12/1/23

LAB-11

Implement Dijkstra's algorithm to compute the shortest path for a given topology.

```
#include < stdio.h >
#include <conio.h>
# define INFINITY 9999
# define MAX 10
void dijkstra (int G[MAX][MAX], int n, int startnode);
int main()
{
  int G[MAX][MAX] ,i,j, n,u;
  printf ("Enter no. of vertices : ");
  scanf ("%d", &n);
     printf ("\n Enter the adjacency matrix\n");
  for(i=0; i<n; i++)
    for(j=0.; j<n ;j++)
      scanf ("%d", &n);
      dijkstra(G, n, u);
          return 0; }
void dijkstra (int G[MAX][MAX] , int n, int startnode)
{

  int cost [MAX][MAX] ,distance [MAX], pred [MAX];
  int visited [MAX] , count, mindistance,
  nextnode ,i, j;
  for (i=0; i<n ; i++)
     for (j=0 ; j<n ; j++)
       if (G[i][j] == 0)
          cost [i][j] = INFINITY;
       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)
{
    mindistance = INFINITY;
    for(i=0; i<n; i++)
    if(distance[i] < mindistance && !visited[i])
    {
    mindistance = distance[i];
    nextnode = i;
    }
    for(i=0; i<n; i++)
    if(distance[i] < mindistance && !visited[i])
    {
    mindistance = distance[i];
    nextnode = i;
    }
    for(i=0; i<n; i++)
    if(!visited[i])
    if(mindistance + cost[nextnode][i] < distance[i])
    {
    distance[i] = mindistance + cost[nextnode][i];
    pred[i] = nextnode;
    }
```

```
        count++;
    }
    for (i=0; i<n; i++)
        if (i != startnode)
    {
        printf ("\n Distance of node %d = %d", i, distance[i]);
        printf ("\n Path = %d", i);
            j = i;
        do {
            j = pred[j];
            printf (" <- %d", j);
        while (j != startnode);
        }
    }
```

Output:

Enter the graph.

```
0 9 2 5
9 0 6 8
2 6 0 0
5 8 0 0
```

Vertex	Distance from Source
0	0
1	8
2	2
3	5

LAB-12.

Using TCP/IP sockets, write a client-server pro
to make client sending the file name & the
server to send back the contents of the requ
file if present.

Client:

```
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName, serverPort))
sentence = input('\nEnter File name: ')
clientSocket.send(sentence.encode())
fileContents = clientSocket.recv(1024).decode()
print('\nFrom Server:\n')
print(fileContents)
clientSocket.close()
```

Output:

```
Enter file name: serverTCP.py
From Server:
connectionSocket, addr = serverSocket.accept()
sentence = connectionSocket.recv(1024).decode()
file = open(sentence, "r")
l = file.read(1024)
connectionSocket.send(l.encode())
print('\nSent contents of ' + sentence)
file.close()
connectionSocket.close()
```

Server:

```
from sokit import *
serverName = "127.0.0.1"
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_STREAM)
serverSocket.bind((serverName, serverPort))
serverSocket.listen(1)
while 1:
    print("The server is ready to receive")
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024).decode()
    file = open(sentence, "r")
    l = file.read(1024)
    connectionSocket.send(l.encode())
    print('Sent contents of' + sentence)
    file.close()
    connectionSocket.close()
```

Output:

```
The server is ready to receive

Sent contents of serverTCP.py
The server is ready to receive.
```

LAB-13

Using UDP sockets, write a client-server program to make client sending the file name & the server to send back the contents of the requested file if present.

Client:

```
from sokt import *
serveName = "127.0.0.1"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
sentence = input("\nEnter file name: ")
clientSocket.sendto(bytes(sentence,"utf-8"),(serveName,
fileContents, serverAddress = clientSocket.recvfrom(2048)
print('\nReply from Server:\n')
print(fileContents.decode("utf-8"))
clientSocket.close()
clientSocket.close()
```

Output:

```
Enter file name: serverudp.py
Reply from Server:
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvFrom(2048)
    sentence = sentence.decode("utf-8")
    file = open(sentence, "r")
    l = file.read(2048)
```

```
serverSocket.sendto(bytes(l,"UTF-8"),clientAddress)
print('\nSent contents of ',end='')
print(sentence)
file.close()
```

Server :

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("127.0.0.1", serverPort))
print("The server is ready to receive")
while 1:
    sentence, clientAddress = serverSocket.recvfrom(2048)
    sentence = sentence.decode("UTF-8")
    file = open(sentence, "r")
    l = file.read(2048)
    serverSocket.sendto(bytes(l,"UTF-8"),clientAddress)
    print('\nSent contents of ',end='')
    print(sentence).
    file.close().
```

Output :

```
The server is ready to receive
Sent contents of serverudp.py.
```