CS3334 – Data Structures – Project

Ryan Dyson Darmawan – 57883349

Compilation method : C++ 11, flag: -static -std=c++0x

1. Brute Force

Submission ID : [304479](http://acm.cs.cityu.edu.hk/oj2/index.php/submission/304479)

Methods and data structures used:

A computer screen shot of a program code

Description automatically generatedFor getting the input of the user, we convert the stringstream into an array (vector) of integers, this has a time complexity of O(n).

A screen shot of a computer program

Description automatically generated

The solution itself is 2 nested loops.

The first loop iterates over each element of the input. It picks out one element of the input array.

The second loop iterates over each element again. If the element of the outer loop is equal to the element currently selected of the inside loop, then a counter is incremented. If the counter is larger than the max, then the max is assigned to the current counter and current element picked of the outer loop.

The Solution is O(n^2) since it loops over the elements of the input array for each element of the array. Meaning, each element iterates through every other element.

The final time complexity for worse case scenario is

O(n) + O(n^2) = O(n^2)

1. Using a sorted list & Stack

Submission ID : [304480](http://acm.cs.cityu.edu.hk/oj2/index.php/submission/304480)

For getting the input of the user, we slightly modify on how we store the elements inside the vector. The vector now contains a pair of integers, one to store the original index of the element. However, this does not change the time complexity of the way we get the input, just the space complexity. Time complexity is still O(n)

A screen shot of a computer program

Description automatically generated

Then we sort the array based on the value of the element. This way all instances of the element will be adjacent to each other. The way I’ve implemented the sorting is by using bubble sort which has a O(n^2) time complexity.

A screen shot of a computer code

Description automatically generatedThe main data structure used in this solution is the double ended queue (implemented with deque). Each element of the deque is a pair of integers, the first containing the very first original index of the same adjacent element, and the second the frequency of the elements. Then we iterate over each element of the sorted array, if the current array is the same as the adjacent element before it, then we increment a counter of the appearances of the current element. If there no same adjacent element, then we push it on to the back of the deque the pair of the index of the first element of all the adjacent same element and the current count. This block has time complexity of O(n).

A screen shot of a computer code

Description automatically generated

Then we push the last element on to the stack aswell. Now we simply find the min index of the element in the stack that its count is the same as the maxCount. Then we simply print the element of the minIndex of the original input array. This block is also O(n).

Via this method, we see that the worst case time complexity is completely based on the implementation of the sorting algorithm.

The final time complexity for worse case scenario is

O(n^2) + O(n) + O(n) = O(n^2)

1. Hash map using unordered map

Submission ID : [304481](http://acm.cs.cityu.edu.hk/oj2/index.php/submission/304481)

For getting the input from the user, we user the exact same method as solution 1 and has a time complexity of O(n).

This time, each element will be a key on the hash map and its corresponding value pair would be the frequency of the element’s appearance.

A computer screen shot of a program code

Description automatically generated

The first loop, counts the frequency of the elements within the vector. The second loops picks which elements has the highest frequency, we simply keep track of the frequency of the elements, and the element itself through max and maxIndex. Then we simply print the result. The time complexity here is O(n).

The final time complexity for worse case scenario is

O(n) + O(n) = O(n)