ECE 250 Project 2: Design Document

Akbar Zafar: 20832368

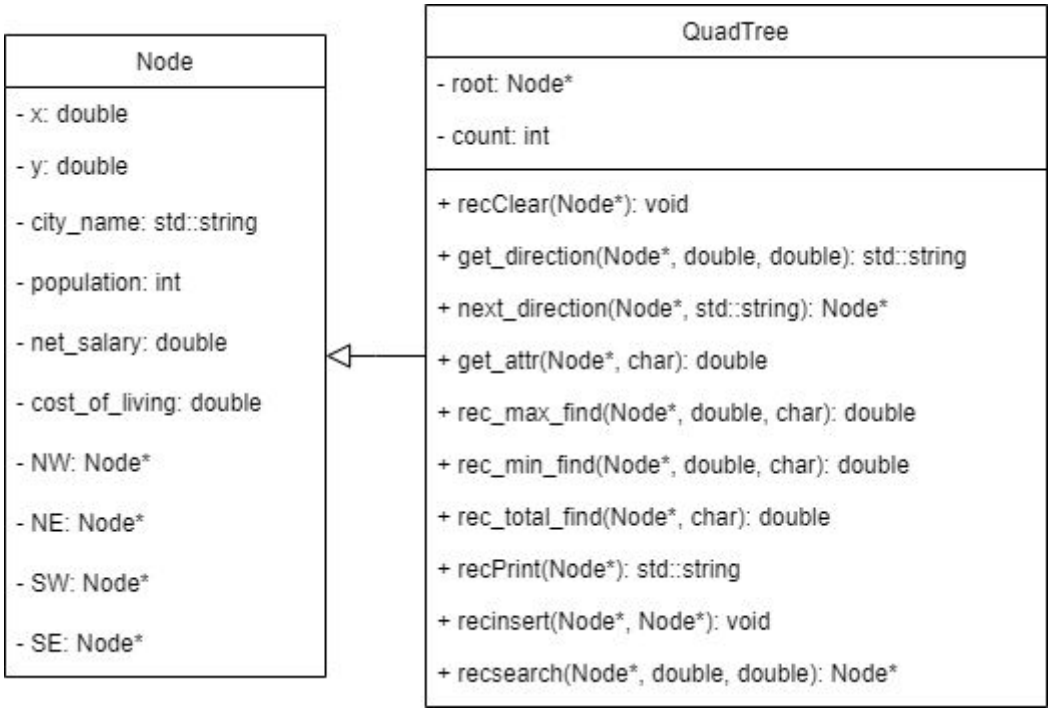
Overview of Classes

The classes used were a Node class and a quadtree class.

The Node class represents a single city, storing data as well as child nodes for cities in the NW, NE, SW, and SE directions.

The Quadtree class allows manipulation on the quadtree including, but not limited to inserting cities, searching for cities, finding max, min, and total, given an attribute.

UML Class Diagram



Details on Design Decisions

The constructor for the Node class takes the longitude(x), latitude(y), city name, population, cost of living, and net salary in order to set the data for a city. Also, the child node pointers are set to nullptr which I use to indicate an empty node slot.

The QuadTree constructor sets the variable to hold the size of the tree to 0 and explicitly sets the root of the tree to nullptr.

The QuadTree destructor calls a recursive function to deallocate all nodes and set to nullptr. In addition, it sets the root to nullptr and resets the count for the size of the quadtree to 0.

Test Cases

Aside from using the provided test cases, I tested my code against specific cases such as clearing an empty tree, searching for a max, min, total attribute in an empty direction, printing an empty tree, and searching for a node in an empty tree.

Performance Evaluation

All calculations are based off assuming we have a balanced tree

Insert: The time to insert would be $O(\log_4(n))$ since the farthest we would have to go to get to the bottom of the tree is the height.

Clear: Since all nodes need to be accessed, the time to clear is O(n).

Print: Since all nodes need to be accessed, the time to clear is O(n).

Search: In the worst case scenario, the node could be at the very bottom of the tree, and in that case the time would be $O(\log_4(n))$.

Max: Since the Max function first uses search, and then searches in all directions(to find the max). The time would be $O(\log_4(n))$. This is because once we find a node, the total number of nodes we need to search is (n/4)-1.

Min: Since the Min function first uses search, and then searches in all directions(to find the min). The time would be $O(\log_4(n))$. This is because once we find a node, the total number of nodes we need to search is (n/4)-1.

Total: Since the Total function first uses search, and then searches in all directions(to calculate the total). The time would be $O(\log_4(n))$. This is because once we find a node, the total number of nodes we need to search is (n/4)-1.