# 

# 

# COP 5536 Fall 19 Programming Project

Milind Jayan

UFID: 81689617

UF mail: [milindjayan@ufl.edu](mailto:milindjayan@ufl.edu)

Submitted to:

Dr. Sartaj Sahini

1. Background

1.1) Problem Description:

The aim of the project is to help manage the construction of buildings using a minheap and a red black tree. Construction can happen on only one building at a time. The building with the least executed time is worked upon till completion or till it has undergone construction for at least five days. If the building is not complete within five days, the building with the least executed time is picked from the heap and worked upon. Ties are broken by selecting the building with the lesser building number.

1.2) Input and output requirements

We are required to create a make document which creates an executable. The name of the executable is risingCity. The program has to read in the file name from the command line and write the output to a file called “output\_file.txt”

The execution of the makefile should compile the code and it should run using the following command:

* java risingCity <input\_file\_name>

1.3) General approach to solve the problem

The project consists of a minheap used to store the building objects ordered by the building’s executed time. The Red Black Tree is used to store the building objects ordered by the building number. Whenever an insert command is encountered, it is first inserted into the red black tree and then into the min heap.

Since, the insertion happens into the red black tree initially, if, a building with the same building number is encountered, the program will exit. If the insertion into the red black tree succeeds, the building is inserted into the min heap.

The building in the root of the minheap is the always chosen to build upon. Once, a building is chosen, it is built on for 5 days or until completion. The next building is chosen only if the current building under construction is built upon for at least 5 days.

Once, a building is completed, the building is removed from the min heap and the red black tree.

I used Java for this project.

2)Structure of the Project and Classes:

The project has 4 main classes:

1. MinHeap
2. Driver
3. RBT
4. Building

2.1)MinHeap

Methods:

* checkEmpty – return true if the min heap is empty
* findParent – returns the parent of an element
* leftChild – returns the left child of an element
* rightChild – returns the right child of an element
* insert – inserts an element into the heap
* swapPos – swaps the position of two elements
* removeMin – removed the min element from the heap
* maintainHeap – maintains the heap property after removing an element or after work has been executed on the element for 5 days.
* build – increments the executed time of the building on the root by one. And checks weather the executed time is equal to total time required for the building to execute and weather the building has executed for 5 days. If the total time and the executed time are equal, the building’s executed time, the building number along with the global counter at that time is printed out. If the building has executed for 5 days, the maintainHeap method is called, which will replace the value in the root with the next smallest building. The function returns a Boolean value of either True or False
  + True is returned if the building being worked upon currently has executed for five days or the building has completed execution.
  + False if the building being currently worked on has not completed five days of construction.

Whenever the building undergoing construction has reached 5 days and is still not complete, then the if some buildings that were read during the meantime would be inserted into the heap.

2.2)RBT

Methods:

* getBuildingNumber – return the building number for a building of the inputted RBT node
* findImbalance – return the imbalance after inserting a node to the RBT
* RRotation – performs right rotation with respect to the node inputted.
* LRotation1 – performs left rotation with respect to the input node.
* maintainTree – recolors and performs rotations if needed after inserting a new node to the tree
* insert – insert a new building into the tree
* findDegree – finds the degree of the given node.
* delete1 – delete a node from the red black tree
* rightSubTreeMin – returns the minimum element from the right sub tree
* printRange – returns the all the buildings in the given range. For every building in the given range, the building’s building number, executed time and total time are printed.
* findNodes – searches for a building in the red black tree. If the building is present, it will print the building number, the building’s executed time and the building’s total time.

2.3)Building

Building class initialisation.

2.4)Driver

Methods:

* findOperation – returns the operation in the current input line

3)Building construction implementation:

The code works by reading the input file as a command line argument. The input text file is read line by line. The following conditions can happen when a line is read in:

* If the global counter at that point of time is less that the input time read in.
  + The construction operation will happen until the global counter is equal to the time read in. The construction operation will increment the executed time of the root of the minheap by 1. The construction method checks the following cases:
    - If the building has completed execution, it will be removed from the data structures
    - If the building has executed for 5 consecutive days, the maintainHeap method is called and the building with next least executed time is trickled up to the root of the heap.
* If the global counter is equal to the read in time. According to the command, that is read in, the following cases can happen:
  + If the command read in is PrintBuilding, all buildings pending construction in the given range are printed out. If there is only one building given, if the building is present, it will be printed out. If there are no buildings in the given range, then,

(0,0,0) will be printed.

* An extra check is kept inside the execution block of the PrintBuilding commands, which is to handle an edge case where a building is deleted when a print statement is encountered, to ensure that the printing happens before the building is deleted.
  + If the command is insert. Based upon the value returned by the build function, the following two cases can occur
    - If the value returned is true, the new building read in from the current line is inserted into both the data structures.
    - If the value returned is false, the new building is not inserted into the data structures, but instead is inserted into a list named as pendingBuildingInsert. This step is required because if the building is inserted into the data structures directly, it would move to the top of the minheap before the building currently being worked on has finished 5 days of construction

The construction of the building will happen irrespective of the command read in.After all the lines are read in, the program will continue executing until the minheap is empty.

Edge Cases:

1. If a print command comes in at the same time as that when a building completes, the print command would be executed first. Then the building would be removed.

4)Instructions to run the code:

The make file present in the submission folder would compile the code. Run the following command to execute the make file:

make -f Makefile

After compilation using the make file, run the following command to run the program:

java risingCity {file\_name}

5)Time Complexity:

MinHeap – The heap has ceil(log2(N+1)) levels. The insert and delete swaps at most once per level. So the order of complexity is O(log N).

Red Black Tree- All red black tree queries are executed in O(log(n) + s) where S is the nodes visited.

Tools Used:

Programming Language – Java

IDE: IntelliJ IDE