DATA STRUCTURES LABORATORY

Course Code: 1BCSL306

VTU 2025 Scheme – Semester 3

PART – B

Open-Ended Experiment

EXPERIMENT – 12

City Connectivity Analysis

Using Graph and DFS

Simple Code • Easy to Remember • VTU Exam Friendly

SEARCH CREATORS

Your VTU Study Partner

Experiment 12

1. Aim

Aim

To develop a menu-driven C program using Graph data structure to represent cities and roads, display the graph, find reachable cities and check connectivity between two cities.

2. VTU Prescribed Experiment

VTU Prescribed Experiment

A transportation department wants to analyse connectivity between cities.
Each city has a City Name and cities are interconnected (not necessarily all) with roads.
Develop a menu-driven C program with the following operations:

a. Create a graph representing N cities with interconnecting roads.

b. Display the graph representation.

c. Print all cities reachable by road from a given city.

d. Check whether any two given cities are connected via road.

3. Suitable Data Structure

The suitable data structure is:

$$\boxed{\text{Graph}}$$

Here:

$$\boxed{Vertex = City}$$

and

$$\boxed{Edge = Road}$$

For easy implementation, we use:

$$\boxed{\text{Adjacency Matrix}}$$

4. Simple Theory

A graph consists of:

$$\boxed{Vertices + Edges}$$

In this problem:

$$\boxed{Cities = Vertices}$$

$$\boxed{Roads = Edges}$$

Example:

$$A \leftrightarrow B$$

means city A and city B are connected by road.

5. Adjacency Matrix

For three cities:

$$A,\ B,\ C$$

if roads are:

$$A - B, \quad B - C$$

then adjacency matrix is:

	A	B	C
A	0	1	0
B	1	0	1
C	0	1	0

Here:

$$\boxed{1 = Road\ Exists}$$

$$\boxed{0 = No\ Road}$$

6. DFS Traversal

To find all reachable cities, we use:

Depth First Search (DFS)

DFS:

1. Visit current city.
2. Mark it as visited.
3. Visit an unvisited connected city.
4. Continue recursively.

7. Algorithm

7.1. Create Graph

1. Read number of cities.
2. Read city names.
3. Initialize adjacency matrix to 0.
4. Read number of roads.
5. Read two city numbers for each road.
6. Set:

$$graph[u][v] = 1$$

and:

$$graph[v][u] = 1$$

7.2. Display Graph

1. Display city names.
2. Display each row of adjacency matrix.
3. Value 1 represents a road.
4. Value 0 represents no road.

7.3. Reachable Cities

1. Read starting city.
2. Set all visited values to 0.
3. Apply DFS from starting city.
4. Display every visited city.

7.4. Check Connectivity

1. Read source city.
2. Read destination city.
3. Perform DFS from source.
4. If destination becomes visited:

$$\boxed{Connected}$$

otherwise:

$$\boxed{Not\ Connected}$$

Easy Exam Memory

Remember:

$$\boxed{City = Vertex}$$

$$\boxed{Road = Edge}$$

$$\boxed{Graph = Adjacency\ Matrix}$$

$$\boxed{Reachability = DFS}$$

$$\boxed{visited[destination] = 1 \Rightarrow Connected}$$

8. C Program – Easy Student Version

```c
#include<stdio.h>
#include<string.h>

#define MAX 10

int graph[MAX][MAX];
int visited[MAX];
char city[MAX][20];
int n;

void dfs(int v)
{
    int i;

    visited[v]=1;
    printf("%s ",city[v]);

    for(i=0;i<n;i++)
    {
        if(graph[v][i]==1 && visited[i]==0)
            dfs(i);
    }
}

void createGraph()
{
    int i,j,e,u,v;

    printf("Enter number of cities: ");
    scanf("%d",&n);

    printf("Enter City Names:\n");

    for(i=0;i<n;i++)
        scanf("%s",city[i]);

    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            graph[i][j]=0;

    printf("Enter number of roads: ");
    scanf("%d",&e);

    printf("Enter road connections using city numbers\n");

    for(i=0;i<e;i++)
    {
        scanf("%d%d",&u,&v);

        graph[u][v]=1;
```

```
        graph[v][u]=1;
    }

    printf("Graph Created\n");
}

void displayGraph()
{
    int i,j;

    printf("\nAdjacency Matrix:\n\n");

    printf("   ");

    for(i=0;i<n;i++)
        printf("%s ",city[i]);

    printf("\n");

    for(i=0;i<n;i++)
    {
        printf("%s ",city[i]);

        for(j=0;j<n;j++)
            printf("%d ",graph[i][j]);

        printf("\n");
    }
}

void reachable()
{
    int i,start;

    printf("Enter Starting City Number: ");
    scanf("%d",&start);

    for(i=0;i<n;i++)
        visited[i]=0;

    printf("Reachable Cities: ");
    dfs(start);

    printf("\n");
}

void connected()
{
    int i,source,destination;

    printf("Enter Source City Number: ");
    scanf("%d",&source);

```

```
    printf("Enter Destination City Number: ");
    scanf("%d",&destination);

    for(i=0;i<n;i++)
        visited[i]=0;

    dfs(source);

    printf("\n");

    if(visited[destination])
        printf("Cities are Connected\n");
    else
        printf("Cities are Not Connected\n");
}

int main()
{
    int ch;

    do
    {
        printf("\n1.Create Graph");
        printf("\n2.Display Graph");
        printf("\n3.Reachable Cities");
        printf("\n4.Check Connectivity");
        printf("\n5.Exit");

        printf("\nEnter Choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                createGraph();
                break;

            case 2:
                displayGraph();
                break;

            case 3:
                reachable();
                break;

            case 4:
                connected();
                break;

            case 5:
                printf("Program Ended\n");
                break;

```

```
            default:
                printf("Invalid Choice\n");
        }

    }while(ch!=5);

    return 0;
}
```

9. Simple Program Explanation

9.1. graph[][]

The array:

```
int graph[MAX][MAX];
```

stores the adjacency matrix.

If:

```
graph[i][j]=1;
```

there is a road.

If:

```
graph[i][j]=0;
```

there is no road.

9.2. city[][]

City names are stored using:

```
char city[MAX][20];
```

Example:

$$\boxed{0 = Tumkur, \quad 1 = Bengaluru, \quad 2 = Mysuru}$$

9.3. createGraph()

For an undirected road:

```
graph[u][v]=1;
graph[v][u]=1;
```

Both are required because:

$$\boxed{A \leftrightarrow B}$$

9.4. dfs()

First:

```
visited[v]=1;
```

marks the current city as visited.

Then:

```
if(graph[v][i]==1 && visited[i]==0)
    dfs(i);
```

visits connected unvisited cities.

9.5. reachable()

It resets:

```
visited[i]=0;
```

and then calls:

```
dfs(start);
```

Hence all reachable cities are displayed.

9.6. connected()

DFS is started from source.

Finally:

```
if(visited[destination])
```

then:

Cities are Connected

Otherwise:

Cities are Not Connected

Easy Exam Memory

Main functions:

$$\boxed{createGraph()}$$

$$\boxed{displayGraph()}$$

$$\boxed{dfs()}$$

$$\boxed{reachable()}$$

$$\boxed{connected()}$$

Remember:

$$\boxed{Create \rightarrow Display \rightarrow DFS \rightarrow Reachable \rightarrow Connected}$$

10. Sample Input and Output

Consider four cities:

$$0 = Tumkur$$

$$1 = Bengaluru$$

$$2 = Mysuru$$

$$3 = Hassan$$

Roads:

$$0 - 1$$

$$1 - 2$$

$$0 - 3$$

Sample execution:

```

1.Create Graph
2.Display Graph
3.Reachable Cities
4.Check Connectivity
5.Exit

Enter Choice: 1

Enter number of cities: 4

Enter City Names:
Tumkur
Bengaluru
Mysuru
Hassan

Enter number of roads: 3

Enter road connections using city numbers
0 1
1 2
0 3

Graph Created
```

11. Graph Representation

```
Enter Choice: 2

Adjacency Matrix:

           Tumkur Bengaluru Mysuru Hassan
Tumkur        0       1        0      1
Bengaluru     1       0        1      0
Mysuru        0       1        0      0
Hassan        1       0        0      0
```

12. Reachable Cities Example

Input:

```
Enter Choice: 3

Enter Starting City Number: 0
```

Output:

```
Reachable Cities:
Tumkur Bengaluru Mysuru Hassan
```

13. Connectivity Example

Input:

```
Enter Choice: 4

Enter Source City Number: 0

Enter Destination City Number: 2
```

Output:

```
Tumkur Bengaluru Mysuru Hassan

Cities are Connected
```

14. Relevant Test Cases

Test	Operation	Expected Result
1	Create 4 cities	Graph is created successfully.
2	Display graph	Adjacency matrix is displayed.
3	Reachable from city 0	All connected cities are displayed.
4	Check connected cities	Displays “Cities are Connected”.
5	Check disconnected cities	Displays “Cities are Not Connected”.

15. Performance

Using an adjacency matrix:

$$\boxed{Space = O(n^2)}$$

DFS traversal:

$$\boxed{O(n^2)}$$

for adjacency matrix representation.

Connectivity is checked using DFS.

16. Result

Result

Thus, a menu-driven C program using Graph data structure was successfully implemented to analyse city connectivity.
The program performs:

- Creation of graph for N cities and roads.
- Display of graph representation.
- Display of all cities reachable from a given city.
- Connectivity checking between two cities.

17. Important Viva Questions

Important Viva Questions

1. What is a graph?
A graph is a collection of vertices and edges.

2. What represents a vertex in this experiment?
A city.

3. What represents an edge?
A road between two cities.

4. Which graph representation is used?
Adjacency Matrix.

5. What does 1 indicate in an adjacency matrix?
It indicates that an edge or road exists.

6. What does 0 indicate?
There is no direct road.

7. Which traversal is used in this program?
Depth First Search.

8. What is DFS?
DFS visits a vertex and continues deeply through connected unvisited vertices.

9. Why is visited[] required?
It prevents a city from being visited repeatedly.

10. What does visited[i]=1 mean?
The city has already been visited.

11. Why are both graph[u][v] and graph[v][u] set to 1?
Because roads are treated as undirected connections.

12. How are reachable cities obtained?
By performing DFS from the starting city.

13. How do we check connectivity?
Perform DFS from source and check whether destination is visited.

14. What is an adjacency matrix?
It is a two-dimensional array used to represent edges between vertices.

15. What is the space complexity of adjacency matrix?

$$\boxed{O(n^2)}$$

18. 1-Minute Exam Revision

Easy Exam Memory

Graph

$$\boxed{Cities + Roads}$$

Vertex

$$\boxed{City}$$

Edge

$$\boxed{Road}$$

Representation

$$\boxed{Adjacency\ Matrix}$$

Reachability

$$\boxed{DFS}$$

Connected

$$\boxed{visited[destination] == 1}$$

19. Easy Code Memory Map

Remember

Remember code in this order:

$$\boxed{graph[][]}$$

$$\downarrow$$

$$\boxed{visited[]}$$

$$\downarrow$$

$$\boxed{dfs()}$$

$$\downarrow$$

$$\boxed{createGraph()}$$

$$\downarrow$$

$$\boxed{displayGraph()}$$

$$\downarrow$$

$$\boxed{reachable()}$$

$$\downarrow$$

$$\boxed{connected()}$$

Most important condition:

$$\boxed{graph[v][i] == 1 \ \&\&visited[i] == 0}$$

SEARCH CREATORS

1BCSL306 – Data Structures Laboratory

PART – B

EXPERIMENT 12

COMPLETED

City Connectivity Analysis

Graph Using Adjacency Matrix and DFS

Create Graph → Display → DFS → Reachability → Connectivity

Your VTU Study Partner