COP 5536 Fall 2019 (Advance Data Structures)

Programming Project

Building Construction using Min-Heap and Red Black Trees

DIKSHANT GUPTA

UFID- 6840-1523

Email: dikshant.gupta@ufl.edu

Project Description

Implemented a software for a company ‘Wayne Construction’ using data structures- Min Heap and Red Black Tree, which keeps the track of all buildings under construction for the new city and the building record has following properties: Building_num, Executed_time, Total_time. This will work on one building at a time and selection of a building is based on the lowest executed time or building Id if the two buildings have similar executed time. The selected building is worked on until complete or for 5 days, whichever happens first. If the building completes during this period, its number and day of completion is output and it is removed from the data structures.

Programming Environment

I have used the JAVA programming environment and used Eclipse IDE to run the code. Also, I have tested my code on the thunder.cise.ufl.edu server.

Compile and Run

Type below commands in the terminal after changing your directory to project directory:

Step 1:
The following file can be compiled by simply firing the make command.
Step 2:
run the main class i.e. risingCity.java by

```
sys>java risingCity <input_file_name>
```

*input file having name input.txt is already placed in the zipped file. One can replace the input instructions or can simply place any other file by making sure the correct filename is specified while running the main class.

Structure of Program:

There are four files (classes) namely:

- risingCity.java (main class)
- Building.java
- RBT.java
- MinHeap.java

• input.txt (Input File)
• makefile

Classes and Function prototype:

1. Class risingCity.java

Main function is contained in this class, and the execution starts with this class only. It reads input from a file and process it according to the given command.

Some of the important methods:
main function– reads input from the file and evoke the construction method to start the construction.

• private static void construction(BufferedReader bfr):
It will start the construction process of a building and process the input from the input file to retrieve the timer and will start the global timer and perform operations on the Building object by selecting a particular Building object based on Executed time and Total time of a Building.

• private static void exe(String inst_set, String str):
This is the main method that performs the necessary logical operations like insertion, deletion to Red black tree and and Minimum Heap. Also, perform operation based on the instruction set provided as input which has a different roles:
-Insert : Will insert a building to the data structures.
-PrintBuilding : Will print all the Buildings which come under given range

• private static Building create_building(int building_iD, int executedTime, int totalTime):
This will create a new Building object and will Insert that object into both Min-Heap and Red Black Tree

• private static int get_time(String line):
This function will return the time of a operation of either Inserting or Printing of a Building which is present in the Input file.

• private static void process_line(String line)
This will split the Instruction Set into two parts, first contains the Instruction i.e. Insert or PrintBuilding and second will contain the building_Id and total_time.

• private static void processInput(String line):

Process the Input line and it will split the timer and the Instruction command.

• private static Building find_in_RBT(int building_iD):
Search a particular building in a Red black tree

2. **Class Building.java-** This class consists of several set & get methods and uses a constructor to initialize executed time of a given building.

Its functions are:
• public int getBuildingId():
This function returns buildingId.

• public void setBuildingId(int buildingId):
This function sets the buildingId.

• public int getTotalTime():
This function returns total time (i.e. Total days for the building to be constructed).

• public void setTotalTime(int totalTime):
This function sets the total number of days count for the Building to complete its work.

• public int getExecutedTime():
This function returns executed time.(i.e. Gives the number of Days the building is worked on since it is being inserted in the Heap)

• public void setExecutedTime(int exTime):
This function sets the execution time.

3. **Class: RBT.java**
This class implements the red black tree with all the necessary functions. All the Building ID's and the execution time of Building ID's stored in this tree.

Methods:
• private void insert(Node node)
Inserts a new node to the Red Black Tree

• private void insert_fix(Node node)
This will take as argument the newly inserted node and maintain red-black property

after insertion.

• void rotateLeft(Node node)
This performs a left rotation operation on the Red Black Tree.

• void rotateRight(Node node)
This performs a right rotation operation on the Red Black Tree.

• void translate(Node target, Node with)
This method is called by the delete() method when the parent and child pointers are to be adjusted.

• public boolean delete(Node z)
This method deletes a particular Building key from a Red Black Tree. If the building is deleted, then it returns true, else, it returns false.

• void fixDelete(Node p)
After the deletion of a node, the Red Black Tree would have to be rebalanced.
public Building findBuilding(int buildingId)
This will search the building in red black tree by the given building ID

• public Building findBuilding(int buildingId)
This will search the building in red black tree by the given building ID. This will be executed in O(log(n))

• public void findBuildingsBetweenRange(Node root, int buildingId1, int buildingId2, List<Building> result)
This will search all the building ID between the given range and it uses the binary search property of a red black tree to traverse through all the nodes in order to find the resulting property. This is executed in O(log(n)+S) where n is number of active buildings and S is the number of triplets printed.

4. **Class: MinHeap.java**

This class implements of the minimum heap tree. It stores the executed time for which each Building has been constructed.

Methods:
• public void insert(Building element)

This method inserts a Building element in the form of a node into the heap. If the newly inserted node is smaller, then it is swapped with the parent and it is made the current node and this continues until the minheapify property is satisfied.

• public boolean isEmpty()
This method return either True or False based on whether Heap is empty or not.

• private void minHeapify(int i)
The node of the index i is compared with its children. If the parent node is smaller than any of the children then it is swapped with that child. Then the minHeapify operation is performed on the exchanged child again to ensure Min heap property.

• public Building extractMin()
This will return the min element from the Heap.

• private void swap(int i, int j)
Swaps two nodes in the heap. It is used in the minHeapify() function.

• private static int parent(int i)
Takes in the position of the current node and returns the position of the parent of the current node.

• private static int leftChild(int i)
Takes in the position of the current node and returns the position of the left child of the current node.

• private static int rightChild(int i)
Takes in the position of the current node and returns the position of the right child of the current node.