Yuhao Zhuang

Homework 3 Report

The Treemap is implemented based on a binary search tree with a unique pointer of root node, and a struct holding the left and right unique child pointers and its own values and keys. A size variable is created to keep track of treemap's size. When Insertion is called, the function calls itself recursively to based on comparing new key's node key to current node key (smaller to the left pointer, larger to the right pointer) and check to see if the node exists, if it doesn't, then the node will be set with its key and value, and size is increased by one. For remove, the function goes to the node containing the key by calling itself recursively and checking whether to go right or left, and when the node is found, if both left and right children exists, the current node's key and value is set to the smallest node on its right subtree. And that node will be deleted by calling Remove. If the node only has either left or right pointer, then move is called to set current node's content to its left or right node and that node will be deleted. The size is then decreased. Floorkey function uses a helper function that takes a pointer and key value as parameters. It checks to see if node's key is smaller, larger or equal to they key in parameter. Every time it is smaller, the function will recurse right to find if any keys larger than this and fit the condition(smaller than the key parameter), and return that node, if not,return the current node. And every time it is larger, the function will recurse left to return if any node smaller than this and fit the condition and return nullptr if not. When the keys are equal, the function stops and return the node, and returns nullptr if node doesn't exist. The CeilKey is implemented in the same concept except when node's key is larger, it will return the node's pointer instead of nullptr, and then it will keep recursing left to find the next smallest node and return those nodes within condition(larger than key parameter) if found. And if node's key is greater, it just recurses the function right ,return node if found, and returns nullptr if nothing is found. Basically every time the node's condition checks, it keeps recursing to see if there is a better choice, and the node will be returned, and if the condition doesn't satisfy, it keeps recursing and return node if found, and nullptr if not found. Other functions like contains value and min,max,get are implemented by just keep recursing the function left or right based on comparing the keys and returning if the conditions are met. For ContainsValue, since values cannot be compared, the function recurses both left and right and return true if find values are equal, false if not.

The 2nd program reads in the keys and values and insert it into a treemap. Rich and cheap is implemented by returning the biggest and smallest values of the treemap using MaxKey, MinKey. All prints the smallest node in the map and then call ceilkey(smallest key+1) each time to find the key larger than the current key until the last element. For the who commands, if it is empty, it uses ContainsKey to check if the key exists and returns the key, and for +, it checks if the largest key in treemap is larger or equal than the key, and then it returns CeilKey(key+1) to find the next larger key. And for -, it checks if the smallest key in treemap is less or equal to the they, and returns Floorkey(key-1) to find the next smaller key.

The treemap implementation is tested by running through different treemaps that have lots of levels and nodes. The test cases used all methods to test if they work and they tried to cover all possible if else, exceptions by setting the treemap to certain condition such as removing until empty or removing when the node has one or both children so that it would execute all the if else, and exception statements to check if everything is done right.

Reference:
Piazza
Lecture Slides
StackOverflow: Reading two columns in CSV file c++