



(S2-20_DSECLZG519)
(Data Structures and Algorithms Design)
Academic Year 2020-2021

Assignment 1 – PS6 - [Library Records] - [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 maximum number of times?
2. Find out if a particular book is available in the library based on the book id.
3. Out of the entire set of books in the library, list the books that have never been checked out by students and delete them from the library system. Take care that the rest of the books are re arranged in the data structure such that their details are not lost.
4. Print a list of all the books and available copies such that the book ids are listed in ascending order.

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.
3. If you were to use a different data structure, which would be the most ideal data structure to replace a binary tree such that the overall time complexity is the least.

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 **inputPS6.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 **promptsPS6.txt** file. The below sample lines should be part of the **promptsPS6** 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 promptsPS6.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 **promptsPS6.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 **outputPS6.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 **promptsPS6.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 function also triggers another function to delete the node and replace it with the most suitable node in order to retain the properties of the tree. The output of this list is put into the **outputPS6.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 **promptsPS6.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 **promptsPS6.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 **promptsPS6.txt** file.

If the book id is found, the below string is appended to the into the **outputPS6.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 **outputPS6.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 **outputPS6.txt** file

Book id xx does not exist.

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

6. **def printBooks(self, bkNode):** The input should be read from the **promptsPS6.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 **outputPS6.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.

1. 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 **inputPS6.txt**.

Sample inputPS6.txt

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

Sample promptsPS6.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
printInventory
```

Sample outputPS6.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.

2. Deliverables

1. Word document **designPS6_<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. **inputPS6.txt** file used for testing
4. **promptsPS6.txt** file used for testing
5. **outputPS6.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_PS6_LibraryRecords.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 you 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.
8. Please note that the design document must include
 - a. The data structure model you chose with justifications
 - b. Details of each operations with the time complexity and reasons why the chosen operations are efficient for the given representation
 - c. One alternate way of modelling the problem with the cost implications.
9. Writing good technical report and well document code is an art. Your report cannot exceed 4 pages. Your code must be modular and quite well documented.

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 **Monday, 21th Jun, 2021**.
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. Copy / Paste's from web resources / or your friends' submission will attract severe penalty to the extent of awarding negative 10 percent. We will not measure the extent of such blatant copy pastes and details of who copied from whom and such details while awarding the penalties. It's the responsibility of the team to solve and protect your original work.**
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