Due 13th October 2022 at 5pm Sydney time

Your solutions must be typed, machine readable PDF files. *All submissions will be checked for plagiarism!*

For each question requiring you to design an algorithm, you *must* justify the correctness of your algorithm. If a time bound is specified in the question, you also *must* argue that your algorithm meets this time bound.

Partial credit will be awarded for progress towards a solution.

Please note that for greedy algorithms we expect a higher standard of proof and justification that your algorithm is correct. More explicitly, this means that a higher percentage of the total marks will be allocated to your proofs and explanations compared to other assignments, and about 10% of the total marks will be specifically allocated towards the clarity and conciseness of your explanations.

Question 1

[30 marks] Antoni is playing a game with some blocks. He starts with a line of n stacks of blocks with heights $h_1, \ldots, h_n$, and in a single move is allowed to split any stack into two (thus increasing the number of stacks by 1). Stacks can only be split *in-place*. That is, when splitting stack h_i, the two resulting stacks become h_i and h_{i+1}. Antoni wins the game by finishing with a sequence of stacks that is in non-decreasing order.

1.1 **[7 marks]** Show that it is always possible to win the game.

1.2 **[23 marks]** Write an algorithm that finds the minimum number of moves required to win. Your algorithm should run in $O(n)$ time.

You are only required to **count** the minimum number of moves. You do not need to actually perform them.

Question 2

[30 marks] UNSW is competing in a programming varsity competition, and $2k$ students have registered. This competition requires students to compete in pairs, and so UNSW has run a mock competition to help determine the pairings. The results for each of the $2k$ students have been stored in an array L, and each result is a non negative integer. The score of each pair is the product of the marks of the two students, and the final score of each university is the sum of all pairs' scores.

For example, if the marks of the students are $L = [1, 4, 5, 3]$, then the pairs $(1, 3)$ and $(4, 5)$ give the largest total score of $1 \cdot 3 + 4 \cdot 5 = 3 + 20 = 23$.

2.1 **[10 marks]** Suppose the marks of 4 students are a, b, c, d where $a \leq b \leq c \leq d$. Show that $ab + cd \geq ac + bd \geq ad + bc$. You can assume that all marks are non-negative integers.

This subquestion should be solved independently of the overall question's context.

2.2 **[12 marks]** Design a $O(k \log k)$ algorithm which determines the maximum score that UNSW can achieve to ensure it has the best chance of winning.

2.3 **[8 marks]** USYD is also competing in this competition and they have also run a mock competition. UNSW gets to choose USYD's team pairings. Design a $O(k \log k)$ algorithm which determines the minimum score that USYD can achieve.

Question 3

[20 marks] Ryno needs your help! He has two directed acyclic graphs X and Y. Each of these graphs have n vertices, labelled $1, \ldots, n$. He wants to know whether there is an ordering of $[1, \ldots, n]$ which is a topological ordering of ***both*** graphs, and if so, what that ordering is.

A trivial example where this ordering does *not* exist is when $n = 2$, and X has an edge between $(1, 2)$ and Y has an edge between $(2, 1)$.

3.1 **[5 marks]** Provide another *non-trivial* example where this ordering does *not* exist. Non-trivial in this case means that an edge cannot appear in one direction in X, and in the opposite direction in Y, as per the example earlier.

3.2 **[15 marks]** Design an $O(n^2)$ algorithm which solves Ryno's problem.

Question 4

[20 marks] Alice and Bob are meeting up at the park. Alice arrives at time $t \in [1, ..., n]$ with probability a_t (where $\sum_{t=1}^{n} a_t = 1$). Bob arrives at time $t \in [1, ..., n]$ with probability b_t (where $\sum_{t=1}^{n} b_t = 1$). Assume the time t is in minutes.

4.1 **[4 marks]** Provide an expression for the probability that Alice arrives k minutes before Bob (where k can be negative).

4.2 **[10 marks]** Design an $O(n \log n)$ algorithm that computes the probability that Alice arrives before Bob.

> How might FFT fit into this problem?

4.3 **[6 marks]** Design an $O(n)$ algorithm to compute the probability that Alice and Bob arrive an even number of minutes apart.

> Note: this is tricky!