

## ASSIGNMENT 3

**Aim:** Represent a Graph using adjacency matrix

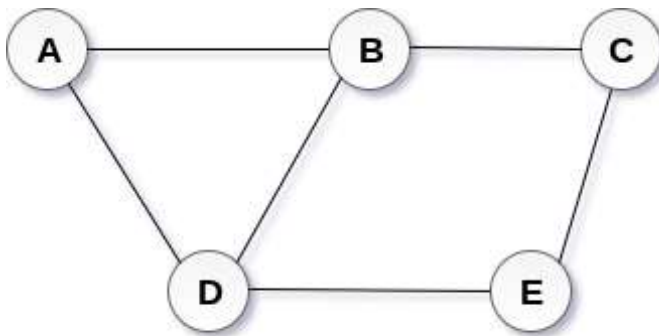
**Objective:** To understand advantages and disadvantages of using adjacency matrix to represent graphs

**Theory:**

Graph

A graph can be defined as group of vertices and edges that are used to connect these vertices. A graph can be seen as a cyclic tree, where the vertices (Nodes) maintain any complex relationship among them instead of having parent child relationship.

A graph  $G$  can be defined as an ordered set  $G(V, E)$  where  $V(G)$  represents the set of vertices and  $E(G)$  represents the set of edges which are used to connect these vertices.



Adjacency Matrix:

Adjacency Matrix is a 2D array of size  $V \times V$  where  $V$  is the number of vertices in a graph. Let the 2D array be  $adj[i][j]$ , a slot  $adj[i][j] = 1$  indicates that there is an edge from vertex  $i$  to vertex  $j$ . Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If  $adj[i][j] = w$ , then there is an edge from vertex  $i$  to vertex  $j$  with weight  $w$ .

The adjacency matrix for the above example graph is:

	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

## Algorithm:

- **Step 1:** Enter all vertices
- **Step 2:** If edge exists between v1 and v2

Adjmat[v1][v2]= cost

Else

Adjmat[v1][v2]= 0

- **Step 3:** EXIT

## Program:

```
#include<iostream>

using namespace std;

class Graph{

int ver=5 ;

string city[5];

int adjmat[5][5];

int val;

public:

Graph(){

cout<<"Enter names of the 5 cities"<<endl;

for(int i=0; i<ver; i++)

cin>>city[i];

}

void create()

{
```

```
for(int i=0; i<ver; i++)
{
for(int j=0; j<ver; j++)
{
if(i!=j && i<=j)
{
cout<<"If route exists between "<<city[i]<<" and "<<city[j]<<" enter Time, else enter
0"<<endl;

cin>>val;

adjmat[i][j]=val;
}
}
}
}

void display(){
for(int i=0; i<ver; i++)
{
for(int j=0; j<ver; j++)
{
if(i!=j && i<=j)
{
if(adjmat[i][j]!=0){
cout<<"Time to travel from "<<city[i]<<" to "<<city[j]<<" is"<<adjmat[i][j]<<endl;
}
}
}
}
```

```
}  
  
}  
  
};  
  
int main()  
{  
    Graph g;  
  
    char choice ;  
  
    int ch ;  
  
    do  
    {  
        cout << "\tMENU" << endl;  
        cout << "\t1. Create graph" << endl;  
        cout << "\t2. display" << endl;  
        cout << "\tenter your choice " << endl;  
  
        cin >> ch ;  
  
        switch ( ch )  
        {  
            case 1:  
                g.create ();  
  
                break;  
  
            case 2 :  
                g.display();  
  
                break;  
  
            default :  
  
                cout << "\tINVALID CHOICE " << endl;
```

```

}

cout << "\tDo you wish to continue (y/n)" << endl;

cin >> choice ;

}

while ( choice == 'y');

}

```

## Output:

```

"C:\Users\n\Desktop\sem2\sd\Assignment 3\assignment 3.exe"
Enter names of the 5 cities
pune
mumbai
surat
manali
panaji
1. Create graph
2. Display
Enter your choice
1
If route exists between pune and mumbai enter time in hours, else enter 0
3
If route exists between pune and surat enter time in hours, else enter 0
9
If route exists between pune and manali enter time in hours, else enter 0
0
If route exists between pune and panaji enter time in hours, else enter 0
6
If route exists between mumbai and surat enter time in hours, else enter 0
7
If route exists between mumbai and manali enter time in hours, else enter 0
0
If route exists between mumbai and panaji enter time in hours, else enter 0
5
If route exists between surat and manali enter time in hours, else enter 0
0
If route exists between surat and panaji enter time in hours, else enter 0
0
If route exists between manali and panaji enter time in hours, else enter 0
0
Do you wish to continue (y/n)
y
1. Create graph
2. Display
Enter your choice
2
Time to travel from pune to mumbai is 3 hours
Time to travel from pune to surat is 9 hours
Time to travel from pune to panaji is 6 hours
Time to travel from mumbai to surat is 7 hours
Time to travel from mumbai to panaji is 5 hours
Do you wish to continue (y/n)

```

## Conclusion:

Graphs are nonlinear data structures that make operations on large data easier.

*Pros:* Representation is easier to implement and follow. Removing an edge takes  $O(1)$  time. Queries like whether there is an edge from vertex 'u' to vertex 'v' are efficient and can be done  $O(1)$ .

*Cons:* Consumes more space  $O(V^2)$ . Even if the graph is sparse(contains less number of edges), it consumes the same space. Adding a vertex is  $O(V^2)$  time.