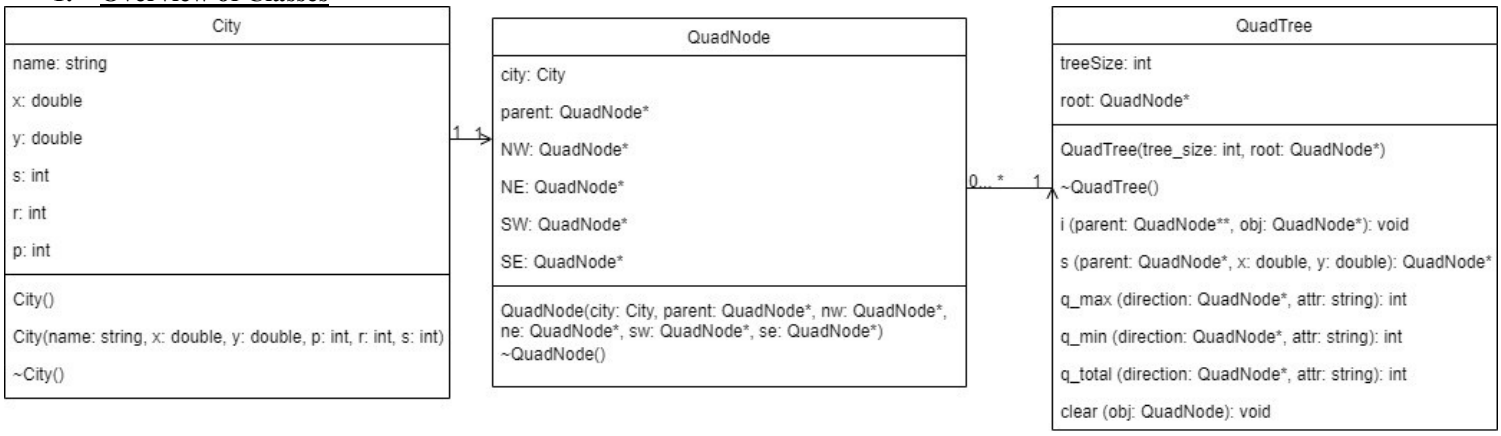
ECE250 – Project 3
QuadTree
Design Document
Qinying Wu, q227wu
Mar 10th, 2020

1. Overview of Classes



Class Name	Description	Member Variables and Functions
City	Object that represents the city	***name: string*** the city name ***x: double*** the longtitude value ***y: double*** the latitude value ***p: int*** the population number ***r: int*** the cost of living ***s: int*** the average net salary
QuadNode	The individual node of the QuadTree	***parent: QuadNode**** the parent node of the current node ***NW: QuadNode**** pointer to the city located in the Northwest direction ***NE: QuadNode**** pointer to the city located in the Northeast direction ***SW: QuadNode**** pointer to the city located in the Southwest direction ***SE: QuadNode**** pointer to the city located in the Southeast direction ***city: City*** the city stored in the current node
QuadTree	The QuadTree object	***treesize: int*** the total number of nodes stored in the tree ***root: QuadNode**** the root node of the tree
		i (parent: QuadNode**, obj: QuadNode*): void inserts a city (obj) into the tree by comparing its the longitude and latitude value with the given (parent) node. Outputs "success" upon successful insertion and "failure" if obj is already existing in tree ***s (parent: QuadNode*, x: double, y: double): QuadNode**** searches for a city in the tree that matches the given x (longitude) and y (latitude) values starting from the given (parent) node. Returns a pointer to the found city if the city is found in the tree, and nullptr otherwise ***q_max (direction: QuadNode*, attr: string): int*** finds the city with the maximum value of the specified attribute (attr representing p, r, s) in the subtree of the given direction. It returns the maximum value found after comparing all the cities in that subtree, or 0 if no cities exist in the subtree of the given direction. ***q_min (direction: QuadNode*, attr: string): int*** finds the city with the minimum value of the specified attribute (p, r, s) in the subtree of the given direction. It returns the minimum value found after comparing all the cities in that subtree, or 0 if no cities exist in the subtree of the given direction. ***q_total (direction: QuadNode*, attr: string): int*** finds the accumulated value of the specified attribute (p, r, s) in the entire subtree of the given direction. It returns the sum of values after visiting all the cities in that subtree, or 0 if no cities exist in the subtree of the given direction. ***clear(obj: QuadNode**): void*** deletes all the nodes under the obj node including the obj node itself.

2. Constructors and Destructor Decisions

Class Name	City	QuadNode	QuadTree
Constructor	Takes in 5 parameters (name: string, x: double, y: double, p: int, r: int, s: int) and assign them to the corresponding member variables	Takes in 6 parameters (city: City, parent: QuadNode*, nw: QuadNode*, ne: QuadNode*, sw: QuadNode*, se: QuadNode*) and assign them to the corresponding member variables	Takes in two parameters (tree_size: int, root: QuadNode*) and assign them to the corresponding member variable
Destructor	Empty since no need to deallocate memory	Dereferences all the pointer member variables to nullptr	Dereference the root to nullptr

3. **Asymptotic Upper Bounds (*Assume uniform hashing for all functions)**

insert **search** **q_total**	Best time: O(1) if the quadtree has ≤1 node (contains only the root node or no node) Average time: O(lgn) if the quadtree is balanced Worst time: O(n) if all the nodes of the quadtree are in the requested direction (for q_total the function is called at the root node)
clear **print**	Best time: O(1) if the quadtree has ≤1 node (contains only the root node or no node) Average and worst time: O(n) if the quadtree has more than one node
q_max **q_min**	Best time: O(1) if the quadtree has ≤1 node (contains only the root node or no node) Worst time: O(n) if all the nodes of the quadtree are in the requested direction and the function is called on the root node
size	Best, average, and worst time: O(1) by printing the treeSize member variable of the QuadTree class

4. **Test Cases**
    a) Insert a city to …
        a. An empty tree
        b. A non-empty tree
            i. Non-duplicated city
            ii. Duplicated city
                1. Insert before clearing the tree
                2. Insert after clearing the tree
    b) Search …
        a. An existing city in the tree
            i. Before clearing the tree
            ii. After clearing the tree
        b. A non-existing city in the tree
    c) q_max, q_min, q_total of an attribute in a direction that has …
        a. no child nodes
        b. only one child node
        c. a subtree of >1 node
            i. all nodes have unique value for the requested attribute
            ii. two or more nodes share the same value for the requested attribute
    d) Clear when …
        a. the tree is empty
        b. the tree contains only one node (root)
        c. the tree contains more than one node
    e) Print when …
        a. the tree is empty
        b. the tree contains only one node (root)
        c. the tree contains more than one node
        d. before and after inserting a city

Examples:

```
size
i city1;1;1;12;12;12
i city2;2;2;13;13;13
i city3;3;3;23;12;4
i city4;4;4;48;5;3
print
size
i city5;4;4;5;5;5
q_max 1;1;NE;p
q_max 1;1;NW;s
q_max 1;4;NE;r
q_total 1;1;NE;p
q_min 3;3;NE;s
q_min 2;2;SE;p
s 1;1
s 1;2
clear
s 1;1
print
size
```