COP5536:Advance Data Structure

Project: Rising City

Name: Tanmay Porwal

UFID: 6658-1115

Email Id: tanmayporwal@ufl.edu

Project Description:

A simplified simulation of Linux's scheduling algorithm (Completely Fair Scheduler) has been implemented. The objective of this project was to help in building of a city giving fair chance to every building present in the city and help in complete scheduling of the buildings in the city.

For implementing this project, red black tree and min heap is used where min heap takes care of the priority(on the basis of a parameter called execution time) to be given while constructing the city.

Input:

An input file which consists of building number and total time required for its completion and also a global time which gives information about when the particular building must be encountered. Each building is given a building number, total time and execution time.

Data structures and Modules used -

Data Structures:

- The heap implementation required the use of a data structure that can be accessed by indices so that the parent and the children nodes can be accessed by the indices (like i, 2i+1 and 2i+2). In addition, the implementation also required storing the building number of the building to which the execution time belongs. So, an ArrayList data structure was used to implement this.

- The red black tree data structure require a node which has parent, left child, right child, colour.

RBT.java:

RBT.class:

This class has the following functions to operate on a Red Black Tree.

bst(Node h): This method is used for insertion in the red black tree.

checkRbt(Node c): This method is used to check for any violation after the insertion of every node in insertbst method.

insertbst(Node newNode,Node current): It is called by bst and checks for a binary search tree property and after taking the right place of the node this is calling checkRbt to check for Rbt violation.

minimum(Node x): Used to find the successor while deleting the node.

rbtDelete(Node z): This method needs to delete the node whose construction is complete.

rbtTrasnsplant(Node u,Node v): This method is used while deleting a node to transplant the sub tree to the position of the node needed to be deleted if required.

rbtXYr(Node gp, Node p, Node u): This handles the XYr case while inserting in rbt tree.

rbtcheckDelete(Node x): check the rbt after any deletion.

rotationL(Node gp, Node p): Does the left rotation at a desired node in the rbt.

rotationR(Node gp, Node p): Does the right rotation at a desired node in rbt.

searchBnum(int bNum, Node root): This method finds and returns a node with a given building number.

Two Node variables are also created:-

root: this is the root of rbt

tnil:it is the null root or external node used in rbt for different purposes.

RBT.java has a inner class Node.class:

It has a constructor which creates a new node with two parameters. Also it assigns values to node such as **bNUm:**Building Number**, colour**, **execTime:**execution time,**left:**left child,**right:**right child,**parent:**parent of the node, **totalTime:**Total time required by the building to complete.

Also a **nil** is defined as an external node.

minHeap.java

minheap class:

insert(Node newNode): It is used to insert a node in the min heap.

checkHeap(ListHeap): It checks for any property violation of min heap after every insertion.

swap(int p, int c): swaps the node on the two indices with each other.

removeMin(): It removes the minimum from the min heap.

checkHeapadv(int p): It checks the min heap after a remove min operation.

risingCity.java

risingCity class:

Print(Node x): It is used to print when PrintBuilding with a single parameter is occurred in the input file.

Print(int x, int y, Node root): It is called when PrintBuilding with 2 parameters is called.

printTree1(Node h, int x, int y): It does an in-order traversal and only prints between the range given in the PrintBuilding statement.

main(String[] args): This is the entry point of the whole project which reads the input file and recognizes all the operations such as insert and Print building also it inserts in the min heap and rbt. It takes care of the global timer and also ensure that at one go a building is only constructed for 5 days and is switched with the one with the lowest execution time which is stored in the 0^{th} position of min heap.