

**Five questions**, marked out of a total of  $(5+15)+10+(5+10)+10+(10+10) = \mathbf{75}$  marks.

Some hints and tips:

- Read the assignment early, so you have time to think about the problems!
- Read **all** the problems and make sure you understand them: sometimes problems can be difficult to understand, but easy once understood.
- You may always assume the solution to a previous part when solving a subsequent part. For instance, if part (a) says provide an algorithm that does X in  $O(N)$  time, then in part (b) you may assume that there is some algorithm that does X in  $O(N)$  time, and treat it as a black box, *even if you don't know how to solve part (a)!* This means you can skip part (a) and still obtain full credit on part (b), if your solution for (b) is correct.
- Concise answers written in clear English text are perfectly acceptable, as are answers written in *clear* pseudocode.

1. (a) [**5 marks**] *Revision:* Describe how to multiply two  $n$ -degree polynomials together in  $O(n \log n)$  time, using the Fast Fourier Transform (FFT). You do not need to explain how FFT works — you may treat it as a black box.

*Hint: Remember that FFT does not multiply polynomials, what does it do? Refer to the lecture slides.*

- (b) In this part we will extend the Fast Fourier Transform (FFT) algorithm described in class to multiply multiple polynomials together (not just two). Suppose you have  $K$  polynomials  $P_1, \dots, P_K$  so that

$$\text{degree}(P_1) + \dots + \text{degree}(P_K) = S$$

- (i) [**5 marks**] Show that you can find the product of these  $K$  polynomials in  $O(KS \log S)$  time.

*Hint: How many points do you need to uniquely determine an  $S$ -degree polynomial?*

- (ii) [**10 marks**] Show that you can find the product of these  $K$  polynomials in  $O(S \log S \log K)$  time. Explain why your algorithm has the required time complexity.

*Hint: consider using divide-and-conquer!*

2. [**10 marks**] 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.

For example, if there are  $N = 3$  coins in the bag with values 1, 4 and 5 (so we could have  $M = 5$ ), then the possible sums are 5, 6 and 9.

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

3. Let us define the Fibonacci numbers as  $F_0 = 0$ ,  $F_1 = 1$  and  $F_n = F_{n-1} + F_{n-2}$  for all  $n \geq 2$ . Thus, the Fibonacci sequence looks as follows: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- (a) [5 marks] 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) [10 marks] 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” found as lecture notes for Topic 0 on the Course Resources page of the course website.*

4. [10 marks] You have  $N$  items for sale, numbered from 1 to  $N$ . Alice is willing to pay  $a[i] > 0$  dollars for item  $i$ , and Bob is willing to pay  $b[i] > 0$  dollars for item  $i$ . Alice is willing to buy no more than  $A$  of your items, and Bob is willing to buy no more than  $B$  of your items. Additionally, you must sell each item to either Alice or Bob, but not both, so you may assume that  $N \leq A + B$ . Given  $N, A, B, a[1..N]$  and  $b[1..N]$ , show that you can determine the **maximum** total amount of money you can earn in  $O(N \log N)$  time.

*Hint: Suppose  $N = A + B$  and pretend you wish to sell all items to Alice, but must choose  $B$  of them to give to Bob instead. Which ones do you want to give to Bob first? Then extend your approach to handle  $N < A + B$  as well.*

5. Your army consists of a line of  $N$  giants, each with a certain height. You must designate precisely  $L \leq N$  of them to be leaders. Leaders must be spaced out across the line; specifically, every pair of leaders must have at least  $K \geq 0$  giants standing in between them. Given  $N, L, K$  and the heights  $H[1..N]$  of the giants in the order that they stand in the line as input, find the *maximum* height of the *shortest* leader among all valid choices of  $L$  leaders. We call this the *optimisation* version of the problem.

For instance, suppose  $N = 10, L = 3, K = 2$  and  $H = [1, 10, 4, 2, 3, 7, 12, 8, 7, 2]$ . Then among the 10 giants, you must choose 3 leaders so that each pair of leaders has at least 2 giants standing in between them. The best choice of leaders has heights 10, 7 and 7, with the shortest leader having height 7. This is the best possible for this case.

- (a) [10 marks] In the *decision* version of this problem, we are given an additional integer  $T$  as input. Our task is to decide if there exists some valid choice of leaders satisfying the constraints whose shortest leader has height no less than  $T$ .

Give an algorithm that solves the decision version of this problem in  $O(N)$  time.

- (b) [**10 marks**] Hence, show that you can solve the optimisation version of this problem in  $O(N \log N)$  time.

Assignment Project Exam Help

<https://tutorcs.com>

WeChat: cstutorcs