Important Information

All Python code you write for this exam must satisfy the following requirements:

- Syntax should satisfy Python 3 requirements
- Use syntax, modules, structures and constructs presented in lectures.
- Avoid using in built libraries that are performing non-trivial operations
- Tasks included here are intended to be discussed with a group of peers; the best learning experience will be gained if you follow this advice
- the difficulty of these questions is **not** representative of the difficulty of the exam, these are designed to be attempted over a longer period of time with discussion to assist your learning and comprehension
- You can implement some of these tasks in code, however it is suggested you attempt them by hand first

Write down any assumptions you make.
The table below is included for your convenience

Question	Points	Score
1	12	
2	23	
3	14	
4	12	
5	14	
Total:	75	

Question 1: [12 marks]

This question is about invariances of problems and how they can help us solve problems.

Imagine a situation with 100 army recruits and one twisted drill sargeant. One night the sargeant tells them he's going to give them the opportunity for a week off if they play his game, but if they lose, it's latrine duty for the week. He will run his game the following morning (giving the recruits the night to prepare a strategy).

The game is this: Each of the 100 recruits stand in a line facing the person in front of them (each 5 metres apart). In this, each recruit can see every recruit in front of them but not the ones behind; the one at the very front sees a wall and the one at the very back can see all the recruits (except themselves).

The sargeant will go around and put one of two helmets on each recruit, a green one or a purple one and they each have to guess what colour their helmet is. If they guess correct it's a week off, if wrong it's latrine duty. The sargeant tells them he'll ask each recruit in turn from the back all the way to the front of the line.

the catch is that when they're lined up they can't communicate with each other, the only thing they can say is which helmet they think is on their head ("purple" or "green").

(a) (4 marks) write out the invariants of this problem

> hint: consider how many helmets of each colour remain after each recruit is considered

(b) (3 marks) Suggest a strategy that allows the most recruits to get time off as possible

> you might be interested in reading this `http://www.cartalk.com/content/prisoners-and-hats?question`. After you read this, have another go at writing up the invariants

(c) (5 marks) Write up an algorithm that any given recruit can use that follows this strategy

should meet all of the standard requirements of algorithms

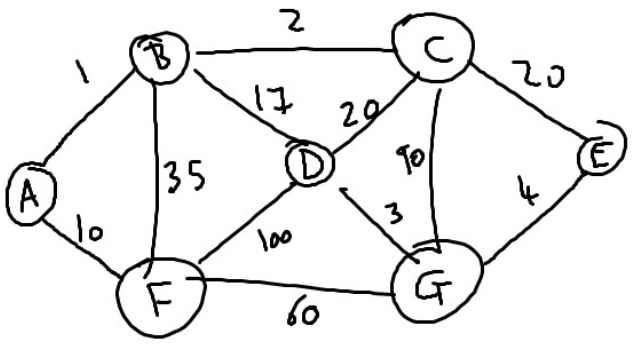
- finite sequence
- effectiveness
- definiteness
- ≥ 0 input(s)
- > 0 output(s)

5

Question 2: [23 marks]

This question is about brute force solutions to problems and how these relate to other kinds of solutions.

Consider the problem of finding a minimal spanning tree for a given graph. (For example the graph below)



Which has the following adjacency Matrix:

```
A:[0,1,0,0,0,10,0]
B:[1,0,2,17,0,35,0]
C:[0,2,0,20,20,0,0]
D:[0,17,20,0,0,100,3]
E:[0,0,20,0,0,0,4]
F:[10,35,0,100,0,0,60]
G:[0,0,90,3,4,60,0]
```


0

(a) (6 marks) Suggest a brute force strategy to find the minimal spanning tree of a graph (such as the one provided)

as a hint consider the problem of finding all subsets of a given set and see if you can find a relation between this problem and the problem of finding a spanning tree.

(b) (6 marks) Suggest a backtracking strategy to find the minimal spanning tree of a graph (such as the one provided)

as a hint a partial solution could be a list of edges currently included in the possible minimal spanning tree

(c) (6 marks) You have been shown a greedy strategy (Prim's Algorithm) that can solve the problem of minimal spanning trees. Explain how this works and, using the invariants of this algorithm, explain why it is always correct.

consider prim's algorithm uses an invariant that K edges are included and there are no cycles. What does that tell us?

(d) (2 marks) What is the minimal spanning tree in this graph?

the minimal spanning tree includes these edges:
<A,B>,<B,C>,<A,F>,<B,D>,<D,G>,<G,E>

(e) (3 marks) Explain what the difference in run time would be for each of these approaches (You do not have to give the complexity for these)

You will likely find that the backtracking avoids a lot of the work of the brute-force solution and the greedy approach is significantly less work than either of them.

Question 3: [14 marks]

This question is about classes of problems and solvability Consider the problem of graph colouring. In this problem you are given a graph of vertices and edges and you wish to assign a colour to each vertex so that it is different to the vertices adjacent to it. Consider the two graphs below:

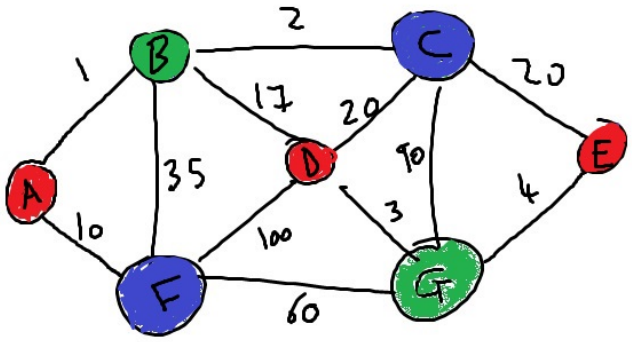


Figure 1: a three-coloured graph with seven vertices which have up to four edges (the same graph as question 2)

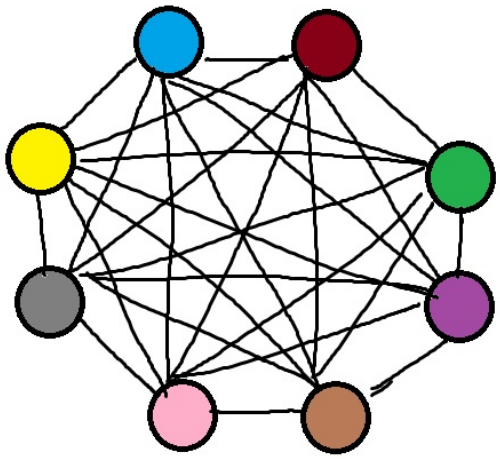


Figure 2: an eight-coloured graph with all eight vertices connected to every other vertex

figure 1 is three-colourable as it is possible to use just three colours to colour every vertex with no adjacent vertices sharing a colour. To do so, A, D and E are red, B and G are green and C and F are blue

figure 2 is eight-colourable as you can use eight different colours to have no colours with adjacent vertices sharing a colour. To do this every vertex has a different colour.

0

(a) (1 mark) What would be the decision problem variant of this problem?

basically, is this graph N-colourable for any given N

(b) (1 mark) What would be a certificate to this problem?

a possible colouring of the graph using N different colours

(c) (6 marks) Suggest a greedy approach to solving this problem

this involves keeping track of the set of colours and the current colours of neighbours. Pick a starting point and see how you go!

(d) (2 marks) Do you think your greedy approach will always work here? Why?

a careful choice of greedy approach should always work, but it will not be optimal

(e) (4 marks) Is the graph colouring problem a P or NP class of problem? Justify your answer.

First start by seeing if you can verify in P time, if so it's NP, if you can think of a P time way to solve it then it is also in P; it's okay if you can't think of a way to do these.

Question 4: [12 marks]

This question is about heaps and trees. Consider the problem of finding the ten lowest values in both a Binary Search Tree and a heap

(a) (4 marks) Suggest an algorithm for finding the lowest ten values in a binary search tree

> Note that you **cannot** assume these are all in a long line
> You should also consider what to do if there are not enough nodes in the tree

(b) (2 marks) Is your algorithm affected by the balance of the tree? In what way?

> If you are uncertain about this, try drawing yourself a few simple trees and compare a perfect tree against a highly unbalanced one.

(c) (4 marks) Suggest an algorithm for finding the lowest ten values in a minHeap

this should simply involve extracting from the min heap a bunch of times

(d) (2 marks) Which of these two algorithms do you think would have better time complexity? Why?

you should be able to answer this if you use your answer about the balance of the tree; hint: heaps are always balanced.

6

Question 5: [14 marks]

This question is about complexity in the context of sorting large lists of numbers.

(a) (4 marks) Explain the factors which affect the time taken by a counting sort

in this case you should consider the range of values that could occur in the list

(b) (4 marks) Explain the factors which affect the time taken by quicksort

in this case you should consider pivot chosen

(c) (4 marks) Explain the factors which affect the time taken by insertion sort

in this case you should consider the relative order of the initial list

(d) (2 marks) Why does complexity focus on large values of N?

Once a sufficiently large value of N is chosen, the highest order term becomes dominant and this allows us to compare algorithms based on their scalability, in smaller values of N the water could be muddied by a single expensive (but constant) operation done once at the start, rather than the behaviour of the algorithm on the whole.

14