D2.

Problem 1.

The clerk should use selection sort as it has many more comparisons than exchanges and we would rather have to do a lot of something that is cheap, rather than a lot of something that is expensive. Additionally we could not use merge-sort since there is only one space to hold the crates and not two or more.

Problem 2.

If we used a simple double for-loop and checked if a certain index were equal to another index, the time complexity would be n^2, which is too high. Instead we can merge-sort (nlog(n)), and then simply go once through the loop and check if any index “i” is equal to “i+1”. All numbers that are the same would be next each other after we sorted. The time complexity would become nlog(n) + n which is simply nlog(n).

Problem 3.

For every GPS coordinate, calculate the distance between the nearest neighbor. Then sort these values and the highest value should be the farthest away.

Problem 4.

We loop through each vote and for every candidate we have an integer variable that starts at 0. While looping, when we see a vote for a candidate we add 1 to the variable for that candidate. The biggest variable will be the candidate with the most votes.

Problem 5.

Instead of sorting multiple arrays, this can be thought of as merge sorting one large array, and we have already split the array up into k pieces. From here we merge-sort each N sized array. Then we merge the arrays together. This will take log(k) * N. log(k) because of the number of merges and N since the arrays are N in length.

Problem 6.

We use a double for-loop. We check if each pair of elements satisfies the conditions and if so, returns them.