

# ASSIGNMENT 3

## Aim:-

There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight takes to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Justify the storage representations used.

## Objective:-

To Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Justify the storage representations used.

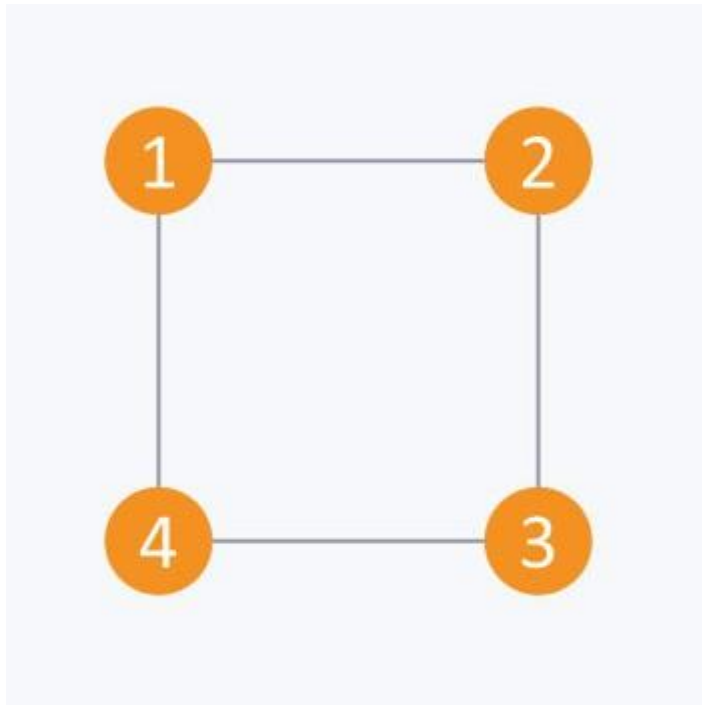
## Theory:-

The other way to represent a graph is by using an adjacency list. An adjacency list is an array A of separate lists. Each element of the array  $A_i$  is a list, which contains all the vertices that are adjacent to vertex i.

For a weighted graph, the weight or cost of the edge is stored along with the vertex in the list using pairs. In an undirected graph, if vertex j is in list  $A_i$  then vertex i will be in list  $A_j$ .

The space complexity of adjacency list is  $O(V + E)$  because in an adjacency list information is stored only for those edges that actually exist in the graph. In a lot of cases, where a matrix is sparse using an adjacency matrix may not be very useful. This is because using an adjacency matrix will take up a lot of space where most of the elements will be 0, anyway. In such cases, using an adjacency list is better.

**Note:** A sparse matrix is a matrix in which most of the elements are zero, whereas a dense matrix is a matrix in which most of the elements are non-zero.



## Algorithm:-

1. Create an array A of size N and type of array must be list of vertices. Initially each list is empty so each array element is initialise with empty list.
2. Iterate each given edge of the form (u,v) and append v to the uth list of array A. Also, If graph is undirected append u to the vth list of array A


## Code:-

```
#include<iostream>
#define MAX 10
using namespace std;
class airport
{
    string city[MAX];
    int distance[10][10];
public :
    int n;
    airport();
    void read_city();
    void show_graph();
};
airport::airport()
{
    n=0;
    for(int i=0;i<MAX;i++)
    {
        for(int j=0;j<MAX;j++)
```

## Skill Development Lab II 2018-19

```
                distance[i][j]=0;
            }
        }
    }
    void airport::read_city()
    {
        int k;
        cout<<"\nEnter the no. of cities: " ;
        cin>>n;
        cout<<"Enter city name:\n";
        for(int k=0;k<n;k++)
        {
            cout<<k+1<<" ] ";
            cin>>city[k];
        }
        for(int i=0;i<n;i++)
        {
            for(int j=i+1 ; j<n ; j++)
            {
                cout<<"\nEnter Distance between "<<city[i]<<" to
" <<city[j]<<": ";
                cin>>distance[i][j];
                distance[j][i]=distance[i][j];
            }
        }
    }
    void airport::show_graph()
    {
        cout<<"\t";
        for(int k=0;k<n;k++)
        {
            cout<<city[k]<<"\t";
        }
        cout<<endl;
        for(int i=0;i<n;i++)
        {
            cout<<city[i]<<"\t";
            for(int j=0;j<n;j++)
            {
                cout<<distance[i][j]<<"\t";
            }
            cout<<endl;
        }
    }
}
int main()
{
    airport obj;
    obj.read_city();
    obj.show_graph();
}
```

## Output Screenshot:-

 "C:\Users\Dell\Downloads\main (1).exe"

```
Enter the no. of cities: 4
Enter city name:
1] Pune
2] Mumbai
3] Chennai
4] Kolkata

Enter Distance between Pune to Mumbai: 1222

Enter Distance between Pune to Chennai: 1212

Enter Distance between Pune to Kolkata: 1311

Enter Distance between Mumbai to Chennai: 3222

Enter Distance between Mumbai to Kolkata: 4232

Enter Distance between Chennai to Kolkata: 211
    Pune    Mumbai    Chennai    Kolkata
Pune      0      1222      1212      1311
Mumbai  1222      0      3222      4232
Chennai 1212      3222      0      211
Kolkata 1311      4232      211      0

Process returned 0 (0x0)    execution time : 133.943 s
Press any key to continue.
```

## Conclusion:-

We Conclude That We Can Use Adjacency List To Show If Route Exists Between Any Particular Cities Or Not.