1.204 Computer Algorithms in Systems Engineering

Spring 2010

Problem Set 1: Intercity Passenger Rail Schedules

Due: 12 noon, Tuesday, February 16, 2010

1. Introduction

This is a 'warm-up' homework that draws on the prerequisite knowledge of Java for this class. If your Java knowledge is rusty, or if you're coming from C++ or C# or C, this homework is intended to exercise the parts of the Java language we'll use most frequently this semester, and will give you a chance to brush up on your skills and get help.

If you are not yet comfortable programming in Java, you may do problem set 1 in another programming language such as C, C++, C# or VB.NET. Please contact the instructor for permission to do this. This option exists only for homework 1. We strongly recommend you do homework 1 in Java, but you do have the option. (Homework 2 is in SQL; homework 3 must be in Java, but it's not due until March 8, which gives you a month to get more comfortable with Java.)

2. Problem statement

The algorithm you'll use in homework 1 is simple; it's up to you to design it, as will be the case during the entire term. It will probably be a 'greedy algorithm', which we'll cover later; you don't need to do any analysis or proofs for it. Use your intuition to find a reasonable way to solve this problem. Amtrak has 20 Acela trainsets.

We have provided the weekday schedule for Acela Express trains between Boston, New York, and Washington. (The departure and arrival times are given to the nearest hour.) You can download the schedule from the course web site. It is under Homework -> Homework 1. You will compute the number of train sets required to cover the existing schedule under different assumptions about turnaround times at terminals.

Model the Acela schedule in a compact fashion that stores departure or arrival times only at the end points (Boston, Washington, and sometimes New York), ignoring times at intermediate stops. Store the schedules in arrays or ArrayLists. If you use arrays, you may dimension them larger than needed. Create appropriate classes and methods, with appropriate access. You do not need to do any error checking on input; the output can be simple and needs no fancy formatting. You may use console output. Input can be hardcoded into the program, or you may use JOptionPanes, files or console input.

3. Assignment

Compute the number of trainsets required to provide the existing service, under varying assumptions of turnaround times at each terminal. Compute the number of train sets

required at turnaround times of 1 hour, 2 hours and 3 hours. The turnaround time is the time from the set's arrival at a final terminal (Washington, Boston, or New York) until its next departure time in the opposite direction. For example, if a train set arrives at a terminal at 11am, with a one hour turnaround it can depart at noon, with a 2 hour turnaround it can depart at 1pm, and with a 3 hour turnaround it can depart at 2pm. The turnaround time provides a buffer for late arrivals and for train servicing. For this program:

- The input is the existing schedule and the three possible turnaround times
- The output is the number of train sets required, and which trips each train set covers. For each trip, output the trip origin, departure time, destination and arrival time.

If a train is running from Boston to Washington it does not incur any turnaround time in New York; this is an intermediate point on the route and the train continues its run in the same direction after a fairly short stop. However, if a train is running only between Boston and New York, or New York and Washington, and it turns around in New York, then it does incur turnaround time.

Homework 1 is a job scheduling problem, of which we'll see other examples. The homework 1 problem is to find the optimal (least) number of machines to do a set of jobs. The machines are train sets, and the jobs are train trips in the schedule. The schedule defines a set of precedence relationships among jobs, in both time and space. Each job has a start time, end time and a pair of locations.

Turn In

1. Place a comment with your full name, Stellar username, and assignment number at the beginning of all `.java` or other programming language's source files in your solution.
2. Place all of the files in your solution in a single zip file.
    a. Do not turn in electronic or printed copies of compiled byte code (`.class` files) or backup source code (`.java~` files). For other languages, do not turn in executable (`.exe`) files. Check with the instructor or TA for details.
    b. Do not turn in printed copies of your solution.
3. Submit this single zip file on the 1.204 Web site under the appropriate problem set number. For directions see **How To: Submit Homework** on the 1.204 Web site.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.
5. After you submit your solution, please recheck that you submitted your .java file. If you submitted your .class file, you will receive **zero credit.**

Penalties

- 30 points off if you turn in your problem set after Tuesday noon but before Thursday (February 18, 2010) noon. You have two no-penalty two-day (48 hrs) late submissions per term.
  No credit if you turn in your problem set after Thursday noon.

1.204 Computer Algorithms in Systems Engineering

Spring 2010