COL351: Analysis and Design of Algorithms **Fall 2024**

Tutorial Sheet 2

Announced on: Aug 01 (Thurs)

Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. Consider the following algorithm for checking whether a given number n is prime.

ALGORITHM 1: Checking primality

Input: An integer $n > 1$.
Output: Yes/No

```
if n equals 2 or 3 then
    return Yes
for i = 2 to ⌊√n⌋ do
    if i divides n then
        return No
return Yes
```

Prove or disprove:

a) The algorithm is correct.

b) The algorithm runs in polynomial time.

2. Consider the following algorithm for calculating a^b where a and b are positive integers.

ALGORITHM 2: `FastPower`(a, b)

Input: Positive integers a and b.
Output: a^b.

```
if b = 1 then
    return a
c := a · a
d := FastPower(c, ⌊b/2⌋)
if b is odd then
    return a · d
return d
```

Suppose each multiplication and division operation can be performed in constant time. Determine the asymptotic running time of `FastPower` as a function of b.

3. Let A and B be two sorted arrays of length n each. Assume that all elements within and across the two arrays are distinct. Design an $\mathcal{O}(\log n)$ algorithm to compute the n^{th} smallest

element of the union of A and B.

4. Design an $\mathcal{O}(\log^2 n)$ algorithm that, given a positive integer n, determines whether n is of the form a^b for some positive integers a and $b > 1$. For the purpose of this problem, you may assume exponentiation to be $O(1)$ time, i.e., computing p^q for two integers p and q takes constant time. Similarly, you can assume that comparison of two integers (i.e., determining whether p equals q, $p > q$ or $p < q$) takes constant time.

5. You are given a sorted (from smallest to largest) array A of n distinct integers which can be positive, negative, or zero. You want to decide whether or not there is an index i such that $A[i] = i$. Design the fastest algorithm you can for solving this problem.

6. ($\star$) You are given an n-by-n grid of distinct numbers. A number is a *local minimum* if it is smaller than all its neighbors. A *neighbor* of a number is one immediately above, below, to the left, or to the right. Most numbers have four neighbors; numbers on the side have three; the four corners have two.
   (a) Prove that a local minimum always exists.
   (b) Use the divide-and-conquer algorithm design paradigm to compute a local minimum with only $\mathcal{O}(n)$ comparisons between pairs of numbers. (Note: since there are n^2 numbers in the input, you cannot afford to look at all of them.)

7. ($\star$) You are given a sequence of n numbers $a_1, a_2, \ldots, a_n$. Design an $\mathcal{O}(n)$ algorithm to find i, j with $i \leqslant j$ such that the sum $a_i + a_{i+1} + \cdots + a_j$ is maximum. Note that the numbers may not be positive.