1. The algorithm divides the set of drones into smaller subsets, recursively finds the minimum distance in these subsets, and then efficiently finds the minimum distance across the subsets. Here are the steps:

Sort the drones based on their x coordinates.

Recursively split the drones into two halves.

Independently find the minimum distance in each half.

Compare the distance near the dividing line to find the overall minimum.

Time Complexity.

Initial x coordinate based sorting taken nlogn time. For later recursion part also taken n logn (n drones and logn recursion). So,

Best Case : Θ (nlogn) Worst Case : Θ (nlog) Average Case : Θ(nlogn)

2. The algoritm implements divide and conquer to find the convex hull of the set of sensors. Here are the steps:

Find the bottom-left sensor as the starting point.

Sort the rest of the sensors based on the polar angle they make with the bottom-lef

Iterate through sorted sensors, forming a hull by retaining sensors that make a left turn. Discard the sensors causing right turn because they are inside the hull.

Time Complexity:

Sorting takes nlogn time. Hull construction takes n time thus;

Best Case : Θ (nlogn) Worst Case : Θ (nlogn) Average Case : Θ (nlogn)

3. The algorithm builds a matrix where each cell represents the cost of converting a subset of DNA sequence into subset of another. This is Needleman-Wunsch algorithm, a type of dynamic programming here are the steps;

Initialize matrix

Fill the matrix based on given costs.

Trace back from dp[m][n] to find sequnce of operations.

Time Complexity:

If we say that matrix dimension is $m \times n$ than the algorithm complexity is $\Theta(mn)$ becuse it takes constant amount time to fill the cells.

Best Case: $\Theta(mn)$ Worst Case: $\Theta(mn)$ Average case: $\Theta(mn)$

4. This algorithm avoids recomputation of the discounts for the same sequence of stores. Uses dynamic programming, stores the results and uses them when needed. Here are the steps;

Define the subproblem as the maximum discount achivable for a spesific subset of stores.

Initialize table with base cases.

Calculate the maximum discount for each subset.

The maximum achivable discount is found in the table entry that corresponds to the subset containing all stores.

Time Complexity:

For each subset of stores, the algorithm iterates over n stores to calculate maximum discount for that subset. So the time complexity;

Best Case: $\Theta(2^n n)$ Worst Case: $\Theta(2^n n)$ Average Case: $\Theta(2^n n)$

We can say that for this particular problem we don't need to use dynamic programming as previous question has $\Theta(2^n)$ time complexity.

5. This greedy algorithm selects antennas based on their coverage ranges while ensuring that no two activated antennas intersecting. The goal is maximizing the number of activated antennas without causing interference. Here are the steps;

Sort all antennas by the end point of their coverage range in ascending order.

Select antennas based on sorted order. Here we make a series of local, greedy choices.

By selecting antennas with the earliest end points, we maximize coverage while avoiding overlap.

Time Complexity:

The initial sorting of antennas takes nlogn time. After sorting, algorithm iterates through antennas once, This takes n time. So the time complexity;

Best Case: $\Theta(n \log n)$ Worst Case: $\Theta(n \log n)$ Average Case: $\Theta(n \log n)$