COP 5536 Fall 2019

Programming Project

Name: Aarthi Kashikar
UF ID: 0366-8968
Email: aarthikashikar@ufl.edu

Language : Java

Problem Statement:

The Wayne Construction is developing a new city where many buildings are constructed. The problem statement is to develop a software to keep track of construction of all the buildings.

The conditions of constructing the city are:

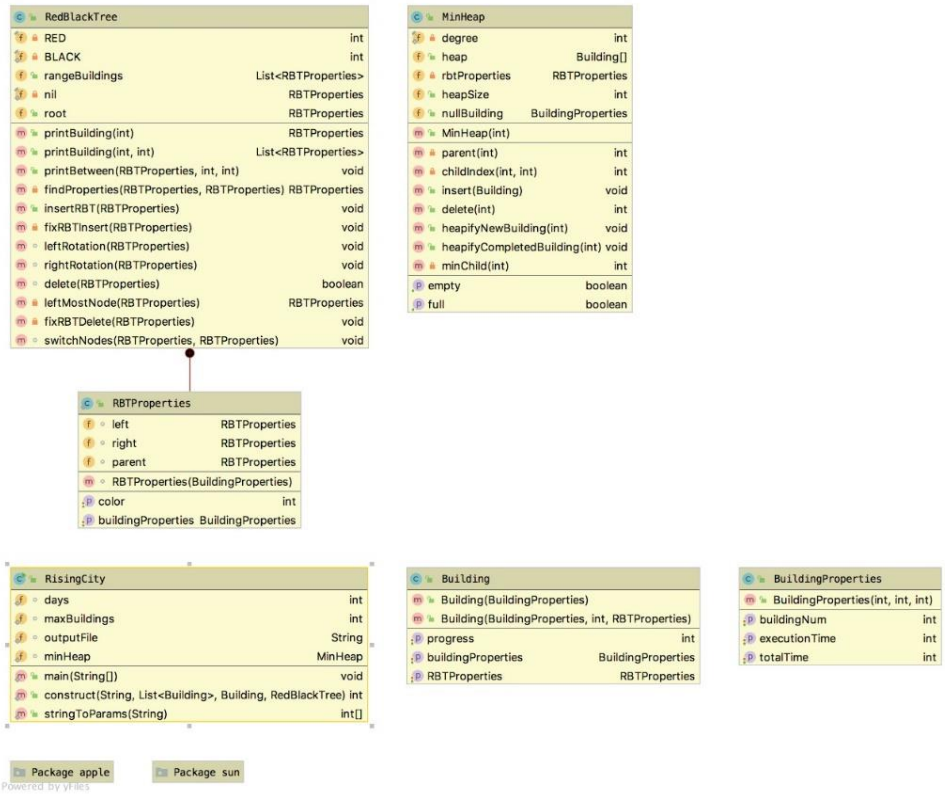
1) Only one building is worked on at a time.
2) When it is time to select a building to work on, the building with the lowest executed_time (ties are broken by selecting the building with the lowest buildingNum) is selected.
3) The selected building is worked on until complete or for 5 days, whichever happens first.
4) If the building completes during this period its number and day of completion is output and it is removed from the data structures. Otherwise, the building's executed_time is updated. In both cases, Wayne Construction selects the next building to work on using the selection rule.
5) When no building remains, the completion date of the new city is output.

Implementation:

The project is written in Java language. The logic applied is to use a global timer that increments every day and the top node of the heap where the building with the least execution time resides is worked on for 5 days or till total time is reached whichever is less. If any new buildings get inserted in mean time, it is inserted and new building with least execution time is taken to construct next. This way all the buildings are constructed one after the other and eventually construction of the new city is completed.

Classes:

Class diagram is as below



Following are all the classes used to develop the application.

BuildingProperties Class :

This is a model class which has the 3 main properties of the building.

```
int buildingNum;
int executionTime;
int totalTime;
```

This class has methods to set and get the properties like buildingNum, executionTime and totalTime.

RBTproperties Class :

This is where the properties of a RedBlack Tree are written. The 2 properties are an object of BuildingProperties mentioned earlier and the color of the node.

```
BuildingProperties buildingProperties;
int color = BLACK;
RBTProperties left = nil, right = nil, parent = nil;
```

Building Class :

This is a model class which has 3 data fields. BuildingProperties object, progress and RBTProperties object. This acts as the main minHeap node with additional fields to maintain progress of construction during the 5-day construction period and an object RBTProperties to form a link between HeapNode and RBTNode.

```
BuildingProperties buildingProperties;
int progress;
RedBlackTree.RBTProperties rbtProperties;
```

This class has methods to set and get the values of the HeapNodes.

RedBlackTree Class:

This class contains all the methods for the operations on RedBlack Tree like insert, delete and many as listed below:

- `public RBTProperties printBuilding( int buildingNum )`

  This method is called when a command has been received to print the BuildingProperties of a building of the buildingNumber passed. This is called from main class which received the commands.

- `public List<RBTProperties> printBuilding(int buildingNum1, int buildingNum2)`

  This method is called when command is received to print BuildingProperties of all the Buildings between the two buildingNumbers passed as parameters. This method is called by main class. This method overloads with the earlier mentioned PrintBuildingmethod. Further printBetween method is called.

- `public void insertRBT(RBTProperties node)`

  This method is called from main class when the insert command is received to insert a new building. There can be Red Black Tree violations while inserting which are taken care by the helper methods.

- `boolean delete(RBTProperties node)`

  This method is called when a building completed its construction and needs to be removed from the tree. There could be some Red Black tree properties that get violated when a node is deleted which are taken care by the helper methods.

Helper methods:

- `public void printBetween(RBTProperties current,int buildingNum1, int buildingNum2)`

  This is a helper class for PrintBuilding(buildingNum1,buildigNum2). This method makes recurrence calls to get the all the building numbers between the two given buildingNumbers. This takes O(log(n)+S) time to fetch the values where n is number of active buildings and S is the number of buildings fetched.

- `private RBTProperties findProperties(RBTProperties findRBTProperties, RBTProperties refRBTProperties)`

  This method is used when a particular node needs to be found in reference to another known node.

- `private void fixRBTInsert(RBTProperties node)`
- `void leftRotation(RBTProperties node)`
- `void rightRotation(RBTProperties node)`
- `void switchNodes(RBTProperties deleteNode, RBTProperties helperNode)`
- `private void fixRBTDelete(RBTProperties node)`
- `private RBTProperties leftMostNode(RBTProperties subTreeRoot)`

MinHeap Class:

- `public void insert(Building building)`

  This method is called when a command is received to enter a new building into the heap. This call is made from main class.The complexity of inserting a building into the heap is O(logn)

- `public int delete(int index)`

  This method is called when the construction of a building is completed and needs to be removed from the heap. The complexity of deleting a building from the heap is O(logn)

Helper methods:

- `public boolean isEmpty()`

  This method is to check if the minheap is empty or not
- `public boolean isFull()`

  This method to check if it is full
- `private int parent(int i)`

  To get the parent of a node
- `private int childIndex(int i,int k)`

  To get the exact index of the child.
- `public void heapifyNewBuilding(int i)`

  When a new building is added, heapify will place it at the top of the heap.
- `public void heapifyCompletedBuilding(int i)`

  This is called when a building construction is completed and deleted and the heap needs to be heapified or when the 5-day construction is completed from the minheap node and next building with smallest execution time needs to be picked up.
- `private int minChild(int i)`

  This method is to get the minimum of the two children of any node.

RisingCity Class:

Below are the global variables initialized:

```
static int days = 0; 'days' is number of 'days' which acts as globarTimer
static int maxBuildings = 2000; set to 2000 as mentioned in the problem statement
static String outputFile="output_file.txt"; outputFile is also set to given name of the output file
static MinHeap minHeap = new MinHeap(maxBuildings);
```

File read:

- In main, first args is checked to see if any value is passed for inputFile. If nothing is passed, an error message is printed and code exits.
- Now the filename is read and if the file is found in the path, all the valid commands are read one after the other and stored into a list called 'commandList'.
- Once all the commands are stored in the list it is traversed to read one command after the other to execute in a for loop till all the commands are read and executed.

Reading all the commands:

- In a for loop two commands are readsimultaneously. First command is called currentCommand and the second command as nextCommand. This is done to know the time that the first and next commands are passed so the construction can be run for those many days without interruption till the next command to be executed. In case, end of the list is reached and there are no more commands left nextCommandTime is

made max value of integer so that there is a limit till when our construction should continue.

- Next the loop runs from currentCommandTime till the nextCommandTime is reached. During this, first the command that has been asked is executed.

Insert command:

- If the command is 'Insert', insert operation is performed. First a check is made if there is already a buildingNumber present. If there is no building with the same buildingNumber, then it is added to a list called buildingList which acts as a queue. This is done so that new building does not disturb the ongoing construction in case it is inserted before the 5 days of construction of the building with least execution time. In case no builing is ongoing construction, then the building in the buildingList is inserted. (the list will have a maximum of 4 buildings as every 5 days a check is made if there are any new buildings in the queue and they are inserted, and the list is emptied)

PrintBuilding command :

- If it is 'PrintBuilding' print operation is performed based on number of parameters in the printBuilding command. If the command contains ',' method that prints between a range of buildingNumbers is called. If the command does not contain ',' it means command is to print details of only a particular buildingNumber.
- Before proceeding to construct the building, first check is made if there is any completed building. If there is any, it is printed and deleted the same from red black tree.

Construct call:

- Now the main construct operation is called. At the same time, it is checked if the construct method returns the number of a building. If it returns -1 it means the building on which construct was being performed did not complete(execution time did not reach total time). If it returns any number other than -1, the building number if printed along with the current day number only when there is no further command or when the next command is not 'PrintBuilding'.

Construct:

- The parameters passed in construct method are nextCommand, the buildingList which contains list of buildings that are yet to insert into heap, building object that is the top node of minHeap on which the construction needs to be performed and redblacktree object.
- Execution time of the building object is incremented for 5 days or total time whichever is lesser.
- Along with this, the progress of 5-day construction is also incremented.
- If the execution time reaches total time of the building, building is deleted from minHeap which heapifies the heap. If the next command is null or it is not print, building is deleted from redblacktree as well. This check is made so as to have this building printed in the next command since PrintBuilding is given preference over delete after completion.
- At this point, buildingList is checked. If any buildings present in the queue are added since construction of the top node of the heap is completed and next building can be taken.

- If execution time did not reach total time but the 5 days construction is completed, the progress is set back to 0. Heapify operation is performed on minHeap to get next building with least execution time. At this point, if there are any buildings that were added during the 5 days period, gets inserted into minHeap.
- If any building completes construction, buildingNumber is returned.