COP 5536: Advanced Data Structures

Email: fnadar@ufl.edu

FLOURA ANGEL NADAR **UFID: 2303-6958**

Project Description:

Wayne Enterprises is developing a new city. They are constructing many buildings and plan to use software to keep track of all buildings under construction in this new city. A building record has the following fields: buildingNum: unique integer identifier for each building, executed_time: total number of days spent so far on this building,total_time: the total number of days needed to complete the construction of the building.

The needed operations are:

1. Print (buildingNum) prints the triplet buildingNume,executed_time,total_time.

2. Print (buildingNum1, buildingNum2) prints all triplets bn, executed_tims, total_time for which buildingNum1 <= bn <= buildingNum2.

3. Insert (buildingNum,total_time) where buildingNum is different from existing building numbers and executed_time = 0. In order to complete the given task, you must use a min-heap and a Red-Black Tree (RBT).

Input:

Input test data should be given in the following format.

time0: Insert(buildingNum, total_time)

time1: PrintBuilding(buildingNum)

time2: PrintBuilding(buildingNum1, buildingNum2)

(*input command for Print() -> PrintBuilding())

Output:

Completion day or status of completion of buildings

Running code:

The class files are already present, otherwise can be created from java file using command

- make.
- java risingCity inputfile_name

Data structures and Modules used - Building Construction

Data Structures:

- The heap implementation required the use of a data structure that can be accessed by indices in order to retrieve data (like (i-1)/2 for parent, 2i+1(for left child) and 2i+2(for right child)). In addition the implementation also required storing the executed time of the process, so an Array data structure was used to implement this with a maximum size of 2000. It uses HeapNodeFeatures to store data of each node.

- The red black tree data structure required to store the process id i.e the building id, total time for execution, color of the node, etc in addition to the parent and child pointers (the left and right node of each node). An array of nested class object RbtNodeFeatures was used to reference it as a node of the tree.

- Both MinHeap and RedBlackTree, have an object reference to each other. This is done to to maintain pointers between corresponding nodes in the min-heap and RBT.

Class Functionalities:

risingCity.java (Main class):

This class is for printing the status of the work completed on a building, it also checks whether further construction of any building is required, and inserts the initial data into Minimum Heap and Red Black Tree.

The class has the following functions to operate on.

- continueBuilding(): If there is not input at a particular time, then the building which has the least execution time sorted in min heap is picked and 5 unit work is done on that.
- insertBuilding(): New building is inserted into min heap and red back tree.
- incrementCounter(): The global counter is incremented for every unit of work done in a day or every passing day.

RedBlackTree.java:

An RBT should be used to store (buildingNums,executed_time,total_time) triplets ordered by buildingNum. A red–black tree is a kind of self-balancing binary search tree having additional information on the color of the node.

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

- avlRRotation() and avlLLRotation(): Inorder to fix the LR problem in Red black tree if there are two red color nodes after each other, the first function performs a right or anti - counterwise rotation, whereas the next is used for clockwise this way the insertion problem is fixed.

- flipColor(RBNode rbn): Two node color are swapped keeping everything else at the same position. Repeated with fixing function till the root node.

- insertReBalancing(RbtNodeFeatures node): Various cases of insertion depending on the color of the node of the parent. The node is fixed with respective to the parent node and then the grandparent node if the node is a black node, as the tree may encounter black node deficiency.

- deleteReBalancing(RbtNodeFeatures node): Cases based on the sibling color of the node to be deleted. Conditions such as if parent node other child(sibling node) is red, left rotation is required. If parents both children are black, recolor both. If only right child of parent is black, recolor and right rotate the node, otherwise if both right and left child is red, recolor and left rotate.

- intervalSearch(): Search the tree in O(log n) for all nodes lying in a range of input.

- decreaseHeight(): When the nodes left or right child at the bottom of the tree is null, then the height of the tree can be decreased.

- delete(): This operation deletes the node selected. This further calls deleteReBalancing().

MinHeap.java:

In a Min Binary Heap, the key at root must be minimum among all keys present in Binary Heap. This class performs construction and operation over a Min Heap. Heap uses array data structure. This class has the following functions:

- parent(int pos): Takes in the position of the current node and returns the position of the parent of the current node.

- swapNodes(int i, int j): Swaps two nodes in the heap. It is used in the heapify() function to swap nodes which does not follow the minimum heap rule.

- heapify(int pos): This operation is performed when an insertion or deletion operation is done on heap to rebalance the heap when the min heap property is violated.

- nodeInsert(HeapNodeFeatures p): inserts a process in the form of a node into the heap and checks if the condition of min heap that the execution time of node above the present node location is less than it.

- deleteMin(): the root node is deleted and the last node data is replaced and the heapify() function is called at index 0.

HeapNodeFeatures.java:

This class is used to define the node properties of Min Heap, also has the constructor function to initialize the node and assign its key.

RbtNodeFeatures.java:

This class is used to define the node properties of RedBlack tree, uses an initial constructor.

Structural Flow of Program:

- The file from where the input is read is inputted in the Main function and read using Buffer Reader and File Reader, followed by reading the command line by line. Incase the input time and the global counter maintained which is incremented for every passing day matches then the command is read.
- If the command is an insert command then the building data is inserted after present work in progress, if it is print command it is executed while pausing work in progress.
- In insert command, the node data is inserted in Min heap and RedBlack tree. In minHeap, it is sorted on the basis of execution time whereas in Red black on basis of building Id.
- Each of the Min Heap and red Back Tree nodes have an object reference to each other(pointer in C++ term). Whenever the condition for them is not satisfied like for MinHeap the execution time of child should be more than parent otherwise it swaps and rebalance itself. In red black tree, if there are two red color nodes following one after the after then insertRebalancing is called.
- Once the building is completed constructing, then the output on what day is completed is printed, followed by removing it from both the trees.
- Incase the building is not completed but the time interval given to it is over, then the building is removed from execution that is deleting it from the trees and then re-inserting it back into both the trees with updated execution time.

- In short, 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. Otherwise, the building's executed_time is updated.
- The building's executed_time is updated. Wayne Construction selects the next building to work on using the selection rule. When no building remains, the completion date of the new city is output.
- The output is stored in a text file called "output.txt" using File Writer library.
- Building id should not be redundant, it should be unique.