Time complexity, Searching, Sorting

1. Solve the following questions about time complexity.
   a. Calculate the time complexity of the following code snippet:

```
int p = 0;
for (i = n / 2; i > 1; i /= 6) {
      for (j = 2; j <= n; j *= 4) {
            for (k = 0; k <= j; k *= 3) {
                  p = p + n / 2;
            }
      }
}
```

   b. Calculate the time complexity of the following code snippet:

```
int p = 0;
for (i = n; i > 1; i -= 1) {
      for (j = 2; j <= n; j += 1) {
            p = p + n;
            if (p > (n ^ i)) {
                  break;
            }
      }
}
for (i = n / 2; i > 1; i /= 6) {
      for (j = 2; j <= i; j *= 4) {
            for (k = 0; k <= j; k *= 3) {
                  p = p + n / 2;
            }
      }
}
```

   c. Explain why the statement, "The running time of algorithm A is at least $O(n^2)$" is meaningless.

2. You have recently joined Facebark as a software engineer. It's a new social media platform where people share pictures of their dogs. Facebark has a lot of functionalities that you have been hired to enhance. For example, users can search another user's profile by name. There is a newsfeed where posts by Facebark friends appear. Each user has a numerical ID and a visible username. The ID works like this: the ID of the first person to register on Facebark is 0, the second person's is 1, the third person's is 2, and so on.

id	0	1	2	3	4	5
username	Woof	Pupper	Max	Tuck	Rocky	Daisy

   a. Suppose, the total number of users in Facebark is n. The previous engineer implemented the searching feature using an algorithm of O(n) time complexity. Write the pseudo-code of that algorithm.

You are trying to figure out the order of the newsfeed posts. The previous engineer's implementation only showed the most recent posts first. However, there have been numerous complaints about users who spam the platform by posting a thousand pictures simultaneously. Therefore, you came up with an idea: the posts by the user with the highest post count will be displayed last. In other words, the posts by users will be shown in ascending order of their post counts. So, you printed the data of the counts of the newest posts:

id	0	1	2	3	4	5	6	7	8	9
post	8	12	3	5	2	17	9	14	2	3

count										

After looking at the data, you thought, "Oh, easy! I can write a linear time algorithm for this!"

b. Describe the algorithm by showing the steps of how you would sort the above list in linear time.

Two days after this implementation, the entire server crashed after running out of memory. You found out that this happened after a few users uploaded 1000 pictures each. But people uploaded thousands of pictures regularly before and this never happened.

c. Do you think the crash was a result of your sorting algorithm? If the answer is yes, provide a possible explanation about how this could have happened.

You have been instructed by your supervisor to change your sorting algorithm. While still aiming for optimal time complexity, you are now more cautious and choose an algorithm that has a time complexity of nlogn in the best, worst, and average cases.

d. Describe this new algorithm by showing the steps of sorting the above list using it.

You are thinking about adding a new feature, where a user can search if someone has posted n pictures recently. For example, Max can search if anyone has posted 17 pictures recently, and Daisy's name will pop up (user id:5, post count:17). Since the post count list will already be sorted, you'll be able to implement this efficiently.

e. Describe this searching algorithm and show the steps on the post count list by searching for post count: 9.

3. After working with the binary search algorithm, as a CSE221 student you want to explore its strength. You have several ideas in your mind. Try to modify the algorithm to implement your ideas.

a. Instead of just returning TRUE or FALSE, you want to return the index of the search key if it exists in the list. Return FALSE or –ve index in case it is not found.

b. What if you want to return the first index of the search key in case there are duplicates. For example: your search key is 13 and there are three 13s in the list.

c. Now, along with the first index, you also want to return the number of times the key appears in the list. Say, the search key is 54 and it occurs 4 times in indices 6,7,8,9. Then you should return (6,4).

d. *[same as question 5]* You have a list of numbers that follows a wave-like pattern. First the numbers follow a decreasing pattern, then after a point, they start increasing again. So there is a minimum element in the list. Can you find that by modifying binary search?

4.

Consider the following list:

Index	0	1	2	3	4	5	6	7
Number	23	2	19	3	7	11	5	13

Will the algorithm given on the right be able to find the search value T=2 for the list above?
If yes, **show** the value of L, R, m in each step of the algorithm. If no, **explain** why not.

```
function binary_search(A, n, T) is
    L := 0
    R := n - 1
    while L ≤ R do
        m := floor((L + R) / 2)
        if A[m] < T then
            L := m + 1
        else if A[m] > T then
            R := m - 1
        else:
            return m
    return unsuccessful
```

Here, ***A*** denotes the list, and ***n*** denotes its size.

5. You are given an array containing N distinct integers in a wave-like sequence. Meaning, the numbers in the beginning are in ascending order, and after a specific position, they are in descending order. For example: [1, 3, 4, 5, 9, 6, 2, -1]

You have to find the maximum number of this sequence. Can you devise an efficient algorithm such that the time complexity will be less than O(N)?

   i. Present your solution idea as a pseudocode/ python code/ flowchart/ step-by-step instructions/ logical explanation in one-two paragraphs.

b. Write the time complexity of your algorithm.

6. For different tasks, we often need to search for things. And most of the time, we sort the dataset before performing search operations. Once, you found the following list of numbers.

[2, 5, 1.2, 6.7, 1.7, 9.3, 2.2, 7.7, 0, –4, –5.1, 2, 5, 5.2]

a. To search an element in an unsorted array, we need O(n) time using linear search. Binary search works in O(lgn) time but the array needs to be sorted beforehand which takes at least O(nlgn) time in general. Why would you then sort the array first and then perform binary search instead of just performing linear search?

b. Can you modify count sort so that it may work with negative integers as well?

c. Can you modify count sort so that it may work with the given list?

d. Say, you are working in a system where you need to sort a very big dataset. The memory available to you can barely accommodate the data. You have two options – merge sort & quick sort. Which one should you choose? Why?

e. Construct an array where quick sort fails to work in O(nlgn) time.

f. Perform simulations of the following algorithms with the given list.

   i. Bubble sort
   ii. Selection sort
   iii. Insertion sort
   iv. Merge sort
   v. Quick sort