Week 10

11/07/2024

PAGE NO :
DATE :

Dijkstra's Algorithm

```
#include <stdio.h>
#include <limits.h>

int main(){
    Printf("Enter number of nodes: ");
    int n;
    scanf("%d", &n);
    int g[n][n];
    Printf("Enter adjacency matrix: \n");
    for (int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            scanf("%d", &g[i][j]);
        }
    }
    int s;
    Printf("Enter source node: ");
    scanf("%d", &s);
    int d[n];
    int v[n];
    for(int i=0; i<n; i++){
        d[i] = INT_MAX;
        v[i] = 0;
    }
    d[s] = 0;
    for(int count=0; count<n-1; count++){
        int u = -1;
        for(int i=0; i<n; i++){
            if(!v[i] && (u==-1 || d[i]<d[u])){
                u = i;
            }
        }
        if(d[u]==INT_MAX) break;
```

DATE :

```
        v[u] = 1;
        for(int i=0; i<n; i++){
            if(g[u][i] && !v[i] && d[u] != INT_MAX
                && d[u]+g[u][i] < d[i]){
            d[i] = d[u] + g[u][i];
            }
        }
    }
    printf("Distance from node %d:\n", s);
    for(int i=0; i<n; i++){
        if(d[i] == INT_MAX){
            printf("INF ");
        }else{
            printf("%d ", d[i]);
        }
    }
    printf("\n");
    return 0;
}
```

Output :-

Enter number of nodes : 5

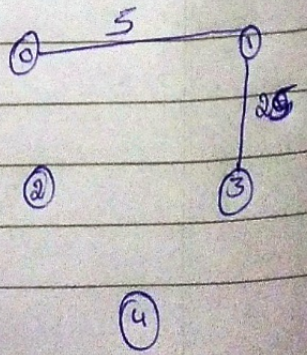
Enter adjacency matrix :

0	5	15	20	9999
5	0	25	30	9999
15	25	0	30	37
20	9999	80	0	35
9999	9999	37	35	0

Enter Source node : 1

Distance from node 1 :

5 0 20 25 57



PAGE NO :
DATE :

ALGORITHM: dijikstras (c[1..n, 1..n], src)

//input : An nxn cost matrix c[1..n, 1..n] with source node
//output : length dist[j] of shortest paths from src to j

```
for j ← 1 to n do
    dist[j] ← c[src, j]
end for
for j ← 1 to n do
    vis[j] ← 0
end for
dist[src] ← 0
vis[src] ← 1
count ← 1
while count ! n do
    min ← 9999
    for j ← 1 to n do
        if dist[j] < min and vis[j] != 1
            min ← dist[j]
            u ← j
        end if
    end for
    vis[u] ← 1
    count ← count + 1
    for j ← 1 to n do
        if min + c[u,j] < dist[j] and vis[j] != 1
            dist[j] ← min + c[u,j]
        end if
    end for
end while
write 'shortest distance is'
for j ← 1 to n do
    write src, j, dist[j]
end for.
```

PAGE NO :
DATE :

Kruskals Algorithm :-

```
#include <stdio.h>

#define MAX 30

typedef struct edge {
    int u, v, cost;
} Edge;

Edge edges[MAX];
int parents[MAX];

int find(int i) {
    while(parent[i])
    i = parent[i];
return i;
}

int uni(int i, int j) {
    if(i != j) {
        parent[j] = i;
        return 1;
    }
    return 0;
}

void kruskals(int c[MAX][MAX], int n) {
    int i, j, u, v, a, b, min, ne = 0, mincost = 0;

    for(i = 1; i <= n; i++)
        parent[i] = 0;
    while(ne < n-1) {
        min = 9999;
```

```
for(i=1; i<=n; i++) {
    for(j=1; j<=n; j++) {
        if(c[i][j] < min) {
            min = c[i][j];
            u=a=i;
            v=b=j;
        }
    }
}
u = find(u);
v = find(v);

if(uni(u,v)) {
    printf("(%d,%d) -> %d\n", a, b, min);
    mincost += min;
    ne++;
}
c[a][b] = c[b][a] = 9999;
}
printf("Minimum cost = %d\n", mincost);
}

int main() {
    int c[MAX][MAX], n, i, j;
    printf("Enter the number of vertices: ");
    scanf("%d", &n);

    printf("Enter the cost matrix:\n");
    for(i=1; i<=n; i++) {
        for(j=1; j<=n; j++) {
            scanf("%d", &c[i][j]);
            if(c[i][j] == 0)
                c[i][j] = 9999;
        }
    }
```

PAGE NO :
DATE :

```
printf("the minimum spaning Tree is:\n");
Kruskals(c, n);

return 0;

}
```

Output:

Enter the number of vertices : 5

Enter the cost matrix :

0	5	15	20	9999
5	0	25	30	9999
15	25	0	30	37
20	9999	30	0	35
9999	9999	37	35	0

The Minimum Spanning Tree is:

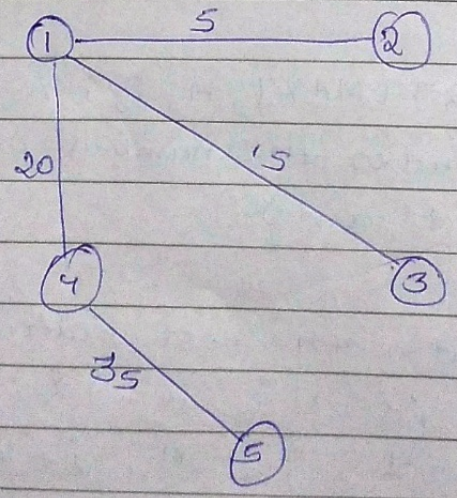
(1,2) → 5

(1,3) → 15

(1,4) → 20

(4,5) → 35

Minimum cost = 75



PAGE NO :
DATE :

Algorithm :- Kruskals(c[1..n, 1...n])
// Input : An nxn cost matrix c[1..n, 1..n]
// output : minimum cost of spaning tree of given undirected graph.

```
ne ← 0
mincost ← 0
for i ← 1 to n do
    parent[i] ← 0
end for
while ne != n-1 do
    min ← 9999
    for i ← 1 to n do
        for j ← 1 to n do
            if c[i,j] < min
                min ← c[i,j]
                u ← i
                a ← i
                v ← j
                b ← j
            end if
        end for
    end for
while parent[u] != 0 do
    u ← parent[u]
end while
if u != v
    write a, b, min
    parent[u] ← u
    ne ← ne+1
    mincost ← mincost + min
end if
c[a,b] ← 9999
c[a,b] ← 9999
end while
return write min(cost)
```