

## DATA STRUCTURE

Type      Graph  
Deadline    **IN CLASS**  
Weighting   TBA

LAB PERFORMANCE

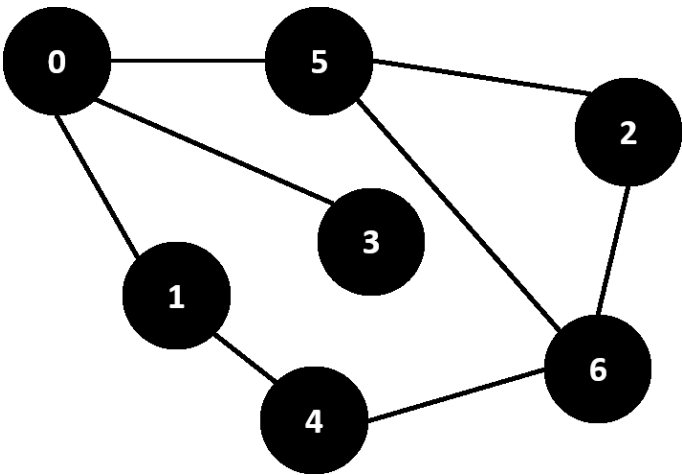
1

### OBJECTIVES

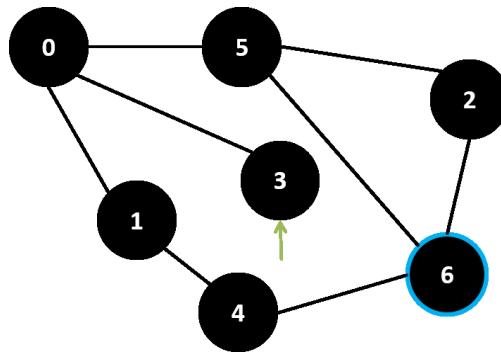
This assessment item is designed to test your skills on constructing graph, find the connected components and traversing through a graph etc.

### ASSESSMENT TASK

1. The first part of this lab aims to construct an **undirected graph** from a given set of edges. The graph nodes have been named using numbers **0** to **n-1**. The graph comes from the console in the following format:
  - First line: number of edges **n**
  - Next **n** lines: list of edges for the nodes (separated by a space)
  - i. You need to construct this graph using Adjacency List/ Adjacency Matrix
  - ii. Print the graph in the same format as in the examples below

Input	Graph	Output
8 0 1 0 3 0 5 1 4 5 2 5 6 4 6 2 6		0: 1, 3, 5 1: 0, 4 2: 5, 6 3: 0 4: 1, 6 5: 0, 2, 6 6: 4, 5, 2

2. The second aim of this lab is to perform a search on the above graph using BFS and/or DFS graph traversal algorithm where you can consider that 3 is the start node and the goal node is 6. The situation is illustrated below:



#### ALGORITHM for BFS:

```

BFS(G, s, g){ //G ≈ graph, s ≈ start node, , g ≈ goal node
    Queue.append(s);
    while(Queue.front()!=g){
        nodeToExpand = Q.front()
        Queue.serve()
        while(adjacents of nodeToExpand not in inQueue){ //inQueue ≈ flag-array
            Queue.append(adjacents of nodeToExpand)
        }
    }
}

```

#### ALGORITHM for DFS:

```

DFS(G, s, g){ //G ≈ graph, s ≈ start node, , g ≈ goal node
    Stack.push(s);
    while(Stack.top()!=g){
        nodeToExpand = Stack.top()
        Stack.pop()
        while(adjacents of nodeToExpand not in inStack){ //inStack ≈ flag-array
            Stack.push(adjacents of nodeToExpand)
        }
    }
}

```

\*\*\* You may use **list** instead of array for the **flag-array** (inQueue or inStack)

### 3. ASSIGNMENT for HOME:

The aim of this assignment is to find the **connected components** (nodes connected to each other either directly, or through other nodes) of a graph. The graph nodes are numbered from **0** to **n-1**. The graph comes from the console in the following format:

- First line: number of lines **n**
- Next **n** lines: list of child nodes for the nodes **0** ... **n-1** (separated by a space)

Print the connected components in the same format as in the examples below:

Input	Graph	Output
9 3 6 3 4 5 6 8 0 1 5 1 6 1 3 0 1 4 2		Connected component: 6 4 5 1 3 0 Connected component: 8 2 Connected component: 7
1 0		Connected component: 0
0	(empty graph)	Connected component:
7 2 6 1 4 3 1		Connected component: 0 Connected component: 2 6 1 Connected component: 4 3 Connected component: 5
4 1 2 3 0 1 2 3 3 0 1 3 0 1 1 2		Connected component: 3 2 1 0

## WHAT & HOW TO SUBMIT

You need to upload through your **VUES** account. You can find the upload link under “*Courses/ DATA STRUCTURE/Lab Performance/*”

### SUBMISSION STEPS:

1. Create a Directory/Folder as following format:

**<Your ID>\_PERFORMANCE-< Performance Number>**

Ex: 14-10380-1\_PERFORMANCE-1

2. If you update your code then the format should be following:

**<Your ID>\_PERFORMANCE-< Performance Number>\_UPDATE-<Update Number>**

Ex: 14-10380-1\_PERFORMANCE-1\_UPDATE-1

3. Put all the files into that Folder and upload the **zipped** format of that Folder

### NOTES

- Your submission will be rejected if uploaded in wrong format
- Only “.zip” file will be accepted.