ALGORITHMS MIDTERM EXAM
FALL 2020-2021

Question 1.(Searching)

A medical doctor gives its daily admitted patients distinct appointment numbers starting from *1*. That is, the k^{th} coming patient gets an id k-*1*. However, some patients do not come to their appointments. Assume that m patients get ids and $n<m$ of them come to their appointment everyday. These patients are stored in an n-dimensional array A sorted by their ids.

Give an $O(Logn)$ time algorithm in pseudocode convention to find the first patient that did not come to his/her appointment. For example, when $A=[1, 2, 3, 5, 7, 8, 10]$, the algorithm must return *4*.

Question 2.(Insertion sort complexity)

A bank keeps record of its customers in an array A[1...n] sorted by account numbers. However, a bank accountant accidentally shifts the array elements k positions to the right in a circular fashion. For example, if *A=[9, 12, 17, 21, 33, 41]*, it becomes *[33, 41, 9, 12, 17, 21]* when shifted *k=2* positions, and becomes *[17, 21, 33, 41, 9, 12]* when shifted *k=4* positions. The accountant realizes the problem and wants to fix it by sorting the array again using insertion sort.

Find the complexity of insertion sort for such an input scenario?

Question 3.(Asymptotic notation)

Suppose that we derive following recursive formulation for the time complexity of an algorithm

$$T(2)=1$$
$$T(n)=T(n^{1/4}) + 1,\ n > 1$$

Solve this recurrence asymptotically. Show your work.

Question 4.(Heap-sort)

We are given an unsorted array *A[1…n]*. Now, imagine its sorted version. The unsorted array has the property that each element has a distance of at most *k* positions, where $0<k<=n$, from its index in the sorted version. For example, when *k* is *2*, an element at index *5* in the sorted array, can be at one of the indices *{3,4,5,6,7}* in the unsorted array. The unsorted array can be sorted efficiently by utilizing a Min-Heap data structure. The outline of the algorithm is given below

- Create a Min Heap of size *k+1* with first *k+1* elements,
- One by one remove min element from the heap, put it in the result array, and add a new element to the heap from remaining elements.

a. Write down the complete algorithm in pseudocode convention to sort the array *A*.

b. Provide a tight asymptotic upper bound time complexity for this algorithm. Show your work.

Question 5.(Dynamic programming)

The dynamic programming algorithm MATRIX-CHAIN_ORDER is illustrated below

```
MATRIX-CHAIN-ORDER(p)
    n ← length[p] − 1
    for i ← 1 to n do
        m[i, i] ← 0
    for ℓ ← 2 to n do
        for i ← 1 to n − ℓ + 1 do
            j ← i + ℓ − 1
            m[i, j] ← ∞
            for k ← i to j−1 do
                q ← m[i, k] + m[k+1, j] + p_{i-1} p_k p_j
                if q < m[i, j] then
                    m[i, j] ← q
                    s[i, j] ← k
    return m and s
```

Given *p = [10, 40, 20, 5, 5]*,

a. Draw both tables m & s and fill in every table entry as returned by MATRIX-CHAIN-ORDER(*p*).

b. Demonstrate how you have calculated the table entry m[2,4] in detail.

Question 6.(Quicksort)

The Quicksort algorithm using Hoare's partitioning is given below

```
H-PARTITION (A, p, r)
    pivot ← A[p]
    i ← p − 1
    j ← r + 1
    while true do
        repeat j ← j − 1 until A[j] ≤ pivot
        repeat i ← i + 1 until A[i] ≥ pivot
        if i < j then exchange A[i] ↔ A[j]
        else return j
```

Draw the recursion tree that represents the recursive calls during the execution of the Quicksort (QS) algorithm for the input array *A=[23,93,4,67,52,9,44,67,1]*.