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

(S1-20_DSECFZG519)

(Data Structures and Algorithms Design)

Academic Year 2020-2021

Assignment 1 – PS12 – [Bio Bubble] - [Weightage 12%]

1. Problem Statement

Given the current pandemic, the IPL organizing committee intends to use the unique player ID and smart card to identify the whereabouts of their players within the secure Bio Bubble setup around the hotel and detect if there have been any breaches. Whenever the player swipes his card in the hotel premises the player ID is recorded. When a player enters into the hotel for the first time, the counter is set to 1. From then onwards, each time a player swipes out of the hotel premises the counter is incremented, and incremented again when he enters back. If the counter is odd, it means the player is in the hotel premises and if the counter is even, it means he is out of the premises.

The IPL organizing committee uses this data to perform the following analytics:

1. How many players are inside the bio-bubble setup today?
2. Number of players that have come today from net practise and are currently in hotel?
3. Check specific player whereabouts?
4. List of players that have swiped (in or out) more than x number of times?
5. Which player ids within a range of IDs attended net practise, the swipe counter for them, and whether they are inside or outside hotel?

Requirements:

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 player node will be:

```
Class PlayerNode:
        def __init__(self, Pid):
                self.PId = Pid
```

self.attrCtr = 1

self.left = None

self.right = None

Operations:

1. **def _recordSwipeRec(self, pNode, Pid):** This function reads from the **inputPS12.txt** file the ids of players entering and leaving the hotel premises. One player 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 player id is already added to the tree, then the swipe counter is incremented for every subsequent occurrence of that player id in the input file. Use a trigger function to call this recursive function from the root node.

2. **def _getSwipeRec(self, pNode):** This function counts the total number of players who came to hotel today. The output is entered in the **outputPS12.txt** file as shown below.
   Total number of players recorded today: ***xx***
   Use a trigger function to call this recursive function from the root node.

3. **def _onPremisesRec(self, pNode, PId):** This function searches for all players that are still on the premises and outputs the total count of such players. The function is triggered with the following tag from the file **promptsPS12.txt.**
   onPremises:
   If there are players found who are on the premises the below string is entered into the **outputPS12.txt** file
   xx players still on premises.
   If there are no players on premises, the below string into the **outputPS12.txt** file
   No players present on premises.
   Use a trigger function to call this recursive function from the root node.

4. **def _checkEmpRec(self, pNode, EId):** This function counts the number of times a particular player swiped today and if the player is currently in the hotel or outside.
   The function reads the player id from the file **promptsPS12.txt** where the search id is mentioned with the tag as shown below.
   checkPlay:12
   checkPlay:22
   The search function is called for every checkPlay tag the program finds in the promptsPS12.txt file.
   If the player id is found with an odd swipe count the below string is output into the **outputPS12.txt** file

Player id ***xx*** swiped **yy** times today and is currently in hotel

If the player id is found with an even swipe count the below string is output into the **outputPS12.txt** file

Player id ***xx*** swiped **yy** times today and is currently outside hotel

If the player id is not found it outputs the below string into the **outputPS12.txt** file

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

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

5. **def _frequentVisitorRec(self, pNode, frequency):** This function generates the list of players who have swiped more than x number of times. The function reads the x number from the file **promptsPS12.txt** with the tag as shown below.

   freqVisit: 5

   The function outputs the below string into the **outputPS12.txt** file.

   Players that swiped more than xx number of times today are:

   Player id, count.

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

6. **def printRangePresent(self, StartId, EndId):** This function prints the player ids in the range **StartId** to **EndId** and how often they have swiped and if they are inside or outside the hotel into the **outputPS12.txt** file.

   The input should be read from the **promptsPS12.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

   Player swipe:

   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 players 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 hotel at least once.

Sample Files:

Input will be taken from the file(**inputPS12.txt**).

Each row will have one player id indicating a single swipe for that player.

Sample inputPS12.txt

```
23
22
41
121
41
22
41
121
```

Sample promptsPS12.txt

```
onPremises:
checkPlay: 12
checkPlay: 22
freqVisit: 3
range: 23:125
```

Sample outputPS12.txt

```
Total number of players recorded today: 4
2 players still on premises.
Player id 12 did not swipe today.
Player id 22 swiped 2 times today and is currently outside hotel
Players that swiped more than 3 number of times today are:
41,3
Range: 23 to 125
Player swipe:
23, 1, in
41, 3, in
121, 2, out
```

Note that the input/output data shown here is only for understanding and testing, the actual file used for evaluation will be different.

2. Deliverables

1. Word document **designPS12_<group id>.docx** detailing your design and time complexity of the algorithm.
2. **[Group id]_Contribution.xlsx** mentioning the contribution of each student in terms of percentage of work done. Download the Contribution.xlsx template from the link shared in the Assignment Announcement.
3. **inputPS12.txt** file used for testing
4. **promptsPS12.txt** file used for testing
5. **outputPS12.txt** file generated while testing
6. **.py file** containing the python code. Create a single *.py file for code. Do not fragment your code into multiple files

Zip all of the above files including the design document and contribution file in a folder with the name:

[Group id] _A1_PS12_BioBubble.zip and submit the zipped file.

Group Id should be given as **Gxxx** where xxx is your group number. For example, if your group is 26, then you will enter G026 as your group id.

3. Instructions

1. It is compulsory to make use of the data structure(s) / algorithms mentioned in the problem statement.
2. Ensure that all data structure insert and delete operations throw appropriate messages when their capacity is empty or full. Also ensure basic error handling is implemented.
3. 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.
4. Make sure that your read, understand, and follow all the instructions
5. Ensure that the input, prompt and output file guidelines are adhered to. Deviations from the mentioned formats will not be entertained.
6. The input, prompt and output samples shown here are only a representation of the syntax to be used. Actual files used to evaluate the submissions will be different. Hence, do not hard code any values into the code.
7. Run time analysis is to be provided in asymptotic notations and not timestamp based runtimes in sec or milliseconds.

Instructions for use of Python:

1. Implement the above problem statement using Python 3.7.
2. Use only native data types like lists and tuples in Python, do not use dictionaries provided in Python. Use of external libraries like graph, numpy, pandas library etc. is not allowed. The

purpose of the assignment is for you to learn how these data structures are constructed and how they work internally.

3. Create a single *.py file for code. Do not fragment your code into multiple files.
4. Do not submit a Jupyter Notebook (no *.ipynb). These submissions will not be evaluated.
5. Read the input file and create the output file in the root folder itself along with your .py file. Do not create separate folders for input and output files.

4. Deadline

1. The strict deadline for submission of the assignment is **27th Dec, 2020.**
2. The deadline has been set considering extra days from the regular duration in order to accommodate any challenges you might face. No further extensions will be entertained.
3. Late submissions will not be evaluated.

5. How to submit

1. This is a group assignment.
2. Each group has to make one submission (only one, no resubmission) of solutions.
3. Each group should zip all the deliverables in one zip file and name the zipped file as mentioned above.
4. Assignments should be submitted via Canvas > Assignment section. Assignment submitted via other means like email etc. will not be graded.

6. Evaluation

1. The assignment carries 12 Marks.
2. Grading will depend on
    a. Fully executable code with all functionality working as expected
    b. Well-structured and commented code
    c. Accuracy of the run time analysis and design document.
3. Every bug in the functionality will have negative marking.
4. Marks will be deducted if your program fails to read the input file used for evaluation due to change / deviation from the required syntax.
5. Use of only native data types and avoiding libraries like numpy, graph and pandas will get additional marks.
6. Plagiarism will not be tolerated. If two different groups submit the same code, both teams will get zero marks.
7. Source code files which contain compilation errors will get at most 25% of the value of that question.

7. Readings

Text book: Algorithms Design: Foundations, Analysis and Internet Examples Michael T. Goodrich, Roberto Tamassia, 2006, Wiley (Students Edition). **Chapters:** 2.3, 3.1