

ECE250 – Project 3: Quadtree Structure Design Document

Chintan Mistry | csmistry@uwaterloo.ca | 20722219 | Mar 6th 2020

Overview of Classes

1. Class: Node

- **Description:** Represents a node in the quadtree. Contains 4 directions which are child nodes and a key to hold city info.
- **Member variables:**

(Private) **NW, NE, SW, SE**: pointers to NW, NE, SW, SE children respectively
(Private) **cityInfo**: pointer to CityInfo Object that is the key of current node

Member Functions:

- **getCityInfo**: returns pointer to the CityInfo object stored in node

2. Class: CityInfo

- **Description:** Class to hold all data relative to a city (name, population, avg net salary etc.)

• Member variables:

(Private) **Name**: city name
(Private) **x, y**: represent longitude and latitude of given city respectively
(Private) **Population, CostOfLiving, AvgNetSalary**: represent Population, Cost of living and average net salary for a given city respectively

Member Functions:

- **getP, getR, getS**: returns value of population, cost of living and average salary for a given city respectively

3. Class: Quadtree

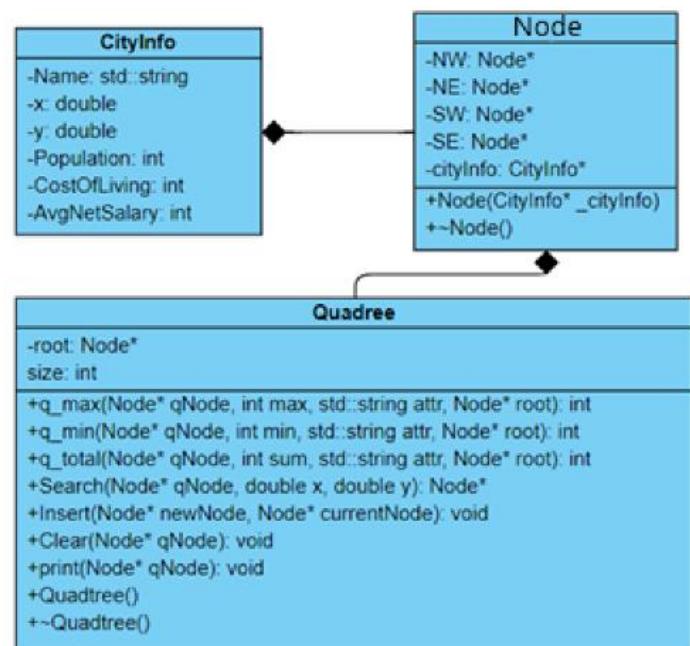
- **Description:** Represents a Quadtree data structure that contains nodes which correspond to different cities

• Member Variables:

- (Private) **root**: pointer to the root node of the Quadtree
- (Private) **size**: holds the current size of the Quadtree (number of nodes in the tree)

• Member Functions:

- **q_max**: Returns the maximum value of a specified attribute under a given node in a specified direction
- **q_min**: Returns the minimum value of a specified attribute under a given node in a specified direction.
- **q_total**: Returns the total sum of a specified attribute under a given node in a specified direction.
- **Insert**: inserts a node into the Quadtree
- **Clear**: deallocates all nodes and CityInfo objects in the Quadtree
- **Print**: prints an inorder traversal of the nodes in a tree
- **getSize**: returns the total numbers in the Quadtree
- **getRoot**: returns a pointer to the root node



Constructors & Destructors

- 1) Node Class: Constructor takes in pointer to a CityInfo object and sets it to the cityInfo member variable for that node. Default destructor.
- 2) CityInfo Class: Constructor takes in all relevant information for a city (x, y, population, name e.t.c) and sets them to their appropriate member variables. Default destructor.
- 3) Quadtree class: Constructor sets root to nullptr and size to 0. Destructor calls clear() method to deallocated nodes.

Performance:

-Insert: Average time complexity is $O(\lg(n))$ as insert recursively traverses the height of tree to find insertion location

-Search: Average time complexity is $O(\lg(n))$ since we are given (x, y) coordinates we can traverse height selecting the appropriate direction based on (x,y) as we move down the tree.

-q_total: Average time complexity is $O(n)$ assuming there are n nodes under the requested subtree.

- print: Average time complexity is $O(n)$ assuming there are n nodes in the tree since we must do constant access operation at each node.

-clear: Average time complexity is $O(n)$ as we need to deallocate each node in the tree.

-size: Average time complexity is $O(1)$ since we have a member variable that increments each time a node is added. (i.e constant accessing operation of member variable)

Best time complexity for all functions would be $O(1)$ and this will occur when the tree is empty.

Worst time complexity for all functions except size is $O(n)$ and this will occur when all subsequent cities follow one direction. (e.g city2 NE from city1, city3 NE from city2, city4 NE from city3)