Nathan Tung
Com Sci 32 (Lec 2)
March 5, 2012

CS32 Homework 4

2. Explain in a sentence or two why the call to map<Point, int>::insert causes at least one compilation error.

→The class Point doesn't have an equality or assignment operator. That operator is needed to assign the "Point" key to the newly-inserted node and to check if the "Point" keys match.

3b. We introduced the function listAllAuxiliary. Why could you not solve this problem given the constraints in part a if you had to implement listAll (with one parameter) as the recursive function?

→We needed a variable to keep track of the string path for the subdomains (that we've traveled thus far), the parameter variable path in this case. The void function has no other way of concatenating and storing the path string without that extra parameter.

4a. What is the time complexity of this algorithm, in terms of the number of basic operators performed? Why?

→There are three for-loops, each embedded in the other. They all have variables that begin from 0 and end at N, with an increment of 1. All the other actions are constants (though possibly multiplied by N in some cases) and they are ultimately dropped. Therefore the efficiency would be $O(N^3)$ or big-O of N-cubed.

4b. What is the time complexity of this algorithm? Why?

→The idea of this second algorithm is similar to the first. The biggest difference comes in one of the for-loops, in which that variable begins from 0 and ends at the variable of the outer-for-loop. Essentially, that particular for-loop is only called about half of N times. Therefore, we can say that its efficiency is N*(1/2)N*N. However, since we drop coefficients and keep only the highest order, the efficiency would still be $O(N^3)$, or big-O of N-cubed.

5. In terms of the number of linked list nodes visited during the execution of this function, what is its time complexity? Why?

→The first part of determining the bigger/smaller map runs in constant time. Then we get to a for-loop which will run through about N times. During that for-loop, we call several functions (like the get function) that loops through presumably around N times again. These functions, however, are not embedded, so the coefficient is eventually dropped. The time complexity turns out to be $O(N^2)$, or big-O of N-squared.