Computer Science 62
Lab 12

Wednesday, Dec 2, 2015

This lab will serve as a warm-up for Assignment 13. You will be working with the provided graph ADT to implement two graph algorithms. The first algorithm is an edge reversal, i.e., the algorithms takes an input graph and returns a graph which is identical to the input but all of the edges are reversed. You will use this function as part of Assignment 13 to test strong connectivity. The second algorithm is breadth-first search. In Assignment 13 you will be implementing Dijkstra's algorithm which is essentially breadth-first search, replacing the queue with a priority queue.

Getting Started

First, copy the starter code over from the usual locations, and read over the API for the graph ADT, described using JavaDocs in `graph.h`. You will also being using a queue for this assignment, which has been provided to you in `queue.h` (its the doubly linked list from a previous lab). Once you understand the API we will be using when working with graphs, read over the JavaDocs in `graph_lib.h`, which contains the specifications for the two functions you will be implementing for this lab.

Edge reversal

Your edge reversal should run in $O(n + m)$ time and must only access the graph through the functions declared in `graph.h`, i.e., you should not try to directly manipulate `struct graph`. There are several ways to implement this method; if you get stuck coming up with an algorithms, you should feel free to ask one of us for help. Make sure you thoroughly test this method! Test cases will be given on the whiteboards, but feel free to create your own (an add them to the whiteboards). You may want to uncomment my debugging code to print the underlying graph.

Breadth-first search

You will be implementing the breadth-first search algorithm, as described in class. In lab you will only be implementing the "basic" breadth-first search algorithm, i.e., you will explore all vertices reachable from a single vertex without restarting to explore the entire graph. Recall, that the "basic" breadth-first search takes a connected graph and produces a spanning tree represented using an array of parents. The entire solution to this problem is contained in the lecture slides, but should make sure you understand every line you are typing! You should thoroughly test this method as well, and compare your results with the examples on the white boards.

Assignment 13

You may freely use code form this lab while implementing Assignment 13.