Due Monday 27th March at 5pm Sydney time

In this assignment we apply the greedy method and associated graph algorithms, including algorithms for network flow. 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, *except* that you may not use network flow algorithms not presented in lectures.
- 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

A demolition company has been tasked with taking down n buildings, whose initial heights are positive integers given in an array $H[1..n]$. The company has two tools at its disposal:

- a small explosive, which reduces the height of a single building by 1, and
- a wrecking ball, which reduces the height of all buildings by 1.

The company has an unlimited supply of the small explosives, which are free to use. However, the wrecking ball costs \$1000 each time it is used.

The company initially has \$1000 dollars. They also get \$1000 whenever they finish the demolition of a building *with the wrecking ball* (reducing its height from 1 to 0), but they receive no money if the demolition of a building is finished using an explosive.

For example, for four buildings of heights $[2, 3, 5, 1]$, the following sequence of actions demolishes all buildings.

- Use a small explosive on building 1, resulting in $[1, 3, 5, 1]$.
- Spend the \$1000 to use the wrecking ball, resulting in heights $[0, 2, 4, 0]$. Buildings 1 and 4 were demolished, so the company receives \$2000.
- Use the wrecking ball again, leaving heights $[0, 1, 3, 0]$. No buildings were demolished, so no money is gained.
- Use an explosive to demolish building 2, giving $[0, 0, 3, 0]$. No money is awarded for completing the demolition of building 2, since the wrecking ball was not used.
- Use explosives three times to demolish building 3.

A sequence of actions is *minimal* if it demolishes all the buildings *and* there is no shorter sequence of actions that does this. The sequence above is *not* minimal.

1.1 **[2 marks]** Find a minimal sequence of actions to demolish all four buildings in the example above, where the heights are 2, 3, 5 and 1. You must provide reasoning for why your sequence is minimal.

1.2 **[6 marks]** Show that there is always a minimal sequence of actions that results in the buildings being demolished where all uses of the explosive occur before any uses of the wrecking ball.

1.3 **[4 marks]** Identify a criterion for whether there is a minimal sequence of actions to demolish all buildings using *only* the wrecking ball, with reasoning to support it.

> Your criterion should be a test which given n and the array H answers YES if there is any minimal sequence using only the wrecking ball, or NO otherwise.

1.4 **[8 marks]** Design an algorithm that runs in $O(n \log n)$ time and determines a minimal sequence of actions to reduce each building's height to 0.

Question 2

You run a jewellery shop and have recently been inundated with orders. In an attempt to satisfy everyone, you have rummaged through your supplies, and have found m pieces of jewellery, each with an associated price. To make things easier, you have ordered them in increasing order of their price. You now need to find a way to allocate these items to $n \leq m$ customers, while minimising the number of customers who walk away with nothing.

2.1 **[8 marks]** Each customer has a minimum price they will pay, since they all want to impress their significant others. Of course, you need to decide who gets what quickly, or else everyone will just leave to find another store.

Given an array $P[1..m]$ of jewellery prices, sorted in ascending order, and an array $M[1..n]$ of customers' minimum prices, design an algorithm which runs in $O(n \log n + m)$ time and allocates items to as many customers as possible, such that the price of the item given to customer i is at least $M[i]$, if they are given an item at all.

For example, suppose the store has $m = 5$ pieces of jewellery with prices $P = [5, 10, 15, 20, 25]$, and the minimum prices of the $n = 3$ customers are $M = [21, 15, 31]$. In this case, the first customer can get the \$25 item, and the second customer can get the \$15 item or the \$20 item. However, the third customer will walk away as the store does not have any jewellery that is priced at \$31 or higher.

2.2 **[8 marks]** With a sigh of relief, and an allocation set up, you go to ring up the first customer's item, just to find out that they can't afford it! In a panic, you get everyone to give you their budgets, and go back to the drawing board.

Given an array $P[1..m]$ of jewellery prices, sorted in ascending order, and array $M[1..n]$ and $B[1..n]$ of customers' minimum prices and budgets, design an algorithm which runs in $O(n^2 m)$ time and allocates items to as many customers as possible, such that the price of the item given to customer i is at least $M[i]$ and doesn't exceed $B[i]$, if they are given an item at all.

For example, the store has $m = 5$ pieces of jewellery with prices $P = [5, 10, 15, 20, 25]$. The minimum prices of the $n = 3$ customers are $M = [21, 16, 31]$, and their budgets are $B = [26, 19, 38]$. In this case, the first customer can get the \$25 item, but the second customer has to walk away as he refuses the \$5, \$10 and \$15 items but cannot afford the \$20 and \$25 items. The third customer will also walk away as the store does not have any jewellery that is priced at \$31 or higher.

2.3 **[4 marks]** There are only $k < m$ days until Valentine's day, and all of your customers need to get their orders before then. Unfortunately, you are only able to process a total of five orders per day. To make matters worse, each customer is only available to pick up their orders on some of the days.

Given an array $P[1..m]$ of jewellery prices, sorted in ascending order, arrays $M[1..n]$ and $B[1..n]$ of customers' minimum prices and budgets, and an array $F[1..n][1..k]$ where

$$F[i][j] = \begin{cases} 1 & \text{if customer } i \text{ is free on day j} \\ 0 & \text{otherwise} \end{cases}$$

design an algorithm which runs in $O(n^2 m)$ and allocates items and assigns collection days to as many customers as possible, such that the price of the item given to customer i is at least $M[i]$ and doesn't exceed $B[i]$ if they are given an item at all, and they are free on their assigned collection day.

For example, the store has $m = 5$ pieces of jewellery with prices $P = [5, 10, 15, 20, 25]$. The minimum prices of the $n = 3$ customers are $M = [21, 16, 31]$ and their budgets are $B = [26, 19, 38]$.

The array $F[1..n][1..k]$ has entries $F[1][1], F[2][1], F[2][2]$ and $F[3][2]$ equal to 1, and everything else set to 0, representing customers 1 and 2 free on day 1, and customers 2 and 3 free on day 2. In this case, the number of purchases is less than 5 on both days, so for the same reasons described in 2.2, only the first customer can get jewellery.

Question 3

You are studying the ancient Antonise language, and are trying to create some kind of alphabet for it. You have scoured the texts and figured out that there are g many glyphs in this language, but you don't know the order of the alphabet, and want to figure out what it might be. Thankfully, you have found a tome with n unique names, all conveniently k glyphs long, which you suspect are in alphabetical order, and hope to find a way to rearrange the glyphs into an alphabet consistent with the order of the names. For example, suppose your tome contained the following $n = 3$ names, consisting of $k = 3$ glyphs each, from $g = 5$ possible glyphs:

Then both  and  would be possible alphabets, as the names are in alphabetical order with respect to either of these. However,  would not, as the names  and  would be out of order with respect to any alphabet where comes before  .

3.1 **[4 marks]** Provide a small example of a tome of at most five names, each of length at most four glyphs, which are not in alphabetical order regardless of how you rearrange the glyphs to form an alphabet.

You must list the names in the order they are found in the tome, and briefly explain why they cannot be in alphabetical order, no matter what order the glyphs are arranged into an alphabet. You may use any symbols for the glyphs in this example, such as the dingbats used in the question, English letters, or numbers.

3.2 **[16 marks]** You are given the number of glyphs g, the number of names n, the number of glyphs in each name k, and a two-dimensional array $S[1..n][1..k]$ containing the numbers 1 through g, where $S[i]$ is an array containing the i^{th} name, and $S[i][j]$ is the index of the j^{th} glyph in that name. The original indexing is arbitary, and the names may not be in alphabetical order with respect to this indexing.

Design an algorithm which runs in $O(nk+g)$ time and finds an alphabet (i.e. reindexes the glyphs) so that the names in S are in alphabetical order if this is possible, or otherwise determines that no such alphabet exists.

An algorithm which runs in $\Theta(n^2k + g)$ time will be eligible for up to 12 marks.