Angel Jimenez

CS 32 – Spring 2018

604-667-905

**2)**

The insert function takes a single int, and compares it to the value stored inside of the node. This works well for most cases but not for the special case of sending a Coord object. C++ has Boolean comparison operators already set so it can handle the int comparison well. The problem arises when we attempt to compare the class Coord object. We failed to implement an operator overload which would tell the compiler how to handle the comparison and the appropriate Boolean value to return.

**4b)**

Constraints: no additional container, must use recursion. Cannot modify any other code.

The two-parameter overload takes to items for input, one item for updating the string and the second for traversing the vector. For our recursive implementation to work, we need to reduce the problem, we must also send enough information for the function to operate. Once the end case is met, our recursion is complete. The end case in this case is if the file() item points to nothing, which means we’re looking at the end of the vector. Then it prints the directory and returns. Beyond the exit condition, a for loop checks all the directories. Essentially, given only 1 parameter, and our constraint of using recursion, we wouldn’t have a method for traversing the vector list.

**5a)**

I believe this problem is O(N^3). My reasoning is due to the three for loops which dominate the other computations. We can ignore the computations before the outer for loop because they’re constant time. The items inside the outer for loop run N times. The items inside the middle for loop run N times. The items inside the inner most for loop run a total of N times and the items inside this for loop are constant time. Therefore, we have 3 for loops nested that run N times each. Thus we get O(N^3)

**5b)**

I believe the time complexity remains at O(N^3). This is largely due to the fact that even though we made improvements to our algorithm, it is still an inefficient algorithm. Fundamentally, the algorithm hasn’t changed, it may have become a bit more efficient, but it is still to the order of three. The items in the outer loop run N times. The Items inside the middle loop run i times, which at max is N/2 therefore the items in the middle loop run N times. Inside this loop is another for loop which runs N times. Therefore, we get O(N^3).

**6a)**

A key piece of information presented is that every input has N elements. Thus we need not worry about which input may dominate the other, they’re relatively equivalent. First outer for loop runs nmin times, which in its worst case is N. Therefore, we assume the contents have N iterations. Inside the for loop, we have several call functions. We can trace them and find they’re N iterations but we know that we’re dealing with linked list. Thus we generalize that in the worst case, our get function goes to the end which can be N large. The same applies to the insert function. Thus we get N\*N which is O(N^2). We see the same thing in the second for loop thus the total is O(N^2).

**6b)**

Both for loops run a max of N times depending on the size of the lists, which can both hold N elements. Assuming worst case scenario, they both hold N elements, therefore the first for loop runs N times. Both insert before functions run N times, and they themselves are of constant time since the insertion is at the beginning. The second for loop also runs N times. Therefore, the time complexity for this function is O(N).