ADS Project

Name: Nidhi Shashi Sharma **UFID: 6843-1215**
UF mail: nidhi.sharma@ufl.edu

The structure of the Project Rising City is as follows:

Class: Building

This class defines the attributes of the building which are as follows:

- **buildingNum**: The Unique building number assigned to each building
- **executed_time**: The number of days for which work has been performed on a building
- **total_time**: The time required for a building to be constructed

Building(int buildingNum, int executed_time, int total_time): Constructor of building Class

Class: MinHeap

The minheap is implemented as an array of Building objects such that the structure of a minheap is always maintained. That is, the parent node is always smaller than its children nodes.

The attributes of the MinHeap are:

- **Building**[]: An array of all the buildings in the minHeap
- **heapSize**: The size of the minheap
- **maxActiveBuildings**: The maximum number of buildings on which work can be in progress at any given time
-

The methods of the MinHeap class are:

- **MinHeap():** Constructor of the class MinHeap
- **getHeapSize():** Function to get the size of MinHeap
- **parentNode(index):** Function to get the position of parent node of a given node
- **leftChildNode(index):** Function to get the position of left child node of a given node
- **rightChildNode(index):** Function to get the position of right child node of a given node
- **checkLeaf(index):** Function to check whether a node is the leaf node of a given node
- **getMinNode():** Function to get the minimum element of the MinHeap i.e. fetch the root of the MinHeap
- **interchangeNodes(index1, index2**): Function to swap the nodes at the given indices. The building at index1 is replaced with building at index2 and vice-versa
- **minHeapifyNode(index**): Function to place a given node in its correct position in the MinHeap by melding the minheap after the insertion of the node at the given index.

- **insertBuilding(Building):** Function to insert a node in the MinHeap. This method inserts the node into the minheap array
- **buildMinHeaps():** Function to construct the MinHeap by recursively calling the minHeapifyNode function
- **removeMinNode():** Removes the root of the MinHeap

Class: RedBlackTree

The attributes of the RedBlackTree are

- RBTreeNode: It is a red-black tree node which is an object of the class RBTreeNode

The methods of the class RedBlackTree are:

- **createNodes(Building):** This method creates the RedBlack Tree node of a given building.
- **insertBuildings(Building):** This method creates a red blacktree node of a building and inserts it into the red black tree.
- **deleteNode(Building):** This method deletes a red blacktree node of a building from the red black tree. It does so by replacing the node to be deleted by the next largest number and then fixing the Red black tree such that it maintains all its properties.
- **deleteFixUpRedBlackTree(RBTreeNode, colour):** This method is called after a node is deleted, which may break the properties of the red black tree, to restore the properties of the tree.
- **replacementNode(RBTreeNode):** This method is called to find the node which will reolace a node that is deleted.
- **FixUpRedBlackTree (RBTreeNode):** This method is called to restore the properties of the red black tree after the insertion of a new node.
- **leftRotation(RBTreeNode):** This method is called to perform left rotation on the given node in order to maintain the property of the red black tree.
- **rightRotation(RBTreeNode):** This method is called to perform right rotation on the given node in order to maintain the property of the red black tree.
- **performRotation(RBTreeNode, parent RBTreeNode,child RBTreeNode):** This method is called to help left rotation and right rotation methods to perform rotattions, in order to maintain the property of the red black tree.
- **fetchUncleNode(RBTreeNode):** This method is used to fetch the uncle node of the given node.
- **fetchParentNode(RBTreeNode):** This method is used to fetch the parent node of the given node.
- **fetchSiblingNode(RBTreeNode):** This method is used to fetch the sibling node of the given node.
- **fetchGrandparentNode(RBTreeNode):** This method is used to fetch the grandparent node of the given node.
- **searchBuilding(RBTreeNode, Building):** This method is used to search a particular building in the red black tree.

- **searchInRange(ArrayList<Building>, RBTreeNode, startIndex, endIndex):** This method is used to search a range of buildings in the red black tree, for the given range.
- **checkRange(buildingNum, startIndex, endIndex):** This method helps the searchInRange method to check the range in which the buildings are to be found.

Class: RBTreeNode

The RBTreeNode represents a red black tree node having attributes:

- Building: An object of building class representing a building that is at nodes of the red black tree.
- leftChildNode: An object of RBTreeNode class representing the left child of the given node
- rightChildNode: An object of RBTreeNode class representing the right child of the given node
- parentChildNode: An object of RBTreeNode class representing the parent of the given node
- colour: A string representing the colour of the RBTree node which is either RED or BLACK.

Class: RisingCIty

This is the main class of the project. This class selects the building which is to be constructed, performs the construction work on the selected building as per the input file passed by reading the inputs from it, inserts the building into the minheap as well as Red-Black Tree and prints the output of this project.

The attributes of this class are:

- **globalCounter**: maintains the count of the number of the day.
- **dayListIndex**: used to access the arraylist of dayNumber in the input file, by its index.
- **buildingInConstruction**: Contains the building which is currently being worked upon.

The methods of this class are:

- **Main:** This is the main method of the RisingCity project. This method reads the input from an input file using a scanner. Then it inserts the buildings in the minheap and the RedBlackTree, after which it selects the building to be worked on as the root of the minheap which is built on the execute_time of the buildings, and the redBlackTree is used to find the building to be worked upon based on their building number, in case of similar execution times.
- **insertBuilding(minheap, RBTree, Building):** This method inserts building in the minheap and RedBlack Tree
- **performActivity(minheap, RBTree, ArrayList<dayCountList>, ArrayList <activityList>):** Perform the activity of inserting the building or printing the building, depending upon the command in the input file.
- **printBuilding(Building, globalCounter):** This method prints the building using the Red Black Tree