix::\n");

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

for(j=0;j < n;j++)

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

//printf("\nEnter the starting node:: ");

//scanf("%d", &u);

dijikstra(G,n,0);

return 0;

}

void dijikstra(int G[MAX][MAX], int n, int startnode)

{

int cost[MAX][MAX], distance[MAX], pred[MAX];

int visited[MAX], count, maxdistance, nextnode, i,j;

char c;

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

for(j=0;j < n;j++)

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

cost[i][j]=0;

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)

{

maxdistance=0;

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

if(distance[i] > maxdistance && !visited[i])

{

maxdistance = distance[i];

nextnode = i;

}

visited[nextnode] = 1;

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

if(!visited[i])

if(maxdistance+cost[nextnode][i] > distance[i])

{

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

pred[i] = nextnode;

}

count++;

}

printf("\nThe output graph in the report.::\n");

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

if(i!=startnode)

{

c = 65 + i;

printf("\nDistance of %c = %d", c, distance[i]);

printf("\nPath = %c", c);

j=i;

do

{

j=pred[j];

c = 65 + j;

printf(" <-%c", c);

}

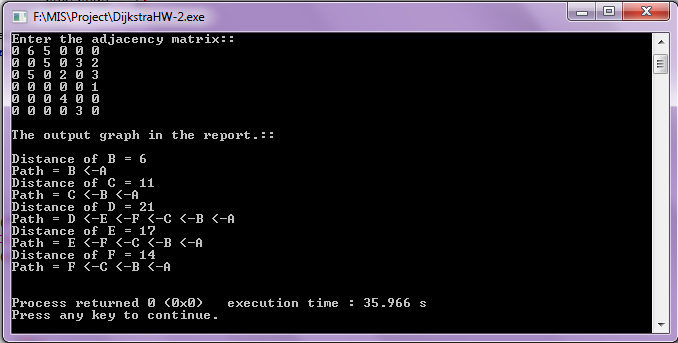
while(j!=startnode);

}

printf("\n\n");

}

**Running the program for the above graph and ‘A’ as the source. Showing here the Input and Output graph in the report:**

****