(S2-18_DSECFZG519)
(Data Structures and Algorithms Design)
Academic Year 2018-2019

Assignment 1 – PS3 - [Traffic Fines] - [Weightage 13%]

1. **Problem Statement**

In this Problem, you have to write an application in Python 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 more than three violations so far, then he has to be booked for further legal action. Also, the police department provides additional bonus to those policemen who have brought in large fine earnings. All policemen who have collected equal to or more than 90% of the highest total fine collected by an individual policeman, shall be awarded the bonus.

The program should help the police department answer the below queries:

1. Find out the drivers who are booked for legal action: All such license numbers are to be output in a file called "violators.txt".

2. Find out the policemen who are eligible for bonus: The list of policemen eligible for bonus must be output in a file called "bonus.txt".

Additionally,

3. **Perform an analysis of questions 1 and 2 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:

```python
class PoliceNode:
    def __init__ (self, policeId, fineAmt):
        self.policeId = policeId
        self.fineAmt = fineAmt
        self.left = null
        self.right = null
```

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 serious violators by looking through all hash table entries and printing the license numbers of the drivers who have more than 3 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. This function returns the updated tree.

6. **def reorderByFineAmount (policeRoot)**: This function reorders the Binary Tree on the basis of total fine amount, instead of police id. This function removes the nodes from the original PoliceTree, and puts it in a new tree ordered by fine amount. Note that if the fine amount in node i is equal to the amount 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 printBonusPolicemen (policeRoot):** This function prints the list of police ids which have earned equal to or more than 90% of maximum total fine amount collected by an individual. The output is pushed to a file called **bonus.txt**. The output will be in the format

   -------------- Bonus -------------

   <license no>, no of violations

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

9. **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 **inputPS3.txt**

Sample inputPS3.txt

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

222 / 1231114 / 100
333 / 1231114 / 100

Sample violators.txt

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

1231114, 3

Sample bonus.txt

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

111, 300

222, 100

3. **Deliverables**

    a. **Zipped PS3_TF_[Group id].py package folder** containing modules and package files for the entire program code and associated functions

    b. **inputPS3.txt** file used for testing

    c. **bonus.txt** containing the list of policemen ids who are eligible for a bonus.

    d. **violators.txt:** containing the list of driving licence numbers against which legal action is required.

    e. **analysisPS3.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 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. 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 6th July, 2019.

    b. The deadline is set for a month from the date of rollout to accommodate for the semester exams. 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

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

        and upload in CANVAS in respective location under ASSIGNMENT Tab.

    d. Assignment submitted via means other than through CANVAS will not be graded.

7. Evaluation

    a. The assignment carries 13 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)