Q-1) In this algorithm we are expected to design a divide conquer algorithm to find closest pair by using given 2D array.

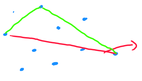
This algorithm works according to x values. So they should be sorted. After sorting it divides two pieces until there are at most three drones. In the base case (there may be 1,2,3 drones), it finds shortest distance by using brute force technique. This was the divide part. In the combination part, we should check if there is closer points between left and right parts. Which is called delta.

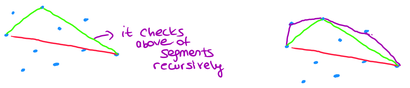
And not to check all the points between left and right parts, it only checks midpoint-delta and midpoint+delta. It checks points by using strip and it checks other points to determine if there are points closer than delta. This combination goes on during back trace of recursive calls. And in the end it finds closest point

* Strip operation, which involves sorting a subset of points takes $O(n)$. However, since the strip only contains points within a small distance from divide line, the actual number points limited.

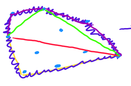
Q-2) To secure perimeter and use minimum number of sensors. We should determine the sensors on the perimeter.

I thought that this problem supposed to be convex hull problem. When I researched that how I can implement this problem by using divide and conquer approach I found quickhull problem. First of all it sorts all the points according to their x coordinates. Because the goal is finding longest distance.





then it does same thing for the below part.



$T(n) = 2\,T(n/2) + O(n)$ ⇒ Finding farthest distance and creating segments

$a=2$, $b=2$, $d = O(n^d)$, $d=1$

Master Theorem

$a = b^d$, $2 = 2^1$ ⇒ $T(n) = O(n \log n)$ recursive function time complexity

main time complexity ⇒ sorting

$T(n) = O(2n \log n)$

Q-3) In this problem we are expected to convert one DNA sequence to the another. And each operation has cost. To achieve this we store data in the 2D dp array (dynamic programming). And for the next step we consider old data from dp array. By comparing and adding new cost to old data we find the minimum cost to apply operation to the data. It goes like that until end of the dp array. According to old data it determines new operation to change data.

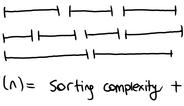
$T(n) = O(m \cdot n)$ which m is the length of first input, n is the length of the second input.

Q-4) In the homework 3, I found all the possible subsets of the stores. Actually this is the sum of combination $\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \dots + \binom{n}{n}$. Then according to shopping places, it calculates maximum discount So, my approach is some. just I found all the subsets by implementing dynamic programming approach. It search dp array, it finds subsets and it add new data to dp array by using old data.

$T(n) = n \left(\sum_{i=1}^{2^n} \right) \rightarrow 1+2+\dots+2^n = \frac{2^n (2^n+1)}{2} = 1^{2n-1}$ (1+2+3 = 3·4/2) it grows exponentially

$T(n) = O(n) \cdot O(2^n) \Rightarrow$ So overall complexity $\Rightarrow T(n) = O(n \cdot 2^n)$

Q-5) To find max antenna from the street, I should find optimal solution by using Greedy algorithm. And the optimal solution is finding first ending point so, to find this, inputs are sorted according to their ending points. After chosing first ending input it will find second first ending input. If they have intersection it continues to search greedy.



$T(n)$ = Sorting complexity + iterating antennas

$= O(n \log n) + O(n)$ it dominates

So over all time complexity is

$T(n) = O(n \log n)$