Ex:10.A **GRAPH REPRESENTATIONS**

AIM:

To write a C program implement adjacent matrix and adjacency list .

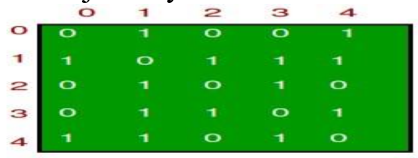
DESCRIPTION:

Graph is a data structure that sconsists of following two components:

1. A finite set of vertices also called as nodes.

2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of a directed graph(di-graph). The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

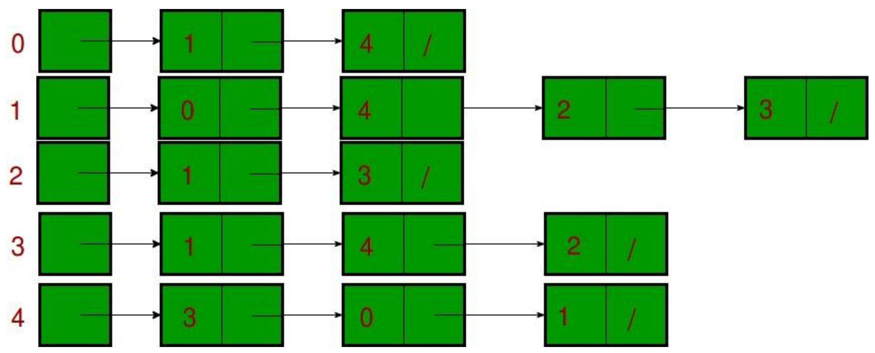
Following two are the most commonly used representations of a graph.
1. Adjacency Matrix
2. Adjacency List
There are other representations also like, Incidence Matrix and Incidence List. The choice of the graph representation is situation specific. It totally depends on the type of operations to be performed and ease of use.
Adjacency Matrix:
Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], 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 weigh t 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

Adjacency List:

An array of linked lists is used. Size of the array is equal to the number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists. Following is adjacency list representation of the above graph..



ALGORITHM:

1. Create a graph with getting no. of vertices and no. of edges

2. Implement adjacency matrix

3. Implement adjacency list

4. Close the program

PROGRAM

```
#include

<stdio.h>

#include

<stdlib.h>

void main() {

int option;

do
```

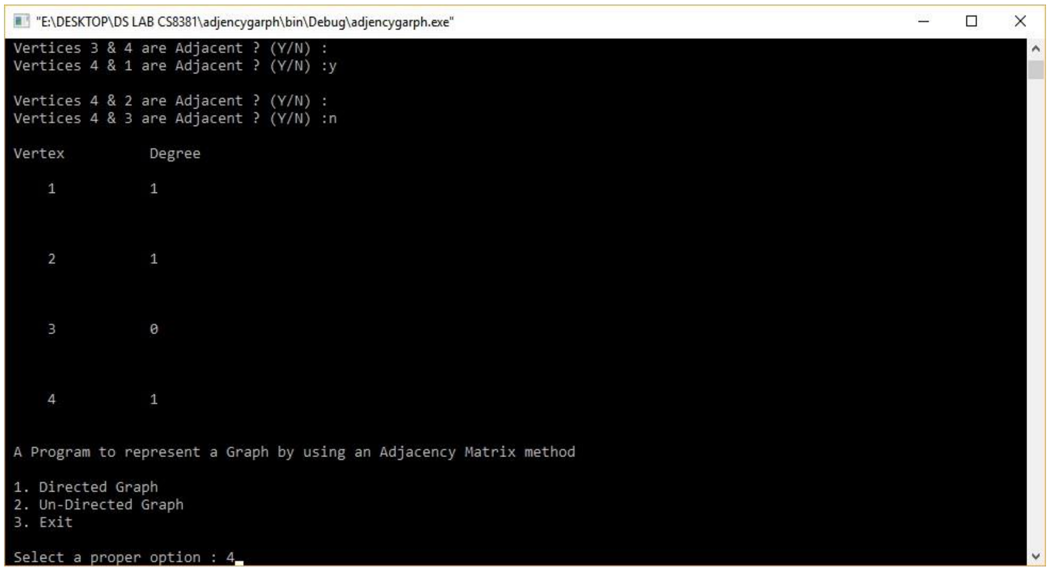
```
   {printf("\n A Program to represent a Graph by using

an ");      printf("Adjacency Matrix method \n ");

      printf("\n 1. Directed Graph ");    printf("\n 2.

Un-Directed Graph ");   printf("\n 3. Exit ");

      printf("\n\n Select a proper option : ");

      scanf("%d", &option);

switch(option)

                {
              case 1 :

dir_graph();

break;             case 2 :

undir_graph();

break;             case 3 :

exit(0);

                } // switch

    }while(1);

} int

dir_graph(

)

{     int adj_mat[50][50];     int n;     int in_deg, out_deg,

i, j;     printf("\n How Many Vertices ? : ");     scanf("%d",

&n);     read_graph(adj_mat, n);     printf("\n Vertex \t

In_Degree \t Out_Degree \t Total_Degree ");     for (i = 1; i

<= n ; i++ )
```

```
    {         in_deg =
out_deg = 0;      for ( j = 1
; j <= n ; j++ )
                 {
            if ( adj_mat[j][i]
== 1 )
in_deg++;}         for ( j = 1 ;
j <= n ; j++ )             if
(adj_mat[i][j] == 1 )
out_deg++;
            printf("\n\n
%5d\t\t\t%d\t\t%d\t\t%d\n\n",i,in_deg,out_deg,in_deg+out_deg);


}return;}
int
undir_graph(
)
{     int adj_mat[50][50];     int
deg, i, j, n;     printf("\n How
Many Vertices ? : ");
scanf("%d", &n);
read_graph(adj_mat, n);
```

```
    printf("\n Vertex \t Degree
");     for ( i = 1 ; i <= n ;
i++ )
    {         deg = 0;         for ( j = 1
; j <= n ; j++ )             if (
adj_mat[i][j] == 1)                 deg++;
printf("\n\n %5d \t\t %d\n\n", i, deg);
    }     return;} int read_graph ( int
adj_mat[50][50], int n )
{     int i, j;     char
reply;     for ( i = 1 ; i
<= n ; i++ )      {
for ( j = 1 ; j <= n ; j++
)
        {
if ( i == j )
            {
adj_mat[i][j] = 0;
                       continue;
            }             printf("\n Vertices %d & %d are
Adjacent ? (Y/N) :",i,j);             scanf("%c", &reply);
if ( reply == 'y' || reply == 'Y' )
adj_mat[i][j] = 1;             else
adj_mat[i][j] = 0;
```

```
                }}

    return;}
```

OUTPUT



"E:\DESKTOP\DS LAB CS8381\adjencygarph\bin\Debug\adjencygarph.exe"

```
Vertices 3 & 4 are Adjacent ? (Y/N) :
Vertices 4 & 1 are Adjacent ? (Y/N) :y

Vertices 4 & 2 are Adjacent ? (Y/N) :
Vertices 4 & 3 are Adjacent ? (Y/N) :n

Vertex          Degree

    1           1



    2           1



    3           0



    4           1


A Program to represent a Graph by using an Adjacency Matrix method

1. Directed Graph
2. Un-Directed Graph
3. Exit

Select a proper option : 4
```

RESULT:

Thus the C program implemented adjacent matrix and adjacency list