Due Monday 10^{th} of July at 6pm Sydney time (week 7)

In this assignment we review some basic algorithms and data structures, and we apply the greedy method. 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

1.1 [12 marks] You are preparing to cross the Grand Canyon on a tightrope, and are building your balance pole out of steel rods. Naturally, you want to do this as cheaply as possible.

You have made a trip to Bunnings, and picked up n rods, the ith of which has length ℓ_i. You need to weld these all together to create your balance pole. Welding rods x and y costs $\ell_x + \ell_y$ dollars, and results in a new rod with length $\ell_x + \ell_y$.

For example, if we have rods with lengths $[2, 1, 3]$:

- After welding rods 2 and 3, costing \$4, we have rods with lengths $[2, 4]$
- We weld the remaining two rods, costing \$6
- The total cost to create the pole is \$10

Note that this is not necessarily the optimal sequence for these lengths.

Design an $O(n \log n)$ algorithm which finds the order you should weld the rods to minimize the total cost of welding.

1.2 [8 marks] Just before you step onto the wire, your pole snaps in half, and you realise going for the cheapest option might not have been the best idea. Fortunately, you brought spare rods and a welding device to the canyon.

As in part 1, you have n rods, the ith of which has length ℓ_i. It is given that n is odd.

Each rod in the pole contributes an *instability*, which is the product of the rod's length and the number of rods it is away from the middle. The order in which the welds are performed does not matter, only the order of the rods in the final pole.

That is, if you weld the n rods to create a pole $r_1, r_2, \ldots, r_n$, then the instability of the pole is given by

$$\sum_{i=1}^{n} r_i \times \left| i - \frac{n+1}{2} \right|.$$

For example, if we create a pole $[2, 4, 6, 5, 2]$, then it has instability

$$\begin{aligned}
&= (2 \times |1-3|) + (4 \times |2-3|) + (6 \times |3-3|) + (5 \times |4-3|) + (2 \times |5-3|) \\
&= (2 \times 2) + (4 \times 1) + (6 \times 0) + (5 \times 1) + (2 \times 2) \\
&= 4 + 4 + 0 + 5 + 4 \\
&= 17.
\end{aligned}$$

Design an $O(n \log n)$ algorithm which finds the order of rods in the pole with minimal instability.

Question 2

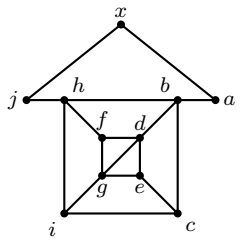
Sam has recently acquired Friendbook, a large social media conglomerate, under dubious circumstances. Since their position as the company's CEO is only probational, they wish to curry favour with the employees by providing raises. However, they don't want to give raises to those who aren't well-known in the company (that would be a waste), and they can't give raises to those who are too well-known (that would raise suspicions). As such, they want to figure out the largest number of people in the company they can give a raise to, without wasting money, and without being caught.

Thankfully for Sam, the company keeps an absurd amount of data on its employees, so they have access to an undirected graph G of all n company employees and who they know within the

company. In particular, this graph is provided as an **adjacency matrix** M such that $M[i, j] = 1$ if and only if employee i and j know each other. An **induced subgraph** H of G is a graph formed from a subset of the vertices of G, where pairs of vertices in H are adjacent if and only if the corresponding vertices are adjacent in G. In other words, an induced subgraph is formed by deleting some (or maybe none) of the vertices of G, and all adjacent edges.

Sam wishes to find the largest possible induced subgraph H of G such that every vertex in H is adjacent to at least k other vertices of H, but is also **not** adjacent to at least k other vertices of H. Such a subgraph is called "maximally nepotistic".

2.1 **[2 marks]** Consider the following graph G of company connections:



With $k = 3$, explain why any maximally nepotistic subgraph of G **cannot** include the vertex labelled x.

2.2 **[2 marks]** Using the graph G defined in 2.1, and again with $k = 3$, find a maximally nepotistic subgraph H of G, and explicitly list the vertices of H.

2.3 **[16 marks]** Given a graph G with n vertices and adjacency matrix M, and integer $k \geq 1$, design an algorithm which finds a maximally nepotistic subgraph of G, if it exists, and lists its vertices. If no such subgraph exists, the algorithm should not output anything. Your algorithm must run in time $O(n^3)$.

Question 3

3.1 **[10 marks]** Santa's assistant has been laid off and it is now your job to ensure that as many **wrapped** presents get delivered to shopping centres as possible.

Santa has an infinite supply of **unwrapped** presents at the North Pole, and can distribute them to k workshops across the globe. At each workshop i, a maximum of a_i presents can be wrapped, and the North Pole can distribute to it at most d_i presents.

The workshops wrap the presents, and then deliver these to m shopping centres. Each shopping centre j can hold up to b_j packages, however, due to some legacy contractual agreements, each shopping centre can only receive presents from certain workshops. You are given the contractual agreements as a two dimensional array C, where $C[i][j]$ denotes that workshop i has an agreement with shopping centre j.

Design an algorithm that runs in $O((k + m)k^2m^2)$ time which finds the maximum number of wrapped presents that can be delivered to shopping centres.

3.2 **[6 marks]** On Christmas Eve, you realise that you should ensure that each of the n children on Santa's nice list receive at least one present. Children are willing to travel at most D distance to a shopping centre to receive a present. You are given the location of all children and shopping centres.

Design an algorithm that runs in $O(nm(k+n))$ time which finds the maximum number of children on the nice list who can receive at least one present.

3.3 **[4 marks]** Let the farthest distance from any child to a shopping centre be F.

Design an algorithm that runs in $O(nm(k+n)\log F)$ time which finds the minimum **integer** value of D such that all n children on the nice list receive at least one present, or returns that no such D is possible.