COMP 9101

Assignment 1

Student ID : z5174741

Student name : GUANQUN ZHOU

Q1 For this question, I think the algorithm consist of 4 steps:

Step1:

We use Merge sort to sort the array and Merge sort is an algorithm. Merge sort is also very stable.

Step2:

For each query, we can use binary search to find the largest index of the element that is less than . And the index is called in every binary search.

Note that in the process of this binary search, if the element that matches is equal to . If so, then we have to continue to run the binary search to search in the group of smaller indexes until we find the largest index of the element that is less than .

Step3:

For each query, we can use binary search to find the largest index of the element that is not greater than . And the index is called in every binary search.

Note that in the process of this binary search, if the element that matches is equal to . If so, then we have to continue to run the binary search to search in the group of larger indexes until we find the largest index of the element that is not greater than .

Step4:

We can use these two index to calculate the number of elements a satisfy

The time complexity for each query is because binary search and there. are n queries. So the time complexity of the whole algorithm is .

Q2:

(a) Firstly, we use Merge sort to sort the array and Merge sort is an algorithm. Merge sort is also very stable.

We assume the sorted array is called T.

For each index from 1 to n, we will use binary search to check if there exist ()

in the array where the range of index is from 1 to (). In this way, we will consider every pair only once. And in case but only appears once in the array, the range is set to be T[1,2..] in every binary search so we can get accurate results.

Thus, this is an algorithm.

(b) I think using hash table is a good option.

Firstly it is an algorithm to establish hash table because we need to iterate the array. of n integers and every integer is used at least once.

Secondly for each index from 1 to n, check if () exists in the hash table. As. searching in hash table requires , for n integers, the time complexity is .

Thus, this is an algorithm.

Q3:  
(a) Assume there are n people present in the party and there is only one celebrity in the party.

These people are numbered from 1 to n

Step1: assume person 1 is the celebrity, we can call this person A.

Step2: for the each person k from 2 to n, we just ask person k if he knows A.

Step3: if person k knows A, then k is not the celebrity and A can continue to be a potential celebrity.

Step4: if person k does not know A, then A is not the celebrity and we let this person k be A, which means this person is the potential celebrity . Then we keep asking the question.

In this way, we ask (n-1) questions and we find a potential celebrity A.

Step5: In order to check if A is the real celebrity. We need ask the other (n-1) people if they know A.

In this way, we ask another (n-1)questions to check if everyone knows A. If everyone knows A, then A can be the celebrity with high possibility. If there exists someone who does not know A, then we can say that there is no celebrity in this party.

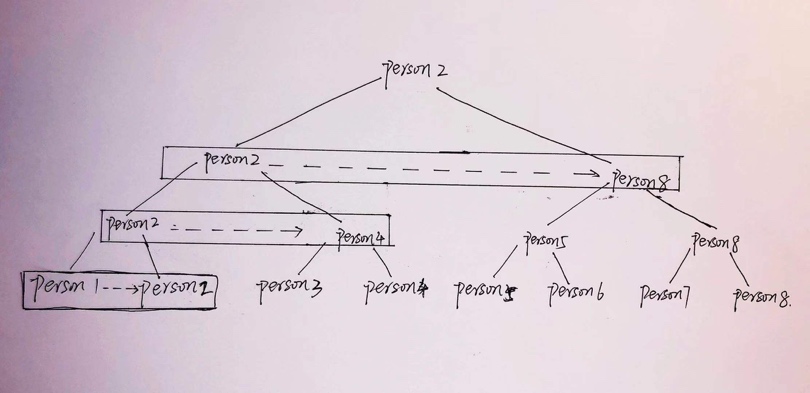
Step6: If we find that everyone knows A, then we need to check if A does not know anyone. So we need to ask A if he knows anyone of the other(n-1)people. We need to ask another (n-1) questions. If the answer is always no, which means A does not know anyone of the other people, then we can say A is the celebrity. If there exists someone is knows by A, then we can that there is no celebrity in this party.

According to 6 steps above, we need at most such questions to find the celebrity, even in the worst case.

(b) We can consider first 4 steps above as a binary tree. we have every single person as a leaf. of the binary tree. For each pair, we can ask the left child a question to find if the left child knows the right child. In this way, we can eliminate haft of the number of persons at each layer and finally we will obtain the potential celebrity A.

The height of tree is , so for step5 and step6 above, we can save questions.

In this way ,the total number of questions will be no more than , even in the worst case.



Assume there are 8 people in the party and people are numbered from 1 to 8

We can see from the graph above, the total number of questions to find the potential celebrity B is (n-1) .For check phase we can save questions which is the height of the tree.

Q4:  
(a)

First we can get

Secondly we can find that when :

So we can say that

We can find when :

So we can say that

There exist positive constant such that

Thus,

(b)

Since both and are monotonically increasing function, so and can also represent the relationship between and

From the above equations, since grows faster than . so will grows faster than .

Thus, .

(c)

Since both and are monotonically increasing function, so and can also represent the relationship between and

According to L’Hˆopital Rule:

So we can say will grow faster than

Thus, .

(d)

According to L’Hˆopital Rule:

So grow faster than

Thus, .

(e)

The range of is [-1,1] thus the range of is [0,1], which means the range of is [1,n]. So is not monotonical function. is a monotonical function. So there is no any positive constant that can realize

Moreover there is no any positive constant that can realize

In this case, I think is not asymptotically comparable with , all three relations( don’t satisfy.

Q5:

According to the Master Theorem, .

Let . The range of is [n,3n], so for any positive constant ,

and for any positive constant , . So we can say that .

According to the second case in the Master Theorem,

According to the Master Theorem, .

Let . In this case , as grows faster than , for , dominates. So there exist a constant , .

According to the first case in the Master Theorem,

According to the Master Theorem,.

Let . Since is monotonically increasing function , will eventually exceed 3. So there exist a constant , f.

For the second requirement, . Since are monotonically increasing function, and can also represent the relationship between .

We can get

For the requirement:

Since is a constant, is a constant. As is monotonically increasing function, with sufficient large , will eventually be greater or equal to .

According to third case in the Master Theorem,

1. In this question, the Master Theorem cannot apply here, so we need to try unwinding the recurrence and sum up the linear overheads.

As , we can get that .

Thus,