Work Integrated Learning Programmes Division
M.Tech (Software Engineering)

(S2-21_SESAPZG519)

(Data Structures and Algorithms Design)

Academic Year 2021-2022

Assignment 1 – PS4 - [Library Books] - [Weightage 12%]

1. Problem Statement

At the school library, there is a need for a new library management system. One of the basic requirements of the system is to keep track of which books are in the library and which ones have been issued / checked-out by students.

In order to maintain this information, each book is given a unique identification number and there can be multiple copies of the same book. Every time a book is checked-out by a student, the count of available copies is reduced by one and the checkout counter is incremented. When the student returns the book, the count of available copies is incremented for that book.

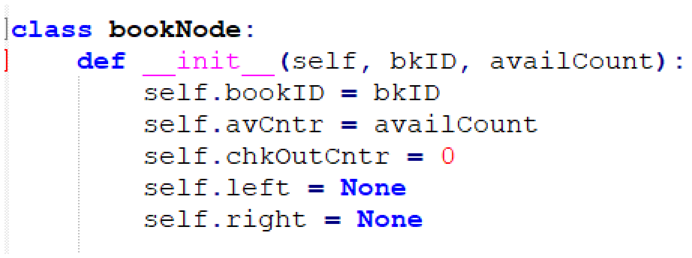
Using the above management system, you are required to help the school answer the below questions:

1. Which three books have been issued the greatest number of times?
2. Which books do not have any more copies left in the library (all copies have been issued)?
3. Find out if a particular book is available in the library based on the book id.
4. Out of the entire set of books in the library, list the books that have never been checked out by students?
5. Print a list of all the books and available copies.

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 book node will be:



```
class bookNode:
    def __init__(self, bkID, availCount):
        self.bookID = bkID
        self.avCntr = availCount
        self.chkOutCntr = 0
        self.left = None
        self.right = None
```

Operations:

1. **def_readBookList(self, bkID, availCount):** This function reads the book ids and the number of copies available from the **inputPS4.txt** file. One book id should be populated per line (in the input text file) along with the number of copies separated by a comma. The input data is used to populate the tree. Use a trigger function to call this recursive function from the root node.

   Sample Input

   100001, 4

   100002, 5

   100003, 15

   100004, 12

   100005, 3

   ....

   100020, 2

2. **def_chkInChkOut(self, bkID, inOut):** This function updates the check in / check out status of a book based on the book id. The function reads the input from the **promptsPS4.txt** file. The below sample lines should be part of the **promptsPS4** file to indicate if a book is checked in or checked out. If a book is checked out, the available count is reduced by one and the checkout counter is incremented by one. If a book is checked in, the available counter is incremented and the checkout counter is not altered.

   Sample promptsPS4.txt entry

   checkOut: 100001

   checkOut: 100002

   checkOut: 100003

   checkOut: 100003

   checkOut: 100001

   checkOut: 100001

checkOut: 100001

checkOut: 100005

checkIn: 100001

checkIn: 100002

checkOut: 100001

…

3. **def _getTopBooks(self, bkNode):** This function is triggered when the following tag is encountered in the **promptsPS4.txt** file

   ListTopBooks

   The function searches through the list of books and the checkout counter and determines which are the top three books that have been checked out the most and lists those books and the number of times they have been checked out into the **outputPS4.txt** file as shown below.

   Top Books 1: ***100001, 5***

   Top Books 2: ***100003, 2***

   Top Books 3: ***100002, 1***

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

4. **def _notIssued(self, bkNode):** This function is triggered when the following tag is encountered in the **promptsPS4.txt** file

   BooksNotIssued

   The function searches the list of books in the system and generates a list of books which have never been checked out. The output of this list is put into the **outputPS4.txt** file as shown below.

   List of books not issued:

   100004

   100020

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

5. **def _findBook(self, eNode, bkID):** This function reads the **promptsPS4.txt** file to get the book id that needs to be searched for availability in the system. The function reads the id from the file **promptsPS4.txt** where the search id is mentioned with the tag as shown below.

findBook: 100005

The search function is called for every findBook tag the program finds in the promptsPS4.txt file.

If the book id is found, the below string is appended to the into the **outputPS4.txt** file

Book id ***xx*** is available for checkout

If the book id is found but all the copies have been checked out, the below string is output into the **outputPS4.txt** file

All copies of book id ***xx*** have been checked out

If the book id is not found it outputs the below string into the **outputPS4.txt** file

Book id ***xx*** does not exist.

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

6. **def _stockOut(self, eNode):** This function is triggered when the following tag is encountered in the **promptsPS4.txt** file

   ListStockOut

   This function searches for books for which all the available copies have been checked out and outputs the list into the **outputPS4.txt** file.

   All available copies of the below books have been checked out:

   100001

7. **def printBooks(self, bkNode):** The input should be read from the **promptsPS4.txt** file where the range is mentioned with the tag as shown below.

   printInventory

   This function prints the list of book ids and the available number of copies in the file **outputPS4.txt**.

   The output file should show:

   There are a total of xx book titles in the library.

   100001, 0

   100002, 5

   100003, 13

   …

   Ensure that you use a traversal method that displays the sequence of books in ascending order of book id.

2. Sample file formats

Sample Input file

Each row will have one book id followed by the number of available copies separated by a comma. The input file name must be called **inputPS4.txt**.

Sample inputPS4.txt

```
100001, 4
100002, 5
100003, 15
100004, 12
100005, 3
....
100020, 2
```

Sample promptsPS4.txt

```
checkOut: 100001
checkOut: 100002
checkOut: 100003
checkOut: 100003
checkOut: 100001
checkOut: 100001
checkOut: 100001
checkOut: 100005
checkIn: 100001
checkIn: 100002
checkOut: 100001
ListTopBooks
BooksNotIssued
findBook: 100006
ListStockOut
printInventory
```

Sample outputPS4.txt

```
Top Books 1: 100001, 5
Top Books 2: 100003, 2
Top Books 3: 100002, 1

List of books not issued:
100004
100020

Book id 100005 is available for checkout.

All available copies of the below books have been checked out:
100001
```

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

3. Deliverables

1. Word document **designPS4_<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. **inputPS4.txt** file used for testing
4. **promptsPS4.txt** file used for testing
5. **outputPS4.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_PS4_Library.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.

4. 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.

5. Deadline

1. The strict deadline for submission of the assignment is **10th April 2022.**
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.

6. 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 E-Learn > Assignment section. Assignment submitted via other means like email etc. will not be graded.

7. 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.

8. Readings

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