Tyler Stovsky

705512370

2. The call to Set<Coord>::insert will cause a compilation error because it uses the ‘>’ comparison operator, which is not defined for the Coord class. We would have to overload this operator for the Coord class in order to get it to compile.

4b. We couldn’t solve this problem with only a one-parameter ListAll because the recursive calls need the previous parts of the path. With only one-parameter, there is no memory of the previous paths so we would be unable to create the overall path.

5a. The time complexity is O(N^3). This is because there are three nested loops which each run from 0 to N. No matter what, each loop must iterate N times.

5b. The time complexity is still O(N^3). Even when taking advantage of the symmetry, the first two loops evaluate to O(N(N - 1)/2), which when looking at the highest order term becomes O(N^2). Then, with the last nested loop, which is O(N), the overall time complexity is O(N^3).

6a. The worst case time complexity is O(N^2). At the bottom of this function, there is a for loop which runs from 0 to N. Within this for loop, there is a call to the “get” function which contains a for loop that runs N times, as well as a call to “insert” which contains a for loop that also runs N times. Since these are separate calls, the inside is O(2N), which means the overall worst case with both nested loops is O(N^2).

6b. The time complexity is O(N\*Log N). The first two four loops both have a time complexity of O(N). The “sort” function has a time complexity of O(N\*Log N). The while loop is O(N), and the last for loop is also O(N). Therefore, the bottleneck is the “sort” function, which means the overall time complexity O(N\*Log N).

6c. The time complexity is O(N). In this function, we start by comparing the sets which is O(1). We have a while loop with a time complexity of O(N), and a separate for loop that also has a time complexity of O(N). So, the overall time complexity is still O(N).