CodeCrunch

CS2010 2016/2017 Sem2: PS1 A

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Emergency Room, v2017 (Subtask A)

Released: Thursday, 19th Jaunary 2017, 12:00 noon
Due: Thursday, 2nd February 2017, 11:59 pm

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 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 in CS2010 Facebook group. If you have access to CS2010 files from senior batch (that is, CS2010 problem sets from year 2011-2016), please refrain from looking at their code verbatim or worse... submit your senior's code. Automatic special checks are done especially on the last/hardest subtask to compare older code with this year's version. Any deviation from this policy will be considered as cheating. If the offender is caught beyond reasonable doubt, he/she will be punished severely, including referral to the NUS Board of Discipline.

Prelude

Due to a hereditary disease your lecturer Ket Fah was born with called Hemophilia, he is quite acquainted with the Emergency room at the hospital and other situations related to his disability. This PS and most of the following PS will be real life problems adapted (not 100% accurate) from these situations. Hopefully this will help CS2010 students come away with a better understanding of the situation of handicapped individuals.

Story

During his many visits to the emergency room at the hospital, Ket Fah will have to wait for the doctor in charge or the housemen working during that particular shift to attend to him. Sometimes the wait is long and sometimes it can be short. After a while he observed that before being sent into the emergency room, the nurse will actually make a preliminary assessment and determine the "emergency level" of the case (a colored sticker representing the level is then pasted on the patient). If the level is determined to be low, then other more urgent cases will be attended to first. If the level is determined to be high, then the case will be given a higher priority. This had to be done due to the shortage of doctors and the high number of cases that can occur at any given time.

Roughly the emergency levels can be classified from level 30 to 100 (made up !), where 100 is highest emergency level and 30 is the lowest. After having their initial assessment, the patient will be wheeled into the emergency and hooked up to machines that will monitor their heart rate etc. Periodically, the nurses will check on the patient to update their emergency level. For example a patient coming in with complaints of chest discomfort might initially have an emergency level of 50, but when he suddenly has difficulties in breathing, his level might be bumped up to 100.

After a patient has been seen by the doctor, he/she will be either warded in the hospital or given treatment and allowed to go home. In either case, they will be removed from the emergency room.

The Actual Problem Description

Given the names of **N** patients admitted into the emergency room, their initial emergency level, and subsequent updates of their emergency level, determine which patient the only doctor on duty (we assume here there is only one) has to give his/her most attention to.

A patient with higher emergency level will have higher priority. If there are more than one patient with the same highest emergency level, this only doctor will give priority to the patient who arrived at the hospital earlier.

The skeleton program EmergencyRoom.java (click to view) that can handle all input/output details is already written for you.

You just need to implement four more methods/functions:

1. `void ArriveAtHospital(String patientName, int emergencyLvl)`
   Insert this `patientName` and his/her initial `emergency level` (`emergencyLvl`) upon arrival at hospital into a suitable data structure of your choice.
   `patientName` is a String that contains only uppercase alphabets with length between 1 to 15 characters.
   The patient names are all unique.
   `emergency level` is an integer between `[30..100]`.
2. `void UpdateEmergencyLvl(String patientName, int incEmergencyLvl)`
   `emergency level` can only go up to `emergencyLvl = 100` and our test data will ensure that this method will not cause a `patientName` to have `emergency level` greater than 100.
   What we guarantee is that `incEmergencyLvl` is an integer between `[0..70]` and before calling this method, `patientName` has arrived at the hospital.
3. `void Treat(String patientName)`
   Upon calling this method, we assume the `patientName` will have been treated by the doctore and no longer need to be in the emergency room.
   We guarantee that before calling this method, `patientName` has arrived at the hospital.
4. `String Query()` Query your data structure and report the name of the patient that the only doctor on duty has to give the most attention to.
   See the priority criteria defined above.
   If there is no more patient to be taken care of, return a String: "The emergency room is empty".

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

1. `ArriveAtHospital("GRACE", 31)`
2. `ArriveAtHospital("THOMAS", 55)`
3. `ArriveAtHospital("MARIA", 42)`
4. `Query()`
   You have to print out `"THOMAS"`, as he is currently the one with the highest emergency level.
   To be precise, at the moment the order is: `(THOMAS, 55), (MARIA, 42), (GRACE, 31)`.
5. `ArriveAtHospital("CINDY", 77)`
6. `Query()`
   Now you have to print out `"CINDY"`.
   The current order is: `(CINDY, 77), (THOMAS, 55), (MARIA, 42), (GRACE, 31)`.
7. `UpdateEmergencyLvl("GRACE", 24)`
   After this event, the one with the highest emergency is still `CINDY` with `emergencyLvl = 77`.
   `"GRACE"` now has `emergencyLvl = 31+24 = 55`, but this is still smaller than `"CINDY"`.
   Note that `"THOMAS"` also has `emergencyLvl = 55` but `"GRACE"` is in front of `"THOMAS"` because `"GRACE"` arrived at the hospital earlier.
   The current order is: `(CINDY, 77), (GRACE, 55), (THOMAS, 55), (MARIA, 42)`.
8. `Treat("CINDY")`
   `"CINDY"` is now treated by the doctor 'instantly', and she will be removed from the emergency room.
9. `Query()`
   Now you have to print out `"GRACE"`, as the current order is: `(GRACE, 55), (THOMAS, 55), (MARIA, 42)`.
10. `Treat("MARIA")`
    For whatever reasons `"MARIA"` suddenly reaches an emergency level of 100 and is treated by the doctor.
    She will now be removed from the emergency room.
11. `Query()`
    The answer is still: `"GRACE"`.
    The current order is: `(GRACE, 55), (THOMAS, 55)`.
12. `Treat("GRACE")`
13. `Query()`
    You have to answer: `"THOMAS"`.
    The current order is: `(THOMAS, 55)`.
14. `Treat("THOMAS")`
15. `Query()`
    You have to answer: `"The emergency room is empty"`.

Subtask A Constraints (35 points)

Time Limit: 1s.

We make things very easy for you: The number of patientes involved in this Subtask A is at least 1 and at most 10.

The Sample Input/Output below is exactly the same as the illustration above. Notice that there are only 4 different patient involved in this sample test case. In our actual test case, we have a few more patients with various `ArriveAtHospital/UpdateEmergencyLvl/Treat/Query` combinations.

Sample Input

```
15
0 GRACE 31
0 THOMAS 55
0 MARIA 42
3
0 CINDY 77
3
1 GRACE 24
2 CINDY
3
2 MARIA
3
2 GRACE
3
2 THOMAS
3
```

Sample Output

```
THOMAS
CINDY
GRACE
GRACE
THOMAS
The emergency room is empty
```

Generating Test Data

The given sample input/output are for illustration purpose and are not enough to verify the correctness of your solution.

You are encouraged to generate and post additional test data in CS2010 Facebook group.

Please use EmergencyRoomVerifier.java (click to view) to verify whether your custom-made test data conform with the required specifications.

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010/R.

Submission (Course)

Select course: CS2010 (2016/2017 Sem 2) - Data Structures and Algorithms II ▼

Your Files:

SUBMIT (only .java, .c, .cpp and .h extensions allowed)

To submit multiple files, click on the Browse button, then select one or more files. The selected file(s) will be added to the upload queue. You can repeat this step to add more files. Check that you have all the files needed for your submission. Then click on the Submit button to upload your submission.

CS2010 2016/2017 Sem2: PS1 B

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Emergency Room, v2017 (Subtask B)

The Actual Problem Description

Please refer to Subtask A for the full problem description.

Subtask B Constraints (additional 40 points)

Time Limit: 4s.

In our test data for Subtask B, we have up to 1 000 000 (that is, 1 Million) commands and up to 200 000 patients in the emergency room waiting to be treated.

But there are two simplifying assumptions in this Subtask B:

1. There is NO call to `UpdateEmergencyLvl(patientName, increaseEmergencyLvl)` method.
   If you observe the Sample Input below that is only valid for Subtask B, there is no command that starts with integer 1. Check the `run()` method in EmergencyRoom.java template code shown in Subtask A that integer 1 implies a call to `UpdateEmergencyLvl(patientName, increaseEmergencyLvl)` method.
2. The method `Treat(patientName)` is always called for the patient currently under the doctor's highest priority (you can view this as `Treat(Query())`).

Hint: **All three** methods (as you do not have to deal with `UpdateEmergencyLvl` yet) must be **sub-linear**, i.e. better than **O(N)**, for example: **O(1)**, **O(log N)**, **O(log^2 N)**, or **O(sqrt N)**.

Note: Automatic plagiarism detection is activated for this Subtask B.

Sample Input

```
14
0 GRACE 31
0 THOMAS 55
0 MARIA 42
3
0 CINDY 77
3
2 CINDY
3
2 THOMAS
3
2 MARIA
3
```

```
2 GRACE
3
```

Sample Output

```
THOMAS
CINDY
THOMAS
MARIA
GRACE
The emergency room is empty
```

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010/R.

Submission (Course)

Select course: CS2010 (2016/2017 Sem 2) - Data Structures and Algorithms II ▼

Your Files:

SUBMIT (only .java, .c, .cpp and .h extensions allowed)

To submit multiple files, click on the Browse button, then select one or more files. The selected file(s) will be added to the upload queue. You can repeat this step to add more files. Check that you have all the files needed for your submission. Then click on the Submit button to upload your submission.

CodeCrunch

CS2010 2016/2017 Sem2: PS1 C

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Emergency Room, v2017 (Subtask C)

The Actual Problem Description

Please refer to Subtask A for the full problem description.

Subtask C Constraints (additional 25 points)

Time Limit: 4s.

In our test data for Subtask C, we have up to 1 000 000 (that is, 1 Million) commands and up to 200 000 patients in the hospital waiting to be treated and frequently updating their emergency level.

Unlike in Subtask B but similar in Subtask A, the method `Treat(patientName)` can now be called for any `patientName` currently in the hospital.

Hint: **All four** methods must be **sub-linear**, i.e. better than **O(N)**, for example: **O(1)**, **O(log N)**, **O(log^2 N)**, or **O(sqrt N)**.

Notes:
1. Automatic plagiarism detection is activated for this Subtask C.
2. The Java source code limit for PS1 is 10 000 Bytes.

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010/R.

Submission (Course)

Select course: CS2010 (2016/2017 Sem 2) - Data Structures and Algorithms II

Your Files:

SUBMIT (only .java, .c, .cpp and .h extensions allowed)

To submit multiple files, click on the Browse button, then select one or more files. The selected file(s) will be added to the upload queue. You can repeat this step to add more files. Check that you have all the files needed for your submission. Then click on the Submit button to upload your submission.