PROJECT REPORT

(Advanced Data Structure, COP 5536)

Name: VIKAS CHAUBEY, **UFID**: 3511 – 5826, **Email**: vikas.chaubey@ufl.edu

Problem Statement:

Wayne Enterprises is developing a new city. Once the city construction starts, the buildings which are under construction, their construction progress has to be tracked using a software. That software has to be developed using a programming language according to the project guidelines and specification provided. The software should be able to track the progress status of all buildings in city being constructed as well as buildings completed and also about the city construction completion

Project Requirement Specification:

(SPEC.1): Developed software should keep track of Building records each building record should have following fields:

1) **buildingNum**: This number is a unique integer identifier for each building.
2) **Executed_time**: This number defines how many days have been spent on building construction so far.
3) **Total_time**: This number define total number of days required to complete the building construction

(SPEC.2): The software needs to perform following operations:

1) When given **Print (buildingNumber) input** command the software should be able to output the whole building record (buildingNume, executed_time, total_time) if the building record exists, otherwise it will give output as (0,0,0).
2) When given **Print (buildingNumber1, buildingNumber2) input** command the software should be able to output all building records with (buildingNumber,executed_time, total_time) between the range of buildingNum1 and buildingNum2 otherwise it will give output as (0,0,0).
3) When given **Insert (buildingNumber,total_time)** input command the software should be able to insert a new record in the system with buildingNum, total_time and executed_time = 0 as the building record fields.in case of insertion it should produce no output. If the building number to be inserted is supplicate then an error should be given as output and program execution should stop.
4) All the outputs should be written in output.txt file.

(SPEC.3): To handle the Building records and perform the operations mentioned in the (SPEC.2) the software should implement a **Min-Heap** and a **Red-Black-Tree** from the scratch.

1) **Min-Heap Specfications:** The min Heap has to store the building records (buildingNumber, executed_time, total_time) and these records should be sorted by executed_time. Duplicity of executed_time should be handled in Min-Heap with a mechanism.
2) **Red-Black-Tree Specifications:** The Red-Black-Tree will also store all the building records and these records should be ordered by Building Number. The pointers between corresponding nodes with same building records should be maintained in Min-Heap and Red Black Tree.

(SPEC.4): Construction guidelines:

1) Construction work will happen only one building at a time
2) The building for construction is selected with lowest execution_time in building records if execution times are same for multiple buildings then building should be selected on the basis of lowest building number
3) Building construction can happen on an individual building until its completed or 5 days maximum whichever is less. If building construction completed, then it should be deleted from the system and its building number and completion day will be given as output. if the construction did not complete after 5 days then the building record is updated and the execution_time field will be incremented by 5 days.
4) Either a building construction completes or 5 working days have been spent on a particular building then in either case a new building has to be selected for construction.

(SPEC.5): All the inputs are provided from an input file which will have set of operations which are defined in SPEC.2 section, the software needs to identify the operations and perform them. The operations can only be performed if number of days since the construction started matches the number of days given in the input command with operation this time is referred to as global time since the construction started. If global time does not match, then system does nothing.

Assumptions:

The software is developed assuming that the active buildings which are under construction will not exceed aa number of 2000.

Technical Specification:

This software is developed using JAVA programming language.

Software Solution, Architecture and Design:

1) **Building Model Class Implementation: In** order to create a building record as mentioned in (SPEC.1) a building model class with name "Building" is created which has following members.

Variables: all the record variables are private and non-static,

private int buildingNum; //this variable gives building number

private int executed_time; //this variable gives number of workdays spent on the building

private int total_time; //this variable gives the total number of days required to complete the building construction

private Building redBlackTreePointer; //this variable is the reference pointer to the corresponding same building record in the red black tree

Constructors: This class has two constructors.

//No-args constructor

public Building ()

//constructor with arguments to set values of all variables in the object

public Building (int buildingNum, int executedTime, int totalTime)

To set the values of private variables public getters and setter methods are defined in the class to get and set the values of all the Building record variables.

```
//getter method for the building number variable
public int getBuildingNum()
//setter method for the building number variable
public void setBuildingNum(int buildingNum)

//getter method for executed_time variable
public int getExecutedTime()
//setter method for executed_time variable
public void setExecutedTime(int executed_time)

//getter method for redBlackTreePointer variable
public Building getRedBlackTreePointer()
//setter method for redBlackTreePointer variable
public void setRedBlackTreePointer(Building redBlackTreePointer)

//getter method for the total_time variable
public int getTotalTime()
//setter method for total_time variable
public void setTotalTime(int total_time)
```

This Building Class completes the SPEC.1 requirements.

2) **Min Heap Implementation:** In order to implement a good software solution, the first question is of choosing a data structure which can save building records. Software program is supposed to store the Building records in a reversed sorted fashion according to execution_time because buildings are picked for construction with least execution time. In order to minimize the fetch operation time of building records MIN-Heap is an ideal choice because Min-Heap will always maintain the Min element at its root position. Also according to specification **Print(buildingNumber)** and **Insert (buildingNumber,total_time)** operations should not take more than **O(log n)** execution time hence to perform these operations Min Heap is ideal because in Min Heap Node search operation and insert operation both takes O(log n) time only hence Min heap is the ideal data structure choice in this scenario.

   MinHeap.class implements MinHeap data structure from scratch in order to store building records and perform print and insert operations in O (log n) time as well as to fetch next building for construction. MinHeap class has following members

   Variables:

   ```
   //variable to point at the first element with index 1 in array, heap implementation considers index 1 as root element in array
   public static final int FrontElement = 1;
   //array variable using which heap is implemented
   private Building [] minHeapArray;
   //variable which gives current size of minheap
   private int minHeapSize;
   ```

//variable which defines the max size of minheap
private int minHeapMaxSize;

Constructors: The MinHeap Class has only one no argument constructor.

// Min Heap constructor with argument to set max size of min heap which in this case could be 2000
public MinHeap(int minHeapMaxSize)

Methods: Method signatures and their description is as follow:

//returns the heap size of min heap
public int getMinHeapSize()

//this method returns the root element which is minimum element in min heap
public Building getMinimumElement

//this method deletes and returns the minimum element node
public Building removeMinimumElementNode()

//this method returns the right child node of the node which is currently at position
private int getRightChild(int positionIndex)

//this method returns the left child node of the node which is currently at position
private int getLeftChild(int positionIndex)

//this method returns the parent node of the node which is currently at position
private int getParentNode(int positionIndex)

//this function returns true if the passed node is a leaf. Otherwise it returns false
private boolean isLeafNode(int positionIndex)

// this function swaps two nodes of the heap
private void swapNodes(int firstNode, int secondNode)

// this function inserts a new node in heap
public void insertNode(Building element)

//this function min-heapifies the whole heap
public void minHeapify()

// this function min heapify the node subtree at given position
public void minHeapifyNode(int positionIndex)

// this function prints the elements of the heap
public void printHeapElements()

3) **Red-Black-Tree Implementation:** The range operation **Print(buildingNum1,buildingNum2)** which outputs all building records between range of buildingNum1 and buildingNum2, according to project specification this operation can maximum O(log(n)+S) time where n is number of active buildings and S is the number of triplets printed, if that operation is performed using minheap then it cannot be executed within the given time bound because min heap does not follow binary search tree properties hence in order to output all mid-range records the whole tree needs to be traversed on the other hand a Red Black tree follows binary search tree property hence this operation could be performed using red black tree in O(log(n)+S) time.

Hence a red-black tree has to be implemented which will also hold all the building records which minheap holds but this also increases the complexity of delete and insert operations of building records because when a building record is inserted it has to be inserted in MinHeap and red-black tree both.the same also applies to delete operation when a building record is deleted from system then it has to be deleted from both data structures. In order to reduce the delete operation complexity , In the building record a reference should be maintained which points to the corresponding node in red-black tree which has the same record, this way when a building record is deleted from minheap, using the pointer corresponding node could also be deleted from the red-black tree, it will reduce the time involved in searching the building record in red-black tree and deleting it separately.

RedBlackTree.class implements the red-black tree from scratch, and it has following members:

Variables:

```
// root of the tree.
static Node rootNode;
```

Node is an inner class within RedBlackTree.class which represents the model of individual nodes of red black tree

```
// This is inner class which represent a single node in the red black tree
        static class Node {

                // this variable points to node data value.
                Building nodeData;

                // this variable points to left child
                Node leftChild;

                // this variable points to right child.
                Node rightChild;

                // this variable points to parent node.
                Node parentNode;

                // character value in the variable represent red or black color of the node.
                char nodeColor;

                //inner class constructor
                public Node(Building data, char color) {
                        this.nodeData = data;
                        this.leftChild = null;
                        this.rightChild = null;
                        this.parentNode = null;
                        this.nodeColor = color;
                }
        }
```

Constructors: RedBlackTree.class has only one no args constructor to initiate class object

```
        //RedblackTree class no args constructor
        public RedBlackTree()
```

Methods: The RedBlackTree.class has following methods:

// This function adds a new element to the red black tree
Public boolean addElement(Building buildingData)

// This method removes an element from the red black tree
public void remove (Building buildingData)

// this function creates a new Node with building data and assign and left and right children.
public Node createNewNode(Building data)

// this method balances the red black tree in case of any insertion
public static void balanceRedBlackTreeAfterInsertion(Node node)

// this function balances the red black tree in case if any node is deleted
private static void balanceRedBlackTreeAfterDeletion(Node node, String color)

// this method performs Right Rotation.
private static void doRightRotation(Node node)

// this method performs the Left Rotation.
private static void doLeftRotation(Node node)

//this method searches whether the given node is present in the tree if its present then it returns the node data
public static Building searchNode(Node node, int buildingNumber)

//this method performs the range search between two nodes and returns a list of nodes which lie between two given nodes
public static ArrayList<Building> doRangeSearchBetweenNodes(ArrayList<Building> list, Node node, int startBuilding, int endBuilding)

// this method displays the elements of the red black tree.
public static void displayTreeElements()

// this method performs the PreOrder Traversal of the red black tree.
private static void preOrderTraversal(Node node)

// this method finds next greater element than the given node.
private static Node findNextBiggerNode(Node node)

//this method checks whether a given node lies in mid of two given nodes
private static boolean isMiddle(int buildingNum, int startBuilding, int endBuilding)

Glossary: MinHeap.class and RedBlackTree.class Implementation together completes requirements of SPEC.2 and SPEC.3 as mentioned in requirement specifications of project above.

4) **Construction Work and Overall Execution flow Implementation:** The software needs to keep track of everyday construction work according to the guidelines given on building construction in SPEC.4 also at the same time it has to read the input file and perform operations given as input which are mentioned in SPEC.2 in the requirement specification section.

This functionality which drives the overall construction work according to guidelines given is implemented in the risingCity.class. This class has the main method which drives the execution of whole program. It reads the input from the input text file, perform appropriate operations as per input and daily construction work guidelines and generates output in the form of text file.

risingCity.class has following member:

Variables:

//variable to keep track of global time, initialized with 0
private static long time = 0;

//this variable keeps track of the size of input operations list
private static int index = 0;

//this variable keeps the reference of the building node on which construction work is being done currently
Private static Building selectedBuilding;

Methods: this class has following methods:

//this method starts the construction of the risingCity

private static void startConstructionOfCity(MinHeap minHeapObj, RedBlackTree redBlackTreeObj, List<String> timeList,List<String> operationList,BufferedWriter bufferedWriter) throws IOException

//this method inserts the building record in red black tree and min heap both

public static boolean insertBuildingRecord(MinHeap minHeapObj, RedBlackTree rbtObj, Building buildingObj)

// this method writes the processed building output in the output file

private static void printBuildingRecord(Building building, long time, BufferedWriter bufferedWriter)

//Main method which starts the program execution
public static void main (String [] args) throws Exception

Execution flow of the Main method in order:

1) (Java risingCity input_file) command is run from the command line.
2) Program execution starts and risingCity.class starts execution variables like time(globaltime indicator for construction) and index are initiated with 0 integer value and selected building is null.
3) In the main method "input_file" path is passed through args[] array, hence args[0] is read to get the command line argument to obtain the complete path of input file.
4) Input file is read sequentially using a buffered reader and each input line is separated into two values globaltime value and operations statement given in input line. the whole file is read line by line and input global time values and operations are saved in separate lists.
5) Initially MinHeap does not have any records hence , hence construction of city is started using awhile loop which keeps on incrementing global time and keep calling "startConstructionOfCity()" method everytime. In this method if global time is matching with input global time then the corresponding operation will be performed.
   For example, suppose input global time value is 1 and construction time value is also 1 and the input operation is "insert" then since the global times are matching hence insert operations will be performed.

   startConstructionOfCity() method performs all input operations given in input file if global times match. The operations are performed in following manner:

   1) **Insert operation**: this operation will insert the record in both MinHeap and RedBlackTree by calling insertBuildingRecord() method.
   2) **Print (BuidlingNumber):** this operation will search the record in MinHeap and return the whole building record and write the output in output file.
   3) **Print (BuildingNum1, BuildingNum2) :** this operation will output the range of building records between given two arguments and write it to output file

6) Once startConstructionOfCity() execution completed program flow returns to while loop in main method , now the daily construction work is carried out on building which is selected.initially till minheap is empty no building is selected and global counter keeps on incrementing until minheap has some record.
7) When MinHeap has some record then it is selected and the construction work is done on that building for 5 days and new building is selected for work .if building construction completes before 5 days then this building is deleted from MinHeap and redblack tree and building record is written as in the output file.
8) This way while loop within main method keeps on running, each loop run is equivalent to one working day, the while loop runs untill all buildings construction is completed and minheap is empty.
9) The corresponding outputs of input operations and building completion work is written to output.txt file.