COP5536: Advance Data Structures Fall 2019

Project Report

Keerthi Suresh

keerthi.suresh@ufl.edu

UFID: 4701-4942

Problem understanding and assumptions:

- There are two tasks Insert, PrintBuilding(x, y), PrintBuilding(x).
  Insert task/command will put a new building for construction.
  PrintBuilding(x, y) prints all the active Buildings from range x to y.
  PrintBuilding(x) prints just an x Building.
- Construction of Building goes for 5 continues days/until the remaining completion time or whichever is less.
- If Building is completed we should print its buildingId and time at which it is completed.
- If there are a PrintBuilding command and completion of the building comes at the same timestamp PrintBuilding takes precedence that is first prints all the undergoing building and deletes.
- After 5 seconds we perform extract min and select the building with least an exceuted time.
- If there are multiple building with least executed time then tie break strategy takes building with lowest building ID(assuming building id is unique if not program terminates with a proper message)
- The input file is not read manually. The input file is read from the command line argument.
- Data structures used: Redblack tree, Heap.

Folder Contents and Program structure.

Program is written in java and after unzipping suresh_keerthi.zip we find the following files:

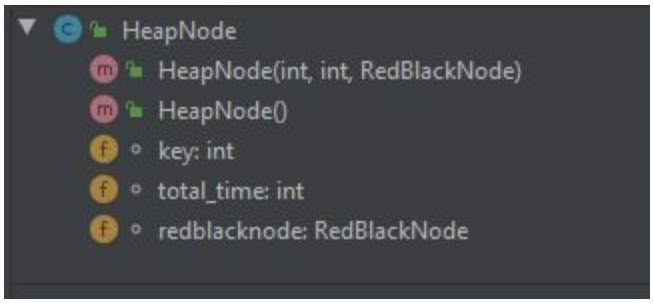
1. risingCity.java
2. MinHeap.java
3. HeapNode.java
4. RedBlackTree.java
5. RedBlackNode.java
6. Build.java
7. Makefile

HeapNode.java

This class holds/constructs a node of the required structure. And have 3 variables:

1. key --- of type integer variable Initially assigned zero for every incoming building.
2. redblacknode --- of type RedBlackNode used as a pointer from heap to the red-black tree.
3. Total_time --- of type integer variable stores to time building has to execute.

This class has a constructor to initialize the mentioned class variables.



MinHeap.java function prototype

This class has all the member function. Performs various tasks.

1. public MinHeap(){
   Description - This is a Constructor
   }
2. public void Insert(int total_time, int executionTime, RedBlackNode node){

   Parameter - int total_time, int executionTime, RedBlackNode node
   Description - inserts a new building into a heap and maintains a pointer to RedBlackNode object(RBT TREE)/
   }
3. public HeapNode extractMin(){
   Description - extracts the minimum element from the heap.
   }

4. public buildheap(){
   Description - Constructs a Heap after extracting minimum element
   }
5. public void remove(){
   Description - removes a node from the heap
   }
6. public void Heapify(){

Description - to rebalance the heap. Rebalence strategy to maintain heap property

}

7. right_node(int i), left_node(int i), fetch_parent(int i){

   Description - to fetch right node, left node and parent respectively

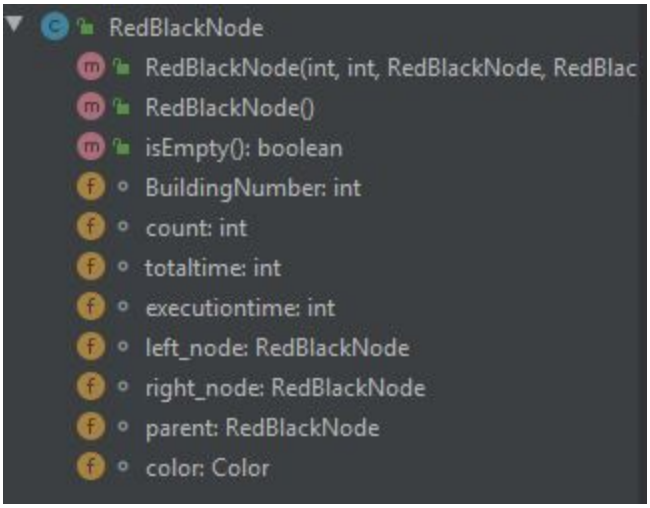
   }

RedBlackNode.java

This class holds/constructs a red-black node. Class variables are:

1. BuildingNumber - stores as a node value.
2. Totaltime - total time(days) to be built.
3. execution time - total time(days) building has undergone construction.
4. Left_node - left child.
5. Right_node - right child.
6. parent - parent of a node.
7. Color - Color of the node.

All these values are assigned in the constructor class.



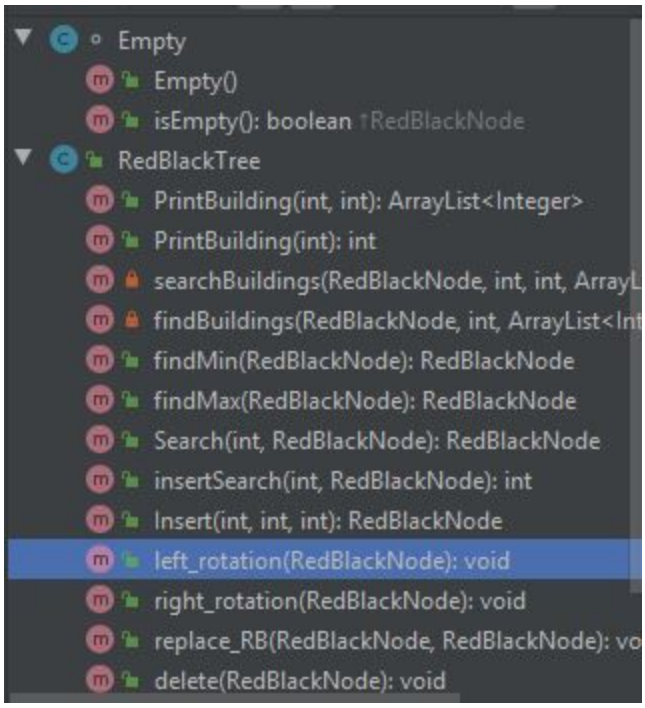
RedBlackTree.java

The prototype of the member function

1. Public int PrintBuilding(int BuildingNumber){
   Parameters - int BuildingNumber
   Description - prints the building number and the time at which construction is completed
   }
2. private void searchBuildings(RedBlackNode root, int building_num_from, int building_num_to, ArrayList<Integer> res){
   Parameters - RedBlackNode root, int building_from, int building_to, ArrayList res
   Description - finds all the buildings in the range building_from to building_true
   }
3. private void findBuildings(RedBlackNode root, int num, ArrayList<Integer> arr){
   Parameters - RedBlackNode root, int num, ArrayList arr
   Description - Searches for a node/building with the passed building number
   }
4. public int insertSearch(int BuildingNumber, RedBlackNode root){
   Parameter - int BuildingNumber, RedBlackNode root
   Description - search for duplicate nodes/building number
   }
5. public RedBlackNode Insert(int BuildingNumber,int total_time,int executionTime){
   Parameter - int BuildingNumber,int total_time,int executionTime
   Desciption - Insert into Redblack tree
   }
6. public void left_rotation(RedBlackNode node){

Parameters - RedBlackNode node
Description - left-rotation operation to balence the tree
}

7. public void right_rotation(RedBlackNode node){
   Parameters - RedBlackNode node
   Description -performs right-rotation as a part of re- balencing strategy
   }
8. delete(RedBlackNode node){
   Parameters - RedBlackNode node
   Description - deletes a node from rbt.
   }
9. Rearrange(RedBlackNode node){
   Parameters - RedBlackNode node
   Description - After Deletion, implements a re-balancing strategy to satisfy red-black tree property
   }



Build.java

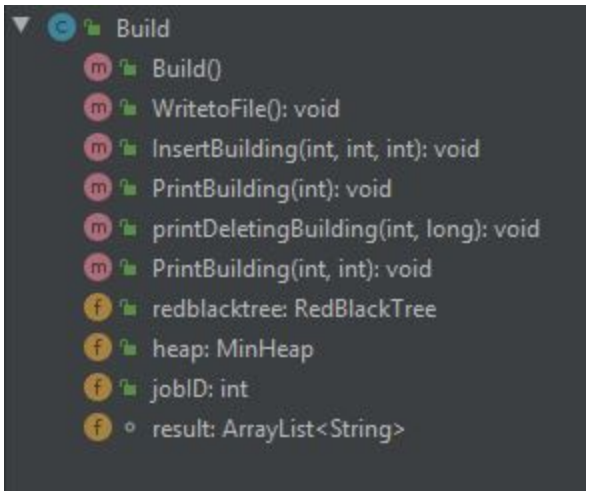
This class acts as a bridge(helper) and has RedBlackTree object minHeap object jobId as class variables.

Member Function of this class and their prototype

1. public Build()

Description - Constructor function to initialize class variables

2. public void WriteToFile()
   Description - writes the output to the file
3. public void InsertBuilding(int ID, int total_time, int executionTime)
   Parameters - int ID, int total_time, int executionTime
   Description - this function is triggered when an insert command is read and this function triggers insert rbtree and insert heap function.
4. public void PrintBuilding(int ID)
   Parameters - int ID
   Description - this function triggers PrintBuilding function of red-black tree in turn returns back with the building_number, execution_time and total_time
5. Public void printDeletingBuilding(int ID,long completed_time)
   Parameters - int ID,long completed_time
   Description - this function prints the details of building_number whose excution time is completed
6. public void PrintBuilding(int id1, int id2)
   Parameters: int id1, int id2
   Description: function overriding concept as we have two printBuilding commands
   prints the building number which are under construction.



Color.java

This file has an enumeration which serves the purpose of representing a group of named constants in a programming language. I used an enum for assigning colors for red-black nodes.

Prototype
public enum Color
    Constants - red, black;

risingCity.java

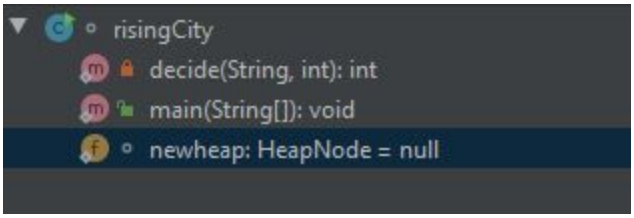
This(rising city) java file contains the main function.
    read file from args[0] using scanner object or FileReader object
    global counter logic is implemented.
    For every building inserted we create a build object.
    Which in turn creates a heap object and redblack object.

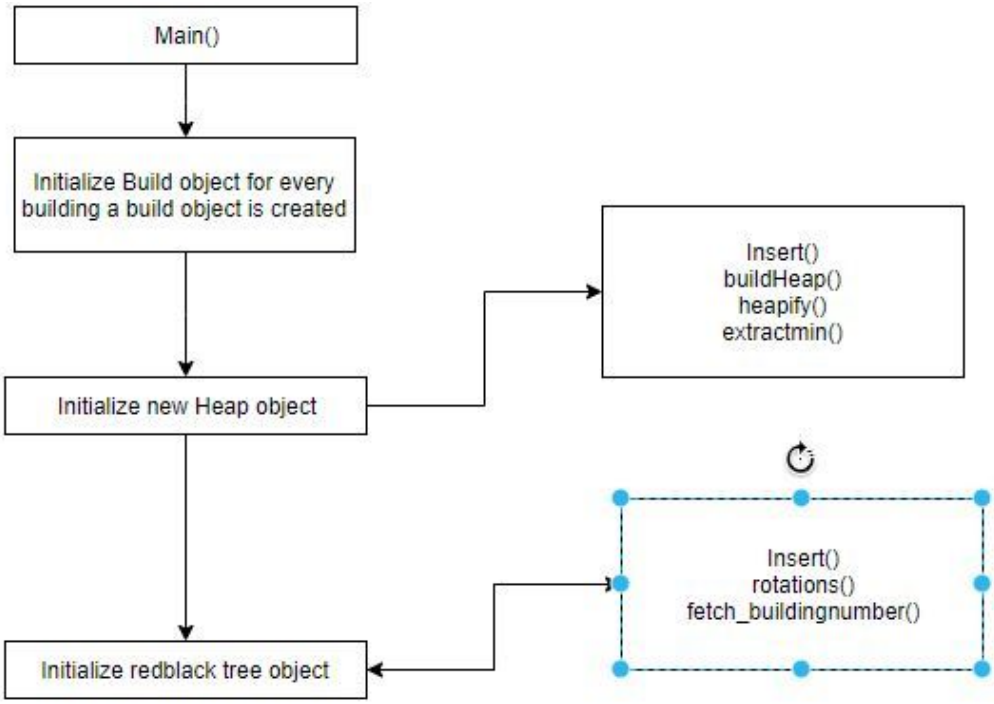
RedBlack object

Key: BuildingNum	Executed time	Total time

    |
    |
    |
    | heap object

Executed time	BuildingNum	Total time





Output

```
(15,1,200),(50,45,100)
(15,45,200),(50,45,100)
(15,47,200),(50,45,100)
(15,50,200),(30,0,50),(50,45,100)
(15,50,200),(30,1,50),(50,45,100)
(15,50,200),(30,5,50),(50,45,100)
(15,50,200),(30,40,50),(50,45,100)
(15,50,200),(30,45,50),(40,45,60),(50,45,100)
(15,50,200),(30,50,50),(40,45,60),(50,45,100)
(30,190)
(15,50,200),(40,50,60),(50,45,100)
(15,50,200),(40,50,60),(50,50,100)
(15,55,200),(40,54,60),(50,50,100)
(15,55,200),(40,55,60),(50,51,100)
(40,225)
(50,310)
(15,410)
```

Reference :

1. COP5536 class lectures and PPT.
2. CLRS 2nd edition.
3. Geeksforgeeks.com (for red-black tree deletion cases)