ADVANCED DATA STRUCTURES

Wayne Enterprises developing a new city

November 28, 2019

Introduction:

The following project aims at implementing specific data structures- namely, Minheap and Red black trees.

Minheap data structure:

- Every parent in a minheap is smaller compared to each of its siblings.
- When an element is inserted into a minheap, it is checked if the element has violated the property of a minheap or not. We use "**reverseHeapify**" in our code to eliminate such violations.
- Deletion is done from the root of the minheap, i.e. the first element is removed (if minheap is maintained in array format). First element of the minheap is replaced with last element, and the last position is made null. "**heapify**" is called to eliminate the violations which could have seeped in due to this change

Red Black trees:

- Red black trees exhibit similar functionality in the sense how elements are stored in a BST (Binary Search Tree)
- Every node is colored either red or black, which is considered true and false respectively in the code. A red node cannot be preceded or succeeded by another red node, and the number of black nodes counted while traversing till the external nodes remain constant for the entire tree
- Insertion and deletion have special cases, which are specifically mentioned in the comments section of the code

Structure of the project

The project is named risingCity.
It has 5 classes namely:

1. RisingCity.java – This is the main class which controls the execution of the program.
    - insert(int, int) – This inserts the nodes into both minheap and RBT.
    - find(String[]) – search for the type of operations given in input file
    - executeOperation(String[]) – perfoms actual execution of either of print or insert operations
    - rangePrint(RBT.Node, int, int) – gives the output for a range of buildings

2. minheapProp.java – This class enlists the following fields for a given building
    - buildingNum
    - executed_time
    - total_time
    - correspondingRBTnode
    - Also, a constructor initializes these fields

3. minheap.java – This class is responsible for creating the minheap which stores executed_time as its keys. Ties are broken by buildingNum.
    - arr – this is a global array which stores the elements of minheap
    - parent(int) – gives parent index of the element
    - left(int) – gives left child index of the element
    - right(int) – gives right child index of the element
    - insert_heap(minheapProp) – inserts into arr elements of type minheapProp
    - reverseHeapify(int) – rearranges the array in accordance with heap property after insert
    - delete_heap() – deletes the minimum element from the array, i.e. the first element
    - heapify(int) - rearranges the array in accordance with heap property after delete

4. RBT_node.java – This class enlists the following fields for a given building
    - buildingNum
    - executed_time
    - total_time
    - correspondingminheapnode
    - Also, a constructor initializes these fields

5. RBT.java – This class is responsible for creating the Red Black tree structure, which stores buildingNum as its keys. Since buildingNum is unique for every building, there are no race situations in this scenario.
    - Node class – this class contains fields like left, right, parent, color and node of type RBT_node
    - rotateLeft(Node) – Rotate the tree using LL rotation
    - rotateRight(Node) – Rotate the tree using RR rotation
    - RBT_insert(Node) – insert a new node to RBT
    - fixInsert(Node) – Rearranges the tree in order to eliminate violations after insert
    - RBT_remove(int) – Remove a node from the RBT
    - delete_fix(Node) - Rearranges the tree in order to eliminate violations after delete
    - nodeExists(Node,int) – Check if the node exists in RBT
    - search(Node,int) – Check for a node in RBT
    - findMin(Node,Node) – Find the minimum of two given nodes
    - findSuccessor(Node) – Finds the replacement of a given node after deletion

Understanding of the project

The basic understanding of this project was implementation of data structures like minheap and Red Black trees. Also, all the functionalities were implemented to the best of my knowledge with algorithm references from Introduction to Algorithms by Cormen, Leirserson, Rivest and Stein.