MATH1005/MATH6005 Discrete Mathematical Models
Final Exam, Semester 1, 2022

“You are a rainbow lorikeet; the problems on this exam are seeds.” AP.

Problem	#1	#2	#3	#4	#5	Total
Score						
Out of	10	10	10	10	10	50

Throughout this exam, and as we did throughout the course, we write $\mathbb{N}$ for the set of positive integers.

Problem 1 (10 marks) (a) Use a truth table to prove or disprove the following logical equivalence:

$$((p \wedge q) \to r) \equiv (p \to (q \to r))$$

(b) Let $\mathcal{X}$ denote the set of all finite graphs. Consider the following statement:

> For all finite graphs G, if G is planar and has no loops, then the chromatic number of G is at most 4.

Write down the structural part of a proof that proceeds via the contrapositive.

Please note, some of the terms in this statement were not introduced in our course. You do not need to know what they mean in order to complete the question.

(c) Let $U = \{x, y, z\}$ and $S = \{(a, W) \in U \times \mathcal{P}(U) \mid a \notin W\}$. Use set-roster notation to describe S.

(d) Consider the following statement:

> For any universal set U and for any $A, B, C \in \mathcal{P}(U)$, we have $(A \setminus B) \setminus C = A \setminus (B \cup C)$.

Either: provide a counterexample to disprove the statement; or use an element proof, the definition of set operations and relations, and any of the logical equivalences below you may need to prove the statement.

Given any statement variables p, q, and r, a tautology t and a contradiction c, the following logical equivalences hold.

1. Commutative laws:	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. Associative laws:	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. Distributive laws:	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. Identity laws:	$p \wedge t \equiv p$	$p \vee c \equiv p$
5. Negation laws:	$p \vee \neg p \equiv t$	$p \wedge \neg p \equiv c$
6. Double negative law:	$\neg(\neg p) \equiv p$	
7. Idempotent laws:	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. Universal bound laws:	$p \vee t \equiv t$	$p \wedge c \equiv c$
9. De Morgan's laws:	$\neg(p \wedge q) \equiv \neg p \vee \neg q$	$\neg(p \vee q) \equiv \neg p \wedge \neg q$
10. Absorption laws:	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. Negations of t and c:	$\neg t \equiv c$	$\neg c \equiv t$

(e) Consider the following statement:

For any universal set U and for any $A, B \in \mathcal{P}(U)$, we have $\mathcal{P}(A \cup B) \subseteq \mathcal{P}(A) \cup \mathcal{P}(B)$.

Either: provide a counterexample to disprove the statement; or use an element proof, the definition of set operations and relations, and any of the logical equivalences below you may need to prove the statement.

Given any statement variables p, q, and r, a tautology t and a contradiction c, the following logical equivalences hold.

1. Commutative laws:	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. Associative laws:	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. Distributive laws:	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. Identity laws:	$p \wedge t \equiv p$	$p \vee c \equiv p$
5. Negation laws:	$p \vee \neg p \equiv t$	$p \wedge \neg p \equiv c$
6. Double negative law:	$\neg(\neg p) \equiv p$	
7. Idempotent laws:	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. Universal bound laws:	$p \vee t \equiv t$	$p \wedge c \equiv c$
9. De Morgan's laws:	$\neg(p \wedge q) \equiv \neg p \vee \neg q$	$\neg(p \vee q) \equiv \neg p \wedge \neg q$
10. Absorption laws:	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. Negations of t and c:	$\neg t \equiv c$	$\neg c \equiv t$

Problem 2 (10 marks) (a) Use induction to prove the following:

For all $n \in \mathbb{N}$, n is a non-negative integer power of 2 or n can be written as a sum of distinct non-negative integer powers of 2.

Note: By completing the problem above, you have proved that every positive integer can be represented by a bit string using binary positional notation.

(b) A certain bank has 31,700 customers. Each customer at the bank has chosen a 4-digit PIN to identify themselves at the ATM. Examples of PINs are 0678, 2189, 9834. Prove that there is at least one PIN shared by four or more customers.

(c) A FAW is a four letter 'word' whose letters are in alphabetical order. Letters are drawn from the standard lower case English 26-letter alphabet, and are allowed to repeat. A 'word' does not have to appear in any dictionary. Examples of FAWs are abcd, ccyz, aooy and yyyy. A FAW is said to contain a repeated letter if there is at least one letter which appears more than once in the word.

If a FAW is chosen at random, what is the probability that it contains a repeated letter? You may give you answer in the form of a mathematical expression (you do not have to compute a final number)

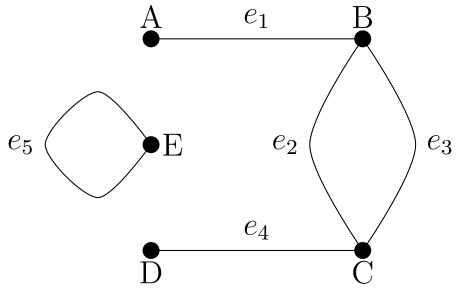
(d) A four-sided die is a tetrahedron with faces marked 1, 2, 3 and 4. When a four-sided die is thrown, one face is not visible—the number on the face that is not visible is said to be the 'number thrown.' Consider a probability experiment in which we toss a red four-sided die and a blue four-sided die simultaneously.

(i) Describe how we may represent outcomes so that outcomes are equally likely, and then use set-roster notation to describe the sample space of the experiment.

(ii) Let T be the random variable on S which assigns to each outcome the absolute value of the difference between the numbers thrown on the two dice. Let $E_1 = \{s \in S \mid T(s) > 1\}$ and let E_2 be the event that a 2 is thrown on the red die. Are the events E_1 and E_2 independent?

Problem 3 (10 marks) Enumerative Combinatorics and Graph Theory

(a) Let G be the graph shown below.



(i) How many different walks in G start at A and end at D? If possible, list them all.

(ii) How many different walks in G are paths which start at A and end at D? If possible, list them all.

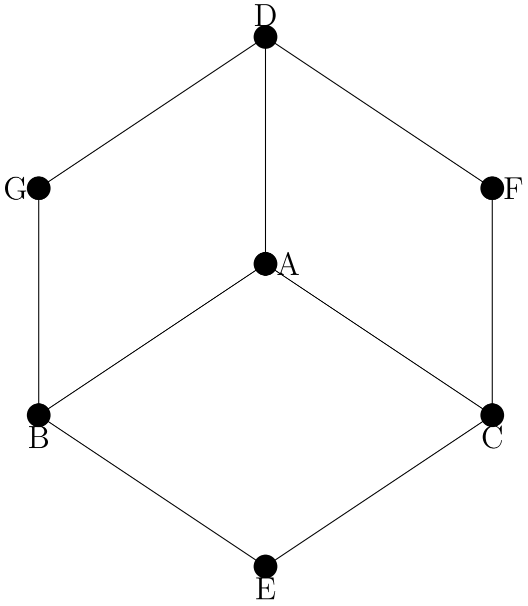
(iii) How many different walks in G are circuits (including the trivial circuits)? If possible, list them all.

(b) Hypercubes were introduced in the course as a family of graphs which may represent a good solution to which problem?

(c) (i) If G and H are graphs, what is an isomorphism between G and H?

(ii) Let $m, n \in \mathbb{N}$ be such that $m \neq n$, and let G be a graph that is isomorphic to the complete bipartite graph $K_{m,n}$. How many different isomorphisms are there between G and $K_{m,n}$? Justify your answer.

(d) Let W be the graph shown below.

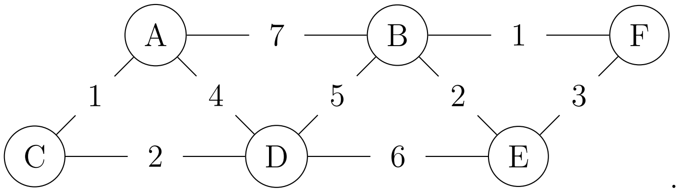


Prove or disprove the following: W has a Hamilton circuit.

Problem 4 (10 marks) (a) Describe the input and output to the Nearest Neighbour algorithm.

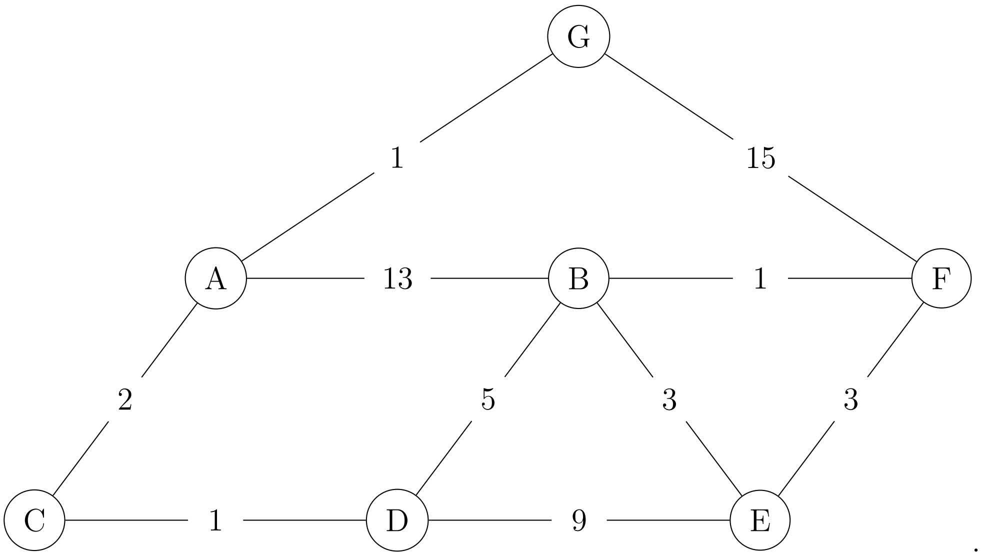
(b) Let G be a connected graph. State a condition on G that is both necessary and sufficient for Fleury's algorithm to be successful at identifying an Euler path in G.

(c) Let G be the weighted graph



Use one of the algorithms discussed in the course to find a minimal spanning tree for G. Write the name of the algorithm, and draw the minimal spanning tree it produces, in the space below.

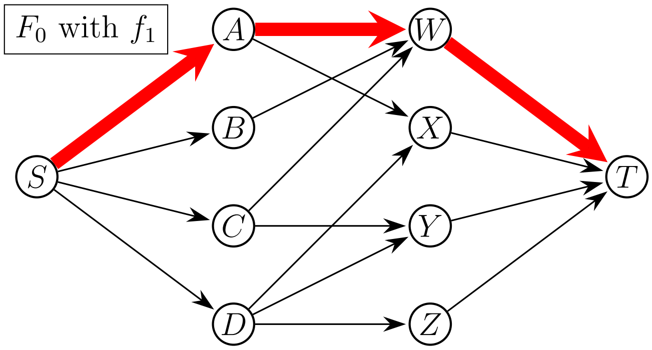
(d) Let H be the weighted graph



Use Dijkstra's algorithm to find the shortest path in H from A to E (pseudocode for the Dijkstra's algorithm is given at the end of the exam paper). To show that you applied the algorithm correctly, you should complete the table below

The vertices are 'locked in' in the following order	
The shortest path found is	

(e) A matching problem has been turned into a maximum flow problem for a transport network using the method described in the course. The resulting directed graph is shown below. Use the vertex labelling algorithm described in the course to solve the matching problem (pseudocode for the vertex-labeling algorithm is given at the end of the exam paper). The first incremental flow is shown on the diagram below and recorded in the first table below. Record each subsequent incremental flow in the first table below (use only as many rows as you need), and record the final matching in the second table.



incremental flow label	path of incremental flow
f_1	$S\ A\ W\ T$

vertex	The vertex, if any, with which it is matched
A	
B	
C	
D	

Problem 5 (10 marks) (a) A freelance computer network consultant, let's call her Yvonne, is employed in weekly contracts. Each week she is either: employed (E), unemployed (U) or training in new technology (T). Yvonne's records support the following assumptions:

- If she's employed this week, then next week she'll be employed with probability 0.80, and training in new technology with probability 0.05.
- If she's unemployed this week, then next week she'll be employed with probability 0.60 and training in new technology with probability 0.20.
- If she is training in new technology this week, then next week she'll be employed with probability 0.70.
- She never trains in new technology for two consecutive weeks.

We can model Yvonne's situation by a Markov process. Make a transition diagram to model Yvonne's situation.

(b) A corporation, BIG CORP INDUSTRIES, seeks to understand which employees perform tasks that are the most important to the corporation's operation. Every employee has submitted a response to the following survey question: "List the names of colleagues whose work is important to yours."

A 'PageRank-like approach' will be used to rank the importance of employees from this data.

(i) Describe a directed graph (What is the set of vertices? When is there an edge from one vertex to another?) related to the survey data that may play the role of a 'webgraph' in a PageRank-like approach to ranking the importance of employees.

(ii) State at least two hypotheses concerning the data and the importance of employees that, if assumed true, would justify the claim that a PageRank-like approach will be an effective way to rank the importance of employees.

(c) Let G be the webgraph with the adjacency matrix A shown below, and suppose that we are using the page rank algorithm with a damping factor of 80% (0.80) to rank the pages in G.

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(i) Draw a picture of G.

(ii) Write down the basic transition matrix T associated to G.

(iii) The modified transition matrix M associated to G is $M = \begin{bmatrix} \frac{3}{60} & \frac{27}{60} & \frac{27}{60} & \frac{3}{60} \\ \frac{27}{60} & \frac{3}{60} & \frac{3}{60} & \frac{27}{60} \\ \frac{3}{60} & \frac{51}{60} & \frac{3}{60} & \frac{3}{60} \\ \frac{19}{60} & \frac{19}{60} & \frac{19}{60} & \frac{3}{60} \end{bmatrix}$.

Your friend says the following: "The page rank vector associated to G is $\mathbf{v} = \begin{bmatrix} \frac{3}{10} \\ \frac{5}{10} \\ \frac{1}{10} \\ \frac{1}{10} \end{bmatrix}$."

In no more than three sentences, explain how you could determine whether or not your friend is correct.

THERE ARE NO MORE PROBLEMS. PSEUDOCODE FOR TWO ALGORITHMS IS ALL THAT FOLLOWS.

Dijkstra's Algorithm

Input:

1. Connected simple graph G. Vertices A, Z from G.
2. Distance function $\mathrm{dist}\colon E(G) \to \mathbb{Q}_+$.

Output:

1. Tree T containing A and Z as vertices.
   T is a subgraph of G.
   The unique path A→Z in T has minimal total distance of all paths A→Z in G.
2. 'Labelling' $L\colon V(T) \to \mathbb{Q}_+$; $L(v) = \mathrm{min.dist}(\mathrm{A} \to v)$.

Method:

```
1. Initialize the tree $T$:  Set $V(T) = \{\mathrm{A}\}$, $E(T) = \emptyset$.
2. Initialize a 'Marking' function $M\colon V(G) \to V(G) \cup \{\mathrm{blank}\}$:
   Set $M(v) = \mathrm{blank}$ for all $v \in G$.
3. Set $L(\mathrm{A}) = 0$. Set 'current vertex' $c$ to A.
While $c \neq \mathrm{Z}$:
    4. For each vertex $v$ adjacent to $c$ but not in $T$:
       If $v$ is unmarked (i.e. $M(v) = \mathrm{blank}$)
       or if $L(v) > L(c) + \mathrm{dist}(\{c, v\})$
       set $M(v) = c$, $L(v) = L(c) + \mathrm{dist}(\{c, v\})$.
    5. From all marked $v \in G \backslash T$ (i.e. $M(v) \neq \mathrm{blank}$ and $v \notin T$)
       (such $v$ are said to be 'on the fringe')
       select one, say $w$, with minimal $L(v)$.
    6. Insert vertex $w$ and edge $\{M(w), w\}$ into the tree $T$.
       (I call this "locking in" $w$ and its lead-in edge.)
    7. Update $c$ to $w$. (i.e. make $w$ the new current vertex.)
End of While Loop

End of Method
```

Vertex labelling algorithm for finding a maximum flow function for a transport network

Input: Transport network D with capacity function C.

Output: A maximum flow function $F_{\max}$ for the network.

Method: Initialise F to the zero flow F_0. Initialize i to 1.

For $i = 1, 2, \ldots$ carry out stage i below to attempt to build an incremental flow f_i.

If stage i succeeds, define $F_i = F_{i-1} + f_i$ and proceed to stage $i+1$.

If stage i fails, define $F_{\max} = F_{i-1}$ and stop.

Stage i:

1. If $i > 1$, mark up the amended edge flows for F_{i-1}.
2. Mark up the levels for F_{i-1}, as explained below.
3. If t is assigned a level, stage i will succeed, so continue.

   If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
   (a) Label each level 1 vertex v with sk_{v}, where $k_{\mathrm{v}} = S((s,v))$. (see below for definition of S)
   (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u, where
      - u is the alphabetically earliest level 1 vertex with $(u,v) \in E(D)$ and $S((u,v)) > 0$,
      - k_{v} is the minimum of $S((u,v))$ and the value part of u's label.

   (c) If t has level 3 or more now work through the level 3 vertices in a similar manner and so on.
5. Let p_i be the path $\mathrm{u}_0\mathrm{u}_1 \ldots \mathrm{u}_n$ where $\mathrm{u}_n = \mathrm{t}$ and for $0 < j \leq n$
   u_j has label $\mathrm{u}_{j-1}k_j$.

   Define f_i to be the incremental flow on p_i with flow value k_n.

End of Method

Levels and labels: At each stage of the vertex labelling algorithm levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.
- If x has level n and $S((x, y)) > 0$ then y has level $n + 1$ provided it has not already been assigned a lower level.

The **label** on a vertex v of level $n \geq 1$ has the form uk, where u is a vertex of level $n - 1$ and $(u, v) \in E(D)$ is an edge on the path for a potential incremental flow through v with flow value k.

The algorithm assigns labels in ascending order of levels, and in alphabetical order within levels.

The spare capacity function S
For vertices u,v of D, where D has capacity and flow functions C, F:

$$S((\mathrm{u,v})) = \begin{cases} C((\mathrm{u,v})) - F((\mathrm{u,v})) & \text{if (u,v)} \in E(D) \\ F((\mathrm{v,u})) & \text{if (v,u)} \in E(D) \\ 0 & \text{otherwise.} \end{cases}$$

When (v,u) $\in E(D)$, $S((\mathrm{u,v}))$ is called a **virtual capacity**.