COMP3121/9101
Algorithm Design

Problem Set 2 – Divide and Conquer

[K] – key questions **[H]** – harder questions **[E]** – extended questions **[X]** – beyond the scope of this course

Contents

§ SECTION ONE: RECURRENCES

[K] Exercise 1. 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) $T(n) = T(n-1) + n$.

[K] Exercise 2. 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) $T(n) = 3T(3n) + n$.

[H] Exercise 3. Consider the following naive Fibonacci algorithm.

Algorithm 1 $F(n)$: The naive Fi

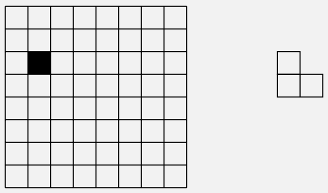
```
Require: $n \geq 1$
  if $n = 1$ or $n = 2$ then
      return 1
  else
      return $F(n-1) + F(n-2)$
  end if
```

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.

§ SECTION TWO: DIVIDE AND CONQUER

[K] Exercise 4. 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".

[K] Exercise 5. Given positive integers M and n, compute M^n using only $O(\log n)$ many multiplications.

[H] Exercise 6. 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 m $[r]]$ is the largest integer among $A[r, 1], A[r, 2], \ldots, A[r, N]$. In ad ny questions. For example, if the hidden grid looks like this:

	Column 1	Column 2	Column 3	Column 4
Row 1	**10**	5		
Row 2	1	**9**		
Row 3	-3	**4**		
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 $\mathbf{O}(\mathbf{N} \log \mathbf{N})$ many questions that produces the array M correctly, even in the very worst case.

[H] Exercise 7. 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.

[H] Exercise 8. 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: there is an $O(n)$ solution that solves the problem; however, the intuition behind that approach is much more involved and will be taught in due time.

[H] Exercise 9. 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: The reasoning resembles a little bit like the celebrity problem from Tutorial 1; compare them in pairs. How many potential candidates can exist that appears in A at least n times?

§ SECTION THREE: KARATSUBA'S TRICK

[K] Exercise 10. Recall that the product of two complex numbers is given by

$$\begin{aligned}(a+ib)(c+id) &= ac + iad + ibc + i^2bd \\ &= (ac-bd) + i(ad+bc)\end{aligned}$$

(a) Multiply two complex numbers $(a+ib)$ and $(c+id)$ (where a, b, c, d are all real numbers) using only 3 real number multiplications.

(b) Find $(a+ib)^2$ using only two multiplications of real numbers.

(c) Find the product $(a+ib)^2(c+id)^2$ using only five real number multiplications.

[K] Exercise 11. Let $P(x) = a_0 + a_1x$ and $Q(x) = b_0 + b_1x$, 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).

[H] Exercise 12. Let $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3$ and $Q(x) = b_0 + b_1x + b_2x^2 + b_3x^3$, 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.

§ SECTION FOUR: CONVOLUTIONS

[K] Exercise 13. Find the sequence x satisfying $x * \langle 1, 1, -1\rangle = \langle 1, 0, -1, 2, -1\rangle$. Is it true that $\langle 1, 1, -1\rangle * x = \langle 1, 0, -1, 2, -1\rangle$? Explain why or why not.

[K] Exercise 14.

(a) Compute the convolution $\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle * \langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle$.

(b) Compute the DFT of the sequence $\langle 1, \underbrace{0, 0, \ldots, 0}_{k}, 1\rangle$.

[K] Exercise 15. Compute directly (i.e. without FFT) and maximally simplify the DFT of the following sequences:

(a) $\langle 1 \underbrace{0, 0, \ldots, 0}_{n-1}\rangle$.

(b) $\langle \underbrace{0, 0, \ldots, 0}_{n-1}, 1\rangle$.

(c) $\langle \underbrace{1, \ldots, 1}_{n}\rangle$.

[K] Exercise 16. You are given a se

$$A = \langle a_0, a_1, \ldots, a_n \;\; 1$$

of length n and sequence

$$B = \langle 1, \underbrace{0, \ldots,}_{k}$$

of length $k + 2$, where $1 \leq k \leq n/4$.

(a) Compute the DFT of sequence B.

(b) Compute the convolution sequence $C = A * B$ in terms of the elements in sequence A.

(c) Show that, in this particular case, such a convolution can be computed in $O(n)$ time.

[K] Exercise 17. You have a set of N coins in a bag, each having a value between 1 and M, where $M \geq N$. Some coins may have the same value. You pick two coins (**without replacement**) and record the sum of their values. Determine what possible sums can be achieved in $O(M \log M)$ time.

Hint: If the coins have values $v_1, \ldots, v_N$, how might you use the polynomial $x^{v_1} + \ldots x^{v_N}$?

[K] Exercise 18. Consider the polynomial

$$P(x) = (x - \omega_{64}^{0})(x - \omega_{64}^{1}) \ldots (x - \omega_{64}^{63}).$$

(a) Compute $P(0)$.

(b) What is the degree of $P(x)$? What is the coefficient of the highest degree term of x present in $P(x)$?

(c) What are the values of $P(x)$ at the roots of unity of order 64?

(d) Can you represent $P(x)$ in the coefficient form without any computation?

[H] Exercise 19. Suppose you have K polynomials $P_1, \dots, P_K$ each of degree at least one, and that

$$\deg(P_1) + \cdots + \deg(P_K) = S.$$

(a) Show that you can find the product of these K polynomials in $O(KS \log S)$ time.

(b) Show that you can find the product of these K polynomials in $O(S \log S \log K)$ time.