IE 170 Laboratory 7: Graph Search

Dr. T.K. Ralphs

Due April 3, 2006

1 Laboratory Description and Procedures

1.1 Learning Objectives

You should be able to do the following after completing this laboratory.

1. Understand the concept and use of a graph.
2. Understand how to construct a graph and the two basic data structures for storing a graph.
3. Understand how to implement a basic graph search.
4. Understand the importance of object-oriented design and code re-use.

1.2 Key Words

You should be able to define the following key words after completing this laboratory.

1. graph
2. vertex
3. edge
4. adjacency list
5. adjacency matrix
6. graph search
7. depth-first search

1.3 Scenario

You have been hired by NeverLost, Inc, a designer and retailer of travel maps. As part of their effort to introduce a new custom map service for members of their travel club, they would like to design a data base for storing map data in the form of a graph, from which various information, such as the shortest route between two given cities, could be extracted. Your immediate task is to implement a data structure that could be used to read in and store a list of destination pairs, as well perform a basic search of the graph once it is constructed. In future work, you may be asked to implement more sophisticated algorithms, such as that of finding the shortest path between two locations. Since you have performed work for other clients and have some code for basic data structures already written, you should try to re-use as much code as possible for this project.

1.4 Design and Analysis

This lab will be focused primarily on design and implementation of a basic graph data structure. Performance analysis will be part of the series of laboratories that follow this one. One of the goals of this lab is to illustrate the efficiencies that can be gained through re-use of object-oriented code. Therefore, we will attempt to re-use two data structures we coded earlier in the course—a linked list and a stack.

1.5 Program Specifications

Your task is to implement a graph class using adjacency lists. The interface for the data structure is contained in the file `graph.h`. To implement the search function, you will need to provide a stack data structure. You can re-use your stack implementation from Laboratory 3 for this purpose.

1.5.1 Program Function

Your graph class should support the interface given in the file `graph.h`. Part of the implementation has been done for you already and is contained in the file `graph.cpp`. Your graph class should include a constructor that reads in a graph from a file and constructs an adjacency list representation.

1.5.2 Algorithms

The algorithm exemplified in this lab is graph search, as described in Lecture 15. The code for the graph search is given to you as part of the lab. However, you will have to provide the stack implementation using your code from Laboratory 3.

1.5.3 Data Structures

The data structures used in this lab are linked lists (for implementing the adjacency lists), a stack (for implementing depth-first search), and an adjacency list representation of a graph.

2 Laboratory Files

The files for this laboratory are in the zip archive `Lab7.zip` available on the course Web site. The archive will unpack into a directory called `Lab7` with `shell` and `data` subdirectories. The shell directory contains the files

1. `main.cpp`: the client program used for testing.
2. `graph.h`: the interface for the graph class and other necessary classes.
3. `graph.cpp`: the implementation file.

The `data` subdirectory contains two test files containing graphs that can be read in and searched to test your code.

3 Laboratory Assignments

3.1 Programming (50 points)

1. Implement a graph class supporting the given interface.
2. Verify that your program compiles and runs correctly using the test files provided for the lab.

3.2 Follow-up Questions (50 points)

1. (10 points) Illustrate the union-find algorithm by constructing the union-find tree for the graph in the file `input1.txt`. Show the result both with the standard quick union algorithm and with the use of compression by halving. What is the difference in performance of the two algorithms on this example?
2. (15 points) 21-1
3. (5 points) CLRS 22.4-1.
4. (10 points) Consider a graph data structure implemented using adjacency lists. Suppose that instead of a linked list, the adjacency list for each vertex is stored as a hash table. Assuming that all edge lookups are equally likely, what is the expected time to determine whether an edge is in the graph using this scheme? What are the advantages and disadvantages of this data structure?
5. (10 points) 22.4-3