ADVANCED DATA STRUCTURES (COP 5536)

PROJECT 1

Madhura Basavaraju - 67941287

PROBLEM STATEMENT:

The project aims to use Min heap and Red Black Tree to schedule construction operations on buildings.

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.

Only one building is worked on at a time. Building to be worked on is chosen based on the execution time. The building with the least execution time is chosen (ties are broken by selecting the building with the lowest building number). A selected building is worked on for only 5 days at a stretch or until building is completed, whichever happens first. When there are no buildings remaining, the completion date of the new city is output.

PROGRAMMING ENVIRONMENT: C++

MIN HEAP:

The Min heap is maintained as an array of nodes. Each node reflects a building. The triplets (buildingNums,executed_time,total_time) are stored in the nodes. Building nodes are ordered by their executed times, with the lowest execution time on the top of the heap.

FUNCTIONS:

- void MinHeapify(int i): Arranges the elements of the heap in such a way the min-heap property is reserved.
- int extractMin(): Returns the element at the top of the heap(building with smallest executed time)
- void insertIntoHeap(Node node): Inserts the given node into the heap.
- int parent(int index): Returns the index of the parent node for the node at the given index in the array
- int rightChild(int index):Returns the index of right child of the node present at the given index.
- int leftChild(int index): Returns the index of left child of the node present at the given index.
- void swap(Node node1, Node node2): swaps node1 and node2.

RED BLACK TREE:

The red black tree is maintained using objects of the class RBTNode, where each object reflects a building and has pointers to its parent, left and right children. In the each node RBT is used store the triplet (buildingNums,executed_time,total_time) ordered by buildingNum.

FUNCTIONS:

- void leftRotate(RBTNode* node): Left rotates the given node
- void rightRotate(RBTNode* node): Right rotates the given node
- void swapcolors(RBTNoode* node1, RBTNode* node2): swaps the colors of the nodes
- void fixRedRed(RBTNode* node): fixes the consecutive red nodes violation in the RBT that originates from the given node.
- RBTNode* BSTreplace(RBTNode* node): Returns the node to replace a deleted node in BST
- void deleteNode(RBTNode* node): Deletes the given node from the RBT

- void levelOrder(RBTNode* node): Prints level order traversal for a given node.
- RBTNode* search(int n): searches for given value, if found returns the node else returns the last node while traversing
- void insert(int bn, int et, int tt): Inserts a building with the triplet value (bn,ex,tt) into the RBT
- void deleteByVal(int n): Deletes the node where the building_number = n from the RBT.
- void printLevelOrder(): Prints the tree in level order starting from the root.
- void updatetree(): Increments the executed time of the building object calling the function.

PROGRAM STRUCTURE:

- The scheduling of buildings logic is executed in the main() function of the program. Both the input file that it must be read from and the output file that must be written to are opened. The global counter is set to zero.
- Until the end of the file of input file is reached the following steps are executed.
  - When the global counter does not match with the arrival time of statement being read from the input file:
    1. Provided the heap is not empty, the topmost building from the min heap is selected. If 5 days have been spent on the topmost building consecutively, and the building is not completed, building is extracted a– extractMin() and re inserted – insertIntoHeap(node) into the heap.
    2. A building on which less than 5 consecutive days have been spent on is selected. If there is only one remaining day to be spent on the building selected, the building is worked on (the work_on_building() function is called) and removed from both data structures after printing details to output file. Else work on the building for that day. Progress to next day is made by incrementing the global counter and the selection process is repeated.
  - If the global counter matches the arrival time of a statement being read from the input file:

    1. The function name is checked. If the function name being read is “Insert”, the insert function for both the data structures is called to insert the building into the min heap and the RBT. The newly inserted building is worked on for that day and the global counter is incremented to proceed to the next day.
    2. If the function name being read is “PrintBuilding. The details of the building/buildings are printed.

       After printing the details of building/buildings if the heap is found to be not empty, a building is chosen to be worked on through the selection process. work_on_the_building() function is called and global counter is incremented to proceed to the next day.

- After the end of the file of the input file is reached and the heap is not empty, until the heap is empty. For each day (for each increment of global counter):
    1. The building to be worked on is extracted from the min heap. The building is checked if it has been worked upon for 5 days consecutively. If 5 days have been spent on the building, the building is extracted from the heap and re inserted into the heap. If the selected building has only one remaining day work_on_the_building() is called and the building is removed. Else the global counter is incremented to proceed to the next day after working on the building for that day.