(S2-19_DSECLZG519)
(Data Structures and Algorithms Design)
Academic Year 2019-2020

Assignment 1 – PS6 - [Testing Priority] - [Weightage 12%]

1. Problem Statement

At the government hospital treating patients for Covid, the management is preparing a token system that gives priority to senior citizens who want to get tested over other patients. As the patients keep coming and registering, they are added to a priority queue from which patient's names are called out for testing. Anticipating a rise in the number of cases across the world, an appointment software has to be developed to manage this priority effectively.

The application should be able to:

a. Take the patient's age and create a patient ID in the following format: <xxxxyy> where

   xxxx is the patient id and

   yy is the patient's age

b. Insert the patient id in the priority queue based on the age of the patient.
c. If x testing counters got vacant, display the next set of patient IDs and corresponding names that should go for testing and remove them from the priority queue.

Asks:

1. **Implement the above problem statement in Python 3.7 using Heaps.**
2. **Perform an analysis for the questions above and give the running time in terms of input size: n.**

Data Structures to be used:

PatientRecord: A list containing the patient information including the patient's name, age and the patient number (assigned by the program).

TestingQueue: A max heap containing the patient id sorted in order of next patient for testing based on the age of the patient.

Patient Record data structure:

```
class PatientRecord:
    def __init__(self, age, name, Pid):
        self.PatId = str(Pid) + str(age)
        self.name = name
        self.age = age
        self.left = None
        self.right = None
```

Functions:

1. **def registerPatient(self, name, age):** This function registers the name and age of the patient entering the hospital and assigns them an ID that is then used to capture the details of the patient in the Patient Record. When the program is executed for the first time, the patient details are loaded from an input file **inputPS6a.txt**. This is analogous to the list of patients present at the hospital before the hospital opens.

   Input PS6a format:

   Surya, 60

   Ajay, 54

   Rishi, 57

   After all records are read from the inputPS6a file, and the queue is sorted, the queue should be output to the file outputPS6.txt in the below format.

   ---- registerPatient ----------

   No of patients added: 3

   Refreshed queue:

   100160, Surya

   100357, Rishi

   100254, Ajay

   Thereafter, new patients will be input through another file **inputPS6b.txt** and identified with the tag **newPatient.**

   newPatient: John, 55

   After every new patient in the input PS6b is inserted into the patient record the heap will have to be refreshed and the new queue should be output to the file **outputPS6.txt** in the below format.

---- new patient entered----------

Patient details: John, 55, 100455

Refreshed queue:

100260, Surya

100357, Rishi

100455, John

100554, Ajay

---

2. **def enqueuePatient(self, PatId)**: This function assigns the patient a place in the max heap depending on their age. The patient id is inserted into the max heap. This function should be called every time a new patient is added and should run a sort after adding the patient id to keep the testing queue updated as per the age condition.

3. **def nextPatient(self):** This function prints the next x number of patient_IDs and their names that are next in line for testing. This function is called when the program encounters a next patient tag from the **inputPS6b.txt** file. The format of the next patient will be

   nextPatient: 1

   The function will read the x number of patient details that should be output. The output is pushed to the file **outputPS6.txt** in the below format.

   ---- next patient : 1 ----------

   Next patient for testing is: 100260, Surya

   ---

4. **def _dequeuePatient(self, PatId):** This function removes from the queue the patient ID that has completed the testing and updates the queue. The function is called from the nextPatient function itself after the next patients name is displayed.

5. Include all other functions required to support the operations of these basic functions.

2. Sample file formats

Sample Input file

The inputPS6a.txt file contains the first set of registrations.

Sample inputPS6a.txt

Surya, 60

Ajay, 54

Rishi, 57

Sample inputPS6b.txt

newPatient: John, 55

nextPatient: 1

newPatient: Pradeep, 45

nextPatient: 2

nextPatient: 1

newPatient: Sandeep, 60

nextPatient: 3

Sample outputPS6.txt

---- initial queue ----------

No of patients added: 3

Refreshed queue:

100160, Surya

100357, Rishi

100254, Ajay

---- new patient entered----------

Patient details: John, 55, 100455

Refreshed queue:

100260, Surya

100357, Rishi

100455, John

100554, Ajay

---- next patient: 1 ----------

Next patient for testing is: 100260, Surya

---

3. Deliverables

a. **A1_PS6_DC_[Group id] package folder** containing modules and package files for the entire program code and associated functions

b. **inputPS6a.txt** file used for testing

c. **inputPS6b.txt** file used for testing

d. **outputPS6.txt** file containing the output of the program

e. **analysisPS6.txt** file containing the running time analysis for the program.

4. Instructions

a. It is compulsory to make use of the data structure/s mentioned in the problem statement.

b. Do not use inbuilt data structures available in Python. The purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.

c. It is compulsory to use Python 3.7 for implementation.

d. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full.

e. For the purposes of testing, you may implement some functions to print the data structures or other test data. But all such functions must be commented before submission.

f. Make sure that your read, understand, and follow all the instructions

g. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.

h. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to test the submissions will be different. Hence, do not hard code any values into the code.

i. Run time analysis is provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

5. Deadline

a. The strict deadline for submission of the assignment is 18th June, 2020.

b. The deadline already accounts for extra days for the students to work on the assignment. No further extension of the deadline will be entertained.

c. Late submissions will not be evaluated.

6. How to submit

a. This is a group assignment.

b. Each group has to **make one submission** (only one, no resubmission) of solutions.

c. Each group should zip **all the deliverables** into one file and name the zipped file as below

d. “ASSIGNMENT1_[G1/G2/…].zip” and upload in CANVAS in respective location under ASSIGNMENT Tab.

e. Assignments submitted via means other than CANVAS **will not be graded**.

7. Evaluation

a. The assignment carries 12 Marks.

b. Grading will depend on

   a. Fully executable code with all functionality

   b. Well-structured and commented code

   c. Accuracy of the run time analysis

c. Every bug in the functionality will have negative marking.

d. Source code files which contain compilation errors will get at most 25% of the value of that question.

8. Readings

Section 2.4,2.6: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition)