ADS Project Report Fall - 19

Chandan Chowdary Kandipati: 6972-9002
chandanc.kandipa@ufl.edu

Project Description:

Waynes enterprises is developing a new city by constructing the buildings in the city. The buildings needs to be managed by using a software. The building in the software consists of the following fields:

buildingNum,executed_time,total_time

The operations that needs to be performed in the software are:

PrintBuilding(buildingNum) – prints the triplet(buildingNum,executed_time,total_time) of the buildingNum1

PrintBuilding(buildingNum1,buildingNUm2) – prints all the triplets in the range of buildingNum1, buildingNum2

Insert(buildingNum,total_time) – inserts the buildingNum triplet to the software.

Data Structures used:

Minheap and red black trees are used . Min heap is built on the execute_time of the building and red black trees are built on the building number of the building.

Programming language used:

I have used Java as my programming language to code the project.

Assumptions made:

The number of buildings that are given as input is less than 1000.

How to run code:

All the files are executed using the makefile and class files are generated. To execute the program following command is used

java risingCity <path_to_input_file>

Output:

The output produced is a file with the name output_file.txt . This file contains the building triplets and the completed buildings.

Object oriented approach:

To implement the building management software, red black trees and min heaps are used. Minheap class is created with minheap implementation. Red black trees is implemented in a class named RBT. The nodes of the heap are represented by a class named Node and the nodes of RBT are represented by a class named RBNode.

Overall Approach:

As the classes of both the data structures are different, a correspondence is maintained between both the classes nodes. Node is used in the minheap class to implement the minheap and RBNode is used in red black tree nodes.

Whenever a new building is being inserted first the building is inserted in the minheap by constructing a new node and inserting into the heap array. Then minheapify is called to minheapify the heap. Once this process is done, the same needs to be inserted into the red black trees as there should be a correspondence between both the trees. This is achieved by inserting the newly created node into the node of the red black trees.

Whenever an insert is made into red black tree, it is restructured so that the tree satisfies the properties of red black trees.

The days counter is incremented from 0. Everyday a check will be performed to see if there is an operation to be done on that day(like insert or print). If there is an insert then the node is inserted in the minheap and also in the red black tree. The days counter is incremented irrespective of the operations. Whenever all the buildings are completed constructing then the program will terminate with heapsize and rbt size 0.

Structure of the project:

risingCity(project folder) →

→ risingCity.java

→RBT.java

Function prototypes of risingCity.java:

1)`public int parent(int I)`

returns the parent of the node at index I

2)`public int leftChild(int i)`

returns the leftchild of node at index i

3)`public int rightChild(int i)`

return the right child of node at index I

4) `public void swap(int x, int y)`

swaps the nodes at position x and y

5)`public void insert(Node node)`

inserts the node into the minheap

6)`public void minHeapify(int I)`

performs minheapify operation on the heap

7)`public Node removeMin()`

removes minimum node from the heap

8)`public void minHeapify()`

caller function to the minheapify method

9)`public void print()`

prints the nodes in the heap

10)`public boolean buildingIsCompleted(Node node)`

check if building is completed

11)`public Node processCurrentBuilding(Node node)`

executed time of current building is incremented by 1

12)`public void printHeapArray()`

prints the heaparray as it is

13)`public static void main(String[] args)`

main entry to the program in which all the process is done

Function prototypes of RBT.java:

1)`private RBNode findRBNode(RBNode node, RBNode rootNode)`

finds the red black node in the tree by taking node and rootnode as arguments

2)`public void insert(Node temp)`

caller function to insert the node into the redblack tree

3)`private void insert(RBNode node)`

inserts the node into the red black tree

4)`private void adjustTree(RBNode node)`

adjusts the tree after inserting a node in to the red black tree to make it satisfy the properties of red black tree

5)`public void leftRotation(RBNode node)`

performs left rotation on the node

6)`public void rightRotation(RBNode node)`

performs right rotation on the node

7)`public void swap(RBNode first, RBNode second)`

swaps the two nodes

8)`public void delete(Node temp)`

caller to delete the node in the rbt

9)`public boolean delete(RBNode node)`

deletes the node from the rbt

10)`public void adjustDelete(RBNode node)`

make adjustments to the tree when deletion is performed to make it satisfy the properties of rbt

11)`RBNode findMin(RBNode node)`
finds the minimum of the nodes in rbt

12)`public List<Node> nodesPrint(RBNode node, RBNode node1,RBNode node2,List<Node> nodesList)`

returns the nodes of the tree as a list that are presented in between the two given nodes

13)`public void inorder(RBNode node)`

performs the inorder traversal of tree

14)`public void print(int buildingNum)`

prints the triplet of buildinNum to the file

15)`public void print_range(int num1,int num2)`

prints the nodes in the range num1 and num2 to the file

Prototype of Node class:

```
class Node{
	int buildingNum;
	int executed_time;
	int total_time;
	int initDay;
}
```

Prototype of RBNode class:

```
public class RBNode {  //Class that represents a node in the red black tree
	public Node building = new Node(0,0,0,0);
	public int color = black;
	public RBNode left = extNode;
	public RBNode right = extNode;
	public RBNode parent = extNode;
}
```