**Data Structures (2028C) -- Spring 2017 – Lab 11**

***Topics covered: Graphs***

*Lab due:* ***Sunday, Apr 8 at 11:55PM for Monday Section***

***Tuesday, Apr 10 at 11:55 PM for Wednesday Section***

**Objective:**

The objective of this lab is to explore creating and traversing graphs

**Task 1:** Create a directed graph *G=(N,E)* using Adjacency List method. Here N is a set of nodes and E is a set of ordered pairs of nodes called edges. An edge(i,j) is directed from i to j.

1. Create a new project. You can name this whatever you like.

Design and implement a graph class using the adjacency list method as explained in the class lecture.

* 1. The class should have the following methods fully implemented.
     1. Constructor – This should have an overload indicating the number of nodes in the graph.
     2. addNode(i) – Add a node with the value i
     3. addEdge(i,j) – Add the edge (i,j) to E.
     4. removeEdge(i,j) – Remove the edge (i,j) from E.
     5. hasEdge(i,j) – Check if the edge (i,j) ∈ E.
     6. outEdges(i) – Return a list of all integers j such that edge (i,j) ∈ E.
     7. inEdges(i) – Return a list of all integers j such that edge (j,i) ∈ E.
     8. displayGraph() – Display the graph as adjacency lists
     9. Destructor

**Task 2:**

****

Figure 1: Input graph

1. Create a program that tests the graph class created in Task 1. Prompt the user for which class method to invoke. This may look like:

|  |
| --- |
| Press 1 to add a node to graph.  Press 2 to add an edge to graph.  Press 3 remove an edge from graph.  Press 4 Find an edge in the graph.  Press 5 Find the out edges of a node  Press 6 Find the in edges of a node  Press 7 Display the graph as adjacency lists  Press 8 to quit. |

1. Use your test program to test all member functions and ensure the class is working correctly. Check both for positive and negative cases (e.g. removing an edge which is not present in the graph).
2. Test all the functions of the graph class on the graph given in Figure 1.
3. Include in the lab report screen shots of the output of all your test results.

Complete this before moving on to task 3.

**Task 3:**

1. Add a function to your graph class namely breadthFirstSearch(i) which starting at node i, which searches all the nodes in a breadth first manner. The function should output the nodes visited in the breadth first order. Update the test program created in Task 2 to check whether breadth first search(BFS\*) is done properly on the graph in figure 1. Include in the lab report screen shots of the output of breadth first search.

**Task 4:**

1. Add a function to your graph class namely depthFirstSearch(i) which starting at node i, which searches all the nodes in a depth first manner. The function should output the nodes visited in the depth first order. Update the test program created in Task 2 to check whether depth first search(DFS\*) is done properly on the graph in figure 1. Include in the lab report screen shots of the output of depth first search.

**Task 5:**

1. Explain in your lab report how you implemented depth first and breadth first searches. More importantly explain what data structures you chose to implement them and the reasoning behind them. Discuss the following in the lab report:
   1. Applications of BFS & DFS
   2. Memory consumption of BFS & DFS
   3. When searching a graph when does one choose BFS/DFS?

**Lab Submission:**

1. Write a lab report including the following information:
   1. A description of the objectives/concepts explored in this assignment including why you think they are important to this course and a career in CS and/or Engineering.
   2. The sections from each task indicated to be included in the lab report.
2. Include all source code from all tasks, input and output files (if any), and any special instructions to compile and run those programs.
3. In a group project, submissions should include what each group member has contributed.
4. Package all files in a single zip folder and upload the file to canopy website within the specified due date.

**Lab Grading:**

1. 10% - Lab attendance
2. 20% - Task 1 has been correctly implemented and meets all requirements.
3. 10% - Task 2 has been correctly implemented and meets all requirements.
4. 15% - Task 3 has been correctly implemented and meets all requirements.
5. 15% - Task 4 has been correctly implemented and meets all requirements.
6. 10% - Task 5: Implementation of BFS & DFS is explained along with the choice of data structure selected. Discussion questions has been properly answered.
7. 20% - Lab report contains all required information and is well written.

If program fails to compile, 0% will be given for that Task.