MONASH University

# 2022 Semester One (June 2022) Examination Period

# Faculty of Information Technology

| | |
|---|---|
| EXAM CODES: | FIT2004 |
| TITLE OF PAPER: | Algorithms and data structures |
| EXAM DURATION: | 2 hours 10 mins |

## Rules

During your eExam, you must not have in your possession any item/material that has not been authorised for your exam. This includes books, notes, paper, electronic device/s, smart watch/device, or writing on any part of your body. Authorised items are listed above. Items/materials on your device, desk, chair, in your clothing or otherwise on your person will be deemed to be in your possession. Mobile phones must be switched off and placed face-down on your desk during your exam attempt.

You must not retain, copy, memorise or note down any exam content for personal use or to share with any other person by any means during or following your exam. You are not allowed to copy/paste text to or from external sources unless this has been authorised by your Chief Examiner.

You must comply with any instructions given to you by Monash exam staff.

As a student, and under Monash University's Student Academic Integrity procedure, you must undertake all your assessments with honesty and integrity. You must not allow anyone else to do work for you and you must not do any work for others. You must not contact, or attempt to contact, another person in an attempt to gain unfair advantage during your assessment. Assessors may take reasonable steps to check that your work displays the expected standards of academic integrity.

Failure to comply with the above instructions, or attempting to cheat or cheating in an assessment may constitute a breach of instructions under regulation 23 of the Monash University (Academic Board) Regulations or may constitute an act of academic misconduct under Part 7 of the Monash University (Council) Regulations.

## Authorised Materials

| | | |
|---|---|---|
| CALCULATORS | ☐ YES | ☑ NO |
| DICTIONARIES | ☐ YES | ☑ NO |
| NOTES | ☐ YES | ☑ NO |
| WORKING SHEETS | ☑ YES | ☐ NO |
| PERMITTED ITEM | ☐ YES | ☑ NO |

if yes, items permitted are:

## Instructions

- This is a **closed book** exam with Specifically permitted items.
- Please **attempt as many questions as possible**.
- **Read each question carefully**. The marks associated with a question is not an indicator of the question difficulty or solution length.
- For algorithms and pseudocode questions, you are allowed to use either:

1. Structured indented text structure.
2. Numbered list.

Best wishes and all the best!

# Instructions

## Information

You can review your exam instructions by clicking the 'Show Instructions' button above.

# Correctness and Complexity

## Question 1

For constants *b* and *c*, consider the recurrence relation given by:

2 Marks

- T(n) = b, if n=1
- $T(n) = 2 * T(n/2) + c * n^3$, if n>1

Which of the following statements is true?

Select one:

- a. $T(n) = \Theta(n^3 * \log n)$
- b. $T(n) = \Theta(n^4)$
- c. $T(n) = \Theta(n^3)$
- d. $T(n) = \Theta(n^6 * \log n)$
- e. $T(n) = \Theta(n^3 * \log n * \log n * \log n)$

## Question 2

Consider the following algorithm, which returns `True` if and only if `m` is a factor of `sum(L)`, where `L` is a list of integers.

4 Marks

```
def sum_factor(L, m):
    """
    Input : A list L of integers, and an integer m
    Output: True if m is a factor of sum(L), and False otherwise.

    For example:
    >>> sum_factor([3, 1, 2, 6], 3)
    True
    >>> sum_factor([4, 1, 2, 6], 3)
    False
    """

    n = len(L)
    s = 0

    for i in range(n):
        s = (s + L[i]) % m

    return s == 0
```

**(a)** Write down a useful invariant for this algorithm.

**(b)** Show that the invariant you wrote is true on loop entry, and each time the loop runs.

**(c)** Now use your invariant to argue that the algorithm is correct.

# Sorting

## Question 3

Consider an input with ***N*** **elements.**

Please select **all correct statements** regarding **comparison-based** sorting algorithms.

2 Marks

Select one or more:

a.
The best-case complexity of insertion sort is O(N).

b.
The best-case complexity of merge sort is O(N).

c.
Heap sort is stable.

d.
Selection sort is stable.

e.
The average-case complexity of selection sort is $\theta(N^2)$.

# Quickselect and Median of Medians

## Question 4

The **Quickselect** algorithm can be run in *O(N)*, provided we have an *O(N)* algorithm to find a **median pivot**.

2 Marks

In your own words, explain why the **median of medians** algorithm can be used for this purpose, even though it only gives an approximation of the median value.

## Question 5

You are interviewing a large number of applicants for a job. There are three rounds of interviews, and for the first round, each applicant you interview is assigned a **unique suitability ranking**. These rankings are represented as an **unsorted list of *N* items.** Each item in the list is of the form **(name, rank)**, where **name** is a string representing the applicant's name, and **rank** is a unique floating point value ranging from 0 to 100. Once you've completed all the interviews in the first round, you want a quick way of calculating who should move on to the second and final rounds of interviews.

3 Marks

- The 50% of applicants who received the lowest suitability rankings will not progress any further with the interviews - they will be told they are unsuccessful in their application.
- The 20% of applicants who received the highest suitability rankings can skip the second round of interviews and instead go straight to the third round.
- The remaining 30% of applicants should progress to the second round of interviews.

**Describe an efficient algorithm using Quickselect** to determine the names of applicants that you know will be proceeding to the second and third round of interviews, after completing the first round. Your algorithm should **run in *O(N)* time**, and you can assume that you have access to a quickselect algorithm which runs in *O(N)* time.

# Graph Structure and Traversal

## Question 6

For each of the following operations, determine its **worst-case big-Θ complexity**.

2 Marks

In this question,

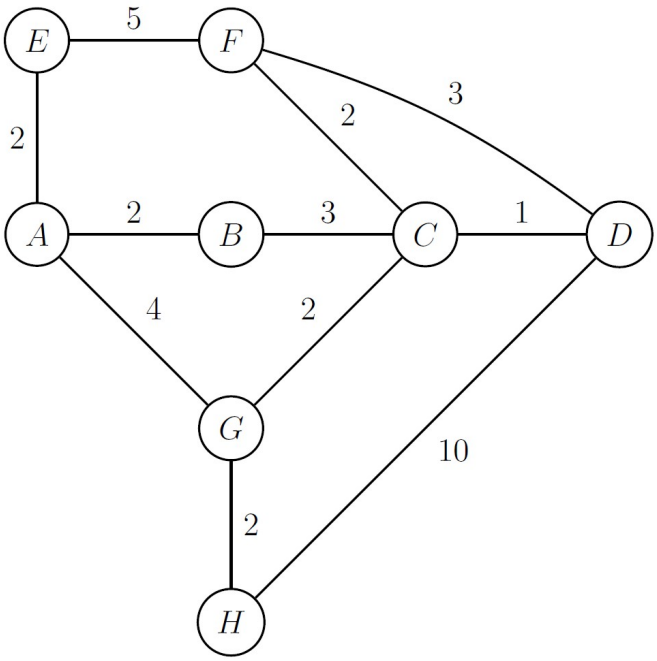
- The graph $G$ is a directed weighted graph.
- $V$ refers to the number of vertices in the graph.
- $E$ refers to the number of edges in the graph.
- $N(A)$ refers to the number of neighbors of vertex $A$.

Assume that in the adjacency list representation, the **interior lists are unsorted.**

| | |
|---|---|
| Time complexity to obtain all incoming edges for vertex $A$ in an adjacency matrix representation. | · Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E) |
| Time complexity to determine if an edge from vertex $A$ to vertex $B$ exists in an adjacency list representation. | · Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E) |
| Time complexity to determine if an edge from vertex $A$ to vertex $B$ exists in an adjacency matrix representation. | · Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E) |
| Time complexity to obtain all incoming edges for vertex $A$ in an adjacency list representation. | · Θ(log V) · Θ(V) · Θ(V^2) · Θ(N(A)) · Θ(V+E) · Θ(log E) · Θ(1) · Θ(E) |

## Information

Consider the **weighted undirected graph** below and answer the following questions.



## Question 7

1 Mark

Perform a **depth-first search** on the graph given above **starting from node A**.

Whenever you have a choice between two nodes, **break ties in ascending alphabetical order.**

Associate each node with the **number for its position in the visiting order**, ie. node A should be associated with *i* if A is *i*-th visited node.

For this question, you can ignore the edge weights.

| | |
|---|---|
| 1st visited node | · D · E · A · H · B · F · G · C |
| 2nd visited node | · D · E · A · H · B · F · G · C |
| 3rd visited node | · D · E · A · H · B · F · G · C |
| 4th visited node | · D · E · A · H · B · F · G · C |
| 5th visited node | · D · E · A · H · B · F · G · C |
| 6th visited node | · D · E · A · H · B · F · G · C |
| 7th visited node | · D · E · A · H · B · F · G · C |
| 8th visited node | · D · E · A · H · B · F · G · C |

## Question 8

1 Mark

Perform a **breadth-first search** on the graph given above **starting from node A**.

Whenever you have a choice between two nodes, **break ties in ascending alphabetical order.**

What is the **maximum height** of the resulting tree from the breadth-first-search you have performed? **Just type the numerical answer.**

For this question, you can ignore the edge weights.

# Dynamic Programming

## Question 9

2 Marks

Recall the following problem from the Dynamic Programming studio:

"You are trying to sell to a row of houses. You know the profits which you will earn from selling to each house *1..n*. If you sell to house i, you cannot sell to houses *i-1* or *i+1*. What is the maximum profit you can obtain?".

Consider instead the **variant of the problem** where **if you sell to house *i*, you cannot sell to houses *i-2*, *i-1*, *i+1* or *i+2***.

Suppose that you have the following DP array, where **cell *i* of the DP array contains the maximum profit you can obtain by selling to a valid subset of the first *i* houses (i.e., a valid subset of {1, 2, ..., *i*})**.

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DP[i] | 12 | 12 | 35 | 35 | 35 | 65 | 65 | 110 | 112 | 120 | 120 | 120 |

Using **backtracking**, determine which houses you should sell to in order to **maximise your profit** (i.e determine the optimal solution to the problem that is **valid for any possible house-profit array** that would result in the DP array above).

For this question you must answer in the following format:

**Write the indices of the houses that you have chosen**, in **ascending order**, separated only by a single comma with no spaces. For example if the answer were houses 7, 8 and 9, you would write 7,8,9

# Shortest Path

## Question 10

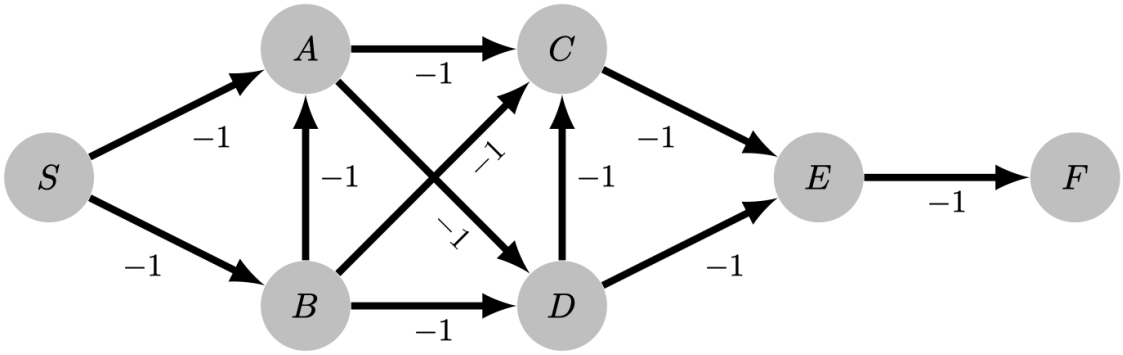
3 Marks

Consider the following version of the **Bellman-Ford algorithm**

**Algorithm 54** Bellman-Ford

```
1: function BELLMAN_FORD(G = (V, E), s)
2:     dist[1..n] = ∞
3:     pred[1..n] = null
4:     dist[s] = 0
5:     for k = 1 to n − 1 do
6:         for each edge e in E do
7:             RELAX(e)
8:     return dist[1..n], pred[1..n]
```

and the following directed graph




Let *S* be the source node for the execution of the Bellman-Ford algorithm.

If the edges are relaxed in the following order (S,A),
(S,B), (B,A), (B,D), (D,E), (A,D), (A,C), (D,C), (E,F), (B,C), (C,E), what is the **value of dist[E]+dist[F] after the first iteration of the outer loop is finished?**

Select one:

- a. dist[E]+dist[F]=-7
- b. dist[E]+dist[F]=-9
- c. dist[E]+dist[F]=-11
- d. dist[E]+dist[F]=-8
- e. dist[E]+dist[F]=∞
- f. dist[E]+dist[F]=-10

## Question 11

Consider a run of the **Floyd-Warshall algorithm** on a **Directed Weighted Graph** *G*.

2 Marks

**Algorithm 56** Floyd-Warshall

```
1: function FLOYD_WARSHALL(G = (V, E))
2:     dist[1..n][1..n] = ∞
3:     dist[v][v] = 0 for all vertices v
4:     dist[u][v] = w(u, v) for all edges e = (u, v) in E
5:     for each vertex k = 1 to n do
6:         for each vertex u = 1 to n do
7:             for each vertex v = 1 to n do
8:                 dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v])
9:     return dist[1..n][1..n]
```

The run produced the following matrix:

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| A | 0 | 14 | None | None | 20 | -9 | 20 | -8 |
| B | 13 | 0 | None | None | 49 | 20 | None | 44 |
| C | 16 | 49 | 0 | 20 | 12 | -6 | -3 | 40 |
| D | 25 | 49 | None | 0 | 30 | 47 | None | None |
| E | None | 39 | None | None | 0 | 40 | 45 | 12 |
| F | None | 11 | 31 | 18 | 25 | 0 | 49 | 39 |
| G | 32 | None | None | None | None | 47 | 0 | 34 |
| H | -10 | 7 | 22 | None | None | 15 | 21 | 0 |

**Select the correct observation(s)** that can be made from the matrix above.

Note: You **do not need to validate the correctness of the matrix** but just make observations based on the given matrix.

Select one or more:

a.
There is a negative cycle in the graph.

b.
There is a cycle including vertices *A* and *G*.

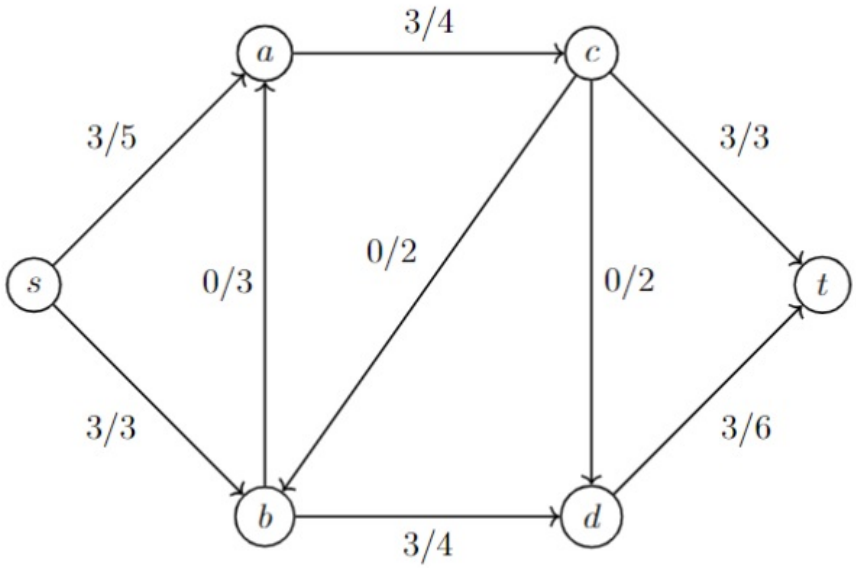
c.
There is an edge from vertex *B* to vertex *E* with a distance of 49.

d.
There is a path from vertex *C* to vertex *F* with a distance of -6.

# Network Flow

## Information

Consider the **flow network** below and answer the following questions.




## Question 12

What is the **maximum possible flow** for the given flow network above? Just type the numerical answer.

1 Mark

## Question 13

A cut partitions the vertices into two disjoint sets, $S$ and $T$, where $S$ contains all the vertices on the source side of the cut, and $T$ contains all the vertices on the sink side of the cut.

1 Mark

Consider the **minimum cut** of the above flow network. Select the **vertices which are in $S$** from the list of vertices below.

Select one or more:

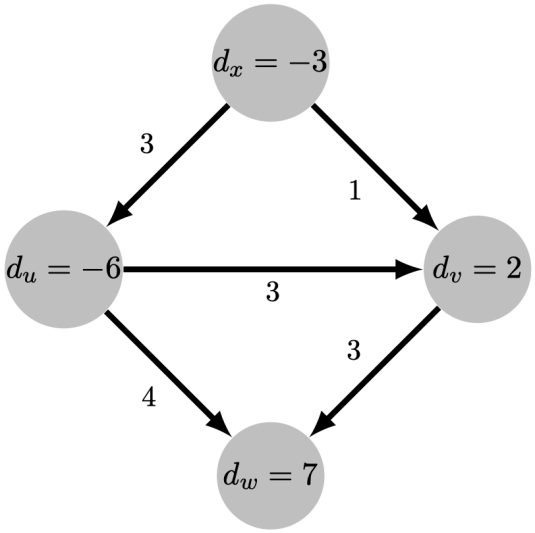
- a. *s*
- b. *a*
- c. *b*
- d. *c*
- e. *d*
- f. *t*

## Question 14

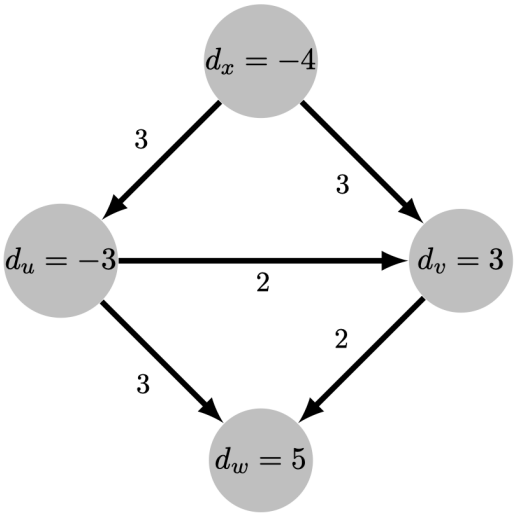
Consider the following two problems of **circulation with demands**, in which the demands are indicated in each vertex, and the capacity in each edge.

3 Marks

Problem 1:




Problem 2:




Which of the problems have **feasible solutions**?

Select one:

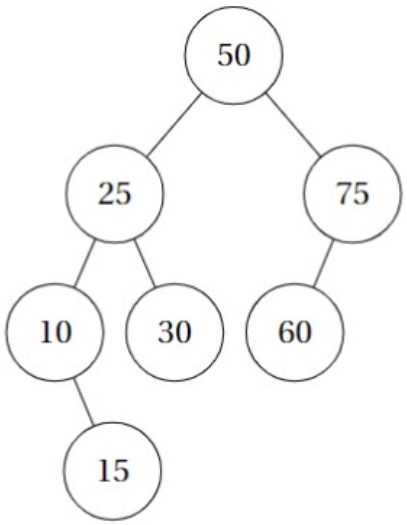
- a. Neither Problem 1 nor Problem 2 has a feasible solution.
- b. Both Problem 1 and Problem 2 have feasible solutions.
- c. Only Problem 1 has a feasible solution.
- d. Only Problem 2 has a feasible solution.

# Efficient Lookup Structures

## Question 15

Consider the following **AVL tree** below:

2
Marks




Which of the following statements are **true**?

Select one or more:

a.
Just deleting 10 without performing rotations would keep the tree balanced.

b.
Just deleting 30 without performing rotations would keep the tree balanced.

c.
The AVL tree is unbalanced.

d.
Just inserting 12 without performing rotations would make the tree unbalanced.

e.
Just inserting 88 without performing rotations would make the tree unbalanced.

## Question 16

Consider a **hash table implemented with separate chaining** for collision resolution.

For a hash table with $N$ items, which of the following data structures would cause the worst-case time complexity of an **insert operation** to be $\Theta(N)$ if the data structure is used to keep the separate chains.

2 Marks

Select one or more:

a. Binary search tree

b. Unsorted array

c. AVL tree

d. Sorted array

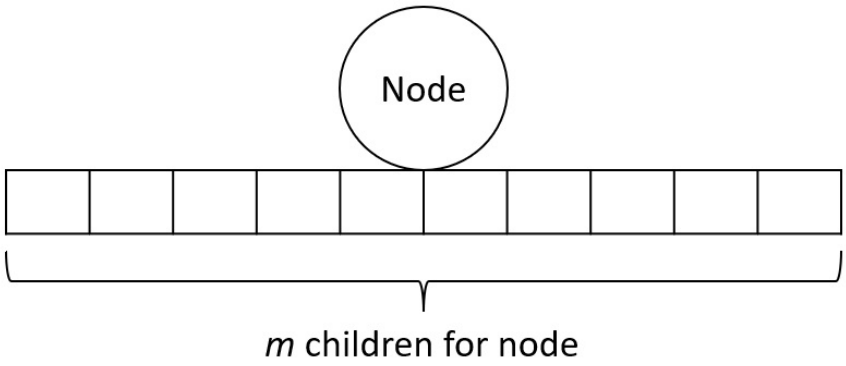
# Retrieval Data Structures for Strings

## Question 17

Assume that you have an alphabet size of ***M*** **unique characters** and let ***S*** **be a string of length** ***N.***

Thus, you **cannot assume the alphabet size to be** ***O(1)*** as discussed in the lecture. A node implemented with this condition, using an **array of size** ***M*** **for the children** is illustrated below.

2 Marks




For nodes implemented in this way, what is the **worst-case space complexity** in terms of *M* and *N* for string *S* of:

A suffix trie.

- Θ(N log M)
- Θ(N M^2)
- Θ(NM)
- Θ(N)
- Θ(N log N)
- Θ(N+M)
- Θ(N^2 M)
- Θ(M)

A suffix tree, with edges using the **[start,end] or [start,length]** representation.

- Θ(N log M)
- Θ(N M^2)
- Θ(NM)
- Θ(N)
- Θ(N log N)
- Θ(N+M)
- Θ(N^2 M)
- Θ(M)

## Question 18

Assume that we are constructing the **suffix array for a string** ***S*** using the **prefix doubling approach**.

We have already sorted the suffixes for string ***S*** according to their **first 2 characters**; with the **corresponding rank array** shown below:

3 Marks

| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Rank | 11 | 10 | 2 | 7 | 2 | 7 | 2 | 9 | 5 | 6 | 1 |

We are now **sorting on the first 4 characters**.

For the suffixes with **ID5** and **ID7, describe in detail** (i.e. all the steps) how will you **compare them on their first 4 characters in** ***O(1)*** and what **the result (order)** from the comparison would be.

# Applications

## Question 19

Consider the following application problem.

3 Marks

You are the transport minister of your country. The recent disaster devastated the road network connecting all of the key locations. Thus you are to **plan out the optimal roads to connect the key locations together,** based on the following criteria:

- You want to **connect all key locations** together; but they **do not need to be directly connected**
- You are given the **importance of directly connecting 2 key locations together,** for all key location pairs. The importance is represented by a number for each pair of locations, and the highest this number is, the most important it is to directly connect the 2 key locations.
- You are given the **cost of directly connecting 2 key locations together,** for all key location pairs.
- For cost savings reasons, you should not build redundant roads: **between any pair of locations, there should be only one possible path.**
- You want to **maximise the importance** of connecting all key locations together.
- If there are multiple ways of achieving the maximum value of the importance, then you should pick the one that **minimises the cost of the selected roads.**

Describe how you would **model this problem as a graph** and **algorithmically solve** this problem **efficiently** using an algorithm you have learnt in the unit.

## Question 20

3 Marks

You are the manager of a super computer and receive a set of requests for allocating time frames in that computer.

- The $i^{th}$ request specifies the desired starting time $s_i$ and the desired finishing time $f_i$.
- A subset of requests is compatible if there is no time overlap between any requests in that subset.

For a set of $N$ requests, give a high-level description of an algorithm with time complexity $O(N * log N)$ to **choose a compatible subset of maximal size** (i.e., you want to accept as many requests as feasible, but you cannot select incompatible requests).

## Question 21

2 Marks

You are coordinating a final year project unit.

There are a total of **123 students** that are enrolled into the unit, and there are **25 topics** to choose from:

- Each student is allowed to select up to **4 preferred topics**, but they would only be **assigned to only 1 topic** at the end.
- Each topic can be assigned to at most **5 students**.

You realised that it is not possible for all students to be doing their preferred topics, as some topics are more popular than others. You would however prioritize to satisfy the preferences of as many students as possible.

Describe how you would model this problem as a **maximum flow problem**; which is then solved using the **Ford-Fulkerson method**.

## Question 22

4 Marks

The engineers in your company believe they have created a phone that is ultra-resistant against falls (perhaps even falls from as high as 150m). To test that hypothesis, your company built two prototypes and asked you to perform the test to determine the maximum height in meters (without considering fractions) that the phone can be dropped without breaking.

There is an unknown integer value $0 \le x \le 150$ such that if the prototype is dropped from heights up to $x$ meters it will not break, while if it is dropped from heights $x+1$ meters or higher it will break. **Your job is to determine $x$.** If one prototype is broken, it cannot be used in the tests anymore.

As a computer science student, you want to optimize your work. For doing so, you develop an algorithm for determining the heights you should drop the prototype at each iteration (it can take into account the result of the previous iterations). No matter what is the value of $0 \le x \le 150$, your algorithm should be correct and determine the value of $x$. For an algorithm $A$ and an unknown integer $0 \le x \le 150$, let **Drop($A$, $x$) denote the number of times you need to drop a prototype** if you use algorithm $A$ and $x$ is the threshold. Let Drop($A$) denote the worst-case performance of $A$, which is given by the maximum of Drop($A$, $x$) over all possible $0 \le x \le 150$.

For an optimal deterministic algorithm A that has Drop($A$) as small as possible, what is the value of Drop($A$)? Just type the numerical answer.

Hint: Since you have two prototypes, it is possible to do better than linear search. But as you only have two prototypes, you need to be very careful if only one prototype remains intact (and thus the search cannot be as efficient as binary search).