CS2521 Assignment

Revision 1.1

Overview and Marking

This project aims to evaluate your ability to create, analyse, implement and evaluate algorithms. It will count for 25% of your overall course mark.

Hand-in

The hand-in date will be 23:59:59, on the 24th of March 2017. Handing in up to 24 hours late will attract a 10% penalty, while handing in up to a week late will attract a 25% penalty, deducted as a percentage of the mark obtained. Work handed in more than a week late will be treated as a "no paper".

Submission should take place via myAberdeen — upload a *zip* file containing a *pdf* report, and source code to the appropriate submission area. Please submit your reports within the relevant *zip* files (**not RAR, ARC, ARJ, LZH or any other format; these will not be marked**), *as PDF documents*. **Reports submitted as word documents, .odt documents, RTF, etc will not be marked.**

Plagiarism

Plagiarism will not be tolerated, it is a serious offence, with punishments ranging from a 0 mark to expulsion. If you are unsure about whether your work counts as plagiarised, please contact me.

1 Tasks

This assessment consists of two overarching tasks related to transportation, namely taxi allocation and passenger routing.

1.1 Taxi Allocation

Assume that you are writing software for a taxi company. For simplicity, assume that you have n taxis available, scattered across town, and that these are required to transport n passengers. Taxis have different fuel efficiencies, and may need to drive further to pick passengers up. Since you're paying the drivers by the hour, we'll treat time and cost interchangeably. Your goal is then to find an optimal allocation of taxis to passengers which minimises time/costs. We can represent an instance of the problem using a 2 dimensional array (a *matrix*) of the following form.

$$\begin{array}{cccc} & p_1 & \dots & p_n \\ t_1 & c_{11} & \dots & c_{1n} \\ \dots & \dots & \dots & \dots \\ t_n & c_{n1} & \dots & c_{nn} \end{array}$$

Here, c_{xy} is the travel time (equivalently cost) for taxi x to transport passenger y to their destination.

Step 1 Subtract the smallest entry in each row from all entries in the row.

Step 2 Subtract the smallest entry in each column from all entries in the column.

Step 3 Draw the minimum number of lines through appropriate rows and columns so that all the zero entries have a line through them.

Step 4 If the number of lines drawn is n (for a $n \times n$ array) then an optimal assignment has been found. This assignment picks a zero from every row such that each zero is in a unique column. If an optimal assignment has not been discovered, proceed to the next step.

Step 5 Find the smallest entry not covered by any line. Subtract this entry from each uncovered row, add it to each covered column, and then return to step 3.

Figure 1: Allocation Algorithm

1.1.1 Tasks

1. Describe a brute-force algorithm which addresses the taxi allocation problem, and identify and prove its time complexity.
2. A common algorithm to address the allocation problem[1] is outlined in Figure 1.
   (a) What is the time complexity for a brute force approach to implementing Step 3? Explain why.
   (b) Provide the pseudocode for, prove the correctness of, and implement, a more efficient version of step 3. Identify/prove your algorithm's time complexity.
   (c) Provide the pseudocode for an algorithm to achieve step 4, prove your algorithm's correctness, and identify/prove this algorithm's time complexity.
   (d) Implement the entire algorithm and empirically evaluate its average time complexity. Such an empirical evaluation should consider the runtime over different numbers of passengers and taxis. Remember also that a single run may not be representative of average algorithm performance.

1.2 Passenger Routing

Consider a graph described by a file containing lines of the form

```
<startNode> <endNode> <weight>
```

Here, the weight is the time taken to travel between a pair of nodes (e.g., by taxi).

1.2.1 Tasks

1. Given a file with lines of the form

   ```
   <startNode> <endNode>
   ```

   where each line represents a single passenger wishing to travel,
   (a) identify the total travel time for all the passengers (by implementing an algorithm to do so). Describe the algorithm you used for this (justifying its use), and empirically evaluate its time complexity, comparing it to its worst case complexity.

[1] Out of interest, this algorithm is used to match students and staff for Level 4 projects.

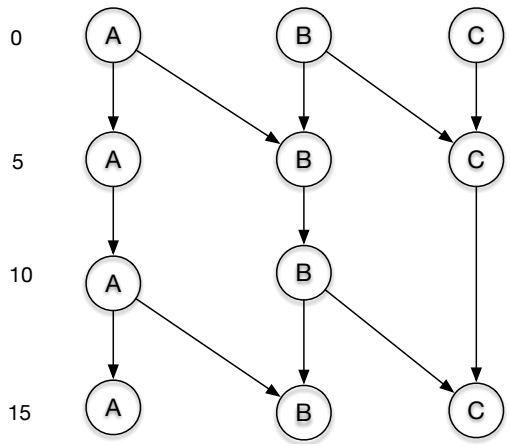

Figure 2: Time Expansion graph.

(b) Can you suggest a $O(n)$ algorithm which can solve this problem? Explain why is such an algorithm infeasible for large maps?

2. Assume that you are given another file with lines of the form

```
<startTime> <every>  <duration> <startNode>  <endNode>
```

Each line represents a public transport option which travels between start node and end node, starting at time startTime, and every `<every>` units of time thereafter, taking duration `<duration>`.

(a) Given a start node, start time, and an end node (a file containing a single `<startNode> <startTime> <endNode>` line[2]), identify the fastest route between the two nodes by outputting a string containing `<startNode> <departuretime> <nextNode>` lines (one per node), specifying which public transport service should be taken. To achieve this, you can use the *time-expansion* method. Here, you create a graph with a node for every arrival or departure at a specific location. There are then two types of edges — *travel edges* represent a transport link from one location to another, and are weighted by travel time. *Stay edges* link nodes at the same location, and are weighted by the time between these departures or arrivals. Figure 2 illustrates this for 3 locations (A,B,C), where busses take 5 minutes to get from A to B and B to C, and leave stations A and B every 10 minutes. You may assume that a startTime of 0 means midnight, and that the schedule for different days is identical.

(b) Given n nodes in the original map, and m lines in the public transport options file, determine what the maximum size of the resultant expanded graph could be, and justify your answer.

(c) Empirically evaluate the average time complexity of your algorithm with regards to different public transport frequencies while routing one passenger.

(d) Finally, assume that you have n passengers wishing to travel between two nodes (using the same format as the `busRouting.txt` file), and n taxis available, starting at nodes `<node1>, ..., <noden>` (these will be given in a file with one node per line). These taxis must travel to the passenger before taking them to their destination (i.e., if the taxi takes a units of time to get to the passenger, and then takes b units of time to transport the passenger, total travel time will be $a + b$). For simplicity, assume that once a taxi has been used to route a passenger, it will not be used again. Write a program to compute the time at which the last passenger(s) arrive at their destination; given the input files, as output, your program should return a single number — the time at which the last passenger(s) arrive at their destination.

[2]This file is used in a later task, you can consider only the first line of the sample file.

2 File Summary

This assignment will make use of the following files.

graph.txt	The description of the physical transport system.
basicRouting.txt	A list of start nodes and end nodes
timetable.txt	public transport timetable.
busRouting.txt	A list of routing requests for public transport.
taxiLocations.txt	A list of starting taxi locations.

You can use the `graph.txt` and `timetable.txt` from myAberdeen. The remaining files uploaded are samples, you should generate the other files yourself for testing your code.

3 Marking

Marks will be allocated as follows.

Question	Answer Format	Mark
Taxi Allocation		
1	Report	5
2a	Report	3
2b	Implementation	5
	Report	3
2c	Report	3
2d	Implementation	5
	Report	6
Passenger Routing		
1a	Implementation	15
	Report	5
1b	Report	5
2a	Implementation	15
	Report	5
2b	Report	5
2c	Report	5
2d	Implementation	10
Other	Quality of source code and of report.	5

Marks will be given for thoroughness within the report (consider for example curve fitting and statistical analyses in your empirical evaluation), insight and novelty. For implementation, marks will be given for good programming practices.

4 Acknowledgements

The graph dataset is based on the Rome dataset of `http://www.dis.uniroma1.it/challenge9/data/rome/rome99.gr`