Due Friday 3rd March at 5pm Sydney time

In this assignment we review some basic algorithms and data structures, and we apply the divide-and-conquer paradigm. There are *three problems* each worth 20 marks, for a total of 60 marks. Partial credit will be awarded for progress towards a solution. We'll award one mark for a response of "one sympathy mark please" for a whole question, but not for parts of a question.

Any requests for clarification of the assignment questions should be submitted using the Ed forum. We will maintain a FAQ thread for this assignment.

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. The required time bound always applies to the *worst case* unless otherwise specified.

You must submit your response to each question as a separate PDF document on Moodle. You can submit as many times as you like. Only the last submission will be marked.

Your solutions must be typed, *not* handwritten. We recommend that you use LaTeX, since:

- as a UNSW student, you have a free Professional account on Overleaf, and
- we will release a LaTeX template for each assignment question.

Other typesetting systems that support mathematical notation (such as Microsoft Word) are also acceptable.

Your assignment submissions must be your own work.

- You may make reference to published course material (e.g. lecture slides, tutorial solutions) without providing a formal citation. The same applies to material from COMP2521/9024.
- You may make reference to either of the recommended textbooks with a citation in any format.
- You may reproduce general material from external sources in your own words, along with a citation in any format. 'General' here excludes material directly concerning the assignment question. For example, you can use material which gives more detail on certain properties of a data structure, but you cannot use material which directly answers the particular question asked in the assignment.
- You may discuss the assignment problems privately with other students. If you do so, you must acknowledge the other students by name and zID in a citation.
- However, you must write your submissions entirely by yourself.
  - Do not share your written work with anyone except COMP3121/9101 staff, and do not store it in a publicly accessible repository.
  - The only exception here is UNSW Smarthinking, which is the university's official writing support service.

Please review the UNSW policy on plagiarism. Academic misconduct carries severe penalties.

Please read the Frequently Asked Questions document, which contains extensive information about these assignments, including:

- how to get help with assignment problems, and what level of help course staff can give you
- extensions, Special Consideration and late submissions
- an overview of our marking procedures and marking guidelines
- how to appeal your mark, should you wish to do so.

Question 1

You have recently been hired to work for a large social media company. The company has staff shortages due to a recent firing spree, and you have been tasked with designing the recommendation algorithm used on the front page! After thinking for several days, you have figured out a way to measure a user's popularity, and now need to design an efficient algorithm to calculate it.

For a given user with n posts, you have access to an integer array $V = [v_1, v_2, \ldots, v_n]$, where v_i is the number of views the user's i^{th} post has in thousands. Since these view counts are used to order posts on a user's profile page, they are already sorted, in *descending* order. The popularity of a user is the largest value of p such that the user has at least p posts that each have at least p thousand views.

For example, suppose a user has $n = 6$ posts, and a view count array $V = [7, 6, 5, 4, 4, 1]$. Then they have at least 4 posts each with at least 4,000 views (indeed, they have five such posts), so $p = 4$ satisfies the criterion. You can confirm that the criterion is also satisfied for $p = 3$, but it is not satisfied for $p = 5$ since there are only three qualifying posts. The user's popularity index is in fact 4, because $p = 4$ is the largest value for which the criterion holds.

1.1 [7 marks] Before you are given full access to the codebase, you need to try and implement your new popularity measure on top of the existing code. Given the integer n and the integer array V for a user as well as an integer value $p \leq n$, you must determine whether that user's popularity is *at least* p. This snippet of code runs every single time the page gets loaded, so it needs to be very efficient.

Design an algorithm that runs in $O(1)$ time and checks whether a user's popularity is at least p.

1.2 [13 marks] The code you modified was written by a now-fired intern, and would simply try every value of p until it found one that worked, for every user. Now that the company has grown to serve millions of users, your boss has demanded that you develop a faster algorithm. Given the integer n and the integer array V for a user, you must efficiently compute the user's popularity index p.

Design an algorithm that runs in $O(\log n)$ time and computes p for a given user.

Question 2

DAC Investments is a finance company that has a team of k bankers. The ith banker manages $M[i]$ dollars worth of investments, and has a performance rating of $P[i]$. The bankers are indexed in *decreasing* order of performance rating.

The profitability of DAC Investments is given by

$$(M[1] + \ldots + M[k]) \times \min(P[1], \ldots, P[k]).$$

For example, if $M = [4, 8, 2, 7, 1]$ and $P = [9, 6, 3, 3, 2]$, then the profitability is

$$(4 + 8 + 2 + 7 + 1) \times \min(9, 6, 3, 3, 2) = 22 \times 2 = 44.$$

2.1 [2 marks] DAC Investments has recently received a letter from the CEO, advising them that his nephew is looking for work. Although the nephew's performance rating is no better than anyone in the team, DAC Investments have been *strongly encouraged* to hire him in place of one of the current team members.

You are given integer arrays $M[1..(k+1)]$ and $P[1..(k+1)]$, representing the amounts managed and performance ratings respectively. In each array, the first k entries are for the current team

members and the last entry is for the CEO's nephew. The performance ratings are listed in decreasing order.

In order to hire the CEO's nephew, DAC Investments will have to remove one of their bankers. How should they select which banker to remove, while maximising profitability?

For the remaining subquestions, suppose DAC Investments has been acquired by a large bank. There are now n employees available, so you are given integer arrays $M[1..n]$ and $P[1..n]$, the latter of which is again in decreasing order. From these n candidates, DAC Investments would like to select k team members.

2.2 **[10 marks]** Design an algorithm that runs in $O(nk)$ time and determines the set of k employees that maximise the profitability of DAC Investments.

2.3 **[8 marks]** Design an algorithm that runs in $O(n \log k)$ time and determines the set of k employees that maximise profitability.

You may choose to skip 2.2, in which case we will mark your submission for 2.3 as if it was submitted for 2.2 also.

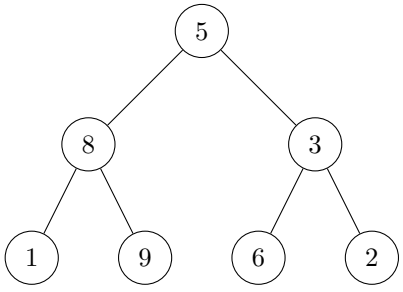
Question 3

A *traversal* of a binary tree is an order in which one can visit all the nodes.

- In-order traversal first recurses on the left subtree, then visits the root node, and recurses on the right subtree.
- Pre-order traversal first visits the root node, then recurses on the left subtree, and finally recurses on the right subtree.
- Post-order traversal first recurses on the left subtree, then recurses on the right subtree, and finally visits the root node.

For example, the tree given below has:

- in-order traversal of $[1, 8, 9, 5, 6, 3, 2]$,
- pre-order traversal of $[5, 8, 1, 9, 3, 6, 2]$, and
- post-order traversal of $[1, 9, 8, 6, 2, 3, 5]$.



3.1 **[8 points]** Given an array P that contains the pre-order traversal of a binary **search** tree of n nodes with distinct integer values, design an algorithm that runs in $O(n)$ time and computes the post-order traversal of the tree.

Note that the tree pictured in the example above is *not* a binary search tree.

3.2 **[12 points]** A binary tree is said to be *height-balanced* if:

- it has zero or one nodes, or
- the heights of its left and right subtrees differ by at most one, and both subtrees are height-balanced.

Given two arrays I and P which contain the in-order and pre-order traversals of a height-balanced binary tree of n nodes with distinct integer values, design an algorithm that runs in $O(n \log n)$ time and computes the post-order traversal of the tree.