ADS Project Assignment Report

Name: Shivani Sanjay Patil
UFID: 5354-9503
UF Email: patil.s@ufl.edu

Program Structure:

Classes

1. **gatorTaxi:**

    - This class contains main method of the program along with logic to take input from text file and writing output to text file.
    - Program takes O(n) space to store nodes in data structures
    - Additional disk space is required to store input and output files.

2. **MinHeap:**

    - This class contains methods to perform minheap operations. These methods are insertNode, removeMin, deleteArbitaryNode, decreaseTripDuration.
    - This class also contains some helper methods, which are private to the class. These methods are heapify, swapElements, getLeftChild, getRightChild, getParent.
    - Space complexity of min MinHeap is O(n).

3. **RBT:**
    - This class contains methods to perform Red Black Tree operations. These methods are insertNode, deleteNode, searchNode, updateNode.
    - This class also contains helper methods, which are printRide, printHelper, searchNodeHelper, inOrderHelper, rebalanceRecolorInsertion, rebalanceRecolorDeletion, switchNodes, leftRotation, rightRotation.
    - Space complexity of RBT is O(n).

4. **Ride:**
    - This class contains ride information like rideNumber, rideCost, tripDuration. It also contains getter and setter method to get and set above parameters.
    - Object of the ride is stored in TreeNode node, it requires space complexity of O (1).

5. **TreeNode:**
    - This class contains node structure. The node contains leftChildNode, rightChildNode, parentNode pointers, key, heapIndex, color and ride.
    - Default and parameterized constructor as well as getter and setter methods for parameters are also declared on this class.

Functions

Class: gatorTaxi.java

1. **public String Print (int rideNumber):**
    - Used to print ride details for the given rideNumber.
    - It prints (0,0,0) if ride doesn't exist.
    - Time complexity if O(log(n)).

2. **public String Print (int rideNumber1, int rideNumber2)**
    - Used to print all the rides with comma separation with ride number ranging from rideNumber1 to rideNumber2 including both.
    - It prints (0,0,0) if there is no ride between given range.
    - Time complexity is O(log(n)+S).

3. **public String Insert (int rideNumber, int rideCost, int tripDuration)**
    - This function is used to add ride with given rideNumber, rideCost and tripDuration in red black tree as well as in minheap.
    - It returns "Duplicate RideNumber" message and terminates the program if already existing rideNumber is provided.
    - Time complexity is O(log(n))

4. **public String GetNextRide ()**
    - This function is used to remove ride with minCost from the minheap, corresponding node is removed from the Red Black Tree as well.
    - It returns the ride removed. And if minheap is empty, it returns "No active ride Requests".
    - Time complexity is O(log(n)).

5. **public void CancelRide (int rideNumber)**
    - This function is used to cancel the ride from database.
    - The ride with provided rideNumber is searched in RBT and if found, it is deleted from RBT and the corresponding node is deleted from the min heap.
    - If ride doesn't exist, nothing is done.
    - Time complexity is O(log(n)).

6. **public void UpdateTrip (int rideNumber, int new_tripDuration)**
    - This function is used to update tripduration for given rideNumber. If ride exists for provided rideNumber, it's tripDuration is updated as per given conditions.
    - If ride doesn't exist, nothing is done.
    - Time complexity is O(log(n)).

Class: MinHeap.java

1. **public void insertNode (TreeNode node)**
    - Used to insert node in minheap. Key here is rideCost and in case of tie, tripDuration is used as tie breaker.
    - Function adds the node at end of ArrayList, calls heapify function to compare rideCost with it's parents and arranges the heap such that min heap property is satisfied.

2. **public Ride removeMin ()**
    - This function removes the smallest element from the heap.
    - It first takes last element, places it at root and call heapify on root.
    - Time complexity is O(log(n)).

3. **Private void heapify (int currRideIndex)**
    - This is helper function, which heapify the heap from index currRideIndex.
    - It checks the left and right children, if any of them are smaller than the current node, swapping is performed.
    - Time complexity if O(log(n)).

4. **public void deleteArbitaryNode (int rideIndex)**
    - This function is used to remove arbitrary node with given rideIndex from the minheap.
    - First the rideCost of the node is decreased to value less than the root, then, it is bubbled up to the root. And then the root is deleted.
    - Time complexity for bubbling up is O(log(n) and remove min requires O(log(n)), so overall time complexity is O(log(n)).

5. **public void decreaseTripDuration (int rideIndex, int new_tripDuration)**
    - This function is used to decrease the tripDuration of ride with rideIndex stored in min heap.
    - This function is called during updateTrip operation. Since, tripDuration is secondary key of min heap, heap is required to be rearranged once the tripDuration of the ride is updated.

6. **private void swapElements (int currentRideIndex, int parentRideIndex)**
    - Used to swap nodes with given ride indexes.
    - While swapping heapIndex is also updated.
    - Time complexity is O (1).
    -

7. **private int getParent (int rideIndex)**
    - Returns the index of parent of the node in min heap.
    - Time complexity is O (1).

8. **private int getRightChild (int rideIndex)**
    - Returns the index of right Child of the node in min heap.
    - Time complexity is O (1).

9. **private int getLeftChild (int rideIndex)**
    - Returns the index of left Child of the node in min heap.
    - Time complexity is O (1).

Class: RBT.java

1. **private void leftRotation (TreeNode currNode)**
    - Performs left rotation around node currNode

2. **private void rightRotation (TreeNode currNode)**
    - Performs right rotation around node currNode.

3. **private void rebalanceRecolorInsertion (TreeNode newNode)**
    - This is helper function which rebalances and recolors the tree if required after the insertion operation.
    - Time complexity is O(log(n)

4. **public void insertNode (TreeNode newNode)**
    - Inserts the node in RBT.
    - Sets the parent as null, left and right child as null, color as 1

- Then add node in normal BST and calls rebalanceRecolorInsertion method
- Time complexity is O(log(n)).

5. **public void deleteNode (int key)**
    - This function is used to delete node from RBT.
    - Node with given key is first searched, if found it is removed in same way deletion is done in BST.
    - Then if color of node removed is black, then only rebalanceRecolorDeletion method is called.
    - Time complexity is O(log(n)).

6. **private void switchNodes (TreeNode node1, TreeNode node2)**
    - This function is used to switch nodes of RBT.

7. **private void rebalanceRecolorDeletion (TreeNode node)**
    - This is helper function which rebalances and recolors the tree if required after the insertion operation.
    - Time complexity is O(log(n)

8. **private TreeNode searchNodeHelper (TreeNode node, int key)**
    - This is recursive function, which goes on checking left child and right child recursively till whole tree is traversed or the desired node is found.
    - Time complexity is O(log(n)).

9. **public TreeNode searchNode (int key)**
    - This function calls searchNodeHelper to find the ride with the key.

10. **Public String inOrderHelper (int rideNumber1, int rideNumber2, TreeNode node)**
    - This function is called by Print method of gatorTaxi class, to print rides between rideNumber1 and rideNumber2.
    - This recursive function traverses the tree in inorder fashion and keeps of adding rides to string if they lie between required range.
    - Time complexity is O(log(n)).

11. **public void updateNode (int key, int new_tripDuration)**
    - Used to update the tripDuration of the ride in tree.
    - First the ride is searched and if found, its tripDuration is updated.
    - Time complexity is O(log(n)), since time to find the ride is O(log(n) and to update tripDuration it’s O (1).

12. **public String printRide (Ride ride)**
    - Used to print the ride in tuple format as specified.

Class: Ride.java

1. **public int getRideCost ()**
    - Getter to method to return rideCost.
    - Time Complexity is O (1).

2. **public int getTripDuration ()**
    - Getter to method to return tripDuration.
    - Time Complexity is O (1).

3. **public int getRideNumber ()**
    - Getter to method to return rideNumber.
    - Time Complexity is O (1).

4. **public void setRideCost (int cost)**
    - Setter method to set rideCost
    - Time Complexity is O (1)

5. **public void setTripDuration (int duration)**
    - Setter method to set tripDuration
    - Time Complexity is O (1).

Class: TreeNode.java

1. **public TreeNode (int key, Ride ride, TreeNode leftChildNode, TreeNode rightChildNode, TreeNode parentNode, int color, int heapIndex)**
    - Parameterized Constructor to initialize class variables with passed values.
    - Time complexity is O (1).

2. **public TreeNode (Ride ride)**
    - Constructor to initialize TreeNode with Ride ride.
    - Time Complexity is O (1)

3. **public Ride getRide ()**
    - Getter to method to return ride stored in the node.
    - Time Complexity is O (1).

4. **public int getMinHeapIndex ()**
    - Getter to method to return heapIndex of the corresponding node in min heap.
    - Time Complexity is O (1).

5. **Public void setMinHeapIndex (int index)**
    - Getter to method to return heapIndex of the corresponding node in min heap.
    - Time Complexity is O (1).

6. **public int getRideCost ()**
    - Getter to method to return rideCost of the ride stored in node.
    - Time Complexity is O (1).

7. **public int getTripDuration ()**
    - Getter to method to return tripDuration of the ride stored in node.
    - Time Complexity is O (1).

8. **public int getRideNumber ()**
    - Getter to method to return rideNumber of the ride stored in node.
    - Time Complexity is O (1).

9. **public void setRideCost (int cost)**
    - Setter method to set rideCost of the ride stored in node.
    - Time Complexity is O (1)

10. **public void setTripDuration (int duration)**
    - Setter method to set tripDuration of the ride stored in node.
    - Time Complexity is O (1).

Steps to run the Program:

1. Unzip the folder.
2. Run ‘make’
3. Run ‘java gatorTaxi <input_file_namr.txt>’

Input file: <filename>
Output file: output_file.txt