Modifiers

To help you with what problems to try, problems marked with **[K]** are key questions that tests you on the core concepts, please do them first. Problems marked with **[H]** are harder problems that we recommend you to try after you complete all other questions (or perhaps you prefer a challenge). Good luck!!!

Contents

§1 Recurrences

Exercise 1.1. [K] Determine the asymptotic growth rate of the solutions to the following recurrences. You may use the Master Theorem if it is applicable.

(a) $T(n) = 2T(n/2) + n(2 + \sin n)$.

(b) $T(n) = 2T(n/2) + \sqrt{n} + \log n$.

(c) $T(n) = 8T(n/2) + n^{\log_2 n}$.

(d) $T(n) = T(n-1) + n$.

Exercise 1.2. [K] Explain why we cannot apply the Master Theorem to the following recurrences.

(a) $T(n) = 2^n T(n/2) + n^n$.

(b) $T(n) = T(n/2) - n^2 \log n$.

(c) $T(n) = \frac{1}{3}T(n/3) + n$.

(d) $T(n) = 3T(3n) + n$.

Exercise 1.3. [H] Consider the following naive Fibonacci algorithm.

Algorithm 1.1 $F(n)$: The naive Fibonacci algorithm

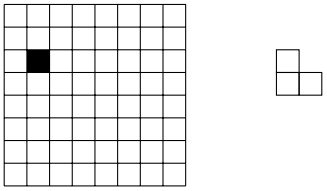
```
Require: n ≥ 1
  if n = 1 or n = 2 then
      return 1
  else
      return F(n − 1) + F(n − 2)
  end if
```

When analysing its time complexity, this yields us with the recurrence $T(n) = T(n-1) + T(n-2)$. Show that this yields us with a running time of $\Theta(\varphi^n)$, where $\varphi = \frac{1+\sqrt{5}}{2}$ which is the golden ratio. How do you propose that we improve upon this running time?

> **Hint —** *This is a standard recurrence relation. Guess $T(n) = a^n$ and solve for a.*

§2 Divide and Conquer

Exercise 2.1. [K] You are given a $2^n \times 2^n$ board with one of its cells missing (i.e., the board has a hole). The position of the missing cell can be arbitrary. You are also given a supply of "trominoes", each of which can cover three cells as below.



Design an algorithm that covers the entire board (except for the hole) with these "trominoes". Note that the trominoes can be rotated and your solution should not try to brute force all possible placement of those "trominoes".

Exercise 2.2. [K] Given positive integers M and n, compute M^n using only $O(\log n)$ many multiplications.

Exercise 2.3. [H] Suppose you are given an array A containing $2n$ numbers. The only operation that you can perform is make a query if element $A[i]$ is equal to element $A[j]$, $1 \leq i, j \leq 2n$. Your task is to determine if there is a number which appears in A at least n times using an algorithm which runs in linear time.

> **Hint —** *A rather a tricky one !! The reasoning resembles a little bit the reasoning used in the celebrity problem (Tutorial Question 1.5): try comparing them in pairs and first find one or at most two possible candidates and then count how many times they appear.*

Exercise 2.4. [H] You and a friend find yourselves on a TV game show! The game works as follows. There is a **hidden** $N \times N$ table A. Each cell $A[i, j]$ of the table contains a single integer and no two cells contain the same value. At any point in time, you may ask the value of a single cell to be revealed. To win the big prize, you need to find the N cells each containing the **maximum** value in its row. Formally, you need to produce an array $M[1..N]$ so that $A[r, M[r]]$ contains the maximum value in row r of A, i.e., such that $A[r, M[r]]$ is the largest integer among $A[r, 1], A[r, 2], \ldots, A[r, N]$. In addition, to win, you should ask at most $2N\lceil \log N \rceil$ many questions. For example, if the hidden grid looks like this:

	Column 1	Column 2	Column 3	Column 4
Row 1	**10**	5	8	3
Row 2	1	**9**	7	6
Row 3	-3	**4**	-1	0
Row 4	-10	-9	-8	**2**

then the correct output would be $M = [1, 2, 2, 4]$.
Your friend has just had a go, and sadly failed to win the prize because they asked N^2 many questions which is too many. However, they whisper to you a hint: they found out that M is **non-decreasing**. Formally, they tell you that $M[1] \leq M[2] \leq \cdots \leq M[N]$ (this is the case in the example above).
Design an algorithm which asks at most $\mathbf{O(N \log N)}$ many questions that produces the array M correctly, even in the very worst case.

> **Hint —** *Note that you do not have enough time to find out the value of every cell in the grid! Try determining $M[N/2]$ first, and see if divide-and-conquer is of any assistance.*

Exercise 2.5. [H] Define the Fibonacci numbers as

$$F_0 = 0, F_1 = 1, \text{ and } F_n = F_{n-1} + F_{n-2} \text{ for all } n \geq 2.$$

Thus, the Fibonacci sequence is as follows:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, \ldots.$$

(a) Show, by induction or otherwise, that

$$\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

for all integers $n \geq 1$.

(b) Hence or otherwise, give an algorithm that finds F_n in $O(\log n)$ time.

Hint — *You may wish to refer to Example 1.5 on page 28 of the "Review Material" booklet.*

Exercise 2.6. [H] Let A be an array of n integers, not necessarily distinct or positive. Design a $\Theta(n \log n)$ algorithm that returns the maximum sum found in any contiguous subarray of A.

Note — Note that there is an $O(n)$ algorithm that solves this problem; however, the technique used in that solution is much more involved and will be taught in due time.

§3 Karatsuba's trick

Exercise 3.1. [K] Let $P(x) = a_0 + a_1 x$ and $Q(x) = b_0 + b_1 x$, and define $R(x) = P(x) \cdot Q(x)$. Find the coefficients of $R(x)$ using only three products of pairs of expressions each involving the coefficients a_i and/or b_j.
Addition and subtraction of the coefficients do not count towards this total of three products, nor do scalar multiples (i.e. products involving a coefficient and a constant).

Exercise 3.2. [K] Recall that the product of two complex numbers is given by

$$\begin{aligned}(a + \mathrm{i}b)(c + \mathrm{i}d) &= ac + \mathrm{i}ad + \mathrm{i}bc + ß^2 bd \\ &= (ac - bd) + \mathrm{i}(ad + bc)\end{aligned}$$

(a) Multiply two complex numbers $(a + \mathrm{i}b)$ and $(c + \mathrm{i}d)$ (where a, b, c, d are all real numbers) using only 3 real number multiplications.

(b) Find $(a + \mathrm{i}b)^2$ using only two multiplications of real numbers.

(c) Find the product $(a + \mathrm{i}b)^2(c + \mathrm{i}d)^2$ using only five real number multiplications.

Exercise 3.3. [H] Let $P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ and $Q(x) = b_0 + b_1 x + b_2 x^2 + b_3 x^3$, and define $R(x) = P(x) \cdot Q(x)$. You are tasked to find the coefficients of $R(x)$ using only twelve products of pairs of expressions each involving the coefficients a_i and/or b_j.

Challenge — Can you do better?

§4 Selection

Exercise 4.1. [K] Suppose we modify the median of medians QUICKSELECT algorithm to use blocks of 7 rather than 5. Determine the appropriate worst-case recurrence, and solve for the asymptotic growth rate.

Exercise 4.2. [K] Given two *sorted* arrays A and B, each of length n, design a $O(\log n)$ algorithm to find the (lower) median of all $2n$ elements of both arrays. You may assume that n is a power of two and that all $2n$ values are distinct.

Exercise 4.3. [K] Given an array A of n distinct integers and an integer $k < \frac{n}{2}$, design an $O(n)$ algorithm which finds the median as well as the k values closest to the median from above and below. You may assume that n is odd.

Exercise 4.4. [H] Let A be an array of n distinct integers, each with an associated weight w_i. All w_i are non-negative, and $w_1 + \ldots + w_n = 1$. The *weighted (lower) median* is the element $A[k]$ satisfying

$$\sum_{A[i]<A[k]} w_i < \frac{1}{2} \text{ and } \sum_{A[i]>A[k]} w_i \leq \frac{1}{2}.$$

Design a $O(n)$ algorithm to find the weighted (lower) median.