CSE 331 - Summer 2015

Programming Assignment 4

Due Tuesday, August 11, 2015 @ 2:10pm

IMPORTANT: The stated deadline on assignments will be strictly enforced. I will go over solutions at the deadline and will not accept submissions after the solutions have been presented.

Relevant code can be found at `https://bitbucket.org/hartloff/cse331-summer2015/` in the `assignments.assignment4` package. It is recommended that you clone the entire repository and pull regularly as all the code for this course will be placed there.

---

1 Space Elevator (100 points)

It's the year 3008 and you are a manager for an interstellar delivery company. Your company has constructed a new method of delivering packages from Earth: A space elevator! The elevator is used to transport the packages to a delivery hub that is just above Earth geosynchronous orbit. From here, the packages are loaded on ships and sent off to their destinations. It is much cheaper to use the space elevator to get the packages to this hub as opposed to having the delivery ships constantly re-entering Earths atmosphere and taking off again. The packages must be sent in the order they are received and Mr. Greedy decides to load packages into the elevator until the next one wont fit which minimizes the number of trips taken by the elevator. You are in charge of operating the space elevator at minimal cost.

The space elevator can be run on a variety of different fuel sources. The most efficient, by far, is dark matter. Conveniently, you have a pet that you rescued from an alien planet named Nibbler who "produces" dark matter. Nibbler produces balls of dark matter at regular intervals and each ball has an integer value for the energy it contains. The space elevator has a lock every 1000 miles so the elevator can be held while awaiting more fuel. The value associated with each ball of dark matter is the number of these 1000 mile intervals that will be covered by burning that ball. You have a dark matter dietitian on staff who tells you the next n values of dark matter drops so you can do some planning on how to use these balls.

You have no way to store dark matter, so each ball must be used as soon as it's produced (in the order you obtain them). Also, you can't exceed the space elevator height without destroying the elevator so you must always make sure there is enough room for each ball to be used and dark matter cannot be broken apart so the entire ball must be used. Since the dark matter will not always get the elevator exactly to the top, you must use an alternate fuel to push the elevator the rest of the way up. This alternate fuel has a cost that is a function of the remaining distance d that the elevator needs to be lifted. After the alternate fuel is used, the elevator quickly returns to the ground before the next ball of dark matter arrives. Your goal is to generate a schedule that chooses when to use the alternate fuel for the next n pieces of dark matter that will minimize cost. For the implementation, the schedule will be a boolean array of size n that is *true* at index i if the alternate fuel is to be used after using dark matter i. You can assume the elevator starts at ground level before drop 0 and that there is no cost for dark matter.

You can leave the elevator part of the way up at the end of the schedule with no cost. You assume you will be in business indefinitely (not like dark matter is going anywhere), but you can

only predict the next n Nibbler drops so you only have to schedule out this far. The last shipment on the elevator will continue with dark matter $n+1$, which we will not be concerned with.

The schedule will be computed by implementing Method 1 and Method 2 in the `Assignment4` class.

Method 1. `public static boolean[] generateSchedule(SpaceElevator elevator);`

Method 2. `public static double computeCost(SpaceElevator elevator, boolean[] schedule)`

For specific implementation details, see the `SpaceElevator` class in the repository.

The elevator is built to only accept 1 type of alternate fuel so there is no choice of which fuel to use, only when to use it. Recall that d is the distance that the elevator must cover to reach the top.

Tasks:

- (40 points) Implement Method 2 to compute the cost of a given schedule for an elevator.
  - This part of the assignment can be used to recover up to 60% of the points lost on PA1. If you score p_1 points on this part, your PA1 grade will be:

$$max(PA1Grade,\ \ 60 * (\frac{p_1}{40}) + (0.4 * PA1Grade))$$

- (40 points) Implement Method 1 to generate a schedule that minimizes the total cost if your alternate fuel is nuclear which has a cost of $c(d) = 100 * d$ each time it's used to push the elevator to the top. The runtime must be polynomial in n (hint: $O(n)$ time is possible.).
  - This part of the assignment can be used to recover up to 60% of the points lost on PA2. Specifically, if you score p_2 points on this part, your PA2 grade will be:

$$max(PA2Grade,\ \ 60 * (\frac{p_2}{40}) + (0.4 * PA2Grade))$$

- (10 points) Implement Method 1 to generate a schedule that minimizes the total cost if your alternate fuel is oil which has a cost of $c(d) = 4 * d^2$ each time it's used to push the elevator to the top. The runtime must be polynomial in n (hint: $O(n^2)$ time is possible.).
- (10 points) Implement Method 1 to generate a schedule that minimizes the total cost if your alternate fuel is steam power which has a cost of $c(d) = 50 * d^3$ each time it's used to push the elevator to the top. The runtime must be polynomial in n (hint: $O(n^2)$ time is possible.).

2 Submission

Submit the file `Assignment4.java` which contains your implementation of Method 1 and Method 2. You may add additional methods or classes as long as they are all contained in the `Assignment4.java` file. All other files and classes in the package will be used exactly as they are given for grading and should not be modified.

The preferred method for submission is the `submit_cse331` command. Emailed submissions will also be accepted.