IE 170 Laboratory 8: Shortest Paths

Drs. T.K. Ralphs and R.T. Berger

Due April 10, 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 the power and generality of graph search in solving many important problems.
3. Understand the importance of object-oriented design and code re-use.
4. Understand the importance of and application of the shortest path problem.

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. graph search
5. shortest path

1.3 Scenario

The public Internet is a collection of networks that interconnects millions of computing devices, or end systems, throughout the world. End systems are not connected directly to the Internet. Instead, end systems are connected to a local or regional network through an Internet Service Provider (ISP), which in turn is connected to the Internet through a national or international ISP, such as UUNET or Sprint. Each component network is comprised of end systems interconnected by intermediate switching devices known as routers. Routers are responsible for forwarding packets along a path through the network from the origin to the destination. Special routers called gateway routers are responsible for the packets that traverse multiple networks. To facilitate the forwarding of packets, each router stores a table that lists for each destination the next router on the path to that destination. The table is constructed and updated based on the results obtained from a routing algorithm. Each component network, or autonomous system (AS), is managed independently, but

each router within an AS must run the same protocol. OSPF (Open Shortest Path First) is a common protocol used for routing packets within an AS. The basic idea of OSPF is that each router constructs a complete topological map (a directed graph) of the entire AS by exchanging information packets. Each router then runs Dijkstra's Algorithm to determine the shortest path to all other routers from itself. The link costs used by the algorithm must be specified by the network administrator. The algorithm is run whenever new link information is received by a router as well as periodically to ensure that the routing table contains valid next hop information. Clearly, the graphs in question here are extremely large and computational efficiency is critical. In this lab, you will implement Dijkstra's Algorithm as it would be implemented within a simple router.

1.4 Design and Analysis

In this lab, you will recycle source code from two previous labs to implement Dijkstra's Algorithm for finding a shortest path in a graph. The two data structures that you will recycle are the graph class from Laboratory 7 and the heap data structure from Laboratory 4. You should first modify your graph class to accommodate weighted graphs and then modify your heap data structure to function as a priority queue. Use these two classes to implement Dijkstra's Algorithm as we discussed in Lecture 16, using the priority queue data structure to maintain the list of unprocessed nodes.

After implementing the needed data structures, add a method to your graph class that finds the shortest path from a given source vertex to all other nodes in a given graph. The method should take the source vertex as an argument and should print the shortest path to each other vertex in the graph, along with the distance of the path.

In the analysis section, you will analyze the running time of finding a shortest paths tree in a randomly generated graph. The following function can be added to your graph class to generate a random graph. Note that you will also have to generate random weights for the edges. How you do this depends on how you decide to implement your class.

```
void Graph::random(int V, double density)
{
  int i(0), j(0), w(0);
  int numEdges(0);

  srand ( time(NULL) );

  //========================================================================
  // Read in number of vertices
  //========================================================================

  setVertNum(V);
  vertices_ = new Vertex* [vertNum_];
  for (i = 0;  i < V; ++i) {
    vertices_[i] =  new Vertex(i);
  }

  //========================================================================
  // Read in edges
  //========================================================================
```

```
  for (i = 0;  i < V; ++i) {
    for (j = 0; j < i; ++j) {
      if (rand() < density * RAND_MAX){
        w = static_cast<int>( (double(rand())/RAND_MAX) * 1000);
        vertices_[i]->getAdjList().push_back(Edge(i, j. w));
        vertices_[j]->getAdjList().push_back(Edge(j, i, w));
        numEdges++;
      }
    }
  }
  setEdgeNum(numEdges);
}
```

1.5 Program Specifications

You should write a client program that will allow you to answer the questions in the analysis section.

1.5.1 Algorithms

The algorithm exemplified in this lab is Dijkstra's Algorithm for solving the shortest path problem with positive edge weights.

1.5.2 Data Structures

The data structures used in this lab are the graph data structure with adjacency lists, and the heap data structure.

2 Laboratory Files

There is one test file called `input1.txt` that can be used to test your shortest path function. You will also be required to generate some random graphs.

3 Laboratory Assignments

3.1 Programming (50 points)

1. Implement a shortest path function for your graph class, as described above.
2. Verify that your program compiles and runs correctly using the test file provided for the lab.

3.2 Analysis and Follow-up Questions (50 points)

1. (10 points) Generate random graphs of various sizes with edge density 10% and find the shortest paths tree for each of them. How big a graph can you find the shortest paths tree for in 1 second? 5 seconds? Try to estimate the growth rate of the running time.

2. (10 points) How much does the solution time vary in finding the shortest paths tree of random graphs of the same size and density? Why do you think this is? Support your assertion with experimental data.

3. (10 points) CLRS 24.3-4

4. (10 points) CLRS 24.3-6

5. (10 points) (T or F) In a shortest path problem, if each arc length increases by k units, shortest path distances increase by a multiple of k. Prove or provide a counterexample.