P3 Design Document

The classes used in the implementation are-

1. *class quad_tree_node*

Member variables	string city_name; double latitude; double longitude; int population; int cost_of_living; int avg_sal; quad_tree_node* north_east; quad_tree_node* north_west; quad_tree_node* south_east; quad_tree_node* south_west;
Member functions	**quad_tree_node**(string name, double x, double y, int pop, int col, int avg);//, quad_tree_node* NE, quad_tree_node* NW, quad_tree_node* SE, quad_tree_node* SW); **bool insert** (quad_tree_node* current, quad_tree_node* node); **int search**(double x, double y); **int attribute**(string attr); **quad_tree_node* search_helper**(double x, double y); **quad_tree_node* direction**(string d);

Member Variables

The variables in the class are the variables that a node of the tree should have.

Member Functions

Constructor- The constructor initializes the variables to the parameters passed in, and pointers are initialized to null.

bool insert- The insert function that is called recursively by the other insert function in the tree class. The function takes in a pointer to the root and a newly formed node initially, and is then called recursively with an updated current. It returns true if successful, false if failed.

int search- Takes in the coordinates of the city, and searches for it recursively. It returns a 1 if successful, 0 if failed.

int attribute- Helper function that takes in the string and returns the suitable integer value of the attribute.

Quad_tree_node* search_helper- Helper search function that works the same way as the search, but returns the pointer to that node instead of an integer.
Quad_tree_node* direction- Helper function that gives suitable pointer according to string passed in it.

2. *class quad_tree*

Member variables	quad_tree_node* root;
Member functions	quad_tree(); **bool insert**(string name, double x, double y, int pop, int col, int avg); **int searchx** (double x, double y); **int q_max**(quad_tree_node* rootz, string attrib); **int q_min**(quad_tree_node* rooty, string attrib); **int q_total**(quad_tree_node* rootx, string attri); **void printx**(quad_tree_node* roott); **int size_of_tree** (); **void clearx**(quad_tree_node* rootl); **quad_tree_node* get_root**();

Member functions
bool insert- Called by the test file with all the attributes. This function calls the insert function of the previous class recursively.
Int searchx- Called by the test file with the coordinates of the class and calls the search function of the node class.
Int q_max- Takes in the pointer to the given city, and the attribute. Then calculates the maximum of the subtree in the direction given from that node.
Int q_min: Works the same was as q_max, but for the minimum of an attribute.
Void clear- clears the tree using post order traversal. This function is called recursively.
Quad_tree_node*get_root()- Returns the root of the tree.

I have not used any destructors in both classes.

TIMING ANALYSIS

The insert function uses recursion to start from the root, and then go down searching for a location to insert the new node. If the tree is just a linked list, it will have the worst insert time of O(n) where n is the height of the tree, as it traverses all the nodes atleast once. On the other hand, the minimum height of the tree in log n, where n nodes have been inserted. Thus, the best case scenario to insert a node is to have go through only log n nodes to find the suitable leaf to insert the node under. Thus, the best case time is O(log n).

Reference:

https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html

https://en.wikipedia.org/wiki/Quadtree