COP 5536: Advance Data Structures
Programming Project Report

Authors:
- Name:- Tanuj Venkata Satya Sridhar Karuturi
- UFID:- 78056734
- Email:- karuturi.t@ufl.edu

File Structure:

After unzipping the submitted file it contains the following files (working directory structure):
- gatorTaxi.java
- makefile
- MinHeapp.java
- RedBlackTree.java
- Ride.java

Definitions:
1. **Ride:** Implementation of a structure which will help us to store the details of each ride in MinHeapp and RedBlackTree. It consists of Id, cost, duration, color of this ride in RedBlackTree(red/black), index in the Heap array and references to its parent, left and right childs
2. **MinHeapp:** Implements a MinHeap with given size, stores nodes in a MinHeap array and contains different operations on MinHeap like insert, flip, delete, extractmin, minheap_check, bubbleUp and bubbleDown to update the minheap and to maintain the minheap property.
3. **RedBlackTree:** Implements a RedBlackTree with a root and extrenal node, considered as BlackNULL. It also has implementation for operations like insert, delete, search of rides and also to get the rootnode i.e, getRoot().

Run the code:
1. Unzip the submitted file
2. Open the file and run command `$make`
3. In command line enter the following command to run `$java gatorTaxi input.txt`

Function Prototypes:

1.MinHeapp:

- **MinHeapp(int capacity)**
  Description: Constructor for MinHeapp class. Initializes the heap array and sets the capacity of the heap.
  Return Type: Constructor has no return type or value.

COP 5536: Advance Data Structures
Programming Project Report

- **getMin()**
  Description: Returns the root of the min heap, i.e., the smallest element.
  Return Type: Ride, gives smallest element.

- **parent(int i)**
  Description: Returns the parent index of the given index in the heap.
  Return Type: int, gives parent index.

- **left(int i)**
  Description: Returns the left child index of the given index in the heap.
  Return Type: int, give left child index.

- **right(int i)**
  Description: Returns the right child index of the given index in the heap.
  Return Type: int, gives right child index.

- **minHeap_check(int i, int j)**
  Description: Checks if the min heap property is satisfied between two nodes in the heap.
  Return Type: Boolean, True- satisfies min heap property, otherwise False.

- **flip(Ride x, Ride y)**
  Description: Swaps two elements in the heap.
  Return Type: void, returns nothing.

- **insertRide(Ride ride)**
  Description: Inserts a new ride into the min heap and maintains the min heap property.
  Return Type: Boolean. true- ride Id not found, false – not found.

- **deleteRide(int index)**
  Description: Deletes a ride from the heap at the given index and maintains the min heap property.
  Return Type: void, return nothing.

- **extractMin()**
  Description: Removes and returns the minimum element from the min heap and maintains the min heap property.
  Return Type: int, gives the minimum element from the heap.

- **bubbleDown(Ride node)**
  Description: Helps to maintain the min heap property by performing bubble down operation on the subtree with root as node in the heap.
  Return Type: void, returns nothing.

- **bubbleUp(int i)**
  Description: Helps to maintain the min heap property by performing bubble up operation from the node at index I to the root in the heap.
  Return Type: void, returns nothing.

2. RedBlackTree:

- **RedBlackTree():**
  Description: This constructor initializes the BlackNULL node as external node which is black in color.
  Return Type: constructor returns nothing.

- **insert(int rideId, int rideCost, int rideDuration):**
  Description: This function inserts a new node with the given rideId, rideCost, and rideDuration into the Red-Black Tree. It calls insertFix(Ride node) to balance the tree according to the Red-Black Tree properties.
  Return Type: Ride, It returns the newly inserted node.

- **insertFix(Ride node):**
  Description: This function takes the newly inserted node as input and performs necessary rotations and recoloring to restore the Red-Black Tree properties. It uses a loop to Iteratively fix the tree until the properties are restored.
  Return Type: void.

- **delete(Ride node, int rideId):**
  Description: This function deletes the node with the given rideId from the Red-Black Tree. After deletion, it calls deleteFix(Ride x) to balance the tree according to the Red-Black Tree properties.
  Return Type: void.

- **deleteFix(Ride x):**
  Description: This function is responsible for fixing the Red-Black Tree after deletion of a node. It takes the replacement node (x) as input and performs necessary rotations and recoloring to restore the Red-Black Tree properties. It uses a loop to iteratively fix the tree until the properties are restored.
  Return Type: void.

COP 5536: Advance Data Structures
Programming Project Report

- **leftRotate(Ride x):**
  Description: This function performs a left rotation on the Red-Black Tree with the given node (x) as the pivot and maintains BST property.
  Return Type: void

- **rightRotate(Ride y):**
  Description: This function performs a right rotation on the Red-Black Tree with the given node (y) as the pivot and maintains BST property.
  Return Type: void

- **minimum(Ride node):**
  Description: This function finds and returns the node with the smallest value in the subtree rooted at the given node.
  Return Type: Ride, gives the smallest ride.

- **replace(Ride u, Ride v):**
  Description: This function replaces the subtree rooted at node u with the subtree rooted at node v.
  Return Type: void

- **getRoot():**
  Description: Gives the root node in the RBT
  Return Type: Ride, root node in the RBT.

- **getBlackNULL():**
  Description: Gives the external dummy node.
  Return Type: Ride, external dummy node.

- **searchRide(int rideId):**
  Description: calls the method search(Ride node, int rideId)
  Return Type: Ride, gives the ride if found

- **search(Ride node, int rideId):**
  Description: searchs for the ride in the red black tree based on the rideId
  Return Type: Ride, gives the ride that is found.

- **searchRide(int rideIdStart, int rideIdEnd):**
  Description: calls the method search(Ride node, int rideIdStart, int rideIdEnd)
  Return Type: String, gives a string that contains all rides in the given range of id's.

- **search(Ride node, int rideIdStart, int rideIdEnd):**
  Description: Search's for all the rides in given range of rideId's and stores them in a string.
  Return Type: String, gives a string that contains all rides in the given range of id's.

COP 5536: Advance Data Structures
Programming Project Report

3. gatorTaxi:

- **main(String[] args):**
  Description: The main class will read the arguments for input file in command line and reads each line from the input line and performs each operation. It stores the results in output_file.txt
  Return Type: void.

<u>Complexity Analysis:</u>

- **Print(rideNumber):**
  Time Complexity: O(log n)
  This operation calls the searchRide operation in the RedBlackTree class, which will search for the Ride with given rideId in the RBT. This is dependent on height of RB tree. So, the time taken will be at most O(log n).
  Space Complexity: O(1)
  Takes constant amount of space to store variables.

- **Print(rideNumber1, rideNumber2):**
  Time Complexity: O(log n + S) (S is the number of the triplets printed).
  This operation calls the searchRide operation in the RedBlackTree class with two rideId's as arguments, which will search for the Rides within the range of given rideId's in the RBT. This is dependent on height of RB tree. So, the time taken will be at most O(log n) and O(s) to give all the triplets.
  Space Complexity: O(S) (S is the number of the triplets printed).
  To store the string representation of all the triplets.

- **Insert (rideNumber, rideCost, tripDuration):**
  Time Complexity: O(log n)
  To insert a new ride into the RedBlackTree, we will search if a duplicate Is present which will take O(log n) and then insert with O(1). We will then restructure the RedBlackTree which again takes O(log n).
  For inserting to the minheap if will take O(1) time to insert and O(log n) to bubbleUp in the min heap.
  In total it will take a maximum time of O(log n)
  Space Complexity: O(1)
  Takes constant amount of space to store variables.

COP 5536: Advance Data Structures
Programming Project Report

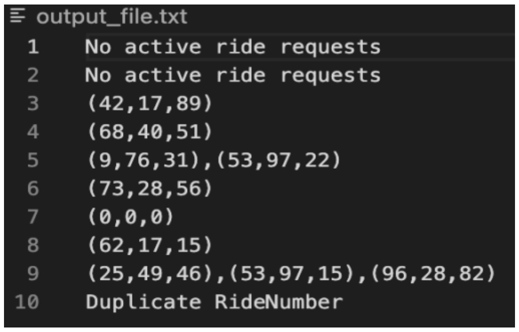
- **GetNextRide():**
  Time Complexity: O(log n)
  In this operation we will return and delete the min ride from minheap which will atke O(1) time to delete and O(log n) time to bubbleDown and restructure the minheap. And also O(log n) time to delete the ride from the RedBlack tree as it is dependent on the height of the tree.
  In total, in worst case it take O(log n) time
  Space Complexity: O(1)
  Takes constant amount of space to store variables.

- **CancelRide(rideNumber):**
  Time Complexity: (log n)
  In this operation we will first search the ride based on the rideId in the RedBlack tree, which takes O(log n). After that we will delete the Ride if it is found in RedBlack tree from minheap and RedBlack tree. Both of these operations will take O(log n) time as these will depend on the height of the RedBlack tree amd height of minheap.
  So, the worst time complexity for this operation will be O(log n).
  Space Complexity: O(1)
  Takes constant amount of space to store variables.

- **UpdateTrip(rideNumber, new_tripDuration):**
  Time Complexity: O(log n)
  We will first search for the ride using rideId which takes O(log n) time. After that for the first case it will take O(log n) to bubbleUp the minheap after changing the duration of the ride. In the second case we will delete and insert a ride in RBT and minheap which will take O(log n) time in both cases. Finally, in last case you will just delete the ride from the minheap and RBT, which will take O(log n) in both cases.
  So, the worst case complexity will be O(log n)
  Space Complexity: O(1)
  Takes constant amount of space to store variables.

<u>Output</u>

The output will be stored in output_file.txt, the below is the output for the testcase given.



```
output_file.txt
No active ride requests
No active ride requests
(42,17,89)
(68,40,51)
(9,76,31),(53,97,22)
(73,28,56)
(0,0,0)
(62,17,15)
(25,49,46),(53,97,15),(96,28,82)
Duplicate RideNumber
```