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

Assignment 1 – PS1 - [Attendance Problem] - [Weightage 12%]

1. Problem Statement

In an attendance monitoring system, the unique integer ID number of every employee is noted whenever the employee swipes his card in the office. First time an employee enters into the office, the attendance counter is set to 1. From then onwards, each time an employee leaves the office premises for tea break or lunch break the counter is incremented, and incremented again when he enters back. If the counter is odd on a day, it means the employee is in the office premises and if the counter is even, it means he is out of the premises.

The organization uses this data to perform the following analytics:

1. How many employees came today?
2. Did a particular employee come today?
3. How often did an employee enter into the office?
4. Which employee moves out of office the greatest number of times?
5. Which employee ids within a range of IDs attended office, the attendance counter for them, and whether they are inside or outside office?

Asks:

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

The basic structure of the employee node will be:

```
class EmpNode:
    def __init__(self, EId):
        self.EmpId = EId
        self.attCtr = 1
        self.left = None
        self.right = None
```

Operations:

1. **def _readEmployeesRec(self, eNode, Eid):** This function reads from the **inputPS1.txt** file the ids of employees entering and leaving the organization premises. One employee id should be populated per line (in the input text file) indicating their swipe (entry or exit). The input data is used to populate the tree. If the employee id is already added to the tree, then the attendance counter is incremented for every subsequent occurrence of that employee id in the input file. Use a trigger function to call this recursive function from the root node.

2. **Def _getHeadcountRec(self, eNode):** This function counts the number of unique IDs stored in the tree and prints the employee headcount for the day into the **output.txt** file as shown below.
   Total number of employees today: ***xx***
   Use a trigger function to call this recursive function from the root node.

3. **def _searchIDRec(self, eNode, eId):** This function searches whether a particular employee has attended today or not. The function reads the search id from the file **promptsPS1.txt** where the search id is mentioned with the tag as shown below.
   searchID:23
   searchID:22
   searchID:11
   The search function is called for every searchID tag the program finds in the promptsPS1.txt file.
   If the employee id is found it outputs the below string into the **outputPS1.txt** file
   Employee id ***xx*** is present today.
   If the employee id is not found it outputs the below string into the **outputPS1.txt** file
   Employee id ***xx*** is absent today.
   Use a trigger function to call this recursive function from the root node.

4. **def _howOften_Rec(self, eNode, EId):** This function counts the number of times a particular employee swiped today and if the employee is currently in the office or outside.
   The function reads the id from the file **promptsPS1.txt** where the search id is mentioned with the tag as shown below.
   howOften:12
   howOften:22
   howOften:11
   The search function is called for every howOften tag the program finds in the promptsPS1.txt file.
   If the employee id is found with an odd attendance count the below string is output into the **outputPS1.txt** file
   Employee id ***xx*** swiped **yy** times today and is currently in office

If the employee id is found with an even attendance count the below string is output into the **outputPS1.txt** file

Employee id ***xx*** swiped **yy** times today and is currently outside office

If the employee id is not found it outputs the below string into the **outputPS1.txt** file

Employee id ***xx*** did not swipe today.

Use a trigger function to call this recursive function from the root node.

5. **def _frequentVisitorRec(self, eNode):** This function searches for the employee who has swiped the most number of times and outputs the below string into the **outputPS1.txt** file.

   Employee id ***xx*** swiped the most number of times today with a count of ***yy***

   Use a trigger function to call this recursive function from the root node. For the sake of the assignment, you need to display any one of the employee ids if there are more than one employee who have entered maximum number of times. For example, if employee id 22 and 23 have both visited 3 times, the output should show either 22 or 23.

6. **def printRangePresent(self, StartId, EndId):** This function prints the employee ids in the range **StartId** to **EndId** and how often they have entered the organization in a file name **outputPS1.txt**. The input should be read from the **promptsPS1.txt** file where the range is mentioned with the tag as shown below.

   range:23:125

   If Input range is given as 23 to 125 the output file should show:

   Range: 23 to 125
   Employee attendance:
   23, 1, in
   41, 3, in
   121, 2, out

   For this purpose, the tree needs to be **inorder traversed** and the id and frequency of the employees in the range must be printed into the file. If the Id is found in the BT, its frequency cannot be zero as the person had entered the organization at least once.

2. Sample file formats

Sample Input file

Each row will have one employee id indicating a single swipe for that employee. The input file name must be called **inputPS1.txt**.

Sample inputPS1.txt

23
22
41
121
41
22

41
121

Sample promptsPS1.txt

searchID:22
searchID:120
howOften:41
howOften:12
range:23:125

Sample outputPS1.txt

Total number of employees today: 4
Employee id 22 is present today.
Employee id 120 is absent today.
Employee id 41 swiped 3 times today and is currently in office
Employee id 12 did not swipe today.
Employee id 41 swiped the most number of times today with a count of 3
Range: 23 to 125
Employee attendance:
23, 1, in
41, 3, in
121, 2, out

2. Deliverables

a. **Zipped A1_PS1_EA_[Group id].py package folder** containing all the modules classes and functions and the main body of the program.
b. **inputPS1.txt** file used for testing
c. **promptsPS1.txt** file used for testing
d. **outputPS1.txt** file generated while testing
e. **analysisPS1.txt** file with answers on the running times (ask number 2)

3. 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 these assignments are that you lean how these data structures are constructed and how do 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.

4. Deadline

a. The strict deadline for submission of the assignment is 20th Dec, 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.

5. 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.

6. 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.

7. Readings

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