COP 5536: ADVANCED DATA STRUCTURES

PROJECT REPORT

Submitted By –

Dipen Jain

UFID: 15521903

UF Email: dipenjain@ufl.edu

Problem description

Wayne Enterprises is developing a new city. They are constructing many buildings and plan to use software to keep track of all buildings under construction in this new city. A building record has the following fields:
buildingNum: unique integer identifier for each building.
executed_time: total number of days spent so far on this building.
total_time: the total number of days needed to complete the construction of the building.

In order to complete the given task, you must use a min-heap and a Red-Black Tree (RBT). A **min heap** should be used to store (buildingNums,executed_time,total_time) triplets ordered by **executed_time**. You mwill need a suitable mechanism to handle duplicate executed_times in your min heap. An **RBT** should be used store (buildingNums,executed_time,total_time) triplets ordered by **buildingNum**. You are required to maintain pointers between corresponding nodes in the min-heap and RBT.

Implementation Details / Logic:

The below algorithm is followed to select buildings to work on and complete the construction:

1. Select a building to work on from the heap. If there is no current building, it selects the next building from the heap and works on it (by increasing executed time)
2. When value of global counter is equal to the input counter - reads and performs the instruction specified.
3. After the instruction is executed, the current worked upon building's execution time is checked if it's equal to it's total time. If the building is complete, the building details (buildingNum, executed_time, total_time) are printed; Else if the building is worked on for 5 days, the building is reinserted into the heap and the global values of consecutive days and current building are reset.
4. The loop is continued until all the buildings are completed and the instructions are completed.
   Example Run – Let the input file be: 0: Insert(1,2)
   1: Insert(2,1)
   2: Print(1)
   The output would be : (1,2,2)
   (1,2)
   (2,3)

Program Prototypes

	Color – Enum
Description	Enumeration for defining RED and BLACK colors of a node.
Values	RED BLACK

	Node - class
Description	Class defining the node structure for Red Black Tree as well as the element structure for MinHeap.
Class members	Node Nil Color color int buildingNum, int executed_time, int total_time, Node left Node right Node parent

	risingCity - class
Description	Main Class implementing the logic to select and construct buildings.
Class members	Node current – This represents the current building that is being worked on. MinHeap h – An object to represent the minheap RedBlackTree t – An object to represent the red black tree. int counter – The global counter to keep track of days int consecDays – Number of consecutive days worked on a building.
Member Functions	void Insert(Node newNode) - Function to insert node into both RBT and MinHeap. - Takes one input argument of Node type. - Return type = null
	void PrintBuilding(StringBuilder output, int bNum) - Function to print the details of a building if it exists else prints (0,0,0) - Input arguments – Output string to be written to file, and the building number for which the details need to be printed.
	void PrintBuilding(StringBuilder output, int b1, int b2) - Function to print the details of a building in a range if it exists else prints (0,0,0) - Input arguments – Output string to be written to file, and the starting building number and the ending building number within the range for which the details of all buildings need to be

	printed.
	Node workBuilding(Node currentNode) - Function to select a building to work on or continue to work on the current building. - Input – takes a current building, if it's null extracts building to work on from minheap and works on it.
	static void main(String[] args) - Main function which reads the input file name from args - Responsible for executing all commands from the input file and also writing the output to the output file.

	RedBlackTree - class
Description	Class defining the data structure for Red Black Tree
Class members	Node Nil – defines the external node Node root – defines the root node for the tree
Member Functions	void nodeTransplant(Node n1, Node n2) - Function to replace node n1 with node n2. - Takes two input arguments of Node type. - Return type = null
	Node minimumNode(Node x) - Function to find the minimum node in the subtree rooted at x. - Input argument – Root of subtree at which the minimum node is to be found. - Returns the minimum node object.
	void leftRotate(Node x) - Function to perform a left rotation at node x by changing parent and child pointers of x and it's parent and it's grandparent(if any) - Input Argument – Takes node at which rotation is to be performed.
	void rightRotate(Node x) - Function to perform a right rotation at node x by changing parent and child pointers of x and it's parent and it's grandparent(if any) - Input Argument – Takes node at which rotation is to be performed.
	void insert(Node newNode) - Function to perform insertion of a newNode in a RBT
	void insertFix(Node z) - Function to fix the double red scenario after an insert. Takes the node at which the anomaly occurs as node z and is the input argument.
	void delete(Node z) - Function to delete the node from the RBT.

	- Input Argument – Node z which is to be deleted.
	void deleteFix(Node x) - This function is used to fix the deletion anomaly caused in the Red black trees. - Input argument – Takes a node x as an input at which the deletion anomaly is to be fixed.
	List<Node> nodesInRange(int b1, int b2) - Function to return list of nodes present in the RBT in between b1 and b2. Takes two building numbers as input arguments and returns a list of nodes as output.
	Node findNode(int b1) - Function to find a node with a given building number. If found, returns the node. If not returns Nil Node.
	Node nodeDetails(int b1) - Function to find a node with a given building number. If found, returns the node. If not returns Nil Node.
	void valuesInRange(Node node, List<Node> list, int b1, int b2) - Function to update values found in the range of building numbers b1 to b2. Takes 4 input arguments node, and a list of nodes and start and end range of building numbers.

	MinHeap - class
Description	Class defining the structure MinHeap.
Class members	Node[] Heap – Array of node references of RBT int size – Stores size of heap final int maxsize – Stores the maxsize of the heap(2000 in this case) static final int FRONT = 1 – Stores the start point of the heap.
Member Functions	int parent(int pos) - Returns the parent index of the element at position (pos)
	int leftChild(int pos) - Returns the index of the left child of the element at position (pos)
	int rightChild(int pos) - Returns the index of the right child of the element at position (pos)
	boolean hasLeftChild(int pos) - Returns a Boolean value if there exists left child for the element at the position(pos)
	boolean hasRightChild(int pos) - Returns a Boolean value if there exists right child for the element at the position(pos)
	void swap(int fpos, int spos)

	- Function to swap element at fpos and spos. The element at spos is placed at fpos and vice versa.
	void insert(Node n) - Inserts a node n in the minheap and the swaps it recursively with parent till it reaches it's right position.
	Node removeMin() - Function to return the minimum element of the minheap. This function removes the minimum element of the heap and returns it and then calls the heapify function to maintain the heap property.
	void minHeapify(int pos) - Function to perform heapify operation on the min heap. In this, the element at position pos is compared with it's left and right children and the minimum element is swapped and this happens recursively until it reaches the correct position.

Conclusions

The project has been implemented with the given described implementation and produces result as specified in the document.