COP 5536 : Advanced Data Structures, Fall 2019
Project Report

Name : NIKHIL MALLADI   UFID : 5835-3253   UF Email : nmalladi@ufl.edu

Implementation

The software for Wayne Enterprises is implemented using Java programming language.

Input and Output

The input to the project is sent through command line which is a text file (.txt) and the output is also written into a text file (.txt)

Environment

Developed in Eclipse IDE.

One Assumption for Print and PrintBuilding Operations

The PrintBuilding takes range of building number as the input, which gives the status of the buildings that are in range of minBuilding and maxBuilding(both included).

Example 20 : PrintBuilding(10,100) :
Expects 2 arguments. prints the status of buildings along with their whose building numbers are in range [10,100] at globalCounter 20.

The Print operation takes single building value as the input and gives the status of building for the requested globalCounter.

Example 40 : Print(20) :
Expects one argument only. Prints the status of building number 20 when global counter value is 40.

Project Structure

Classes, Class variables and methods

1. *risingCity.java*

- The class variables defined in this class are

File file	: The input file, taken as an argument.

BufferedReader br : It is used to read the input text file line by line.
BufferedWriter out : It is used to write into the output file.
BuildingHeap<BuildingPropertiesNode> minHeap : Used to access the BuildingHeap methods and properties, which implement the minHeap data structure of type BuildingPropertiesNode.
RedBlackTree rbTree : This object is used to initialise an object for RedBlackTree. The nodes inserted into the RedBlackTree are of type BuildingPropertiesNode.
int localCounter : This variable is used to get the current day on which construction of the city is going on.
BuildingPropertiesNode[] bufferQ : The buffer used to store the BuildingPropertiesNode when there is a building whose construction is not executedForFiveDays. So, the incoming building has to wait until the minNode in the heap finishes its execution for at least 5 days or finishes its construction before 5 days.
int rear = 0 , front = 0 : The pointers to update the bufferQ which follows Queue(FIFO) data structure.

public static void main(String[] args)
Project execution starts from this method. The method takes a text file as an input from command line argument and uses it to process it. After the initial processing of the first two lines is done i.e by extracting the buildingProperties using the *tearDownBuildingPropertyValues* we start construction of the city incrementing localCounter day by day and adjusting the minHeap for that day. The insertion into the minHeap is NOT done until the minimum node in minHeap is not done with execution for 5 days or finished construction before 5 days. Once, the execution is over, the node at the front in the Queue is inserted into the min heap and the front, rear pointers are updated accordingly.

tearDownBuildingPropertyValues(String , int , int)
This method is used to extract the building Property values like Building Number, executionTime and totalTime from the lines read from the input file. The parameters taken are operation (String) , valueStartIndex (int) , globalCounter (String)

2. *BuildingPropertiesNode.java*

- A node class having the properties of building like buildingNum, executedTime, totalTime and globalCounter.

- The class variables defined in this class are
public int buildingNum : The building number of buildings
public int executedTime : The executed time of a building
public int totalTime : The totalTime for the building to complete construction
public boolean executedForFiveDays : A flag to know if a building is constructed for five days consecutively, default is false
public int resetNumber : The number to keep track of the number of days a building is constructed consecutively, default is 1 maximum 6 (used 1 indexing)

BuildingPropertiesNode parent : The parent reference in RedBlackTree

BuildingPropertiesNode left	: The left child reference in RedBlackTree
BuildingPropertiesNode right	: The right child reference in RedBlackTree
public static final int BLACK = 0	: The color of a node in RedBlackTree
public static final int RED = 1	: The color of a node in RedBlackTree

- The methods implemented in this class are

  BuildingPropertiesNode(int buildingNum, int executionTime, int totalTime)
  The constructor takes input as three arguments, which is used when creating the BuildingPropertiesNode while inserting into the min heap.

  BuildingPropertiesNode()
  Here, there is one more constructor which is used by RedBlackTree when inserting a new node into the tree. So, since there are two constructors with the same name, this class uses the concept of constructor overloading.

3. *BuildingHeap.java*

- BuildingHeap is the heap of BuildingPropertiesNode which we follows the minimum heap data structure.
- The minimum heap is based on the executionTime of the nodes in min heap. In case there are two nodes with the same execution times, heap is adjusted based on the building number values. The building number values are always unique in heap.
- The class variables defined in this class are

  public int currentSize : The current size of the heap

  public int heapCapacity : The maximum capacity the heap can have

  public BuildingPropertiesNode[] node : The min heap where all buildingPropertiesNode are processed.

- The methods defined in this class are

  public BuildingHeap(int heapCapacity)
  The constructor method that takes the maximum heap capacity as the argument. This is the maximum number of buildings the heap is expected to have at any point of time.

  public void insert(BuildingPropertiesNode x)
  This method is used to insert a newly identified identified building into the heap. As, the data structure used is min heap, the insertion is done at the end of heap(i.e at currentSize) and from there we perform bubbleUp operation.

public BuildingPropertiesNode removeMin()
The removeMin method is used to remove the minimum node from the min heap. As, in a min heap the removal is done by swapping the last node in heap with minimum node and then apply heapify or sinkdown operation from minimum node. This method also follows the same procedure for removing a node from the min heap.

public void sinkDown(int k)
The sinkDown method is used to maintain the heap property throughout the heap. The heap property the BuildingHeap is following is, the node with minimum execution time occupies the minimum node i.e node[1] position in heap. In case the execution times are equal between the nodes, the node with minimum buildingNumber among them takes the minNode position.

public void swap(int a, int b)
The swap method is used to swap two nodes that are in indices ‘a’ and ‘b’.

public boolean isEmpty()
This method returns true if the BuildingHeap is empty

public int heapSize()
This method gives the current size of the heap which is <= 2000.

public void updateMinHeap(int localCounter, RedBlackTree rbTree)
This method is used to update the min heap based on the localCounter variable. Whenever this method is called, the heap is adjusted based on construction flags of the minNode. The construction flags are executedForFiveDays and resetNumber. The executedForFiveDays flag is set to true if the minimum building is constructed for five days continuously. If there is a call to this method at this instant the flag is reset to false, using the resetNumber. The resetNumber keeps track of number of consecutive days a building is constructed for. If the executedTime is equal to the totalTime of the building, then *removeMin()* is called. Otherwise, if the executedTime of the minNode is not equal to the totalTime for construction, but the building is executedForFiveDays then the sinkDown(1) is called to select the next building based on selection criteria.

public BuildingPropertiesNode extractMin()
The method returns a BuildingPropertiedNode which is minimum among all the nodes. The node will always be located at the index 1 position.

4. *IRedBlackTree.java*
   - The Interface class which shows the high level implementation details of the RedBlackTree.
   - There will be no class variables as this is an interface, the methods declared in interface are
     public void insert(BuildingPropertiesNode z);
     public void remove(BuildingPropertiesNode nodeToRemove);
     public BuildingPropertiesNode search(int buildingNum);
     public void printBuildingsStatusInRange(int minimumBuilding, int maximumBuilding);

5. *RedBlackTree.java*
- This class implements the IRedBlackTree interface and has all the implementation details of the methods defined in the IRedBlackTree.
- The class variables used in this are

  private BuildingPropertiesNode nil : This is initialised to a new BuildingPropertiesNode object.

  private BuildingPropertiesNode root = nil : Root node of the tree.

  private boolean newLine = false : newLine is when printing buildings in a given range. If there are no buildings in the tree in the given range, then there will be no extra line printed. This is to make the output look consistent without extra lines.

  private boolean commaAppearance = false : commaAppearance is used while printing buildings in range. The Buildings triplets should appear one after the other with comma separated. The comma should be only for the intermediate nodes but not for the last node. So, this flag is also used to make the output look consistent without extra commas.

  public void leftRotate(BuildingPropertiesNode x)
  This method is used to rotate the tree in the left direction over the node x. It internally calls the leftRotateFixup method to update the numLeft and numRight values over the node x.

  private void rightRotate(BuildingPropertiesNode y)
  This method is used to rotate the tree in the right direction over the node y. It internally calls the rightRotateFixup method to update the numLeft and numRight values over the node y

  public void insert(BuildingPropertiesNode z)
  Insert method is used to insert a new node into RedBlackTree. The insert method follows the insertion procedure of BinarySearchTree to insert the node into tree. Inserting a new node will be *BLACK* by default. Once, the insertion of the node is done properly the RedBlackTree properties are then satisfied. So, this method calls insertFixup internally to fix the violations in the RedBlackTree.

  public void remove(BuildingPropertiesNode nodeToRemove)
  Remove operation is to remove a node from RedBlackTree. Remove operation takes nodeToRemove object as the argument and performs search operation to remove the node based on nodeToRemove.buildingNum as the key value. Once the removal is performed removeFixup method is called to fix RedBlackTree violations.

  public BuildingPropertiesNode search(int buildingNum)
  The search method returns the node which matches the key buildingNum in the RedBlackTree. If there is no matching node the method returns null.

  public void printBuildingsStatusInRange(int minimumBuilding, int maximumBuilding)

This method is called by PrintBuilding operation, to return the buildings statuses that are in the range of minimumBuilding and maximumBuilding(both included).