# CS 240

Data Structures and Algorithms

Spring 2014

## 1 Lab 11 (Project 2)

## 2 Grading

- Five percent of this lab will be following submission instructions on both parts of the lab assignment.
- Ten percent of this lab assignment will be the result of your evaluation of each other as a team – evaluation forms will be available when the assignment is due.
- Ten percent of this lab assignment will be the result of your implementation's performance relative to other students in the class. This means your code will be tested in a series of automated tests. This will be done using the static library you will create. The only parts that will be tested for performance will be those marked with a * in the description of each method below.
  - Top team – 10 points
  - Team 2 – 9 points
  - Team 3 – 8 points
  - Team 5 – 7 points
  - Teams 5 and 6 – 6 points
  - Teams 7 and 8 – 5 points
  - Teams 9, 10, and 11 – 4 points

  - Teams 12, 13, and 14 – 3 points
  - Teams 15, 16, and 17 – 2 points
  - Teams 18+ – 1 point

- The remainder will be based upon successful completion of the assigned tasks.

# 3 Assignment

You will be building an all-inclusive graph class/library. This library will have to perform several tasks. Please see the accompanying header file for function signatures.

- Construct – Construct an empty graph.
- Delete – Delete a graph.
- Read A Graph From A File – The first line of the file will contain either the word `directed` or `undirected` and will describe the types of edges the graph contains. The second line of the file will contain the number of vertices, which will need to be labeled as $1..n$ in your program. The third line is the number of edges in the graph, E. The next E lines of the file will contain:

  ```
  v1 v2 weight
  ```

  where v1 and v2 are integers (from $1..n$ that each represent a vertex). The first number is the vertex from which the edge emanates (if applicable) and the second number is where the edge terminates. The weights are doubles. *Do not make any assumptions about the values of these weights.*
- Write A Graph To A File – Using the same format as was defined for reading from the file, write a graph to a file.
- Empty – Check to see if the graph is empty.
- Add Edge – Add an edge to a graph. An edge is defined by the names of two nodes and a weight.

- Add Vertex – Add a vertex to a graph. This will increase the number of vertices by one, the name for this vertex must be the next consecutive integer value.

- Count Connected Components – Count the number of components in the graph that are connected to one another. Components are connected when there is a path from each vertex to every other vertex in the component.

- Tree Check – Determine whether or not this graph is a tree. When we talk about a tree for an undirected graph, the graph must be connected and may not contain any cycles (acyclic).

- Depth First Traverse – Print to a file, a traversal for the graph using a Depth First approach, starting from the passed source. When considering children, consider them in numerical order. The name of the file is passed, and the file should contain only integers, one per line representing the traversal in the order nodes are explored.

- Breadth First Traverse – Print to a file, a traversal for the graph using a Breadth First approach, starting from the passed source. When considering children, consider them in numerical order. The name of the file is passed, and the file should contain only integers, one per line representing the traversal in the order nodes are explored.

- Closeness – Determine the closeness of two nodes in the graph. For this piece, the closeness of nodes is defined as the minimum number of edges that need to be taken to navigate a path from one node to the other in the graph. If the nodes are not connected, return -1.

- *Partition – Determine if there is a way to partition the nodes in the graph. The nodes can be partitioned if we can categorize each node into two groups such that any path from one node to another switches between groups with each step. For example, if we were to name the groups Red and Blue, and we were to follow any path in the graph, then if we listed the groups of each node in the path their pattern would either be:

  ```
  Red -> Blue -> Red -> Blue -> ...
  ```

  or

`Blue -> Red -> Blue -> Red -> ...`

- *MST – Print a minimum spanning tree of the graph to a file. If the graph is disconnected, list each component and a minimum spanning tree for that component. The tree should be printed on one line as {`V, E`}, where $V$ is the set of vertices, and $E$ is the set of edges. Note that $V$ and $E$ themselves are sets and should be enclosed in brackets. Each edge is defined as a 3-tuple `(v1,v2, w)`.

- *StepAway – Print all vertices to a file, one line at a time, whose "closeness" (as defined above) matches the provided argument. Note that it is possible that the provided argument has a value of `-1`.

# 4 Teamwork

There is a teamwork component to this lab. You will be required to work in a team (teams of two – when possible try to make sure your teammate is in your scheduled lab session). Additionally, since you must be able to work with other students than just one in the class, you must choose a different student to work with than the one you worked with for Lab 10 (Project 1).

# 5 Submission

Further submission instructions for the overall project will be posted as the due date approaches.

**The due date for this assignment is May 13, 2014 at 11:59:59pm.**

Note that all of you have at least one final exam, it is highly recommended that you begin work on this assignment as early as possible since you will likely have a difficult time coordinating your schedules as finals approach.

**There will be no late assignments accepted.**