Advanced Data Structures - COP 5536 Fall 2023

Programming Project Report

GatorLibrary Management System

Name: Vaishnavi Chilakamarthi

UFID: 99936597

Email: vchilakamarthi@ufl.edu

Introduction

The GatorLibrary project is designed to efficiently manage books, patrons, and borrowing operations for a fictional library. The system utilizes a Red-Black tree data structure for book management and Binary Min-heaps for book reservations. This report outlines the implementation details, challenges faced, and the structure of the program.

Project Structure

- ***gatorLibrary.py***

This file contains code to read the file name through the command line arguments, extract each command from the text file and call respective functions in the GatorLibrary class. All the commands after the Quit() command are not executed.

- ***GatorHelper.py***

This is a helper class that creates an output file and writes the output into it. It calls the respective function in the GatorRedBlack class that has the main logic for the Red Black Tree implementation.

The GatorLibrary Class handles the following operations:

- InsertBook() : updateColorFlips() from the GatorRedBlack class is called to update colorFlips counter after every Insert operation. Also writes the value returned by the GatorRedBlack class into the output file.
- DeleteBook() : updateColorFlips() from the GatorRedBlack class is called to update colorFlips counter after every Delete operation. .Also writes the value returned by the GatorRedBlack class into the output file

- BorrowBook() : Writes the value returned by the GatorRedBlack class into the output file.
- ReturnBook() : Writes the value returned by the GatorRedBlack class into the output file.
- ColorFlipCount() : Writes the value returned by the GatorRedBlack class into the output file
- PrintBook() : Writes the value returned by the GatorRedBlack class into the output file
- FindClosestBook() : Writes the value returned by the GatorRedBlack class into the output file
- Quit() : Writes the string “Program Terminated!!” In the output file

- ***GatorRedBlack.py***

This is the main file which contains the logic. This class contains the following functions:

- __init__(): Parameters - bookID: int, title: str, author: str, availability: str

  This is the constructor function that initializes the following class variables

  - bookID = This value is received as the parameter
  - parent = None : Initial parent is None
  - left = Initial left subtree is None
  - right = Initial right subtree is None
  - color = 1 : Initial color is Red
  - bookName = This value is received as the parameter
  - authorName = This value is received as the parameter
  - availabilityStatus = This value is received as the parameter
  - borrowedBy = None : No borrower is set initially
  - self.reservationHeap = MinHeap(20) : Initialized with the min heap with a maximum size of 20
- insertNode(): Parameters - bookID: int, title: str, author: str, availability: str

  This function creates a new Node with the values received as parameters using the binary search tree logic.
- fixInsert(): Parameters - k: Node

This function checks the tree starting from the Node k and performs color balancing and rotations.

- RR(): Parameters - x: Node

  Tis function performs Right-Right rotation on the subtree with x as the root node

- LR(): Parameters - x: Node

  Tis function performs Left-Right rotation on the subtree with x as the root node

- borrowBook(): Parameters - patronID: int, bookID: int, patronPriority: int

  This function allots the book with the bookID to the patron with the patronId if the book is available. Else, the function adds the patron into the reservation Heap of the book based on the priority and the timestamp.

- returnBook(): Parameters - patronID: int, bookID: int

  This function removes the patronId from the borrowedBy attribute of the book and allots the book to the next patron in the reservation heap. If the reservation heap is empty, it sets the availability to Yes.

- DeleteBook(): Parameters - book: Node

  This function deletes the Node with the specified book ID and cancels all the reservations made by the patron in the reservation heap.

- getColorFlipCount()

  This function returns the value of the colorFlips attribute of the tree.

- printBook(): Parameters - bookID: int

  This function takes the bookID and prints all the details like id, title, author, availability, reservations, borrowed by.

- printBooks(): Parameters - bookID1: int, bookID2: int

  This function takes a range of int values and prints the details of all the books present in the given range(inclusive of bookID1 and bookID2)

- findClosestBook(): Parameters - bookID

  This function takes an int value and prints the details of the book with the closest bookID value. If there is a tie, it prints the details of both the books.

- get_node(): Parameters - val: int

  This function takes a value and fetches the Node which val as the bookID

- delete_node(): Parameters - val: int

  This function takes a value and calls the delete_node_helper() function that deletes the node with val as the bookID

- delete_node_helper(): Parameters - node: Node, key: int

  This function deletes the node with val as the bookID. While doing so it checks the Tree for the Red Black Tree properties and calls __rb_transplant() function

- __rb_transplant(): Parameters - u: Node, v: Node

  This function effectively replaces the subtree rooted at node u with the subtree rooted at node v, adjusting the parent-child relationships accordingly.

- maximum(): Parameters - node: Node

  This function takes a Node and find the largest value in the subtree

- updateColorFlips():

  This function traverses through the Red Black Tree and compares it with the colorState which contains the previous color states. The function then computes the number of color flips and updates the color Flips count and color state of the Tree.

- print_tree()

  This function is used to print the tree while debugging. This function calls the __printCall() which has the main logic to print

- __printCall(): Parameters - node: Node , indent: chr , last: bool

  This is a recursive function to print the nodes in the tree along with the color and the bookID. This function is required while debugging.

- ***MinHeap.py***

This helper class implements a MinHeap for managing book reservations and waitlists. Key functions include:

- __init__(): Parameters - maxsize: int

  This function initializes the Heap with the following class variables:

  - maxsize: Maximum size of the heap
  - size: Current size of the heap
  - Heap: List to store heap elements

    - FRONT: Index of the front element in the heap

- parent(): Parameters - pos: int

  This function computes the parent index of a given position.

- leftChild(): Parameters - pos: int

  This function Computes the left child index of a given position.

- rightChild(): Parameters - pos: int

  This function Computes the right child index of a given position.

- isLeaf(): Parameters - pos: int

  This function checks if the given position is a leaf Node. It returns True if it is a leaf node. Else it returns False.

- swap(): Parameters - fpos: int, spos: int

  This function swaps elements at two positions in the heap.

- minHeapify(): Parameters - pos: int

  This function maintains the min-heap property by recursively fixing the heap at a given position.

- insert(): Parameters - element: tuple

  This function inserts an element into the heap and maintains the min-heap property. The element is a tuple containing patronPriority, patronID, time values in order.

- getValues():

  This function returns all the values present in the Heap

- minHeap():

  This function maintains the heap property at every position in the Heap by calling the minHeapify() function

- remove()

  This function removes the minimum element from the Heap and returns that element.

Input and Output Requirements

- Input is read from a text file specified as a command-line argument.

- Output is written to a text file named as concatenation of input_filename + "_output_file.txt".

Program Execution

To execute the program, use the following command in the terminal:

```
python gatorLibrary.py 'file_name.txt'
```

Ensure that 'file_name.txt' contains the commands for the desired testcase. After execution, an output file named "file_name_output_file.txt" will be generated, containing the program's output.

Alternatively, you can use the make command for streamlined execution. Running: *'make run'* command will execute the program with the filename specified in the Makefile as input, and the output will be saved in a file with the corresponding name.

If you wish to override the filename, you can use the following command:

```
make run TEST_CASE='new_file_name.txt'
```

Replace 'new_file_name.txt' with the desired testcase file name. This command will execute the program with the specified file, and the output will be saved in "new_file_name_output_file.txt".

Testcase1 explanation

Testcase:

```
InsertBook(101, "Introduction to Algorithms", "Thomas H. Cormen", "Yes")
InsertBook(48, "Data Structures and Algorithms", "Sartaj Sahni", "Yes")
PrintBook(48)
InsertBook(132, "Operating System Concepts", "Abraham Silberschatz", "Yes")
InsertBook(25, "Computer Networks", "Andrew S. Tanenbaum", "Yes")
BorrowBook(120, 48, 2)
BorrowBook(132, 101, 1)
InsertBook(73, "Introduction to the Theory of Computation", "Michael Sipser", "Yes")
InsertBook(12, "Artificial Intelligence: A Modern Approach", "Stuart Russell", "Yes")
InsertBook(6, "Database Management Systems", "Raghu Ramakrishnan", "Yes")
BorrowBook(144, 48, 3)
BorrowBook(140, 48, 3)
BorrowBook(142, 48, 2)
BorrowBook(138, 12, 4)
BorrowBook(150, 12, 3)
BorrowBook(162, 12, 1)
ReturnBook(120, 48)
```

```
FindClosestBook(9)
DeleteBook(12)
ColorFlipCount()
InsertBook(125, "Computer Organization and Design", "David A. Patterson", "Yes")
InsertBook(180, "Introduction to Software Engineering", "Ian Sommerville", "Yes")
BorrowBook(111, 73, 3)
BorrowBook(52, 73, 1)
InsertBook(115, "Operating Systems: Internals and Design Principles", "William Stallings", "Yes")
BorrowBook(153, 25, 2)
PrintBooks(10, 150)
InsertBook(210, "Machine Learning: A Probabilistic Perspective", "Kevin P. Murphy", "Yes")
BorrowBook(171, 25, 3)
BorrowBook(2, 132, 2)
FindClosestBook(50)
BorrowBook(18, 101, 2)
InsertBook(80, "Software Engineering: A Practitioner's Approach", "Roger S. Pressman", "Yes")
BorrowBook(210, 210, 1)
BorrowBook(43, 73, 1)
InsertBook(60, "Introduction to Computer Graphics", "David F. Rogers", "Yes")
PrintBook(210)
InsertBook(4, "Design Patterns: Elements of Reusable Object-Oriented Software", "Erich Gamma", "Yes")
InsertBook(2, "Introduction to the Theory of Computation", "Michael Sipser", "Yes")
BorrowBook(34, 210, 2)
InsertBook(65, "Computer Networks: Principles, Protocols, and Practice", "Olivier Bonaventure", "Yes")
ColorFlipCount()
DeleteBook(125)
DeleteBook(115)
DeleteBook(210)
ColorFlipCount()
DeleteBook(25)
DeleteBook(80)
ColorFlipCount()
Quit()
```

The tescase states 27 as output for the color third flip count count. However, the program outputs 25 as the values with the following series of insertions and deletions.

```
R---- 101(BLACK)
Color Flips Count - 0
```

```
R---- 101(BLACK)
     L---- 48(RED)
     R---- 132(RED)
Color Flips Count - 0

R---- 101(BLACK)
     L---- 48(BLACK)
    |    L---- 25(RED)
     R---- 132(BLACK)
Color Flips Count - 0

R---- 101(BLACK)
     L---- 48(BLACK)
    |    L---- 25(RED)
    |    R---- 73(RED)
     R---- 132(BLACK)
Color Flips Count - 2

R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 25(BLACK)
    |   |    L---- 12(RED)
    |    R---- 73(BLACK)
     R---- 132(BLACK)
Color Flips Count - 2

R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 12(BLACK)
    |   |    L---- 6(RED)
    |   |    R---- 25(RED)
    |    R---- 73(BLACK)
     R---- 132(BLACK)
Color Flips Count - 5


R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 6(BLACK)
    |   |    R---- 25(RED)
    |    R---- 73(BLACK)
     R---- 132(BLACK)
Color Flips Count - 7
```

```
R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 6(BLACK)
    |   |    R---- 25(RED)
    |    R---- 73(BLACK)
     R---- 132(BLACK)
         L---- 125(RED)
Color Flips Count - 8

R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 6(BLACK)
    |   |    R---- 25(RED)
    |    R---- 73(BLACK)
     R---- 132(BLACK)
         L---- 125(RED)
         R---- 180(RED)
Color Flips Count - 8

R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 6(BLACK)
    |   |    R---- 25(RED)
    |    R---- 73(BLACK)
     R---- 132(RED)
         L---- 125(BLACK)
        |    L---- 115(RED)
         R---- 180(BLACK)
Color Flips Count - 8

R---- 101(BLACK)
     L---- 48(RED)
    |    L---- 6(BLACK)
    |   |    R---- 25(RED)
    |    R---- 73(BLACK)
     R---- 132(RED)
         L---- 125(BLACK)
        |    L---- 115(RED)
         R---- 180(BLACK)
             R---- 210(RED)
Color Flips Count - 11

R---- 101(BLACK)
     L---- 48(RED)
     |     L---- 6(BLACK)
     |     |     R---- 25(RED)
     |     R---- 73(BLACK)
     |           R---- 80(RED)
      R---- 132(RED)
           L---- 125(BLACK)
           |     L---- 115(RED)
           R---- 180(BLACK)
                 R---- 210(RED)
Color Flips Count - 11

R---- 101(BLACK)
      L---- 48(RED)
     |     L---- 6(BLACK)
     |     |     R---- 25(RED)
     |     R---- 73(BLACK)
     |           R---- 80(RED)
      R---- 132(RED)
           L---- 125(BLACK)
           |     L---- 115(RED)
           R---- 180(BLACK)
                 R---- 210(RED)
Color Flips Count - 11

R---- 101(BLACK)
      L---- 48(RED)
     |     L---- 6(BLACK)
     |     |     R---- 25(RED)
     |     R---- 73(BLACK)
     |           L---- 60(RED)
     |           R---- 80(RED)
      R---- 132(RED)
           L---- 125(BLACK)
           |     L---- 115(RED)
           R---- 180(BLACK)
                 R---- 210(RED)
Color Flips Count - 11

R---- 101(BLACK)
      L---- 48(RED)
     |     L---- 6(BLACK)
     |     |     L---- 4(RED)
     |     |     R---- 25(RED)
     |     R---- 73(BLACK)
     |          L---- 60(RED)
     |          R---- 80(RED)
      R---- 132(RED)
           L---- 125(BLACK)
          |     L---- 115(RED)
           R---- 180(BLACK)
                R---- 210(RED)
Color Flips Count - 11

R---- 101(BLACK)
      L---- 48(BLACK)
     |     L---- 6(RED)
     |    |     L---- 4(BLACK)
     |    |    |     L---- 2(RED)
     |    |     R---- 25(BLACK)
     |     R---- 73(BLACK)
     |          L---- 60(RED)
     |          R---- 80(RED)
      R---- 132(BLACK)
           L---- 125(BLACK)
          |     L---- 115(RED)
           R---- 180(BLACK)
                R---- 210(RED)
Color Flips Count - 16

R---- 101(BLACK)
      L---- 48(BLACK)
     |     L---- 6(RED)
     |    |     L---- 4(BLACK)
     |    |    |     L---- 2(RED)
     |    |     R---- 25(BLACK)
     |     R---- 73(RED)
     |          L---- 60(BLACK)
     |         |     R---- 65(RED)
     |          R---- 80(BLACK)
      R---- 132(BLACK)
           L---- 125(BLACK)
          |     L---- 115(RED)
           R---- 180(BLACK)
                R---- 210(RED)
Color Flips Count - 19

R---- 101(BLACK)
      L---- 48(BLACK)
     |     L---- 6(RED)
     |    |     L---- 4(BLACK)
     |    |    |     L---- 2(RED)
     |    |     R---- 25(BLACK)
     |     R---- 73(RED)
     |          L---- 60(BLACK)
     |         |     R---- 65(RED)
     |          R---- 80(BLACK)
      R---- 132(BLACK)
           L---- 115(BLACK)
           R---- 180(BLACK)
                R---- 210(RED)
Color Flips Count - 20

R---- 101(BLACK)
      L---- 48(BLACK)
     |     L---- 6(RED)
     |    |     L---- 4(BLACK)
     |    |    |     L---- 2(RED)
     |    |     R---- 25(BLACK)
     |     R---- 73(RED)
     |          L---- 60(BLACK)
     |         |     R---- 65(RED)
     |          R---- 80(BLACK)
      R---- 180(BLACK)
           L---- 132(BLACK)
           R---- 210(BLACK)
Color Flips Count - 21

R---- 48(BLACK)
      L---- 6(BLACK)
     |     L---- 4(BLACK)
     |    |     L---- 2(RED)
     |     R---- 25(BLACK)
      R---- 101(BLACK)
           L---- 73(RED)
          |     L---- 60(BLACK)
          |    |     R---- 65(RED)
          |     R---- 80(BLACK)
           R---- 180(BLACK)
                L---- 132(RED)
Color Flips Count - 23

R---- 48(BLACK)
```

```
     L---- 4(BLACK)
    |     L---- 2(BLACK)
    |     R---- 6(BLACK)
     R---- 101(BLACK)
          L---- 73(RED)
         |     L---- 60(BLACK)
         |    |     R---- 65(RED)
         |     R---- 80(BLACK)
          R---- 180(BLACK)
               L---- 132(RED)
Color Flips Count - 24

 R---- 48(BLACK)
     L---- 4(BLACK)
    |     L---- 2(BLACK)
    |     R---- 6(BLACK)
     R---- 101(BLACK)
          L---- 65(RED)
         |     L---- 60(BLACK)
         |     R---- 73(BLACK)
          R---- 180(BLACK)
               L---- 132(RED)
Color Flips Count - 25
```

Conclusion:

In conclusion, the GatorLibrary project provides an efficient solution for library management, ensuring accurate book tracking, reservations, and patron interactions. The Red-Black tree and Binary Min-heap structures enable effective data organization and retrieval.