Name: Yiyan Yang

zID: z5183946

# Question 1

## Question Part

You’re given an array A of integers, and must answer a series of questions, each of the form: “Given two integers, and , how many elements of the array A satisfy ”. Design an expected algorithm that answers all these queries.

## Solution

Firstly, we create a hash table to record the number of occurrences of each element of with time expected as every search and insertion cost time.

Secondly, follow the processes listed below for every question to answer all the queries in time.

1. Calculate for each pair of and by dividing by . If is not an integer, the answer for this question is 0, because is an array of integer.
2. If is an integer, then find the value of the in the hash table we calculated before, the answer is then the value. Moreover, if the element is not in the hash table, just return 0. The reason is that the hash table records the number of exists in .

The complexity of this algorithm is then .

# Question 2

## Question Part

You are given an array of integers and another integer .

a) Describe an algorithm that determines whether or not there exist two elements in whoses absolute diﬀerence is exactly .

b) Describe an algorithm that accomplishes the same task, but runs in expected time.

## Solution

1. At first, we need to sort the array S using an -cost sorting algorithm, Merge Sort.

Secondly we run binary search twice for each element () to find whether or is in the sorted array to find if there exists at least one element whose absolute difference to element is exactly . It is known that the complexity of binary search is and we search for n times.

The complexity of sorting and searching are both .Hence, this is an algorithm.

1. We could use hash table to optimize the efficiency of searching certain element to and avoid sorting process, and then find the answer by the similar method as that in (a).

At first this algorithm requires a hash table to store all the elements in with insertion in time for elements, so the complexity of this part is .

Then we search for each element () to find if or is in the hash table in for at most times. Hence the complexity is .

After combining the processes of constructing a hash map and searching for the answer, we have got an expected O(n) algorithm.

# Question 3

## Question Part

You are an assistant news broadcaster reporting on a cycling race containing n cyclists. You have been given the task of computing the excitement factor of a given race, which is calculated as follows:

• The excitement factor starts at zero.

• Any time one of the first n 2 cyclists is overtaken by any other cyclist, the excitement factor increases by 1.

Unfortunately, you have only been given the starting and finishing order of the cyclists. From this data, you need to calculate the minimum possible excitement factor for the whole race. You may assume than n is a power of 2 and is strictly greater than 1.

## Solution

We use greedy algorithm to solve this problem and the process is to simulate a least excitement case and calculate the total excitement.

For the purpose of clarity, we use to represent the starting order and to represent the finishing order.

We assume that the least excitement case is that the cyclist in the finishing order would go straight from their starting place to the final position and won’t be surpassed by anyone during the race as fluctuation of rank results in increasement in excitement of the whole game.

In the algorithm, we use iteration to count each cyclist from the first player to the last in the final order array. When considering player , we promote him (or her) from the starting place by manipulating the starting order array . We assume that the first to players have been promoted to corresponding position, so the process of promotion is to shift moving back every player in the range of to the by step, and then place the player into . While simulating the order change, we calculate the excitement using the rule provided.

To illustrate with an example:

Starting Order:

**1 2 3 4 5 6**

Finishing Order:

**2 3 6 4 5 1**

1st Iteration: 1

**2 1 3 4 5 6**

2nd Iteration: 2

**2 3 1 4 5 6**  
3rd Iteration: 3

**2 3 6 1 4 5**

4th Iteration: 3

**2 3 6 4 1 5**

5th Iteration: 3

**2 3 6 4 5 1**

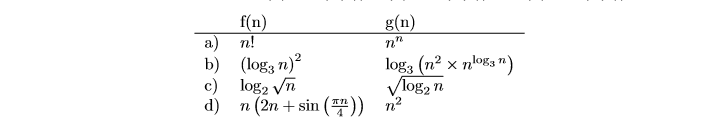
6th Iteration: 3

**2 3 6 4 5 1**

With excitement of **3**.

# Question 4

Read the review material from the class website on asymptotic notation and basic properties of logarithms, and then determine if



.

We using induction to prove:

.

.

.

.

.



.

.

.

.

.

.

# Question 5

## Question Part

Determine the asymptotic growth rate of the solutions to the following recurrences. If possible, you can use the Master Theorem, if not, find another way of solving it.

## Solution

1. We have a = 3 and b = 2, then . In addition, . Thus, the first case of Master Theorem applies so we have
2. Note that a = 3 and b = 4, then . On the other hand, . In order to satisfy the third condition of Master Theorem, we need to find c for . We have . For Thus We have
3. There is a = 5 and b = 2, then . On the other hand, there is . According to second case of Master Theorem, we have .
4. By expanding the recurrence we get: