COP 5536 – Advanced Data Structures

Project Report – Rising City

Kartik Rode

UFID:8971-1791

rode.kartik@ufl.edu

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.

Input:

The input file has different instructions denoting what action needs to be performed. It can either be Insert or one of the two variant of the Print statement. Insert method contains BuildingID and totalTime as parameters while print statement can take two arguments.

Data Structures:

- The Min Heap is implemented with arrays. The parent value is accessed by i , left child by 2i and right by 2i+1. We have implemented this as array of objects where each object has its own properties like BuildingID, executionTime and totalTime where the min heap is implemented on executed time of the buildings.
- The Red Black tree is also used to store BuildingID, executionTime and totalTime in which we design the binary search tree on the basis of BuildiingID and also contain the color value and pointer to left and right child.

minHeap.java

This class exhibits the functionality of a min heap and uses array data structure for it. It stores buildings as different objects in the array. This class has following functions:

- Parent(int pos): Takes the index of the current node and returns the position of its parent in heap.
- leftChild(int pos) : Takes the index of the current and computes the left child of the node specified.

- rightChild(int pos): Takes the index of the current node and computes the index of the right child of the given node.
- Exchange(int I, int j): Swaps the two nodes specified at I and jth index of the heap array.
- RemoveMin(): Returns the minimum element of the heap and calls heapify on the first element of the heap.
- minHeapify(int pos):This operation is performed when we delete the minimum element from the heap and we need to rebalance the minHeap.
- constructionInProgress(Node2 obj): This method is responsible for increasing the execution time of the building that is currently being worked on . If the current building is null, we call removeMin Method to extract the min element from heap.
- Insert(Node node): This is used to insert a new node in the minHeap. It uses exchange method when the value of parent is larger than child node.

RedBlackTree.java

This class exhibits the functionality of a red black tree. It also stores buildings as different objects. This is a binary search tree which uses buildingID for comparison.

- Insert(Node node): This method is used to insert a node in the RBT. By default it colors the node as Node and then call TreeFixer method to fix the colors and to rotate the nodes.
- TreeFixer(Node node): This method is used to fix the colors and rotate the nodes in order to fix the violations of RBT . Based on the position of the node, it calls rotateToLeft and rotateToRight.
- rotateToLeft(Node node): This method rotates the RedBlackTree to the left.
- rotateToRight(Node node) : This method performs right rotation operation on the Red Black Tree.
- Swap(Node target, Node x): This method is called by the delete() method when the parent and child pointers are to be adjusted.
- findNode(Node findNode, Node node): This method is used when we want to find a particular node with respect to root.
- Delete(Node z) : This method is used to delete a particular node from RBT. In order to replace the deleted node we can perform swap or find the minimum element of the right subtree of the node.
- treeMinimum(Node subtreeRoot): This method is used to return the smallest element from the right subtree.
- deleteFixer(Node x): This method is used to fix the colors and rotations after deletion.
- printTree(Node node): This method is used to print the tree in in-order traversal.

risingCity.java

This is the main class of the solution. It takes file as an argument and parses the input and perform the operations accordingly. It has the driver code for the rest of the operations. Following are operations performed by it:

- Main method: This method instantiates objects of redBlackTree and minHeap class. It then parses the input present in the input file and sets localCounter and globalCounter based on it. This is used to decide when the input file needs to be parsed again. At each value of globalCounter, it calls constructionInProgress method so that we are always working a building and increasing its execution time. The condition to break from while loop of globalCounter is to check whether the RedBlackTree is empty and there is no further input to read. It also checks for the condition whether a building has been worked for 5 days or if the total time is equal to executed time. Based on the operation specified after the time value in the file, we perform insert or print operation method on the data.
- Print(int x): This method takes the buildingID as the input and print its current executed time and total time from the data structures in which we have stored it.
- Print(int x, int y): This method two buildingID as input and prints all the buildings that have buildingID either equal to them or in between them. This method stores the result in an array of Nodes. First, it finds whether the node is present in RBT by using FindNode method. If it is found, it is added to the array and finally the results are printed.

Node.java

This is the class for a building for RedBlackTree which has buildingID, executed time, totalTime and color. It has a constructor which initializes the values for different data members.

Node2.java

This is the class for building value stored in MinHeap. Along with building values it also has a reference to Node class object which has the same values.