Work Integrated Learning Programmes Division
M.Tech (Data Science and Engineering)

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

Assignment 1 – PS11 - [Traffic Fines] - [Weightage 12%]

1. Problem Statement

In this Problem, you have to write an application in Python 3.7 that keeps track of traffic fines.

All violations are noted by a traffic policeman in a file as a record <license number of driver, fine amount>. At the end of each day, files from all traffic policemen are collated. If a driver had been charged with greater than five violations, then he has to be booked for further legal action. Also, the police department wants to take strict action on policemen who are not performing their duties. All policemen will have to book at least 10 violators in a given period to avoid further action.

The program should help the police department generate reports on the below queries:

1. Generate a list of license numbers which are repeat offenders.

2. Generate a list of policemen who are not meeting the monthly targets.

3. Top 10 policemen that have made the most bookings and total fine amount they have collected.

Additionally,

4. Perform an analysis of questions 1, 2 and 3 and give the running time in terms of input size, n.

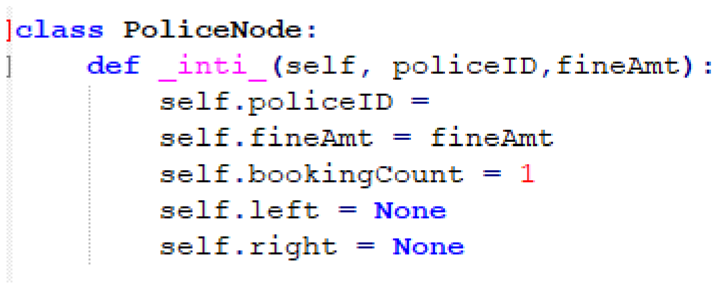
Use hash tables for keeping track of drivers (and their violations), and a binary tree for keeping track of policemen (and their bookings).

Data structures to be used:

DriverHash: A separately chained hash table indexed by license numbers where each entry is of the form < license number, number of violations>. A simple hash function h(x) = x mod M, where M is the size of hash table can be used for this.

PoliceTree: This is a Binary Tree of entries of the form <police ID, total fine amount>

The basic structure of the Police Node will be:



```
class PoliceNode:
    def _inti_(self, policeID,fineAmt):
        self.policeID =
        self.fineAmt = fineAmt
        self.bookingCount = 1
        self.left = None
        self.right = None
```

Functions:

1. **def initializeHash (self):** This function creates an empty hash table that points to null.

2. **def insertHash (driverhash, lic):** This function inserts the licence number lic to the hash table. If a driver's license number is already present, only the number of violations need to be updated else a new entry has to be created.

3. **def printViolators (driverhash):** This function prints the repeat violators by looking through all hash table entries and printing the license numbers of the drivers who have greater than and equal to 5 violations onto the file **violators.txt**. The output should be in the format

   --------------Violators-------------

   <license no>, no of violations

4. **def destroyHash (driverhash):** This function destroys all the entries inside the hash table. This is a clean-up code.

5. **def insertByPoliceId (policeRoot, policeId, amount):** This function inserts an entry <policeId, amount> into the police tree ordered by police id. If the Police id is already found in the tree, then this function adds up to the existing amount to get the total amount collected by him and increments the bookingCount by 1. This function returns the updated tree.

6. **def reorderPoliceTree (policeRoot)**: This function reorders the Binary Tree on the basis of total number of bookingCount. This function removes the nodes from the original PoliceTree, and puts it in a new tree ordered by bookingCount. Note that if the count in node i is equal to the count in node j, then the node i will be inserted to the left of the node j. This function returns the root node of the new tree.

7. **def printPolicemen (policeRoot):** This function prints the list of police ids which have not met the target of atleast 10 bookings. The output is pushed to a file called **police.txt**. The output will be in the format

   -------------- Police list -------------

   Police id, no of violations

8. **def printTopTen(policeRoot):** This function prints the list of top 10 police ids based on total number of bookings. The output is pushed to a file called **police.txt**. The output will be in the format

   -------------- Police Top 10 -------------

   Police id, no of bookings, total fine amount

9. **def destroyPoliceTree (policeRoot):** This function is a clean-up function that destroys all the nodes in the police tree.

10. **def printPoliceTree (policeRoot):** This function is meant for debugging purposes. This function prints the contents of the PoliceTree in-order.

2. Sample file formats

Sample Input file

Every row of the input file should contain the <police id> / <license number> / <fine amount> in the

same sequence. Save the input file as **inputPS11.txt**

Sample inputPS11.txt

```
111 / 1231114 / 100
111 / 1214413 / 200
222 / 1231412 / 100
222 / 1231114 / 100
333 / 1231114 / 100
```

Sample violators.txt

```
--------------Violators-------------
```

```
1231114, 3
```

Sample police.txt

```
--------------Police list-------------
111, 2
222, 2
-------------- Police Top 10 -------------
555, 15, 2100
```

3. Deliverables

a. **A1_PS11_TF_[Group id] package folder** containing modules and package files for the entire program code and associated functions
b. **inputPS11.txt** file used for testing
c. **police.txt** containing the list of policemen ids who have underperformed.
d. **violators.txt:** containing the list of driving licence numbers against which legal action is required.
e. **analysisPS11.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** and name the zipped file as below

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

e. Assignment 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 2.3,2.5,3.1: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition)