Practical no. 6

Represent a given graph using adjacency Matrix / list to perform dfs and using adjancency list to perform bfs. Use the map of the area around the college as graph identify the prominent Landmarks as nodes and perform dfs and BFS on that.

OR

You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connectdifferent pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem bysuggesting appropriate data structures

/*
#Quick Revision Notes:-

- **Prims Algorithm :- It is use to find MINIMUM COST SPANNING TREE**

Spanning Tree :- No of Vertex = V then Edges= V-1 (Graph should be Connected in Nature)
1. Start with an arbitrary vertex.
2. Add it to the minimum spanning tree (MST).
3. Find the minimum-weight edge connected to the MST.
4. Add the vertex connected by the edge to the MST.
5. Repeat steps 3 and 4 until all vertices are included in the MST.
 # Note: At each

```
#include <iostream>
#include <stdlib.h>
using namespace std;

int cost[10][10], i, j, k, n, qu[10], front, rear, v, visit[10],
visited[10];
int stk[10], top, visit1[10], visited1[10];

int main()
{
    int m;
    cout << "Enter number of vertices : ";
    cin >> n;
    cout << "Enter number of edges : ";
    cin >> m;

    cout << "\nEDGES :\n";
    for (k = 1; k <= m; k++)
    {
```

```
    cin >> i >> j;
    cost[i][j] = 1;
    cost[j][i] = 1;
}

//display function
cout << "The adjacency matrix of the graph is : " << endl;
for (i = 0; i < n; i++)
{
    for (j = 0; j < n; j++)
    {
        cout << " " << cost[i][j];
    }
    cout << endl;
}

cout << "Enter initial vertex : ";
cin >> v;
cout << "The BFS of the Graph is\n";
cout << v<<endl;
visited[v] = 1;
k = 1;
while (k < n)
{
    for (j = 1; j <= n; j++)
        if (cost[v][j] != 0 && visited[j] != 1 && visit[j] != 1)
        {
            visit[j] = 1;
            qu[rear++] = j;
        }
    v = qu[front++];
    cout << v << " ";
    k++;
    visit[v] = 0;
    visited[v] = 1;
}

cout <<endl<<"Enter initial vertex : ";
cin >> v;
cout << "The DFS of the Graph is\n";
cout << v<<endl;
visited[v] = 1;
k = 1;
while (k < n)
{
    for (j = n; j >= 1; j--)
        if (cost[v][j] != 0 && visited1[j] != 1 && visit1[j] != 1)
        {
            visit1[j] = 1;
            stk[top] = j;
            top++;
```

```
            }
        v = stk[--top];
        cout << v << " ";
        k++;
        visit1[v] = 0;
        visited1[v] = 1;
    }

    return 0;
}
```

Output:

Enter number of vertices : 2
Enter number of edges : 2

EDGES :
3
4
5
6
The adjacency matrix of the graph is :
0 0
0 0
Enter initial vertex : 1
The BFS of the Graph is
1
0
Enter initial vertex : 2
The DFS of the Graph is
2
0