Rising Building Project Report
COP 5536 - Advanced Data Structure

Ramandeep Singh
UFID 8019-7991 ra.singh@ufl.edu

Change History

Version	Date	Changes	Author
0.1	26 Nov 2019	Initial Draft	Ramandeep Singh

TABLE OF CONTENT

1. Introduction

This project is a software to keep track of the building to be constructed. A building record has three fields - Building Number, Executed Time and Total Time. These records are maintained in min-heap and red-black tree data structure. Insert operation inserts the record in mishap, whereas print building prints the record to an output file.

2. Objective

The objective of this software is to keep track of the building construction done by Wayne construction. Min heap is implemented which keeps the record in such a way that it follows the min-heap property based on executed time. In case we have two or more buildings with the same execution time, the building is picked based on a lower building number. The red-black tree has building records based on building numbers only. All the insert and remove operations are done in min-heap and are coordinated with a red-black tree to satisfy the property of min-heap and red-black tree simultaneously.

3. Code Functionality

Node is defined as an array of objects which store the information of the building.
The global counter variable is maintained to keep a check on the total number of days construction has been done.
The reset counter is maintained and is reset every 5 days. It keeps a check on the number of daily work is done on a particular building.
Class main has the min-heap functionality implemented

1. Swap will swap the position of parent with its children based on the condition
2. Insert will insert the record in the min-heap.
3. minHeap will run through the whole tree and call createMinHeap function to maintain the min-heap property by swapping the building to be

constructed based on execution time and building number if execution time is the same.

4. Remove function will remove the building from the min-heap if the construction is complete.

createMinHeap Functionality

1. If the node to be worked on is a leaf node do nothing.
2. If not check the execution time of the root node. If the execution time of the root node is less than both its children, the construction continues.
3. If the execution time of parent is more than the execution time of any of the children it is swapped by the child which has less execution time. It loops through the whole tree to make sure min-heap functionality is maintained.
4. If the execution time of the parent is more than the execution time of its children but the execution time of its children is the same, the one with a small building number is swapped with the parent. It loops through the whole tree to make sure min-heap functionality is maintained.
5. If the execution time of the parent is equal to the execution time of its children, then the parent is swapped with the child with lower building number It loops through the whole tree to make sure min-heap functionality is maintained.

Red Black Tree Functionality

1. First node inserted will be a black node.
2. SearchBuilding method searched the node to be deleted which comes as an in parameter. It returns the node to delete function if found.
3. InsertNode inserts the new node into the red black tree. Based on the position where it is inserted and its parent, grandparent and uncle's color we do rotations to maintain the red black tree functionality.
4. Fix function has all the rotations calling functions and is called by Insert-Node function to fix once a new node is inserted.
5. rotateLeftChild does the left child rotations and is called by the fix function.
6. rotateRightChild does a right child rotation and is called by the fix function
7. Transfer does a swap function between two nodes.
8. PrintTree function prints the buildings in range. It called the LoopPrint

9. LoopPrint returns the triplet (building number, execution time and total time to the printTree function.
10. deleteNode deletes a particular node from Red Black Tree.
11. deleteNodeFixup - Once a node is deleted it does a fix-up to maintain the red black tree property.

Main Functionality

1. A global counter is maintained and incremented by 1 denoting 1-day construction on building.
2. It reads the input file. The first value in the input file denotes what happens on that particular day. It can be an insert or print building.
3. It loops through the whole input file until all lines are read.
4. After the first insert happens, the construction of the building starts.
5. In case of a new insert at a point when construction is happening in another building, it will continue till 5 days.
6. After every 5 days, it checks whether we have another building with less or equal execution time.
7. If it finds a building with less execution time it swaps with that building and construction starts for that building for the next 5 days.
8. If it finds a building with execution time equal to its own execution time it checks the building numbers. Since the building numbers are unique, the one with lower building numbers among all with the same execution time his picked.
9. The construction of a building will continue unless its execution time is equal to total time.
10. Once the construction is complete remove function is called in min heap which will remove the building from min-heap.
11. Construction on a building happens for 5 days. There can be a situation when construction is completed before 5 days. Then the building is removed from min-heap and the next building will be picked based on the execution time.