Gator Library Management System

Lohit Bhambri (lohit.bhambri@ufl.edu)

How to run the project

Firstly you need to navigate to the project directory where `Main.py` file is available.
To test and run all the test-case associated with this project, use the following command:

```
python Main.py input1.txt
python Main.py input2.txt
python Main.py input3.txt
python Main.py input4.txt
```

It will generate the following output files

```
input1_output1.txt
input2_output2.txt
input3_output3.txt
input4_output4.txt
```

Node.py

Method	Description	Time Complexity
__init__	Initializes a Node object for Red-Black Tree.	O(1)

Attributes:

- `value` : Represents the book details using the `Books` class.
- `left` : Points to the left child node.
- `right` : Points to the right child node.
- `parent` : Points to the parent node.
- `color` : Represents the color of the node ("Red" or "Black").

Note: The time complexity for the `__init__` method is O(1) since it involves simple assignments and object creation.

Books.py

Method	Description	Time Complexity
__init__	Initializes a book with given book details.	O(1)
add_reservation	Adds a reservation to the book's reservation list and sorts it.	O(log n)
remove_reservation	Removes and returns the earliest reservation from the book.	O(log n)

Note: The time complexity for sorting in `add_reservation` is O(n log n), where n is the number of reservations. The removal operation in `remove_reservation` is O(log n) due to the min-heap properties.

RedBlackTree.py

Method	Description	Time Complexity
__nodeInsertion	Inserts a node into the Red-Black Tree.	O(log N)
__colorConflictResolution	Resolves conflicts arising from red-red violation during insertion.	O(log N)
insertionNode	Inserts a new node with given book details.	O(log N)

Method	Description	Time Complexity
__searchNode	Searches for a node with a given bookId.	O(log N)
__getSuccessor	Returns the in-order successor of a given node.	O(log N)
__getReplacementNode	Returns the replacement node for deletion.	O(log N)
__getSibling	Returns the sibling of a given node.	O(1)
__rotateRight	Performs a right rotation to fix coloring during deletion.	O(1)
__rotateLeft	Performs a left rotation to fix coloring during deletion.	O(1)
__hasRedChild	Checks if a given node has a red child.	O(1)
__fixDoubleBlack	Fixes double black violations during deletion.	O(log N)
__swapValues	Swaps values between two nodes.	O(1)
__deleteRBTNode	Deletes a node from the Red-Black Tree.	O(log N)
delete	Deletes a node with the given bookId.	O(log N)

Note: The time complexities are stated in terms of the height of the Red-Black Tree (log N), where N is the number of nodes in the tree.

MinHeap.py

Class	Description
MinHeap	Implementation of a Min Heap data structure.

Attributes

- `heap` : List representing the heap.

- `size` : Current size of the heap.
- `capacity` : Maximum capacity of the heap.
- `reservation_map` : Dictionary mapping priority numbers to their indices in the heap.

Methods

Method	Description	Time Complexity
`__init__(self, capacity)`	Initializes the MinHeap with the given capacity.	O(1)
`get_left(self, index)`	Returns the index of the left child of a node.	O(1)
`get_right(self, index)`	Returns the index of the right child of a node.	O(1)
`get_parent(self, index)`	Returns the index of the parent of a node.	O(1)
`insert(self, new_reservation)`	Inserts a new reservation into the heap and maintains the heap property.	O(log N)
`swap(self, index1, index2)`	Swaps elements at two indices in the heap.	O(1)
`find_min_index(self, index1, index2)`	Finds the index of the minimum element between two indices.	O(1)
`heapify_up(self, index)`	Maintains the heap property by moving an element upwards in the heap.	O(log N)
`heapify_down(self, index)`	Maintains the heap property by moving an element downwards in the heap.	O(log N)
`extract_min(self)`	Extracts the minimum element from the heap and maintains the heap property.	O(log N)
`delete_key(self, index)`	Marks an element at a given index as deleted and adjusts the heap.	O(log N)
`delete_reservation(self, priority_number)`	Deletes a reservation with the given priority number from the heap.	O(log N)

Method	Description	Time Complexity
`__str__(self)`	Returns a string representation of the MinHeap.	O(N)

Note: Time complexities are mentioned in the individual method descriptions.

TestCases.py

Method	Description	Time Complexity
__init__	Initializes the TestCases object.	O(1)
__inorderTraversal	Performs an inorder traversal of the Red-Black Tree.	O(n)
__chkForAvailability	Checks the availability status and returns "Yes" or "No".	O(1)
__chkForReservationHeap	Checks the reservation heap and returns a list of patron IDs.	O(n)
InsertBook	Inserts a book into the Red-Black Tree.	O(log n)
PrintBook	Prints details of a specific book.	O(log n)
PrintBooks	Prints details of books within a given range.	O(k log n)
BorrowBook	Borrows a book or adds a reservation.	O(log n)
ReturnBook	Returns a book, handles reservations.	O(log n)
Quit	Terminates the program.	O(1)
DeleteBook	Deletes a book and cancels reservations.	O(log n)
FindClosestBook	Finds the closest books to a target ID.	O(n)

Method	Description	Time Complexity
ColorFlipCount	Returns the color flip count of the Red-Black Tree.	O(1)
write_to_output	Writes the result to the output file.	O(1)

Note: The time complexity for Red-Black Tree operations depends on the height of the tree (log n), where n is the number of nodes in the tree. The time complexity for `PrintBooks` is O(k log n), where k is the number of books in the specified range. The time complexity for `FindClosestBook` is O(n), where n is the number of books in the Red-Black Tree.

Main.py

Method	Description	Time Complexity
__main__	Main entry point for the program. Reads input, executes	O(N)
	functions, and handles program termination.	

Attributes:

- `sys.argv` : Command-line arguments.
- `TestCases` : Class containing test cases.
- `test_instance` : Instance of the `TestCases` class.
- `input_file_name` : Name of the input file.
- `outputFileName` : Default output file name.
- `lines` : List containing lines from the input file.

Functions:

- `__main__` : Main entry point for the program. Reads input, executes functions, and handles program termination.

Note: The time complexity for the `__main__` function is O(N), where N is the number of lines in the input file, as it processes each line sequentially.