Terna Engineering College
Computer Engineering Department

Program: Sem VII

Course: Artificial Intelligence & Soft Computing (AI&SC)

Experiment No. 03

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per the following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Blackboard access available)

Roll No. 50	Name: AMEY THAKUR
Class: BE-COMPS-50	Batch: B3
Date of Experiment: 10-08-2021	Date of Submission: 10-08-2021
Grade :	

Aim: To Implement uninformed search methods using C or Java.

B.1 Document created by the student:

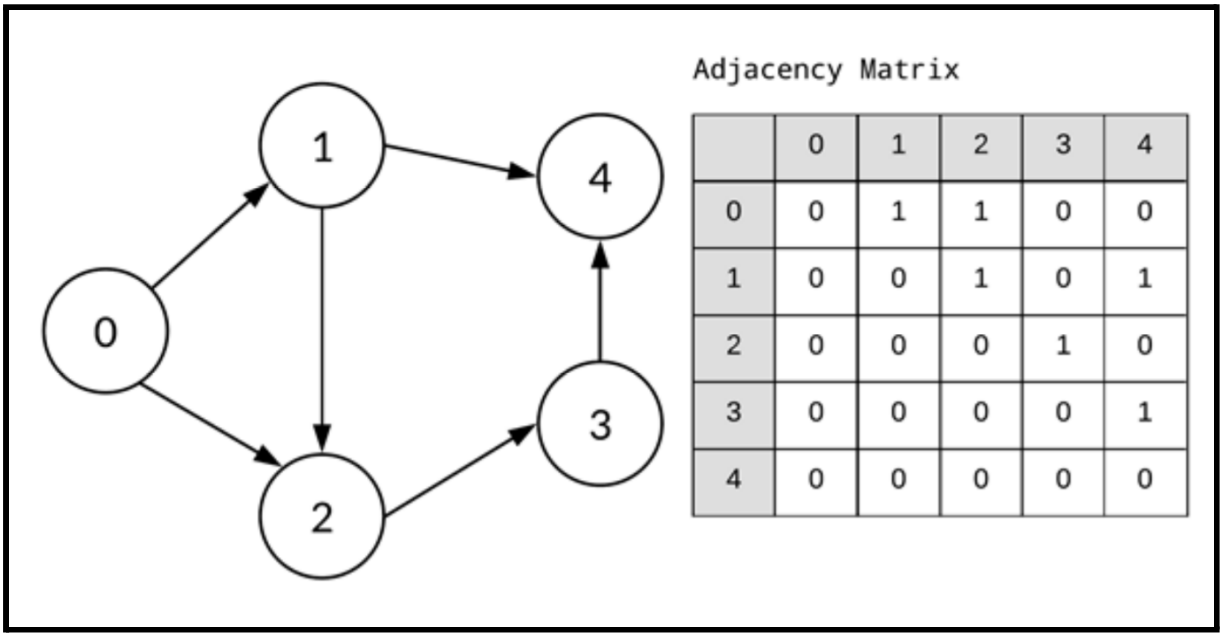
(Write the answers to the questions given in section 4 during the 2 hours of practice in the lab here)

1. **BFS.C**

```
#include<stdio.h>

int a[20][20], q[20], visited[20], n, i, j, f = 0, r = -1;

void bfs(int v);


int main() {

 int v;

 printf("\n Enter the number of vertices: ");

 scanf("%d", &n);


 printf("\n Enter graph data in matrix form: \n");
```

```
for(i=0; i<n; i++)
{
  for(j=0;j<n;j++)
  {
   scanf("%d", &a[i][j]);
  }
}

printf("\n Enter the starting vertex: ");
scanf("%d", &v);
for(i=0; i<n; i++)
{
   q[i] = 0;
   visited[i] = 0;
}
bfs(v);
printf("\n The node which are reachable are: \n");

for(i=0; i<n; i++)
{
   if(visited[i])
   {
      printf("%d\t", i);
   }
}
 return 0;
}
```

```
void bfs(int v)
{
 for(i=0; i<n; i++)
 {
   if(a[v][i] && !visited[i])
   {
      q[++r] = i;
   }
 }
   if(f <= r)
   {
      visited[q[f]] = 1;
      bfs(q[f++]);
   }
}
```



Adjacency Matrix

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

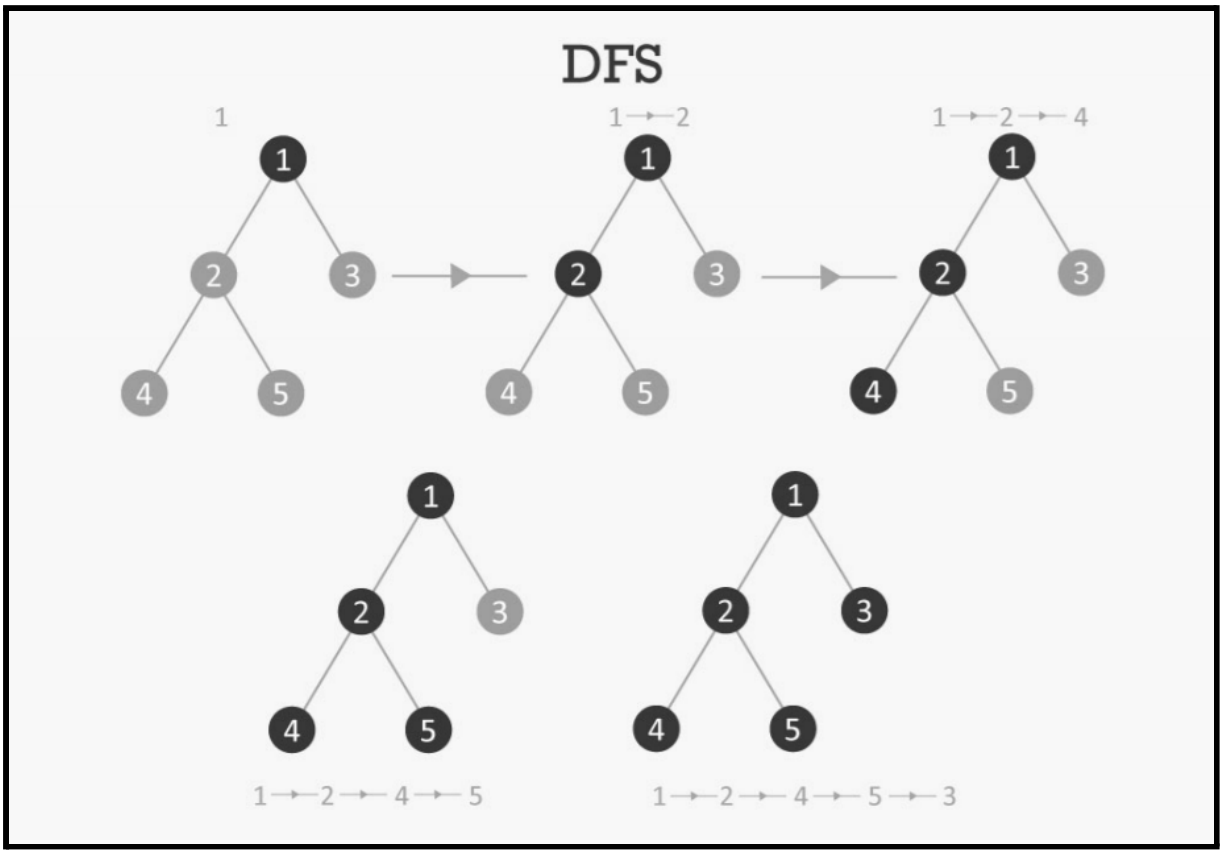
```
C:\TURBOC3\BIN>TC

 Enter the number of vertices: 5

 Enter graph data in matrix form:
0 1 1 0 0
0 0 1 0 1
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0

 Enter the starting vertex: 0

 The node which are reachable are:
1       2       3       4
```

2. **DFS.C**

```
#include <stdio.h>
#include <stdlib.h>
int source,V,E,time,visited[20],G[20][20];

void DFS(int i)
{
        int j;
        visited[i]=1;
        printf(" %d->",i+1);
        for(j=0;j<V;j++)
        {
```

```
            if(G[i][j]==1&&visited[j]==0)
            DFS(j);
        }
}

int main()
{
        int i,j,v1,v2;
        printf("\tGraphs\n");
        printf("Enter the no of edges: ");
        scanf("%d",&E);
        printf("Enter the no of vertices: ");
        scanf("%d",&V);
        for(i=0;i<V;i++)
        {
            for(j=0;j<V;j++)
            G[i][j]=0;
        }
        for(i=0;i<E;i++)
{
        printf("Enter the edges (format: V1 V2) : ");
        scanf("%d%d",&v1,&v2);
        G[v1-1][v2-1]=1;
}
for(i=0;i<V;i++)
{
        for(j=0;j<V;j++)
```

```
        printf(" %d ",G[i][j]);
        printf("\n");
}
        printf("Enter the source: ");
        scanf("%d",&source);
        DFS(source-1);
return 0;
}
```



```
C:\TURBOC3\BIN>TC
        Graphs
Enter the no of edges: 4
Enter the no of vertices: 5
Enter the edges (format: V1 V2): 1 2
Enter the edges (format: V1 V2): 1 3
Enter the edges (format: V1 V2): 2 4
Enter the edges (format: V1 V2): 2 5
 0  1  1  0  0
 0  0  0  1  1
 0  0  0  0  0
 0  0  0  0  0
 0  0  0  0  0
Enter the source: 1
 1-> 2-> 4-> 5-> 3->
```

B.2 Observations and learning:

(*Students are expected to understand the selected topic Prepare a flow of the steps defined in the paper)*

The depth-first search (DFS) algorithm begins with the first node of graph G and proceeds deeper until we reach the goal node or the node with no child nodes. The algorithm then reverses its path from the dead end to the most recent node that has yet to be fully examined.

Stack is the data structure that is employed in DFS. The procedure is similar to that of the BFS algorithm. Edges that go to an unvisited node are referred to as discovery edges in DFS, whereas edges that link to an already visited node are referred to as block edges.

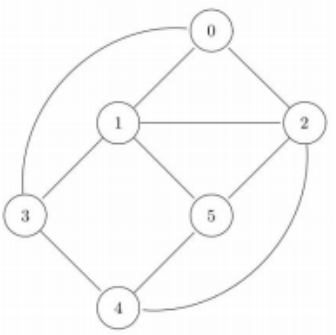
B.3 Conclusion:

(Students must write the conclusion as per the attainment of an individual)

Hence we've successfully implemented a program for an uninformed search method, the DFS algorithm.

B.4 Question of Curiosity:

Q1) Which sequence corresponds to that depth first search for the graph given below. The search starts at vertex 0 and lexicographic ordering is assumed for the edges emanating from each vertex.



A. 0 1 2 4 3 5
B. 0 1 2 5 4 3
C. 0 1 2 3 4 5
D. 0 1 3 4 2 5

ANS: A. 0 1 2 4 3 5

Explanation: A correct sequence of DFS traversal is 0 1 2 4 3 5 There is no edge between 2 and 3 and also 5 which is connected to 2 is unvisited.

Q2) Given a rooted tree, one desires to find the shortest path from the root to a given node v. Which algorithm would one use to find this shortest path?
A.DFS
B. BFS
C. Either BFS or DFS

ANS: C. Either BFS or DFS

Explanation: A tree has a unique path between any two pairs of nodes. Any traversal strategy would give us the path (which is the shortest path)

Q3) Consider a graph G. Let T be a BFS tree with root r. Let d(u,v) denote the length of the shortest path between the nodes u and v. If v is visited before u in the breadth first search traversal, which of the following statements is true ?
A. d(r,v) > d(r,u)
B. d(r,v) = d(r,u)
C. d(r,v) < d(r,u)
D. insufficient information to comment on d(r,v) and d(r,u)

ANS: D. Insufficient information to comment on d(r,v) and d(r,u)

Explanation: u being traversed after v does not tell us if they are on the same level or not. The correct relation would be d(r,v) <= d(r,u)