Yash Tibrewal	UFID: 86029690
Advanced Data Structure	ADS Project

Index

1. Compilation and Execution
2. Environment
3. Project Files
4. Project Keywords and descriptions
5. File Function Descriptions
6. Contact

Compilation and Execution

After downlading and unzipping the folder.

javac GatorLibrary.java
java GatorLibrary test1.txt

To check if the test cases are running

javac RedBlackTreeTest.java
java RedBlackTreeTest

Environment

Tested on College Server

Program Terminated!!thunder:~/ads_p> java --version
openjdk 11.0.20.1 2023-08-24
OpenJDK Runtime Environment (build 11.0.20.1+1-post-Ubuntu-0ubuntu122.04)
OpenJDK 64-Bit Server VM (build 11.0.20.1+1-post-Ubuntu-0ubuntu122.04, mixed mode, sharing)

Tested on Personal Computer

PS G:\My Drive\Masters\Masters Study Material\Advanced Data Structures\Project\gatorLibrary> java --version
java 20.0.2 2023-07-18
Java(TM) SE Runtime Environment (build 20.0.2+9-78)
Java HotSpot(TM) 64-Bit Server VM (build 20.0.2+9-78, mixed mode, sharing)

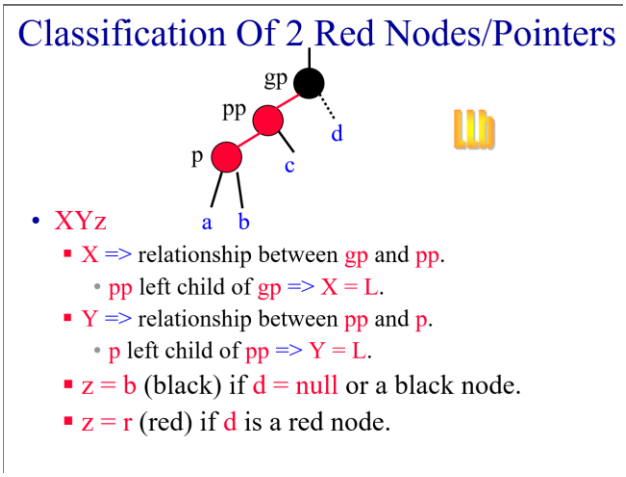
Project Keywords and descriptions

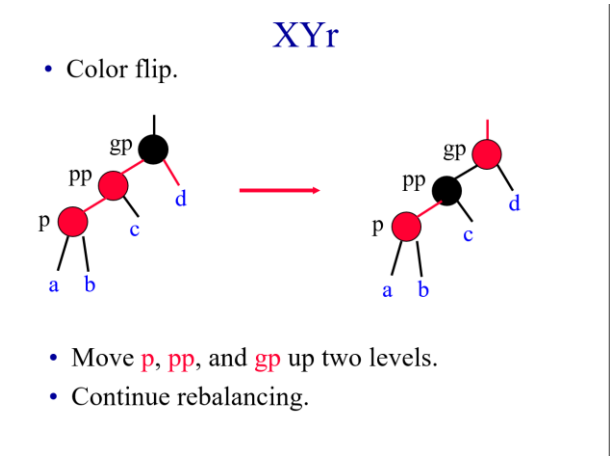
Naming Convention Used in the project

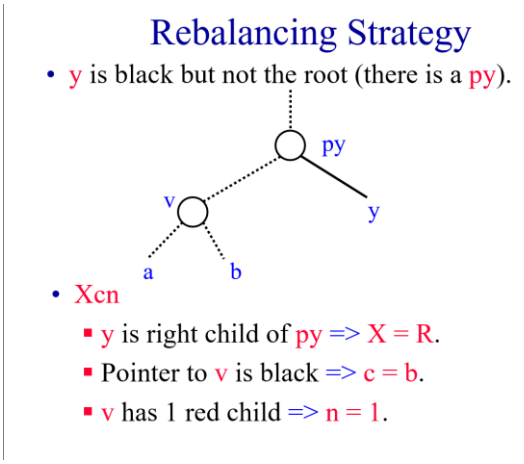
I have used lowerCamelCasing for functions and variables which can be differentiated by the () characters depending on what they are. I have used UpperCamerCasing for the classes.

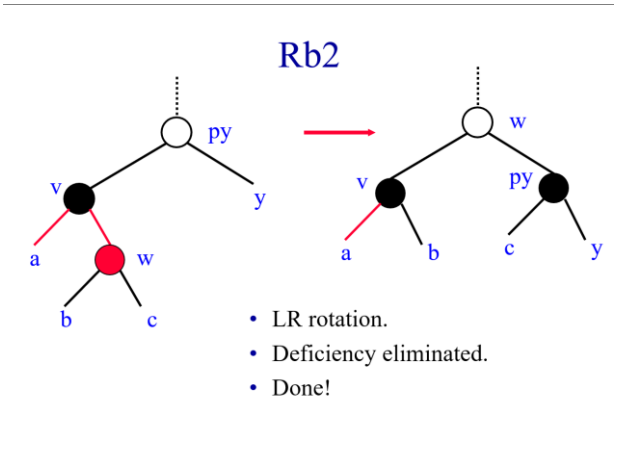
Every file contains functions and each function is verbose to self-understand the meaning of what the function is trying to do. If that is not the case, there is documentation above it in a multiline Java documentation.

All the naming convention for the nodes are taken from the power point presentation shared by the professor. Pasting the screenshots for better readability of the code.

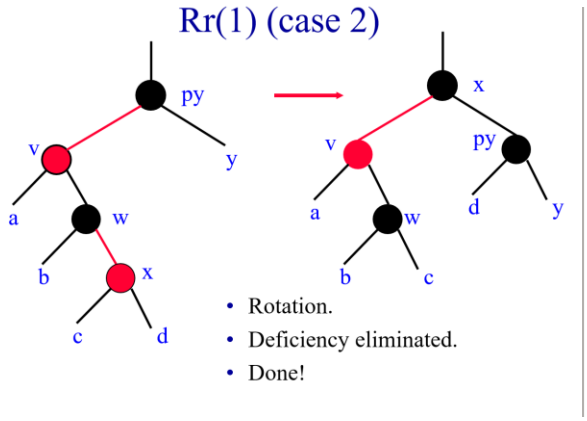


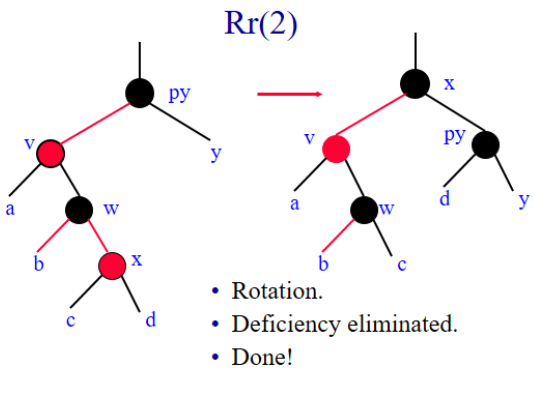






Same naming conventions are used in the files. i.e. for node v in the slide I have used nodeV. Or for example like node c in the slide is referred to as nodeC in the code. For cases, I have added CaseOne, or CaseTwo in the end of the functions as the naming convention.



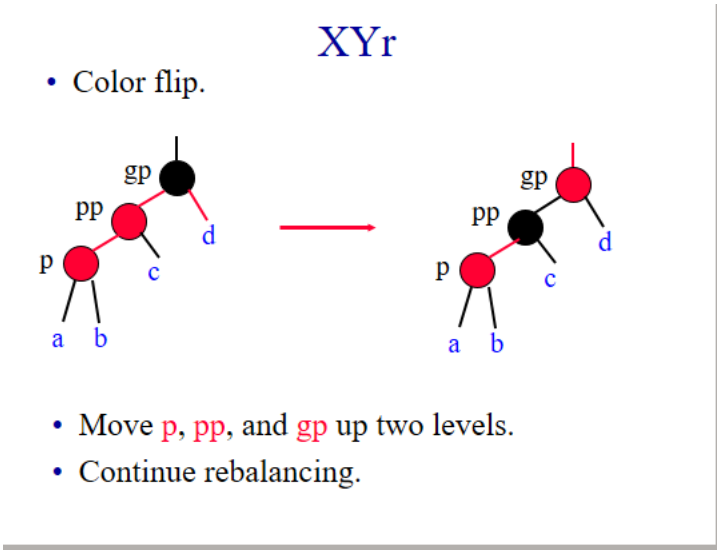


My test cases cover all the cases shown in the power point presentation and the mirror images of those cases, i.e. Rb0,Lb0 both cases, Rb1, Lb1 both cases, Rb2, Lb2, Rr(0,1,2), Lr(0,1,2).

However, my algorithm **DOES NOT** work if the tree has the above structure **BUT** with nodeW as the left chid of nodeV, as I was not sure how to solve it. Similarly for the other mirror images of the same too.

Observe that the nodeX could also be the left child of nodeW and the code does not give the correct output for the same.

Color Flip:



In this project, color flip is ONLY counted when there is a color flip action during the insert.

Inversing a node color is NOT counted as a color flip.

NEITHER does the total number of nodes flipped color before and after the tree is counted.

Project Files

Java Files (.java)	Description
Book	Contains the Book Data structure
BookComparator	Contains the comparison logic between 2 books
BookManager	Contains properties to use a R.B. tree Data Structure with Books to manage the books
Controller	Contains the top level cases which are read from the file, e.g. “InsertBook” etc.
FileProcessor	Handles the File reading and writing
FileProcessorException	Exception raised during File reading and writing
GatoryLibrary	Main Function to run the code.
RedBlackTree	Contains the R.B. Data structure and public functions to use a red black tree.
RedBlackTreeNode	Represents a node in the R.B. Tree data structure
RedBlackTreeTest	Contains state tests for the R.B. Tree for cases
ReservationHeap	Has the Heap Data Structure implemented for Patron reservation of the Book.
ReservationHeapNode	Represents a node in the Reservation Heap.
ReservationHeapTest	Contains test cases for the Reservation Heap.
HeapSizeFullException	Exception when the Heap size is full.
NoElementInHeapException	Exception when there is no element in the heap.

File Function Descriptions

The documentation for every function is done on the code for better understanding while going through the code base instead of mentioning it here.

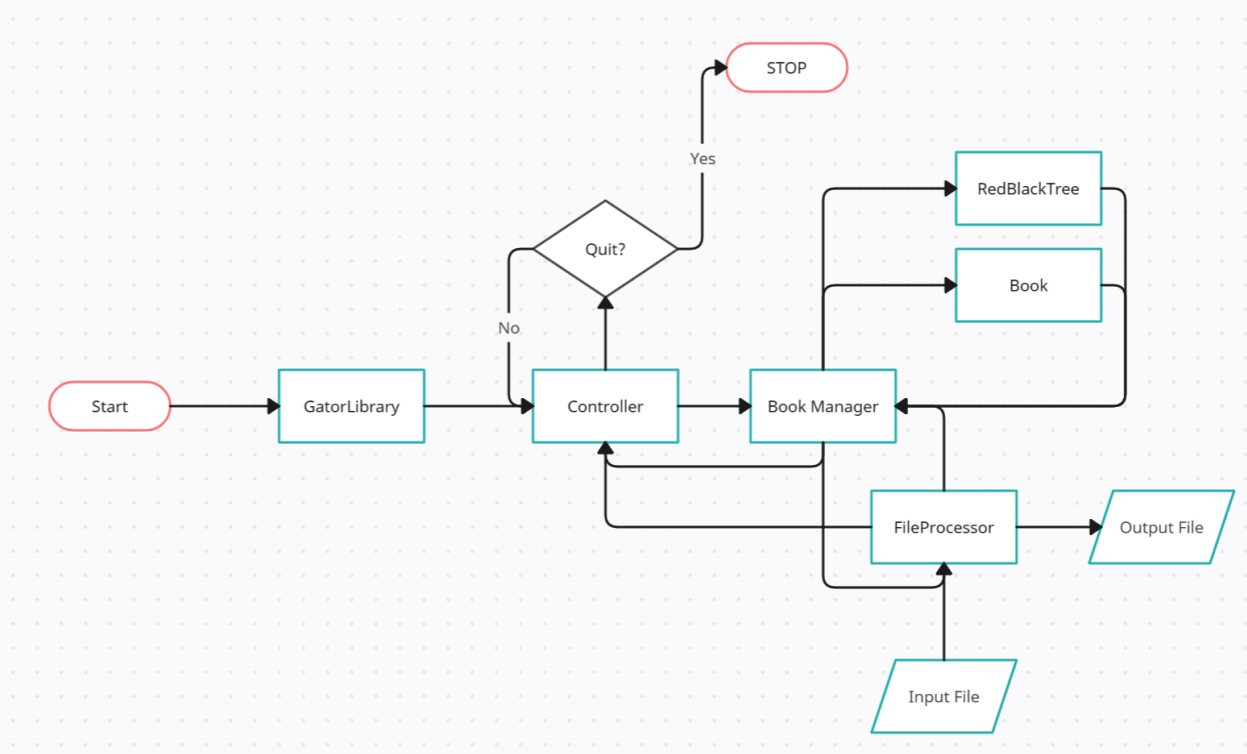
Time Complexity

None of the functions implemented in the project take more then O(logn) time.

As far as borrow function is concerned, since I am using milliseconds as the smallest unit of time at the time of borrow, and the systems can read the borrow requests in nano seconds, it is always a problem when there are multiple patrons with the same priority number and exactly the same time borrowing, it messes up with the algorithm tie breaking and does not help with first come first serve. To solve this issue, **I introduced ONE millisecond delay** for every time a Patron borrows a book.

Hence if the code is tested for time complexity, borrow function might result in bottleneck for large n.

Program and Data Flow Diagram



Important Notes

1. Please clean the .class file before compiling for the new time.
2. Please read the on purpose delay while a patron borrows a book.

Contact

Mobile Number

Phone. +1 352 871 3427

Whatsapp +91 88790 34882

Email

Personal: yashkush.tibrewal@gmail.com

College: yashtibrewal@ufl.edu