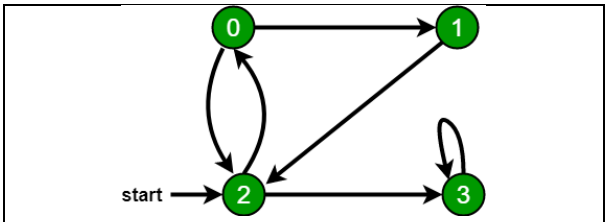
# EEE 210: Software Engineering
# Lab 10 Exercises for Week 15 (16 Apr. – 22 Apr.), Spring 2018

***Note:***

- Project folder nomenclature: Lab10_*yourname*
- After completion, zip your project folder and upload it to your Moodle account by the end of today's session.
- Any queries during the lab should be discussed merely with the Instructor/TA.
- No report is required for this project.
- You are **not allowed** to use the Internet while doing these exercises. Consider this as a quiz!

***Exercise 1:*** **Breadth First Search (BFS) for a Graph**

Unlike trees, graphs may contain cycles, i.e. we may come back to a node that was already traversed. For example, in the following graph $G_1$, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Breadth First Traversal of the following graph is 2, 0, 3, and 1. For simplicity, it is assumed that all vertices are reachable from the starting vertex.



Example of directed graph $G_1$.

(a) Implement the Graph class that represents a directed graph using an adjacency list. It should contain a parameterized constructor `Graph(int v)`, method `addEdge(int v, int w)` to add an edge `(v,w)` into the graph, and method `BFS(int s)` that traverses the graph from the given starting vertex s.

Use the following driver program to be included in the Graph class to test the above methods:

```
/* Driver program to test above functions */
    public static void main(String args[]){

        Graph g = new Graph(4);
        g.addEdge(0, 1);
        g.addEdge(0, 2);
        g.addEdge(1, 2);
        g.addEdge(2, 0);
        g.addEdge(2, 3);
        g.addEdge(3, 3);
        System.out.println("Following is BFS " + "(starting from vertex 2)");
        g.BFS(2);
    }
```

The output of BFS should be as shown below:

```
Following is BFS (starting from vertex 2)
2 0 3 1
```

(b) Modify BFS in part (a) to perform depth first search (DFS) instead. For instance, the result of applying DFS on graph $G_1$ would be 2, 0, 1, and 3. The output should look like:

```
Following is DFS (starting from vertex 2)
2 0 1 3
```

Good luck!