CS 32 – Homework 4

2.) We must recall that the one argument form of insert, tries to evaluate the position as to where the item should be placed. It does this by comparing the value at the current position, so as to position the argument before the bigger value, i.e. arranging it in ascending order. However, when trying to use the one argument form of insert with Complex, we have to realize, that for the complex object to be successfully inputted, it needs to have overloaded the > operator, which the class doesn’t, and this is why we receive an error.

3.b) For the solution to be successfully recursive, we need to be able to break the problem down further into simpler sub problems, which requires the Domain\* d parameter to access the subdomains of domain (i.e. make the problem smaller), but at the same time we also need to edit the labels of the domain which requires the string path parameter, which thus means to solve this problem recursively we need at least 2 parameters and nothing lower.

4.a) The time complexity of this algorithm is O(N3) because in each instance of the function, we get 3 levels deep essentially as we have 3 for loops. All the if statements are of constant runtime, but notice that each of the for loops run until we hit N, and therefore each for loop has a runtime of O(N) and as we go higher out of the innermost for loop, we see the relationship between the embedded for loops, and see that the overall out for loop has a total time complexity of O(N3), which for the higher order ends up being the overall time complexity of the algorithm.

4.b) The time complexity remains at O(N3) because, in the worst case scenario, we see that i will be N and so the problem becomes analogous to 4.a). In the best case scenario i will be zero and the problem reduces to a time complexity of just O(N); however, we are looking for an average case scenario. In such a case we can assume that i will become N/2 and the two inner for loops will then run in such a way that we get an inner time complexity of O(0.5N2,) which for our purposes is equal to O(N2). Combining this with the outer for loop, we get an overall time complexity of O(N3) for this problem.

5.a) The time complexity for visiting each item in the linked list for our first implementation of interleave is about O(N2). The reason being, looking at the first for loop, we see that k varies essentially from 0 to N. Then within this for loop, we use get(int index, ItemType value) again goes through the linked list k to N-1 times , which in the worst case will be about N itself(twice over because it does so for both sequences, technically making it O(2N) within the for loop. It should be noted that the second half of the program just copies over the remaining values from the bigger sequence into our result, which also performs at an O(N2) time complexity. Therefore both for loops in this algorithm essentially are of O(N2), making the overall algorithm of O(N2) since we are not concerned about coefficients.

5.b) The time complexity for visiting each item in the linked list for the second implementation of interleave is about O(N), which is better than our first implementation of interleave. The reason being is that our first for loop loops through both our sequences for about N times, while insertBefore only looks at the value at our current position in the linked list and inserts the value there, which is a constant time O(k) function. The second for loop, fills in the remaining items from the bigger sequence, which is again at worst the same as our first for loop ~ O(N). The swap is of O(N) as well, thus making our overall time complexity O(N) without care of coefficients.