Artiom Arutiunov

ID: 504597668

Professor Smallberg

5/24/16

Homework 4

2) The problem with creating a Set containing the Bomb class is that the insert function calls the find function to ensure that the passed insert value is not already present in the set; but since the find function calls for the node’s pointer to compare the values between the two bomb objects, we get a compilation error because the compiler has no way to know whether the bomb is equal to the target “value” we are searching for.

3B) I wouldn’t be able to implement the function with one parameter recursively because I need a string parameter and a pointer parameter to advance through the branches of the tree and print out the entire directory. Without the string node, there would be no way for me to include the previous directories such as “/My Pictures/Fun”, and without the pointer there would be no way to advance through the tree and eventually hit an edge case for the recursive function. Ultimately, I need both parameters in order to implement the function with the entire directory of strings that accumulate in the leaf nodes.

4A) The time complexity of this algorithm is O(N^3). This is because there are 3 for loops that all execute and iterate based off of the value of N, one nested into the other. Although there are several lines of code which in of themselves are have a time complexity value of N (bestMidPoint[i][i] = -1), (int minDist = maximum possible integer), and (int d = dist[i][k] + dist[k][j]), these statements don’t matter in the scope of the N^3.

4B) Although in this algorithm in middle for loop has the condition (j < i), this does not change the fact that the algorithm still has a time complexity of N^3. This is because the complexity created by the outer for loop and the middle one is still N^2, just at a smaller value. However, once the algorithm reaches high levels of iterations, it still behaves as N^3 (the innermost for loop still iterates to N each time).

5A) The worst-case time-complexity of this algorithm is O(N^2). The worst-case scenario in this algorithm is when set1 and set2 are not the same set, and the result set is not set1 or set2 either. This way, we iterate through the for loop toward the bottom of the algorithm N amount of times, and within each iteration we call set’s get function (which has an average time complexity of (0.5\*N). Thus when we consider the outer for loop and the corresponding “get” function called, after dropping coefficients we yield a time complexity value of O(N^2).

5B) The time complexity of this algorithm is simply O(N log N). This is because the vector function “push\_back” has a time complexity value of O(1), and that is done N times for set1. The same process is then repeated for set2, giving us a value so far of O(N + N). The sort function called is an O(N log N) algorithm, which dominates the O(N + N) result we had previously. Deleting the resulting nodes has a value of O(N), and copying the unique items from vector v into set result takes O(N) since v previously copied all its elements from sets set1 and set2 that have a size N. The ultimate result is O(N log N) because this has the highest degree power and will come to dominate the other values.