CS2010 PS2 - Scheduling Deliveries v2 (with R-option)

Released: Thursday, 06 September 2012
Due: Saturday, 15 September 2012, 8am

Collaboration Policy. You are encouraged to work with other students or teaching staffs (inside or outside this module) on solving this problem set. However, you **must** write the Java code **by yourself**. In addition, when you write your Java code, you **must** list the names of every collaborator, that is, every other person that you talked to about the problem (even if you only discussed it briefly). This list may include certain posts from fellow students in CS2010 IVLE discussion forum. Any deviation from this policy will be considered cheating, and will be punished severely, including referral to the NUS Board of Discipline. It is not worth it to cheat just to get 15% when you will lose out in the other 85%.

R-option. This PS has R-option at the back. This additional task usually requires understanding of data structure or algorithm *beyond* CS2010. A self research on those relevant additional topics will be needed but some pointers will be given. CS2010R students *have to* attempt this R-option. CS2010 students can choose to attempt this R-option too for higher points, or simply leave it.

Last Year's Story. June 2011 was part of the long NUS 'University holiday' period last year. Although NUS teaching staffs are not actually on holiday during University holiday[1], June 2011 was considered a lighter month for Steven. Therefore, Steven and his wife, Grace, decided to enroll in birth classes at a certain hospital in Singapore (name omitted to avoid indirect advertising).

During one of the session, we were escorted to do a 'hospital tour' (this is the topic of the next PS3). One of the room type that is shown during the hospital tour is the 'delivery suites'. There are several rooms that are designed for women to deliver their babies. Those rooms have special delivery bed, pain relief systems: laughing gas 'Enthonox', epidural anaesthesia, and lots of other stuffs to make the mother-to-be as comfortable as possible.

Steven spotted a big white board in the reception desk and saw several pregnant women names but only one obstetrician[2]/gynaecologist[3] (let's just called him/her 'the doctor') on duty at that time. So Steven is quite sure that the doctor must be praying that not all pregnant women gave birth at the same time[4]. In such situation, the doctor must decide which one of these women to give more attention based on their 'dilation status' (see below).

In the 'three stages of labor' (this is the topic of that will be discussed in more details in PS5), the longest part is stage one: Dilation. Let's omit the detailed description and let's just say that dilation is measured in centimeters: Usually starts from about 3.0 cm (the start of labor) to around 10.0 cm (the baby is ready to be 'pushed' out/born). But for this problem, we will use millimeters so that we can work with integers, i.e. 10.0 cm = 100 mm, 3.0 cm = 30 mm. Anyway, if you are interested, please read this article: `http://en.wikipedia.org/wiki/Childbirth`.

[1] They have to prepare for the upcoming semester, or to continue with their research duties, or for Steven's case – prepare the Singapore International Olympiad in Informatics (IOI) team.

[2] Medical specialty dealing with the care of all women's reproductive tracts and their children during pregnancy (prenatal period), childbirth, and the postnatal period.

[3] Medical practice dealing with the health of the female reproductive system, or simply said "the science of women".

[4] In practice, each woman already have her own private doctor standby so the situation described in PS2 is really overly exaggerated.

The Actual Problem. Given N pregnant women names that are about to give birth and their initial dilation (in millimeters), determine which woman that the only doctor on duty has to give his/her most attention to. A woman with higher dilation status has higher priority. If there are more than one woman with the same highest dilation status, this only doctor will give priority to the woman who arrived at the hospital earlier.

The skeleton program `SchedulingDeliveries.java` is already written for you, you just need to implement four more methods/functions:

- `void ArriveAtHospital(String womanName, int dilation)`
  Insert this `womanName` and her initial `dilation` upon arrival at hospital into a suitable data structure of your choice. `womanName` is a String that contains only uppercase alphabets with length up to 15 characters. The women names are all unique. `dilation` is an integer.

- `void UpdateDilation(String womanName, int increaseDilation)`
  Medically, `dilation` can only go up to around `dilation = 100` millimeters. However, to simplify this problem, we will not bother with the maximum value. This method will simply update the dilation status of `womanName` from `dilation` to `dilation + increaseDilation` even if this causes `dilation` $>$ `100` millimeters. You can safely assume that the method: `ArriveAtHospital(womanName, any positive value` $\geq$ `30)` is already called prior to calling this method. You can also assume that `increaseDilation` is always nonnegative.

- `void GiveBirth(String womanName)`
  Medically, it takes several minutes or even hours from dilation around 100 millimeters until the baby is actually born. Some mothers can actually deliver the baby even if her dilation is still less than 100 millimeters (but not an ideal situation). Again, to simplify this problem, we assume that upon calling this method, the `womanName` gives birth in 'that instant' and no longer need to be taken care by the only doctor on duty.

- `String Query()`
  Query your data structure and reports the name of the woman that the only doctor on duty has to give the most attention to. See the priority criteria defined above. If there is no more woman to be taken care of, return a String: "The delivery suite is empty".

Example:
Let the chronological sequence of 15 events are as follows:

1. `ArriveAtHospital(''GRACE'', 31)`
2. `ArriveAtHospital(''ASTRID'', 55)`
3. `ArriveAtHospital(''MARIA'', 42)`
4. `Query()`
   You have to print out "ASTRID", as she is currently the one with highest `dilation`.
   To be precise, at the moment the order is: (ASTRID, 55), (MARIA, 42), and (GRACE, 31).
5. `ArriveAtHospital(''CINDY'', 77)`
6. `Query()`
   Now you have to print out "CINDY".
   The current order is: (CINDY, 77), (ASTRID, 55), (MARIA, 42), and (GRACE, 31).
7. `UpdateDilation(''GRACE'', 24)`
   You still have to print out "CINDY" because "GRACE" now has `dilation = 31 + 24 = 55`, 22 millimeters smaller than "CINDY". Note that "ASTRID" also has `dilation = 55` but "GRACE" is in front of "ASTRID" because "GRACE" arrived at the hospital earlier.
   The current order is: (CINDY, 77), (GRACE, 55), (ASTRID, 55), and (MARIA, 42).

8. `GiveBirth(‘‘CINDY’’)`
   “CINDY” now gives birth ‘instantly’, and she is no longer in the doctor’s radar.
9. `Query()`
   Now you have to print out “GRACE”, as the current order is: (GRACE, 55), (ASTRID, 55), and (MARIA, 42).
10. `GiveBirth(‘‘MARIA’’)`
    Suddenly “MARIA” reaches dilation 100 millimeters and gives birth instantly.
11. `Query()`
    The answer is still: “GRACE”.
    The current order is: (GRACE, 55) and (ASTRID, 55).
12. `GiveBirth(‘‘GRACE’’)`
13. `Query()`
    You have to answer: “ASTRID”.
    The current order is: (ASTRID, 55).
14. `GiveBirth(‘‘ASTRID’’)`
15. `Query()`
    “The delivery suite is empty”.

Subtask 1 (50 points). In this subtask, there is no call to `UpdateDilation` method. The method `GiveBirth` is always called for the woman currently under the doctor’s highest priority (you can view this as `GiveBirth(Query())`. For this simpler Subtask 1, you can assume that `dilation` value when method `ArriveAtHospital(womanName, dilation)` is called is always an integer between `[30..100]` millimeters. The number of women involved in this subtask is ≤ 10.

Subtask 2 (Additional 25 points). Unlike Subtask 1, this time you *also* have to deal with `UpdateDilation` method and women `GiveBirth` in *any* order as shown in the example above. However, the number of women involved in this subtask is still ≤ 10.

Subtask 3 (Additional 25 points). Your program must be able to solve Subtask 2 and use some form of the efficient Binary Heap data structure discussed in lecture (or use Java `PriorityQueue`) in order to handle up to ≤ 20000 women arriving in the hospital and frequently updating their dilation status (‘impossible’ in real life). Hint: All four methods must be at most $O(\log n)$.

Note: The test data to reach 75 points: `Subtask1.txt` and `Subtask2.txt` are given to you. You are allowed to check your program’s output with your friend’s. You are encouraged to generate and post additional test data in IVLE discussion forum.

R-option (Additional 10 bonus points for CS2010 students or just 100 points for CS2010R students). Solve **Subtask 3** above **without** using any form of Binary Heap data structure, i.e. you cannot use Java `PriorityQueue` and you cannot write your own Binary Heap. Other data structures outside Binary Heap are allowed.

The requirement for this R-option is *contradictory* with the requirement for **Subtask 3**. Normal CS2010 students who want to get 10 bonus point have to submit `SchedulingDeliveriesR.java` **on top of** their `SchedulingDeliveries.java`. If both codes solve **Subtask 3**, those normal CS2010 students can gain the maximum of 110 points.

For CS2010R students (especially those who do not take the normal CS2010): Since your PS1R is already (very) heavy, you only need to submit one `SchedulingDeliveriesR.java` only (actually simpler than the requirement of **Subtask 3**) to get 100 points. However, if you still keen to get 110 points, you will need to submit two codes.