COMP3711: Design and Analysis of Algorithms

Tutorial 2

HKUST

Question 1

Give asymptotic upper bounds for $T(n)$ by recursion tree approach. Make your bounds as tight as possible.

(a)

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(n/2) + n \qquad \text{if } n > 1 \end{aligned}$$

(b)

$$\begin{aligned} T(1) &= T(2) = 1 \\ T(n) &= T(n-2) + 1 \qquad \text{if } n > 2 \end{aligned}$$

(c)

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(n/3) + n \qquad \text{if } n > 1 \end{aligned}$$

Question 1

(d)

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 4T(n/2) + n \qquad \text{if } n > 1 \end{aligned}$$

(e)

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 3T(n/2) + n^2 \qquad \text{if } n > 1 \end{aligned}$$

(f)

$$\begin{aligned} T(1) &= 0, \quad T(2) = 1 \\ T(n) &= T(n/2) + \log_2 n \qquad \text{if } n > 2 \end{aligned}$$

Question 2

Given a sorted array $A[1..n]$ of n distinct integers (positive or negative), give an algorithm to find the index i such that $A[i] = i$, if such an index exists. If there are many such indices, the algorithm may return any one of them. Solve this problem in $O(\log n)$ time.

Question 3

Let $A[1..n]$ be an array of n elements. A *majority element* of A is any element occuring more than $n/2$ times (e.g., if $n = 8$, then a majority element should occur at least 5 times). Your task is to design an algorithm that finds a majority element, or reports that no such element exists.

(a) Suppose that you are not allowed to order the elements, the only way you can access the elements is to check whether two elements are equal or not. Design an $O(n \log n)$-time algorithm for this problem.

(b) Design an $O(n)$ algorithm for this problem. Similar to (a), you are still only allowed to use equality tests on the elements.