**01000010001111010101010101101010101010101001000101110011010101011001101110000101010100100011110101010101011010101010101010010001011100110101010110011011100001010101001000111101010101010110101010101010100100010111001101010101100110111000010101010010001111010101010101101010101010101001000101110011010101011001101110000101010100100011110101010101011010101010101010010001011100110101010101010010011101000101111000010101010010001111010001011110010001011110001010101001000111101000101**

**INSTITUTE OF**

**INFORMATION AND COMMUNICATION TECHNOLOGY (IICT)**

|  |  |  |
| --- | --- | --- |
| ***Assignment (2)***  On    **DIJKSTRA’S algorithm**   |  |  | | --- | --- | | **Course Name:**  **Course No:** | **Data Structure and Algorithm**  **ICT 5102** |   **Submited By:**  Sk. Md. Zubayer-Al-Mahmud  Student ID: 1017311014  Session: Oct, 2017  PGD in IICT ,BUET  C:\Users\workgroup\AppData\Local\Microsoft\Windows\INetCache\Content.Word\logoBIRN.PNG  Bangladesh University of Engineering and Technology (BUET) |

|  |  |
| --- | --- |
| **[PAGE INDEX]** | |
| Content: | Page no. |
|  |  |
| 1. **DIJKSTRA’S algorithm** | 2 |
| 1. **DIJKSTRA’S Graphical view** | 3 |
| 1. **Implement DIJKSTRA’S algorithm Code** | 10 |
| 1. **Output Program of DIJKSTRA’S algorithm Code** | 12 |
|  |  |

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

Dijkstra(v1, v2):

for each vertex v: // Initialization

v's distance := 0.

v's previous := none.

v1's distance := 0.

List := {all vertices}.

while List is not empty:

v := Remove List vertex with maximum distance.

mark v as known.

for each unknown neighbor n of v:

dist:= v's distance + edge (v, n)'s weight.

if dist is larger than n's distance:

n's distance := dist.

n's previous := v.

reconstruct path from v2 back to v1, following previous pointers.

1. **DIJKSTRA’S Graphical view to step by step for the longest path:**
2. **Example: Initialization**

4

1

6

5

5

2

2

3

3

3

**Distance(source) = 0**

0

0

0

0

0

0

1. **Example: Update neighbors' distance**

6

5

5

2

2

3

3

3

**Distance(B) = 6**

**Distance(c) = 5**

0

6

5

0

0

0

1

4

1. **Example: Remove List vertex with maximum distance(B) and update neighbors**

6

5

5

2

2

3

3

3

**Distance(C) = 6+5=11**

**Distance(E) = 6+3=8**

**Distance(F) = 6+2=7**

0

6

11

7

0

8

1

4

1. **Example: Remove List vertex with maximum distance(C) and update neighbors**

6

5

5

2

2

3

3

3

**Distance(D) = 11+2=13**

**Distance(F) = 11+3=14**

0

6

11

14

13

8

1

4

1. **Example: Remove List vertex with maximum distance(F) and update neighbors**

6

5

5

2

2

3

3

3

**Distance(E) = 14+3=17**

0

6

11

14

13

17

1

4

1. **Example: Remove List vertex with maximum distance(E) and update neighbors**

6

5

5

2

2

3

3

3

**Distance(D) = 17+4=21**

0

6

11

14

21

17

1

4

1. **Example: Remove List vertex with maximum distance(D) and final path**

6

5

3

3

0

6

11

14

21

17

4

1. **Implement DIJKSTRA’S algorithm Code for the longest path:**

#include<stdio.h>

#define MAX 6

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

int main()

{

printf("\tSk. Md. Zubayer-Al-Mahmud\n\tID: 1017311014\n\tPDG in ICT, BUET.\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("In the given Graph have 6 vertices \nThese are: A, B, C, D, E, F \nAnd Source Vertex: A\n");

int G[MAX][MAX]={{0,6,5,0,0,0},{0,0,5,0,3,2},{0,5,0,2,0,3},

{0,0,0,0,0,1},{0,0,0,4,0,0},{0,0,0,0,3,0}};

int n=6,u=0;

dijkstra(G,n,u);

return 0;

}

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

{ char s[6]={'A','B','C','D','E','F'};

int distance[MAX],pred[MAX];

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

//pred[] stores the predecessor of each node

//count gives the number of nodes seen so far

//initialize pred[],distance[] and visited[]

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

{

distance[i]=G[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n-1)

{

maxdistance=0;// max =0;

//nextnode gives the node at maximum distance

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

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

{

maxdistance=distance[i];

nextnode=i;

}

//check if a better path exists through nextnode

visited[nextnode]=1;

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

if(!visited[i])

if(maxdistance+G[nextnode][i]>distance[i])//.......

{

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

pred[i]=nextnode;

}

count++;

}

//print the path and distance of each node

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

if(i!=startnode)

{

printf("\nDistance from node %c to node %c = %d",s[0],s[i],distance[i]);

printf("\nPath= ");

j=i;

int k=0;

char p[6]={};

do

{

j=pred[j];

p[k]=s[j];

k++;

}while(j!=startnode);

// print by for loop

int m;

for (m=k-1;m>=0;m--)

{

printf("%c --> ",p[m]);

}

printf("%c",s[i]);

printf("\n");

}

}

1. **Output Program of DIJKSTRA’S algorithm Code for the longest path:**

