COMP3121 Algorithms and Programming Tech 2022T3 assignment 1
Chen En (z5335039)

Question 2 Counting Occurrences
[30 marks] Answer the following:

2.1 [6 marks] Let X = [$x_1, \ldots, x_n$] be a sorted array of n positive integers. Design an algorithm to count the number of occurrences of each distinct integer in X in worst case O(n) time.

Given by the question, we know that X is a sorted array with consecutive positive integers. We are using linear search which goes through each element of the array till we search the end element of the data set. To implement this, we set a counter to keep track of the numbers of occurrences that we meet each distinct integer while iterate through X.

Algorithm
Firstly, we initialize counter variable with 0. Iterate from $x_2\ to\ x_n$, the counter increment while the current element is not equal to the previous element that we were checking ($x_i \neq x_{i-1}$ that i is the current index and x_i present the current element.)
Case 1: If $x_i \neq x_{i-1}$, we increment counter by one, then compare next pair of elements. (Take x_{i+1} as new current value and compare with x_i which becomes the previous element now.)
Case 2: If $x_i = x_{i-1}$, we then compare next pair of elements. (Take x_{i+1} as new current value and compare with x_i which becomes the previous element now.)

The procedure keep repeating till we compare the last element x_n, then return the value of the counter.

Time complexity
Since we iterate through all n elements in X once, the time complexity is O(n).

2.2 [12 marks] Let Y = [$y_1, \ldots, y_n$] be an unsorted array of n positive integers. Design an algorithm to count the number of occurrences of each distinct integer in Y in expected O(n) time.

We use the Hashmap to store the number of occurrences of each distinct integers in array Y. By using the hash table, we traverse through the elements from y_1, to y_n and keep track of the elements that we have visited in a hash set.

For each element y_i in Y, if y_i is not a key in the Hashmap, then add a new key-value pair (y_i, 1) into Hashmap.
Otherwise, if y_i is already a key in the Hashmap, then increase the value of the key yi by 1.
The Hashmap is the number of occurrences of each distinct integer in Y.

Time Complexity
Since we only traverse all the elements once in Y, and the complexity of the operations of Hashmap is O(1), the time complexity is the size of Y, which is O(n).

2.3 [12 marks] Let Z = [$z_1, \ldots, z_n$] be a sorted array of n positive integers. You are also given an integer k ≤ n, the number of distinct integers in this array. Design an algorithm to count the number of occurrences of each distinct integer in Z in worst case O(k log n) time

Apply the divide and conquer algorithm, and use binary search to count the number of occurrences of each distinct integer in Z.

Since Z is sorted, we can use binary search to find the leftmost and rightmost indices of the distinct number.

For the leftmost index, we set the lower bound to 0 and upper bound to n-1 at first. In each iteration in the binary search, we check the middle of lower bound and upper bound (m).

If Z[m] is greater than Z[0] then set the upper bound to m-1. Otherwise if Z[m] is less than Z[0] then set the lower bound to m+1. If the above conditions were not met, which means Z[m] is equal to Z[0], then we check if Z[m-1] < Z[0]. If Z[m-1] < Z[0] then the leftmost index is m. Otherwise set the upper bound to m-1.

We can use a similar approach to find the rightmost index of the distinct number.

We perform binary search twice on array Z, hence the time complexity of the above algorithm is O(log n)

After finding the leftmost and rightmost indices, we can calculate the number of occurrences of the distinct number, which is rightmost index - leftmost index + 1. The next distinct number is Z[rightmost index+1]. We can use the above algorithm to find the number of occurrences of Z[rightmost index+1].

Time Complexity
Since finding the next distinct number cost O(1) time, and there are k distinct numbers, the time complexity of finding all the occurrences of k distinct numbers is k * O(log n), which is O(klogn).