Nathan Tsai

Smallberg

304575323

CS 32

CS32 Homework 4

2) The class Coord in the call Map<Coord, int>::insert does not have an equals operator or an assignment operator. If you want to call insert and assign the Coord key to a node, you need to check if the Coord keys match. In order to check of the keys match, the Coord class must have an equals operator. This is the ultimate root cause of the problem. It is alright to just call Map<int, double>::insert because both the int class and the double class have an assignment operator and an equals operators already defined for them. Furthermore, the insert function calls the find function, which has a != operator. Without a bool operator!= function, there is no way to compare two Coords.

3b) In order to track how far you’ve gone down the tree (the path for the subclasses), you need to pass in a string parameter that keeps track of the “distance traveled.” Because the function is void and has recursive calls, it is impossible to keep track of the distance traveled without the extra parameter which serves as a way to store the string that records the distance traveled.

4a) Because there are three for loops each nested inside of the previous for loop, and because they all range from 0 to N, with an increment of 1 (i++, j++, k++), and because all the other statements are constant and/or irrelevant when N becomes big, the efficiency is **O(N^3)**.

4b) This is similar to 4a). However, the second nested for loop loops from 0 to the variable of the outer for loop. In other words, this second for loop is only processed about N/2 times. So, since all three for loops are nested within each other consecutively, you multiply how many times each for loop is processed. Since for loops 1 and 3 are the same as 4a), which is them being called N times each, the efficiency is calculated as follows: (N)\*(N/2)\*(N) = (1/2)\*(N^3). However, as N gets big, the coefficients become negligible, so the efficiency is still **O(N^3)**.

5) The first half of the code that determines which Map is bigger and which Map is smaller is constant run time O(1). In the second half of the code, there is a single for loop, which loops from 0 to the size of the smaller array. However, inside of the for loop, several functions are called that also have loops inside of them, just as the get call, which loops through an array probably once (which means the get can be treated as a for loop). Even though there are many calls in the for loop that have loops in them, and there exists more than one get call, they aren’t embedded, so the coefficient can ultimately be ignored, because as N gets big, the effect of the coefficient on the overall efficiency becomes negligible. Therefore, the efficiency is N\*N, which results in **O(N^2)**.