(S1-19_DSECLZG519)
(Data Structures and Algorithms Design)
Academic Year 2019-2020

Assignment 1 – PS5 - [Doctor Consultation] - [Weightage 12%]

1. Problem Statement

In this Problem, you have to write an application in Python 3.7 that keeps track of incoming patients in a hospital.

Dr. Kumar runs a hospital for senior citizens which sees a large number of patients coming in everyday. In order to avoid inconvenience to the aged patients, the rule is that the oldest patient is seen first by the doctor. As the patients keep coming and registering, they are added to a priority list from which patient's names are called out for consultation. Since it is not practical to implement this system using pen and paper, an appointment software has to be developed. This software will use Heaps as data structures to keep track of incoming patients and prioritizing them.

The application should be capable to:

1. Take the patient's age and creates a patient ID in the following format: <xxxxyy> where
   xxxx is the patient id and
   yy is the patient's age
2. Insert the patient id in the priority list based on the age of the patient.
3. Display the next patient ID and name in line to meet the doctor and remove this patient from the priority list.
4. Perform an analysis of question number 2 and 3 and give the running time in terms of input size, n.

Data Structures to be used:

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

ConsultQueue: A max heap containing the patient id in the sequence of the next consultation 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 **inputPS5a.txt**. This is analogous to the list of patients present at the hospital before the doctor arrives.

   Input PS5a format:
   Surya, 60
   Ajay, 54
   Rishi, 57

   After all records are read from the inputPS5a file, and the queue is sorted, the queue should be output to the file outputPS5.txt in the below format.
   ---- initial queue ----------
   No of patients added: 3
   Refreshed queue:
   100160, Surya
   100357, Rishi
   100254, Ajay

   Thereafter, new patients will be input through another file **inputPS5b.txt** and identified with the tag **newPatient.**

   newPatient: John, 55

   After every new patient in the input PS5b is inserted into the patient record the heap will have to be refreshed and the new queue should be output to the file **output5.txt** in the below format.
   ---- new patient entered----------
   Patient details: John, 55, 100355
   Refreshed queue:
   100260, Surya
   100357, Rishi
   100455, John
   100254, 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 consultation queue updated as per the age condition.

3. **def nextPatient(self):** This function prints the patient_ID and name of the patient that is next in line to meet the doctor. This function is called when the program encounters a next patient call from the **promptsPS5.txt** file. The format of the next patient will be

   nextPatient

   The output of this function should be pushed to the file **outputPS5.txt** in the below format.

   ---- next patient ----------

   Next patient for consultation is: 100260, Surya

   ---

4. **def _dequeuePatient(self, PatId):** This function removes from the queue the patient ID that has consulted the doctor 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 inputPS5a.txt file contains the first set of registrations.

Sample inputPS5a.txt

Surya, 60

Ajay, 54

Rishi, 57

Sample inputPS5b.txt

newPatient: John, 55
nextPatient
newPatient: Pradeep, 45
nextPatient
nextPatient
newPatient: Sandeep, 60
nextPatient

Sample outputPS5.txt

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

No of patients added: 3

Refreshed queue:

100160, Surya

100357, Rishi

100254, Ajay

```
---------------------------------------------

---- new patient entered----------
Patient details: John, 55, 100355
Refreshed queue:
100260, Surya
100357, Rishi
100455, John
100254, Ajay
---------------------------------------------
---- next patient ----------
Next patient for consultation is: 100260, Surya
---------------------------------------------
```

3. Deliverables

a. **Zipped A1_PS5_DC_[Group id].py package folder** containing modules and package files for the entire program code and associated functions
b. **inputPS5a.txt** file used for testing
c. **inputPS5b.txt** file used for testing
d. **outputPS5.txt** file containing the output of the program
e. **analysisPS5.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. It is compulsory to use Python 3.7 for implementation.
c. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full.
d. 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.
e. Make sure that your read, understand, and follow all the instructions
f. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
g. 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.
h. 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 20th Dec, 2019.

b. 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 the deliverables and name the zipped file as below

d. “ASSIGNMENT1_[BLR/HYD/DLH/PUN/CHE]_[B1/B2/…]_[G1/G2/…].zip”

e. and upload in CANVAS in respective location under ASSIGNMENT Tab.

f. Assignment submitted via means other than through 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)