2. The insert function uses the “==” operator to check if the value inserted is already present, and the == operator is well-defined for integers and strings. But for the Coord class, there is no well-defined == operator for object variables, so we need to define one ourselves for insert to work.

3b.

In part a, we were able to output paths that had previous values because we stored those values into the parameter path that we passed as we called recursively. But without that extra parameter, we have no way of storing the compound Menu Item and passing it into the next call.

4a.

For the loop with int k, the statements are run N times. This loop is inside a loop with int j, so the k loop is run N times. Now this loop is inside a loop with int I and is run N times. Therefore, it’s N\*N\*N = N^3

O(N^3)

4b. For the 2 outer loops, we are basically traversing the two 0 + 1 + 2 + … + N – 1, which is pretty much N^2. Those loops together contain another loop that runs N times, so we are adding N N^2 times, which is just N^3.

so still O(N^3)

5a.

For worst case, let’s suppose s1 and s2 are completely distinct and that result is empty. This means that result with be reassigned s1’s elements, so result will now have N elements. Now, we are reinserting N elements from s2 into s1, and to do so, we will end up going from 0 to 1 to 2 to … N – 1, which simplifies to N times N – 1 divided by 2.

So O(N^2)

5b. Let’s call v s3.

Always assume worst case scenario, so we’re pushing N items from s1 into s3, then another N from s2 to s3. We’re told that s3 is sorted using an O(N log N) algorithm. Then, we’re erasing the old value of \*this, which had N elements. Then, we’re inserting all of s3, which should have 2N worst case scenario, into result. Since we only care about the leading term, we only care about the NlogN, not the other N terms.

Therefore, it’s O(N Log N).

5c.

The lines before the for loop is only comparing the references to the sets, so no nodes are visited. For the for loop, we are visiting all N elements in v because of the get function. Then we have O(1) from inserting an element to the back, but because we’re doing this N times, we also get O(N). So simplifying, we get

O(N)