Qiyao Zhou

Z5379852

Question 2

2.1:

First, since X is a sorted array, duplicate numbers will only appear consecutively. Therefore, an empty hash map can be created to store the number of occurrences of each integer in the array.

Next, iterate through the X array starting from x_1 and record its index (index (0)), then each time go backwards to make an equality judgement (i.e., judge x_i? = x_{i+1}), if the two values are judged to be equal, then continue to iterate backwards, otherwise, put xi into the hash map with the value (index (x_i) - index (0) + 1), then replace index (0) is replaced by the index of x_{i+1}.

The final hash map is the result of the number of occurrences of each integer.

The worst case is when the n integers in the array are not the same as each other, then it takes O(n) to traverse the array, O(n) to judge, O(n) to write the data and O(n) to replace the index, the total time complexity is 4O(n)= O(4n) = O(n).

2.2:

First, an empty hash map can be created to store the number of occurrences of each integer in the array.

Next, iterate through the Y array starting from y_1, each time looking for a keyword in the hash map, if there is no y_i in the map, then y_i is set as the key and its value is recorded as 1, if there is y_i in the map, then the direct value count is added by 1.

The final hash map is the result of the number of occurrences of each integer.

In this way, the whole algorithm is expected to complete the array traversal in O(n), and the keyword lookup and value modification are also recorded as O(n), so the total time complexity is 3O(n) = O(3n) = O(n).

2.3:

First, considering that Z is a sorted array and there are k independent integers known, it is also possible to create an empty hash map to store the number of occurrences of each integer in the array.

Next, traverse the Z array from z_1 (for convenience, the point traversed is denoted as z_a), each time starting from the traversed position to the end of the array using the bifurcation method to find the furthest point z_b in the array that is equal to the traversed value, and then add the value of za as a key to the hash map, the value of which is (index (z_b) - index (z_a) + 1), and the next traversal is from z_b +1.

The final hash map is the result of the number of occurrences of each integer.

In this case, the algorithm requires at most k iterations, each of which requires a bifurcation (logn), with a total time complexity of O(klogn).