THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

Tamaki Campus

COMPSCI.220.S1.T - Algorithms and Data Structures

ASSIGNMENT 2 – GRAPH ALGORITHMS

Out: Friday 30 March, 2007

Due: Thursday 3 May, 2007

This assignment is worth **100 marks** representing **8.33%** of your total course grade.

Requirements

1. (50%) MAZE: Write a Java program to solve the following problem. You are given a maze, represented as a graph in standard adjacency lists format, a source node s and a target node t. You must find a path from s to t (that is, a path out of the maze) if one exists, and otherwise report failure. You should take input from standard input and output to standard output.

   Hint: for maximum marks on the speed test, you may want to consider using randomization to defend against hard inputs dreamed up by the lecturer. However, correctness is more important than speed!

2. (50%) APSP: Write a Java program to solve the all-pairs shortest path problem. Your program should be able to handle digraphs with loops, and with arcs of negative weight. The program should take as input a file in standard weighted adjacency matrix format, given as a command line argument. The program should terminate and print an error message if it finds a cycle of negative total weight. It should output, to standard output, the all-pairs distance matrix otherwise.

Notes on programming

- Java should be used for all programming questions unless special permission is obtained from the lecturer. You may freely use the graph classes available from my handouts directory. This is strongly recommended.
- Your answer to each question should be a single Java file (containing all nonstandard classes you use). You may assume that the markers have access to all standard libraries and also the graph classes in my handouts directory.
- Marks will be awarded for correctness and for running time in comparison to the lecturer's answer. A sample input and output file for each question is in my handouts directory, and your program's output should be formatted in exactly the same way.
- Your programs will be tested by running them on several input files, some of which will contain tricky small graphs and some of which may contain very large graphs. Marks will be allocated for correctness and speed of the programs. Programs that fail the correctness test will not be eligible for the speed test.
- The markers may check the output with a text comparison program, so it must be in EXACTLY the right format.

Representation of digraphs

We represent digraphs using one of three standard formats, described in the textbook. Sample files showing these formats are available from my handouts directory.

Here is a more formal description of our standard formats. We assume the nodes of each digraph are are labelled $0, 1, \ldots, n-1$. A digraph file consists of a certain number of subfiles appended one after the other, followed by a file consisting of a single line whose only entry is the character 0. Each of the small files is an adjacency list or (weighted) adjacency matrix for a

digraph and has the following form. The first line gives the order n of the digraph. There are n more lines, $L_0, \ldots, L_{n-1}$. Each line L_i consists of a finite list $a_0, a_1, \ldots,$ of integers, successive items being separated by a space. Item a_j is:

- (adjacency list case) the $(j+1)$-th neighbour of i;
- (adjacency matrix case) 0 or 1, depending on whether there is not or is an arc from node i to node j;
- (weighted adjacency matrix case) 0 if there is not an arc from node i to node j, and w if there is an arc of weight w.

Submission

The due date is Thursday, 3 May 2007, 8:30 p.m. (ADB time) [then penalty linearly grows in time from 0% to 50% on 5 May 2007, 8:30 p.m.; no submission afterwards].

Submit your assignment to the Assignment Dropbox (ADB) system. You have to electronically hand in

- a Java program maze.java for Q1.
- a Java program apsp.java for Q2;