**Due 27th 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 max flow algorithms you must clearly detail any graphs you are constructing. This means describing all vertices, edges, capacities, sources, and sinks. Diagrams (either neatly hand-drawn or in LaTeX) are also encouraged as a supplement. In terms of quoting known max flow algorithms, you may only quote the ones taught in lectures (i.e. Ford-Fulkerson or Edmonds-Karp). To justify the correctness of a max flow algorithm, you must explain why your algorithm result satisfies all conditions imposed by the problem. Roughly 10% of the total marks will be specifically allocated towards the clarity and conciseness of your explanations.

## Question 1 *Arc Competition*

[**20 marks**] Arc is hosting a training session for the members of various societies at UNSW. There are $n$ students who are Arc members, and $m < n$ registered societies. Each of the $n$ students is a member of at least 1 and at most 4 of the $m$ societies. Each of the $m$ societies must send exactly 1 student to represent them at the training session.

In order to keep the crowd diverse, Arc would like to avoid all of the student representatives studying degrees from the same faculty. Each of the $n$ students is enrolled in a single degree offered by 1 of the $k \leq n$ faculties at UNSW. For every faculty $i \in [1..k]$, at most $u_i$ students belonging to it will be allowed to attend the event.

Design an $O(nm)$ algorithm that determines if it is possible to find a selection of students to attend the event such that all of these criteria are met. If it is possible, also identify which $m$ students will attend.

## Question 2 *Pirate Gathering*

[**30 marks**] Somewhere in the Caribbean there are $V$ islands labeled $1..V$ and $N$ pirate ships named $1..N$. Additionally there are $E$ two-way routes, the $i^{th}$ of which allows a ship to sail between island $u_i$ and island $v_i$. However, the pirates are fiercely territorial and hence each direction of a route can only be used by one pirate ship. That is, if ship $x$ has traveled from island $u$ to island $v$, another ship $y$ can still travel from island $v$ to island $u$.

The pirate ships are initially on unique islands $\{h_1, h_2, \ldots, h_N\}$, and wish to gather on island $T \in [1, ..., V]$.

**2.1** [**18 marks**] Design an $O(NE)$ algorithm which determines whether or not it is possible for every pirate to reach the designated island, $T$.

**2.2** [**6 marks**] In an attempt to prevent the gathering, the government has now imposed a restriction that at most $s_i$ pirate ships are allowed to depart from island $i$.

Design an $O(NE)$ algorithm which determines whether every pirate can still reach the designated island. You may reference your answer from Question 2.1 and only provide the details and justification of any modifications.

You may choose to skip Question 2.1 in which case your solution to Question 2.2 will be submitted for both parts.

**2.3** **[6 marks]** The location of the designated island $T$ has been leaked to the government and it may no longer be safe. The pirates want to know if there any other options where all pirates can gather, given the same restrictions as Question 2.2. Design an $O(VEN)$ algorithm that counts the number of islands are suitable for the gathering (including island $T$).

You may reference your answer from Question 2.2 and only provide the details and justification of your modifications, which must also include an updated time complexity justification.

## Question 3 *CSE Labs*

**[30 marks]** The School of CSE is scheduling some labs for its classes. There are $n$ classes that need to be scheduled into one of the $n$ available labs, and they would like all of the labs to run at the same time. A single class cannot occupy multiple labs, and multiple classes also cannot share a lab.

**3.1** **[4 marks]** For a class to use a lab, there has to be at least 1 seat for each student. Given an array $C[1...n]$ where $C[i]$ indicates the number of students in class $i$, and an array $L[1...n]$ where $L[j]$ indicates the number of seats in lab $j$, design an $O(n \log n)$ time algorithm that assigns a lab room to each class. Your algorithm should also identify if an assignment is not possible.

**3.2** **[7 marks]** It turns out that classes are very picky about which labs they want. In particular, you are now provided an additional array $B[1..n][1..n]$ with

$$B[i][j] = \begin{cases} \texttt{True} & \text{class } i \text{ likes lab } j \\ \texttt{False} & \text{otherwise} \end{cases}$$

Given $C[1..n]$, $L[1..n]$ and $B[1..n][1..n]$, design an $O(n^3)$ algorithm to assign a lab room to each class. Your algorithm should also identify if an assignment is not possible.

**3.3** **[12 marks]** Unfortunately an issue with the building has affected the power and now there are only $m < n$ labs available. The school has no choice but to schedule some classes at different times. Given the arrays $C[1...n]$, $L[1...m]$, $B[1..n][1..m]$ defined as in the previous parts, design an $O(n^3)$ time algorithm that determines whether it is possible to schedule the classes with $T < N$ different time slots available.

**3.4** **[7 marks]** Each of the classes also requires one of $k \leq n$ tutors to run the lab. Each tutor is only allowed to run a maximum of 3 labs, and can only run 1 lab at a time. The tutors' availability is given as an array $D[1..k][1..T]$ with

$$D[i, j] = \begin{cases} 1 & \text{if the } i\text{th tutor is available at timeslot } j \\ 0 & \text{otherwise} \end{cases}$$

For some number of timeslots $T < n$, design an $O(n^3)$ algorithm that assigns a room, time, and tutor to each lab if a schedule is possible.

## Question 4 *Spy Escape*

**[20 marks]** Agency X has sent $n$ spies to Sydney for a secret mission. Before the mission begins, Agency X has prepared $m > n$ secret hideouts throughout the city, of which $n$ of them contain

a single emergency escape pod. The hideouts are connected via a network of tunnels, and each hideout can only accommodate one spy at a time. The emergency escape pods can also only accommodate one spy. Everyday, the spies can crawl through at most one tunnel to reach a new hideout.

The tunnels are represented by an adjacency matrix $T[1..m][1..m]$, where

$$T[i][j] = \begin{cases} \texttt{True} & \text{hideout } i \text{ has a tunnel to hideout } j \\ \texttt{False} & \text{otherwise} \end{cases}$$

**4.1** **[14 marks]** A few days after the mission began, all spies have been compromised. The spies are currently scattered around the hideouts in Sydney and must make it to an emergency escape pod within $D$ days to avoid capture.

Design an $O(nm^2D)$ algorithm which determines whether or not all spies can successfully escape.

> Try to first solve this for $D = 1$, and then consider how would you could extend this for $D = 2$ and greater.

**4.2** **[6 marks]** Despite the compromise, Agency X still wants to wrap up some tasks in Sydney before the spies escape. The spies will die if they are not done in D days.

Design an $O(nm^2D \log D)$ algorithm to determine the minimum number of days needed for all spies to successfully escape.