Register for Certification exam (https://examform.nptel.ac.in/2023_01/exam_form/dashboard)

Course outline

How does an NPTEL online course work? ()

Week 0 ()

Week 1 ()

Week 2 ()

- Trouble Sort (unit?unit=25&lesson=26)
- The Meeting Place Cannot Be Changed (unit?unit=25&lesson=27)
- Magic Ship (unit?

Week 2: Assignment 2

Your last recorded submission was on 2023-02-07, 15:58 IST Due date: 2023-02-08, 23:59 IST.
Question 1 & 2

Consider the following sorting algorithm:

```
InsertionSort(A,n) // Sort array A of size n
for (pos = 0; pos < n; pos++)
    nextpos = pos
    while (nextpos > 0 && A[nextpos] < A[nextpos-1])
        swap(A[nextpos],A[nextpos-1])
        nextpos = nextpos-1
return A
```

1) What will be the time complexity of the given **InsertionSort** algorithm if the input array consists of **n** identical elements? ***1 point***

- $O(log\ n)$
- $O(n)$
- $O(n\ log\ n)$
- $O(n^2)$

2) For input array `A = [1, 6, 5, 4, 3, 2, 7]` , number of swapping `(swap(A[nextpos],A[nextpos-1]))` will be performed by the given **InsertionSort** algorithm?

10

1 point

Question 3 & 4

Consider the following function that takes a list `L` of integers as input and returns a list.
In the code given below, the function `rev(L, i, m)` takes a list `L` , indices `i` and `m` and reverses the segment `L[i],L[i+1]...L[m]` of `L` and returns the updated list. For instance, if `L = [0, 1, 2, 3, 4, 5, 6, 7, 8]` then, `rev(L, 4, 7)` will return L = `[0, 1, 2, 3, 7, 6, 5, 4, 8]` .

```
def mystery(L):
    m = 0
    n = len(L)
    for i in range(0, n):
        m = i
        for j in range(i, n):
            if (L[j] > L[m]):
                m = j
        L = rev(L, i, m)
    return L
```

3) What will `mystery(L)` return? ***1 point***

- Reversed list `L`
- List `L` in the same order as the input
- **List `L` sorted in descending order**
- List `L` sorted in ascending order

4) If we generalize this to inputs of size `n` , then the best upper bound for the running time of the procedure is__ . ***1 point***

- $O(n)$
- $O(n \log n)$
- **$O(n^2)$**
- $O(n^3)$

5) Consider a list `L` of `n` sorted numbers that are circularly shifted `k` positions to the right. ***1 point***

For example, `[-1,0,3,4,9,12]` is a sorted list.

`[9,12,-1,0,3,4]` : circularly shifted `2` positions to the right.

`[3,4,9,12,-1,0]` : circularly shifted `4` positions to the right.

What will be the complexity of the **most efficient algorithm** to search for the smallest element in `L` for the two cases listed below?

I. Value of `k` is not known.
II. Value of `k` is known.

- I.$O(n)$ II.$O(1)$
- I.$O(log\ n)$ II.$O(log\ n)$
- I.$O(n)$ II.$O(log\ n)$
- (selected) I.$O(log\ n)$ II.$O(1)$

6) Let `M` be an `n × m` integer matrix in which the entries of each row are sorted in increasing order (from left to right) and the entries in each column are also sorted in increasing order (from top to bottom) and the first element of each row is greater than the last element of the previous row. What will the asymptotic complexity of an efficient algorithm (based on the binary search) be to find if any integer **x** is present in the matrix `M` or not? ***1 point***

- $O(m+n)$
- $O(log\ m * \ log\ n)$
- (selected) $O(log\ m + \ log\ n)$
- $O(n\ log\ m + \ m\ log\ n)$

7) Consider a list `L` of `n` integers with many duplicates, such that the number of distinct integers in `L` is $log\ n$. We use the below algorithm to sort this list. What will be the worst-case asymptotic complexity of this algorithm? ***1 point***

Algorithm

1. Iterate over all elements of `L` and create an array `A1` of all distinct elements in `L` .
2. Sort the array `A1` created in step 1. (If the size of `A1` is `s` , then this step takes $O(s\ log\ s)$)
3. Create another array `A2` containing all zeros equal in size to `A1` .
4. For each element `e` in `L` .
    a. Using binary search find position `p` of `e` in `A1` .
    b. Increment the value at position `p` in `A2` .
5. Initialize a new list `Ls` .
6. (Create a sorted list using `A1` and `A2`) For each index `i` of array `A1` .
    a. Append `A2[i]` times value at `A1[i]` to list `Ls`

- $O(n\ log\ n)$

○ $O(n\ log^2\ n)$

○ $O(n + log\ n)$

◉ $O(n\ log\ log\ n)$

Question 8 & 9

Consider an array A with n distinct integers where `n>2`. In an array, for some index `p` between `0 < p < n-1`, the values in the array elements increase up to index `p` from index `0` and then decrease the remainder of the way until position `n - 1`.

Consider the following algorithm **find_max** to return the index of the maximum element of array `A` :

```
function find_max(A):
    low = 0
    high = length(A)-1
    while True:
        mid = (low + high)//2
        if (Condition1):
            low = mid
        elif (Condition2):
            high = mid
        else:
            return mid
```

8) What should be the `Condition1` and `Condition2` so that the algorithm correctly returns the required output? ***1 point***

○
```
Condition1 -> A[mid-1] < A[mid]
Condition2 -> A[mid-1] > A[mid]
```

○
```
Condition1 -> A[mid-1] < A[mid]
Condition2 -> A[mid] < A[mid+1]
```

○
```
Condition1 -> (A[mid-1] > A[mid]) and (A[mid] > A[mid+1])
Condition2 -> (A[mid-1] < A[mid]) and (A[mid] < A[mid+1])
```

◉
```
Condition1 -> (A[mid-1] < A[mid]) and (A[mid] < A[mid+1])
Condition2 -> (A[mid-1] > A[mid]) and (A[mid] > A[mid+1])
```

9) What is the time complexity of the above mentioned **find_max** algorithm? ***1 point***

- ○ $O(n)$
- ◉ $O(log\ n)$
- ○ $O(log\ log\ n)$
- ○ $O(n\ log\ n)$

Question 10 & 11

You have a deck of shuffled cards with positive integer values. There are 2 sub-ordinate below you and two sub-ordinate below them, and it goes on.

- The job of the sub-ordinate is to split the deck of cards that they received and give it to two sub-ordinate of them. If they receive a deck of cards from their subordinates, they merge it in ascending order and give it their higher level.
- If a subordinate received only two cards, then he/she himself/herself arranges them in ascending order and gives them back to the superior.
- If a subordinate received only one card, then he/she will give back that to the superior.

10) If the deck of card's values are `[44, 69, 98, 103, 103, 150, 151, 194]` then how many people (including you) are required to sort these cards?

15

1 point

11) If we follow the above approach in an algorithm to sort an array with **n** elements, what would be the worst-case time complexity for that algorithm? Consider that merging of two sorted arrays (each of size n) takes $O(n)$ time. ***1 point***

- ○ $O(n)$
- ○ $O(n^2)$
- ◉ $O(n\ log\ n)$
- ○ $O(n^2\ log\ n)$

You may submit any number of times before the due date. The final submission will be considered for grading.

Submit Answers