CS 32 Lecture 2 – Professor Smallberg Leo Gretzinger

HW#4 Write Up 3/4/19

Question 2:

Set<Coord>::insert causes an error because insert uses a comparison operator that is never defined for the Coord object.

Question 3b:

You would not be able to recursively implement the one-parameter function because when analyzing the lower sub-menu-items, you would have no way of knowing the names of the previous menu items. For example, at the plain menu item with just “Window,” the spec requires that listAll print out File/New/Window, but there would be no way of accessing the File and New names without a recursive “path” string variable.

Question 4:

1. This algorithm is O(N^3). This is because the k (third) loop has complexity O(N-j), then the j (second loop) has complexity O((N-i)\*(N-i)), then the i loop has complexity O(N\*N\*N) = O(N^3). Essentially, with nested for-loops, barring recursion, the complexity is O(N^m) where m is the number of loops. All the other lines are O(1), which don’t multiple the O value.
2. This algorithm is O(N^3), as well for very similar reasons as the previous algorithm in 4b. The only difference is that the j (second) loop only iterates i times. This creates less iterations in total and therefore a smaller constant of proportionality, but this is irrelevant when determining the power of N in the complexity. All the other lines are O(1), which don’t multiple the O value.

Question 5:

1. The worst case time complexity for the unite function is O(N^2). This is because in the worst case, either the “else if” or the “else” in unite are used, calling the operator= function in Set.cpp. This function uses the Copy and swap technique, which calls the Set copy constructor which has a one for loop and is therefore O(N) and the swap function which has all O(1) statements. After the if-else statements, the unite function has a for loop that iterates until the size of set2, which is N, making the complexity of the loop O(N). Then there are three lines inside the loop. The constructor makes a default construction of ItemType, which presumably has no loops within it. Then, the get function is called from Set.cpp, and no matter what this function iterates through half the Set in question, making the complexity of the function O(N/2), where the ½ is negligible. The last line in the for loop calls the insert function from Set.cpp, which calls a function findFirstAtLeast. findFirstAtLeast loops through the entire Set in the worst case, making its complexity O(N). The rest of the insert function has O(1) calls, making the complexity of insert O(N). Within the for loop, the complexity is 2O(N) after adding get and insert, and multiplying this by the complexity of the for loop, the complexity would be 2O(N^2), where the 2 is negligible. As a result, the overall time complexity for the unite function is O(N^2).
2. The time complexity of this function is O(NlogN). All of the for and while loops in this function are of time complexity O(N) (the last for loop is O(2N)) and the sort algorithm uses O(NlogN). When adding all of these terms up, the overall complexity would just be O(NlogN) because the all the rest of the O(N)’s have a smaller order and are therefore negligible.
3. The time complexity of this function is O(N). In the worst case scenario, the else if or the else is called and therefore the assignment operator which is an O(N) function. Then the while loop consists of O(1) statements. Lastly, the for loop consists of insertBefore calls, which is O(N). Therefore, the O(N) defines the complexity for the algorithm.