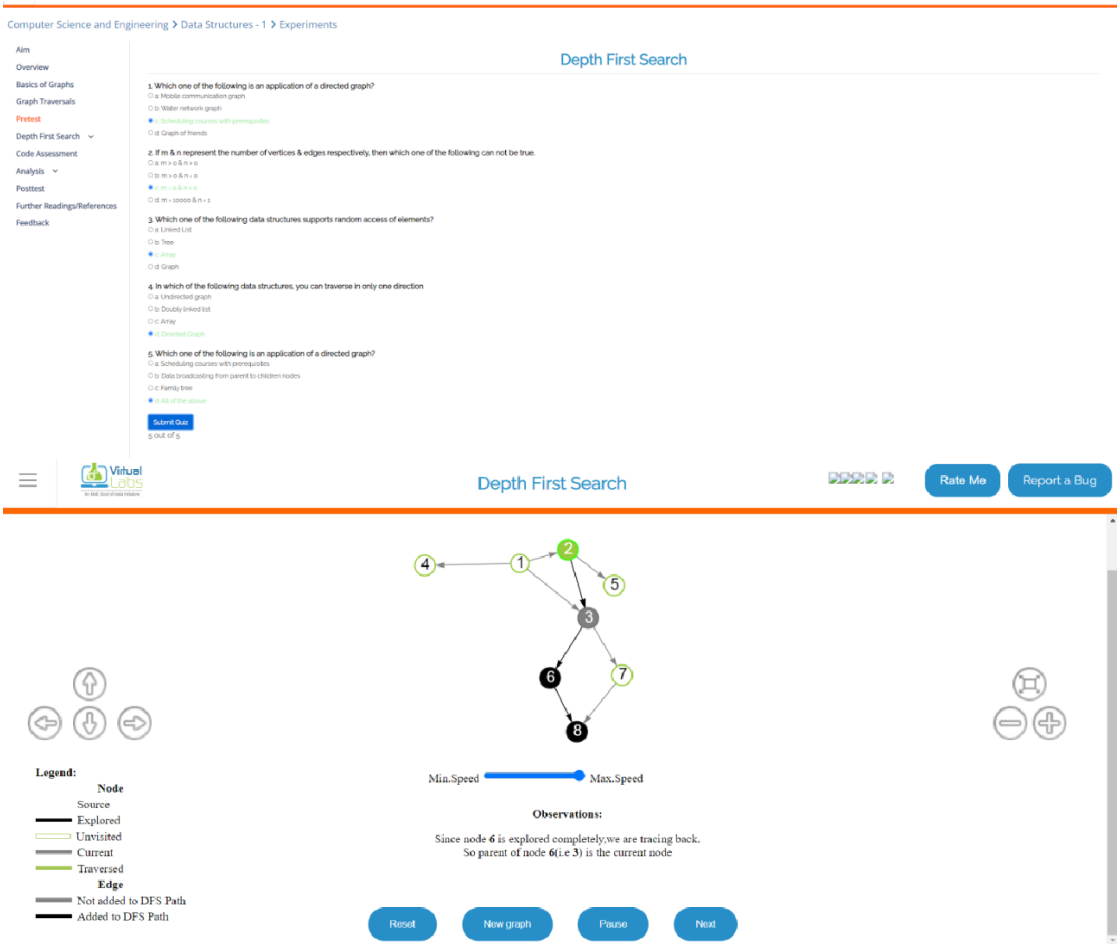
Batch: A1 **Roll No.: 16010123012**

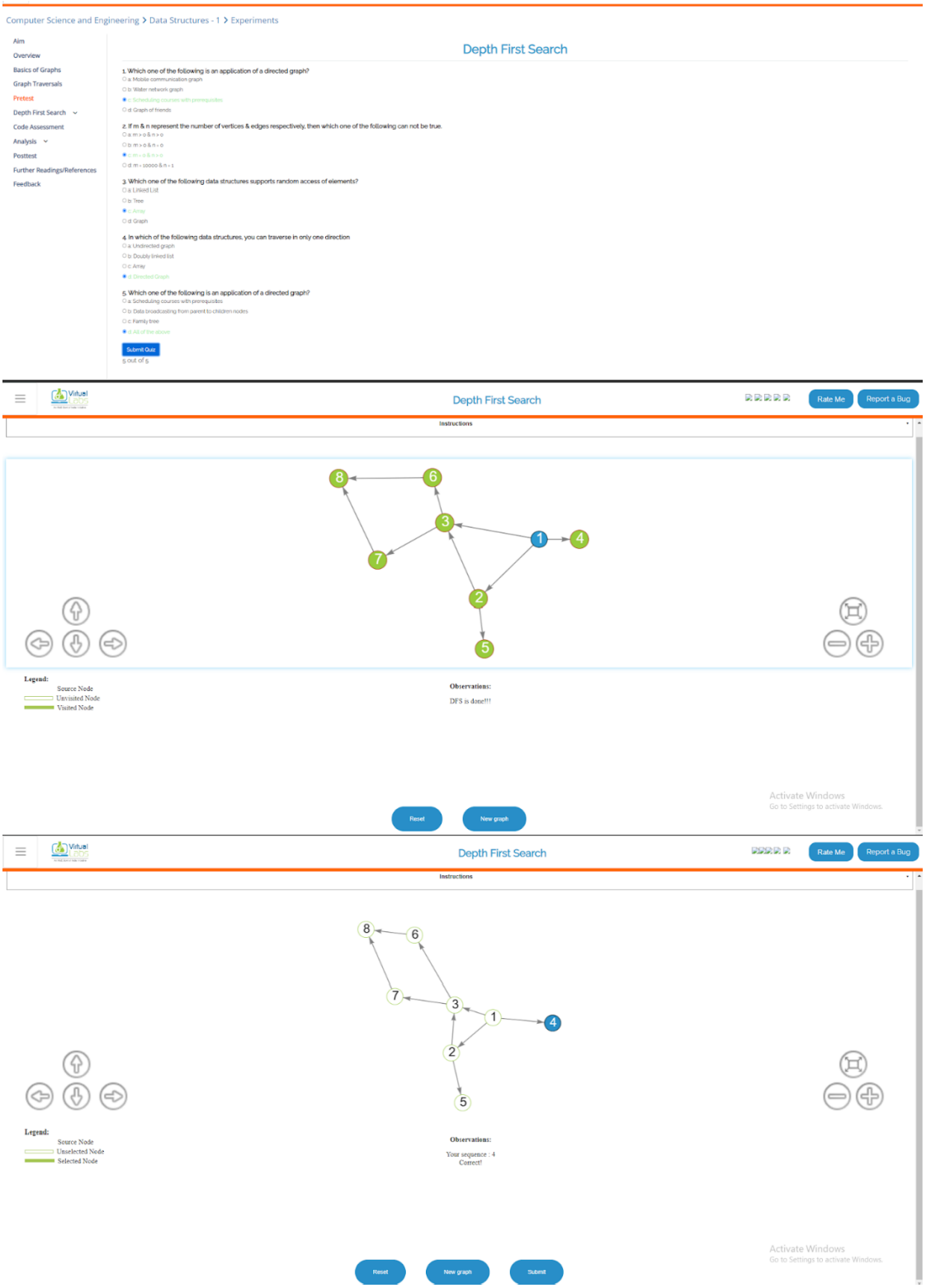
Experiment / assignment / tutorial No. 8

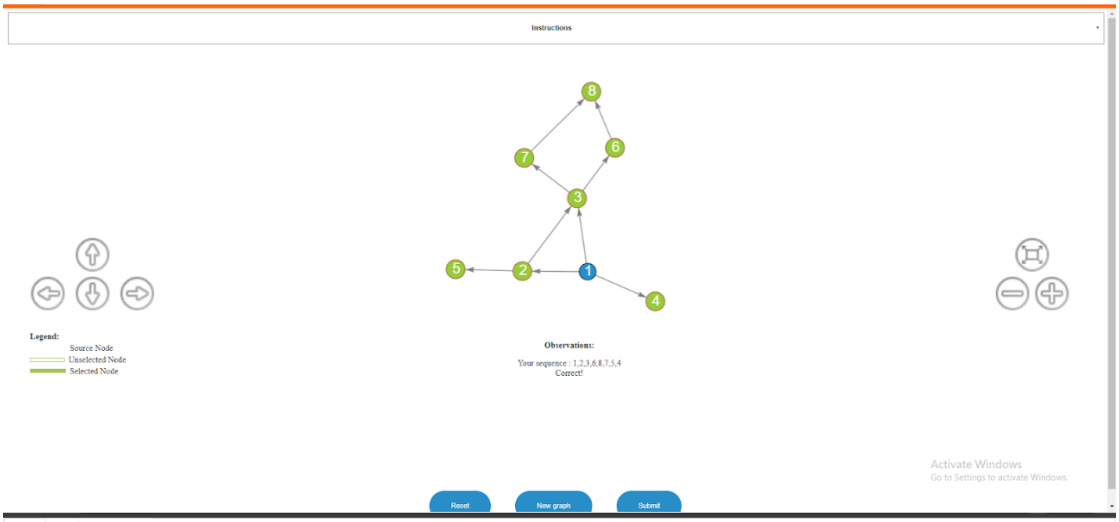
Grade: AA / AB / BB / BC / CC / CD /DD

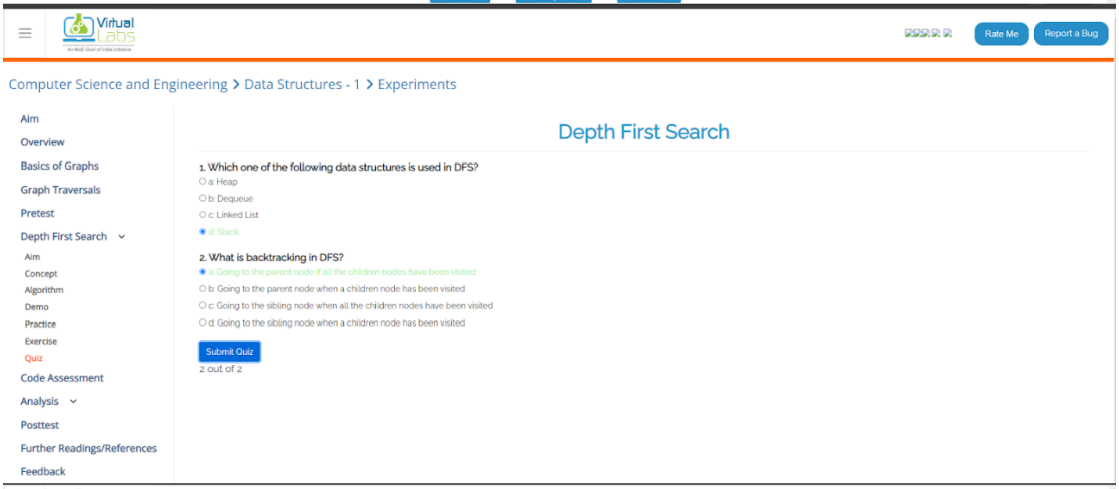
Signature of the Staff In-charge with date

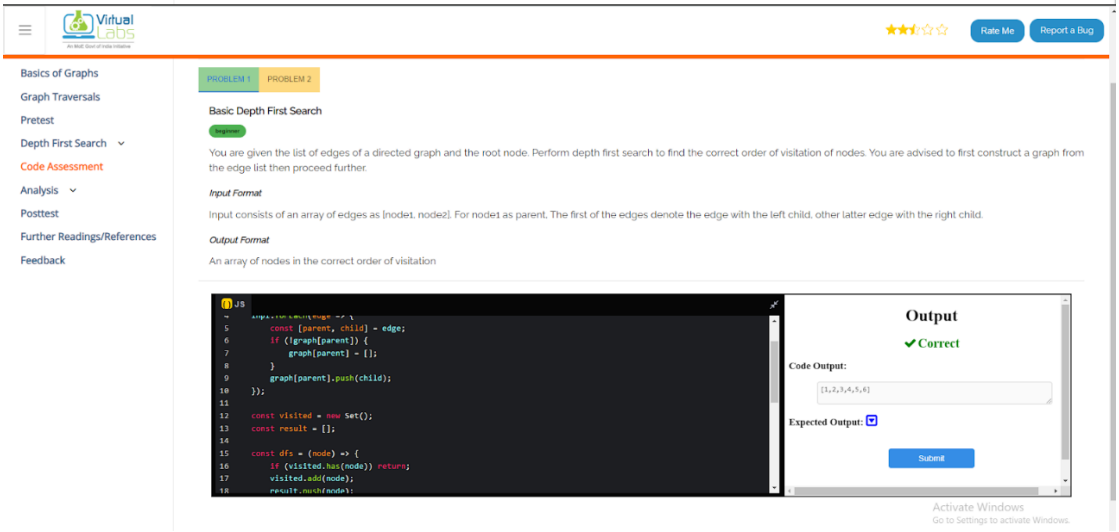
Title: Study of Graph traversal methods

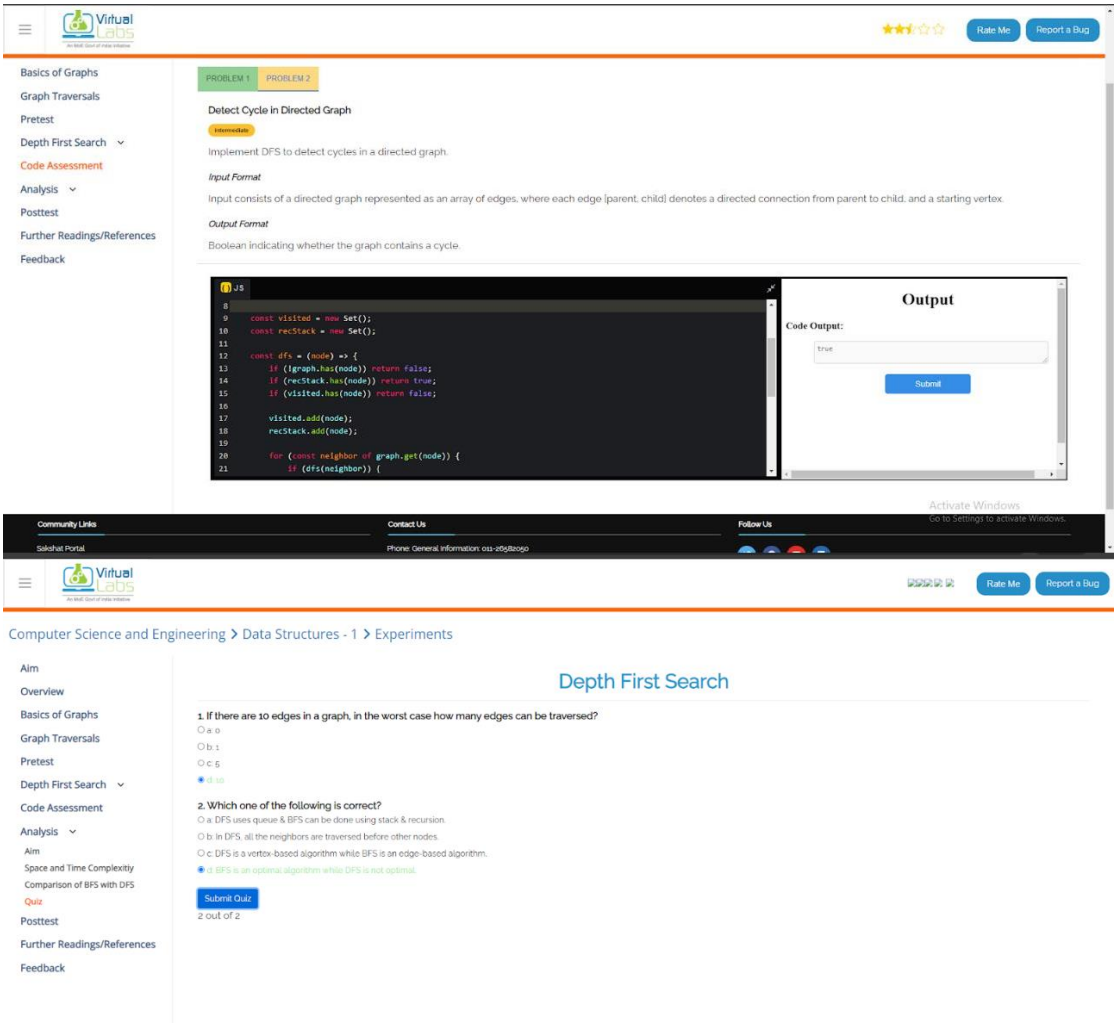



Computer Science and Engineering > Data Structures - 1 > Experiments
Depth First Search
Aim
Overview
Basics of Graphs
Graph Traversals
Pretest
Depth First Search
Code Assessment
Analysis
Posttest
Further Readings/References
Feedback
1. Which one of the following is an application of a directed graph?
a. Mobile communication graph
b. Water network graph
c. Scheduling courses with prerequisites
d. Graph of friends
2. If m & n represent the number of vertices & edges respectively, then which one of the following can not be true.
a. m > 0 & n > 0
b. m > 0 & n = 0
c. m = 0 & n = 0
d. m = 10000 & n = 1
3. Which one of the following data structures supports random access of elements?
a. Linked List
b. Tree
c. Array
d. Graph
4. In which of the following data structures, you can traverse in only one direction
a. Undirected graph
b. Doubly linked list
c. Array
d. Directed Graph
5. Which one of the following is an application of a directed graph?
a. Scheduling courses with prerequisites
b. Data broadcasting from parent to children nodes
c. Family tree
d. All of the above
Submit Quiz
5 out of 5
Depth First Search
Rate Me
Report a Bug
Instructions
8
6
3
1
4
7
2
5
Legend:
Source Node
Unvisited Node
Visited Node
Observations:
DFS is done!!!
Reset
New graph
Activate Windows
Go to Settings to activate Windows.
Depth First Search
Rate Me
Report a Bug
Instructions
8
6
7
3
1
4
2
5
Legend:
Source Node
Unselected Node
Selected Node
Observations:
Your sequence : 4
Correct!
Reset
New graph
Submit
Activate Windows
Go to Settings to activate Windows.

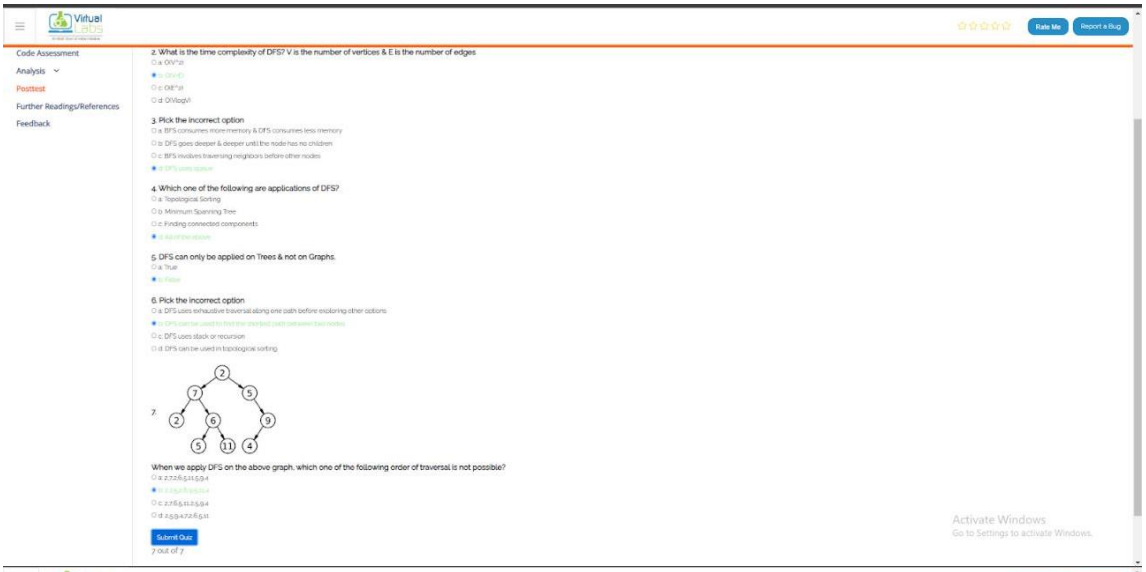

Instructions
Legend:
Source Node
Unselected Node
Selected Node
Observations:
Your sequence : 1,2,3,6,8,7,5,4
Correct!
Reset
New graph
Submit

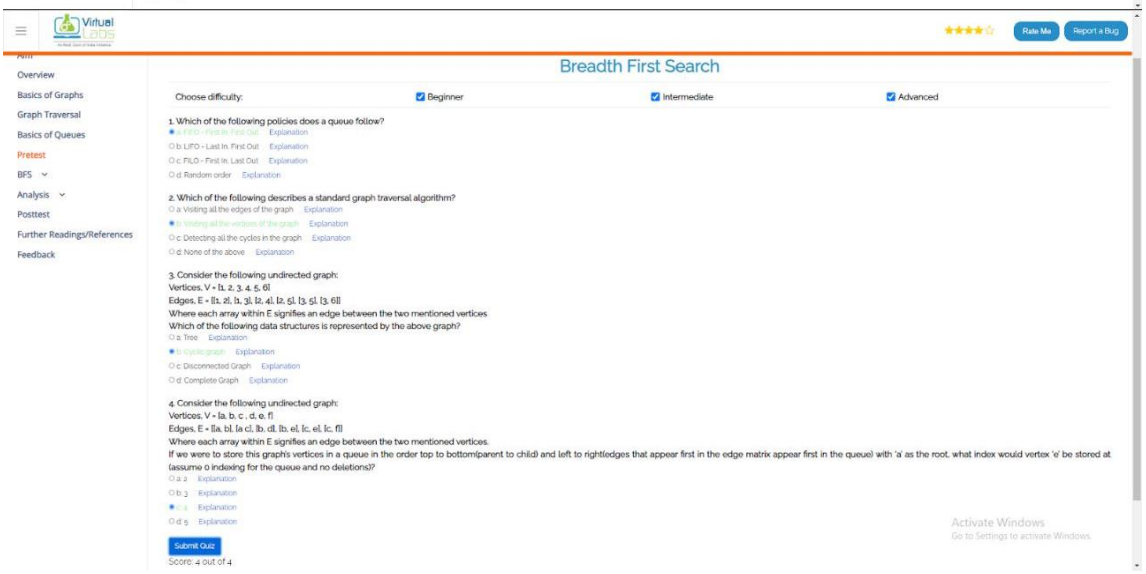

Virtual Labs
Rate Me
Report a Bug
Computer Science and Engineering > Data Structures - 1 > Experiments
Aim
Overview
Basics of Graphs
Graph Traversals
Pretest
Depth First Search
Aim
Concept
Algorithm
Demo
Practice
Exercise
Quiz
Code Assessment
Analysis
Posttest
Further Readings/References
Feedback
Depth First Search
1. Which one of the following data structures is used in DFS?
a: Heap
b: Dequeue
c: Linked List
d: Stack
2. What is backtracking in DFS?
a: Going to the parent node if all the children nodes have been visited
b: Going to the parent node when a children node has been visited
c: Going to the sibling node when all the children nodes have been visited
d: Going to the sibling node when a children node has been visited
Submit Quiz
2 out of 2


Virtual Labs
Rate Me
Report a Bug
Basics of Graphs
Graph Traversals
Pretest
Depth First Search
Code Assessment
Analysis
Posttest
Further Readings/References
Feedback
PROBLEM 1
PROBLEM 2
Basic Depth First Search
You are given the list of edges of a directed graph and the root node. Perform depth first search to find the correct order of visitation of nodes. You are advised to first construct a graph from the edge list then proceed further.
Input Format
Input consists of an array of edges as [node1, node2]. For node1 as parent. The first of the edges denote the edge with the left child, other latter edge with the right child.
Output Format
An array of nodes in the correct order of visitation
Output
Correct
Code Output:
Expected Output:
Submit


Virtual Labs
Rate Me
Report a Bug
Basics of Graphs
Graph Traversals
Pretest
Depth First Search
Code Assessment
Analysis
Posttest
Further Readings/References
Feedback
PROBLEM 1
PROBLEM 2
Detect Cycle in Directed Graph
Intermediate
Implement DFS to detect cycles in a directed graph.
Input Format
Input consists of a directed graph represented as an array of edges, where each edge [parent, child] denotes a directed connection from parent to child, and a starting vertex.
Output Format
Boolean indicating whether the graph contains a cycle.
JS
const visited = new Set();
const recStack = new Set();
const dfs = (node) => {
if (!graph.has(node)) return false;
if (recStack.has(node)) return true;
if (visited.has(node)) return false;
visited.add(node);
recStack.add(node);
for (const neighbor of graph.get(node)) {
if (dfs(neighbor)) {
Output
Code Output:
true
Submit
Community Links
Sakshat Portal
Contact Us
Phone General Information: 011-26582030
Follow Us
Activate Windows
Go to Settings to activate Windows.

Virtual Labs
Rate Me
Report a Bug
Computer Science and Engineering > Data Structures - 1 > Experiments
Aim
Overview
Basics of Graphs
Graph Traversals
Pretest
Depth First Search
Code Assessment
Analysis
Aim
Space and Time Complexity
Comparison of BFS with DFS
Quiz
Posttest
Further Readings/References
Feedback
Depth First Search
1. If there are 10 edges in a graph, in the worst case how many edges can be traversed?
a. 0
b. 1
c. 5
d. 10
2. Which one of the following is correct?
a. DFS uses queue & BFS can be done using stack & recursion.
b. In DFS, all the neighbors are traversed before other nodes.
c. DFS is a vertex-based algorithm while BFS is an edge-based algorithm.
d. BFS is an optimal algorithm while DFS is not optimal.
Submit Quiz
2 out of 2





Code:

```
#include <stdio.h>

#include <stdlib.h>
int visit[20] = {0};
int v[20] = {0};
typedef struct node
{
    int data;
    struct node *prev;
    struct node *link;
} node;

typedef struct queue
{
```

```
    struct node *rr;
    struct node *fr;
} que;

int dequeue(que *q)
{
    node *temp;
    if (q->rr != NULL)
    {
        temp = q->rr;
        int d = temp->data;
        q->rr = temp->prev;
        if (q->rr != NULL)
            q->rr->link = NULL;
        else
            q->fr = NULL;
        return d;
    }
    return 0;
}

void enqueue(int ch, que *q)
{
    node *nnode;
    nnode = (node *)malloc(sizeof(node));
    nnode->data = ch;
    nnode->link = NULL;
    nnode->prev = NULL;
    if (q->fr == NULL)
    {
        q->fr = nnode;
        q->rr = nnode;
    }
    else
    {
        nnode->link = q->fr;
        q->fr->prev = nnode;
        q->fr = nnode;
    }
}

void display(que *q)
{
    node *temp;
    temp = q->fr;
    while (temp != NULL)
```

```
    {
        printf(" %c", temp->data);
        temp = temp->link;
    }
}

void dfs(int t, int a[20][20], int n)
{
    int i, j;
    printf("%d->", t);
    visit[t - 1] = 1;
    for (i = 0; i < n; i++)
        if (a[t - 1][i] == 1 && visit[i] == 0)
            dfs(i + 1, a, n);
}

void bfs(int t, int a[20][20], int n, que *q)
{
    int i, j;
    printf("%d->", t);
    int temp;
    enqueue(t, q);
    v[t - 1] = 1;
    while (q->fr != NULL)
    {
        temp = dequeue(q);
        for (i = 0; i < n; i++)
        {
            if (a[temp - 1][i] == 1 && v[i] == 0)
            {
                enqueue(i + 1, q);
                printf("%d->", i + 1);
                v[i] = 1;
            }
        }
    }
}

int main(void)
{
    printf("Enter number of vertices:\n");
    int n, i, j, e, p, q;
    scanf("%d", &n);
    int a[20][20];
    for (i = 0; i < n; i++)
    {
```

```
        visit[i] = 0;
        for (j = 0; j < n; j++)
            a[i][j] = 0;
    }
    printf("Enter number of edges:\n");
    scanf("%d", &e);
    printf("\nEnter 1 for undirected graph and 0 for directed graph:");
    int t;
    scanf("%d", &t);
    for (i = 0; i < e; i++)
    {
        printf("Enter edge vertex(p,q):\n");
        scanf("%d%d", &p, &q);
        a[p - 1][q - 1] = 1;
        if (t == 1)
            a[q - 1][p - 1] = 1;
    }

    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
            printf("%d ", a[i][j]);
        printf("\n");
    }
    printf("Enter Element from where you want to start dfs and bfs:");
    int d;
    scanf("%d", &d);
    printf("\n DFS:\n");
    dfs(d, a, n);
    que q1;
    q1.fr = q1.rr = NULL;
    printf("\n BFS:\n");
    bfs(d, a, n, &q1);
    return 0;
}
```

Output:

```
Enter number of vertices:
4
Enter number of edges:
4

Enter 1 for undirected graph and 0 for directed graph:1
Enter edge vertex(p,q):
4 3
Enter edge vertex(p,q):
7 5
Enter edge vertex(p,q):
8 6
Enter edge vertex(p,q):
9 3
0 0 0 0
0 0 0 0
0 0 0 1
0 0 1 0
Enter Element from where you want to start dfs and bfs:3

 DFS:
3->4->
 BFS:
3->4->
```