Zewei Zhong 122090814

Report for CSC3100 Assignment 1

Problem 1: Let's Sort

Language: Java 8

All algorithms of sorting can solve the problem if not concerning time and space limits, such as bubble sort, insertion sort and merge sort. Besides, Python itself has function sort() and sorted() that can sort the elements in a list and C++ also has libraries regarding the sorting vectors. All such sorting methods could be possible solutions to the problem.

I solved the problem using merge sort. By using mergeSort1 and mergeSort2 (In a version of assignment, it says when id is 3 or 4, I have to sort the element "a", which I considered means sorting the value of "a" instead of the location, otherwise can be done in only one method. I found the problem after it gave me 2 WA in test cases 4 and 5 in OJ system. I simply change the output while didn't merge the 2 functions.), we use recursion to divide the given array into small pieces by resetting the beginning and ending index. After all the "division" process, use 4 different merge method (merge1, merge2 …) to merge the small pieces in correct order defined by the 4 methods in the problem.

The reason why I use merge sort is that it provides the least time complexity among all algorithms (I currently haven't learnt any sorting methods in Java which is both natural and available in this assignment), which is O(n*log(n)), faster than all other algorithms especially when number of objects to be sorted are large. That's why it is chosen although it is the hardest to write (almost 200 line of codes).

Problem 2: Detecting Tyranids

Language: Python 3.11

There may be 3 methods to solve the problem. The first method is simply denoting all k's and sum them up using 6 for-loop in for-loop structure, which has the worst time complexity of O(n^6). The second method is using a 10^5 times 10^5 table to and use the location to denotes n and m and value in the location to denotes k. Finally sum them up. This method is faster (O(n)) but cost huge internal memory for a 2-dimensional array of 10^5 times 10^5.

The method I use is finding the rectangles at 2 sides of a given location in the given rectangle. Then multiply all length of sides of each rectangle and the given k value. Sum all such values and mod 998244353 to obtain the result. This method is provided in Sample Input 1 and Sample Output 1.

The method costs very little memory space because it directly obtains an integer result after obtaining each line of the input. Besides, time complexity of the method is O(n), which is not higher than the other 2 methods. Only one for-loop is used in the method. By iteratively summing up all those integers, we have the result after all input is read.

However, codes in Java with the same method provide wrong answer (WA) in the test cases 8, 9 and 10 in the OJ system and I did my best to optimize it (from 5, 7, 8, 9, 10 WA to 8, 9, 10 WA). I turned to Python to avoid discussion of data types in the question, by which I only have to write 6 lines of codes in total to obtain all ACs.