University of Dhaka

Department of Robotics and Mechatronics Engineering

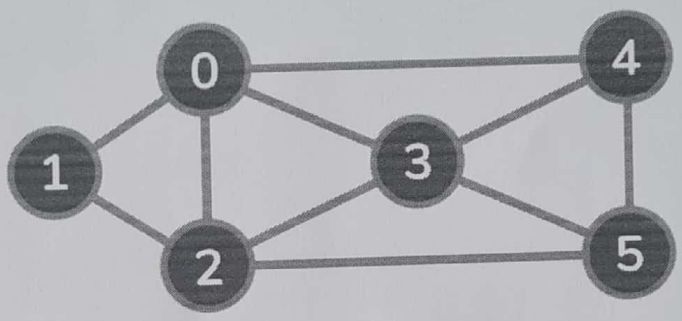
3rd Year 1st Semester B.Sc. Lab Final Examination, 2021

RME 3111: Artificial Intelligence Lab

Full Marks: 40 Time: 90 minutes

Problem 1: **[10]**

Show the step by step execution of Breadth-First -Search (BFS) to compute the shortest path from node 1 to 5.



Problem 2: (Omitted from the exam) **[15]**

In this problem, you need to write a code for computing the BFS traversal given an unweighted graph. The graph has **n** nodes indexed **0** to **n-1**. It is given in the form of an adjacency list. Return the BFS traversal starting at the node indexed **0**. Follow the order of nodes as in the adjacency list.

Input Format:

The first line contains an integer T denoting the number of test cases.

For each test case, the input has the following lines:

- The first line contains the integer **n**.
- The next **n** lines describe the adjacent nodes for the **i**th node.
  - The first integer **m** denotes the number of adjacent nodes.
  - The next **m** integers denote adjacent nodes.

Output Format:

For each test case, the output has one line with space-separated integers denoting the BFS.

Sample Input

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

Expected Output

```
0 1 2 3
0 1 2 3
```

Constraints

1 <= T <= 10
1 <= n <= 500
0 <= m <= n

Problem 3: **[15]**

Here you need to modify the code you write for Problem 2. This time the queue has to be implemented using two stacks (i.e. **you cannot use the queue data structure** ☹). The input and output should be exactly the same as Problem 2.

Here is how:

Let the queue to be implemented be **q** and stacks used to implement **q** be **stack1** and **stack2**. The **q** can be implemented in the following way:

This method makes sure that oldest entered element is always at the top of **stack1**, so that deQueue operation just pops from **stack1**. To put the element at top of **stack1**, **stack2** is used.

enQueue(q, x):

- *While **stack1** is not empty, push everything from **stack1** to **stack2**.*
- *Push **x** to **stack1** (assuming size of stacks is unlimited).*
- *Push everything back to **stack1**.*

deQueue(q):

- *If **stack1** is empty then error*
- *Pop an item from **stack1** and return it*

A. Write a code to solve the given problem. [9]
B. What is the run time complexity of the enQueue operation? [3]
C. What is the run time complexity of the deQueue operation? [3]