Problem 1

For this question, let F denote the set of well-formed formulas over a set Prop of propositional variables.

(a) Show that the logical equivalence relation, ≡, is an equivalence relation on F.

To prove that ≡ is an equivalence relation, just show that it satisfy Reflexivity(R), Symmetry(S) and Transitivity(T)

(R): for any well-formed formula(wff) φ on F

$$v(\varphi) = v(\varphi) \text{ holds for all truth assignments } v$$

So φ ≡ φ holds, ≡ is reflexive

(S): for any wff φ and ψ on F; if φ ≡ ψ holds, then:

$$v(\varphi) = v(\psi) \text{ holds for all truth assignments } v$$

So v(ψ) = v(φ) holds for all truth assignments v

Thus ψ ≡ φ, the relation ≡ is symmetric.

(T): for any wff φ, ϕ and ψ on F;

If φ ≡ ϕ and ϕ ≡ ψ holds, then:

$$v(\varphi) = v(\phi) \text{ and}$$
$$v(\phi) = v(\psi) \text{ holds}$$

Then v(φ) = v(ψ) holds for all truth assignments v

Thus φ ≡ ψ, the relation ≡ is Transitive.

In conclusion, ≡ is

(b) List four elements in [⊥], the equivalence class of ⊥.

the elements ϕ in [⊥], equivalence class of ⊥ must satisfy that for all truth assignments:

$$v(\varphi) = \text{False}$$

listing 4 possible solutions as below:

(1) $(\bot\wedge\bot)$

(2) $(\bot\wedge\phi)$ (ϕ is any wff on F)

(3) $(\bot\vee\bot)$

(4) $((\bot\wedge\bot)\wedge\bot)$

(c) For all $\phi, \phi', \psi, \psi' \in F$ with $\phi \equiv \phi'$ and $\psi \equiv \psi'$; show that:

(i) $\neg\phi \equiv \neg\phi'$

Since $\phi \equiv \phi'$ then:

$$(\phi \leftrightarrow \phi') \text{ is tautology}$$

Thus for all truth assignments

$$v(\phi \leftrightarrow \phi') = \text{True holds}$$

Then ϕ and ϕ' are either both true or both false, listing all possible truth assignments as below:

ϕ	ϕ'	$\neg\phi$	$\neg\phi'$	$\phi\leftrightarrow\phi'$	$\neg\phi\leftrightarrow\neg\phi'$
T	T	F	F	T	T
F	F	T	T	T	T

Thus $(\neg\phi\leftrightarrow\neg\phi')$ is True for all truth assignments and it is a tautology, $\neg\phi \equiv \neg\phi'$ holds

(ii) $\phi \wedge \psi \equiv \phi' \wedge \psi'$

Since $\phi \equiv \phi'$ and $\psi \equiv \psi'$ then:

$$\text{Both } (\phi \leftrightarrow \phi') \text{ and } (\psi \leftrightarrow \psi') \text{ are tautology}$$

Similarly, listing all possible truth assignments as below, $\phi(\psi)$ and ϕ' (ψ') are either both true or both false:

ϕ	ϕ'	ψ	ψ'	$\phi \wedge \psi$	$\phi' \wedge \psi'$	$((\phi' \wedge \psi) \leftrightarrow (\phi' \wedge \psi'))$
T	T	T	T	T	T	T
T	T	F	F	F	F	T
F	F	T	T	F	F	T
F	F	F	F	F	F	T

Thus $((\phi' \wedge \psi)\leftrightarrow(\phi' \wedge \psi'))$ is True for all truth assignments. So it is a tautology and $\phi \wedge \psi \equiv \phi' \wedge \psi'$ holds.

(iii) $\phi \vee \psi \equiv \phi' \vee \psi'$

Similarly, listing all possible truth assignments as below, $\phi(\psi)$ and ϕ' (ψ') are either both true or both false:

ϕ	ϕ'	ψ	ψ'	$\phi \vee \psi$	$\phi' \vee \psi'$	$((\phi' \vee \psi) \leftrightarrow (\phi' \vee \psi'))$
T	T	T	T	T	T	T
T	T	F	F	T	T	T
F	F	T	T	T	T	T
F	F	F	F	F	F	T

Thus $((\phi' \vee \psi)\leftrightarrow(\phi' \vee \psi'))$ is True for all truth assignments. So it is a tautology and $\phi \vee \psi \equiv \phi' \vee \psi'$ holds.

(d) Show that $F_{\equiv}$ together with the operations defined above forms a Boolean Algebra. Note: you will have to give a suitable definition of a zero element and a one element in $F_{\equiv}$.

Defining a zero element and a one element:

For any equivalence class [ϕ] on $F_{\equiv}$, according to the complementation of Boolean Algebra:

$$x \vee x' = 1$$

$$x \wedge x' = 0$$

Then $$[\phi] \vee [\neg\phi] = 1$$

$$[\phi \vee \neg\phi] = [\top]$$

Thus $$1 = [\top]$$

Also $$[\phi] \wedge [\neg\phi] = 0$$

$$[\phi \wedge \neg\phi] = [\bot]$$

$$0 = [\bot]$$

Proof:

To show that $F_{\equiv}$ together with the operations forms a Boolean Algebra, we need to prove laws below

1) For any [x], [y] on $F_{\equiv}$, as $[x], [y] \in F_{\equiv}$, it can be proved that:

$$[x] \vee [y] \in F_{\equiv}$$

$$[x] \wedge [y] \in F_{\equiv}$$

$$[\neg x] \in F_{\equiv}$$

2) commutativity: For any [x], [y] on $F_{\equiv}$

$$\begin{aligned}[x] \vee [y] &= [x \vee y] \\ &= [y \vee x] \\ &= [y] \vee [x]\end{aligned}$$

Similarly, $[x] \wedge [y] = [y] \wedge [x]$

3) association: For any [x], [y], [z] on $F_{\equiv}$

$$\begin{aligned}([x] \vee [y]) \vee [z] &= [x \vee y] \vee [z] \\ &= [(x \vee y) \vee z] \\ &= [x \vee (y \vee z)] \\ &= [x] \vee [y \vee z] \\ &= [x] \vee ([y] \vee [z])\end{aligned}$$

Similarly, $([x] \wedge [y]) \wedge [z] = [x] \wedge ([y] \wedge [z])$

4) distributivity: For any [x], [y], [z] on $F_{\equiv}$

$$
\begin{aligned}
[x] \vee ([y] \wedge [z]) &= [x] \vee [y \wedge z] \\
&= [x \vee (y \wedge z)] \\
&= [(x \vee y) \wedge (x \vee z)] \\
&= [x \vee y] \wedge [x \vee z] \\
&= ([x] \vee [y]) \wedge ([x] \vee [z])
\end{aligned}
$$

similarly, $[x] \wedge ([y] \vee [z]) = ([x] \wedge [y]) \vee ([x] \wedge [z])$

5) identity: For any [x] on $F_{\equiv}$

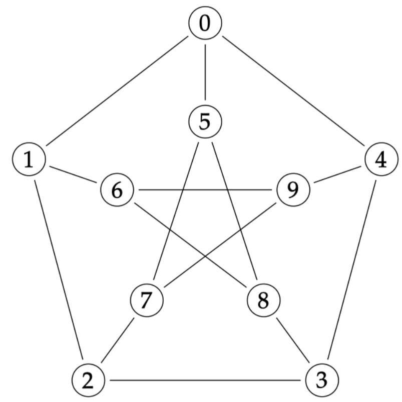
$$
\begin{aligned}
[x] \vee 0 &= [x] \vee [\bot] \\
&= [x \vee \bot] \\
&= [x] \\
[x] \wedge 1 &= [x] \wedge [\top] \\
&= [x \wedge \top] \\
&= [x]
\end{aligned}
$$

6) complementation:

since complementation is used to define 1 and 0, so it has been proved.

Problem 2

This is the Petersen graph:



(a) Show that the Petersen graph not contains a subdivision of K_5.

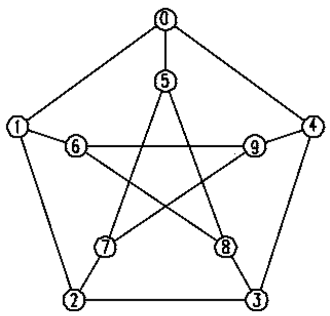
In the Petersen graph, the degree of each vertex $\deg(v_p) = 3$.

For K_5, the degree of each vertex $\deg(v_{k5}) = 4$.

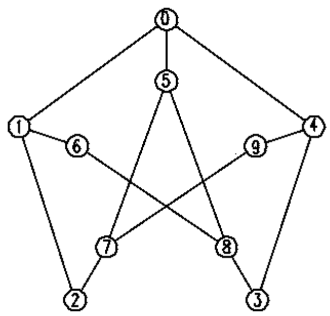
According to the specification of subdivision, the degree of a vertex cannot increase with subdivision.

Hence, Petersen graph not contains a subdivision of K_5.

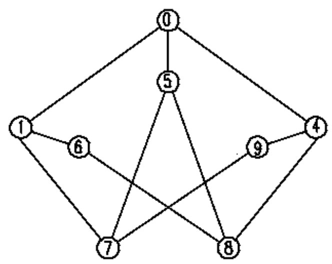
(b) Show that the Petersen graph contains a subdivision of $K_{3,3}$.



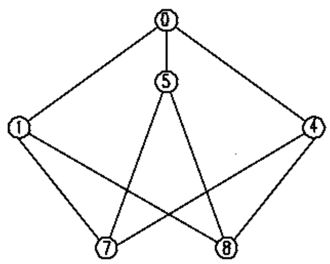
graph_1



graph_2



graph_3



graph_4

Starting from the Petersen graph as graph_1 shows and follow steps below can get a $K_{3,3}$

1. remove the edges 6 - 9 and 2 - 3, then we get the graph_2
2. remove the vertices 2 and 3, and replace them with edges 1 – 7 and 4 – 8, then we get graph_3
3. remove the vertices 6 and 9, and replace them with edges 1 – 8 and 4 – 7. Then the remained vertices and edges form a $K_{3,3}$ as graph_4 shows.

Problem 3

Harry would like to take each of the following subjects: Defence against the Dark Arts; Potions; Herbology; Transfiguration; and Charms. Unfortunately some of the classes clash, meaning Harry cannot take them both. The list of clashes are:

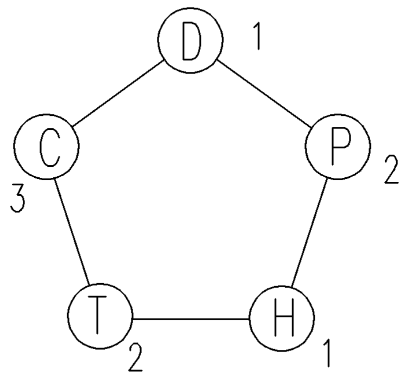
- **Defence against the Dark Arts clashes with Potions and Charms**
- **Potions also clashes with Herbology**
- **Herbology also clashes with Transfiguration, and**
- **Transfiguration also clashes with Charms.**

Harry would like to know the maximum number of classes he can take.

(a) Model this as a graph problem. Remember to:

(i) Clearly define the vertices and edges of your graph.

model the problem with a graph G(V, E) as below:



* vertex D stands for the course Defence against the Dark Arts
* vertex C stands for the course Charms
* vertex T stands for the course Transfiguration
* vertex H stands for the course Herbology
* vertex P stands for the course Potions
* 1, 2, 3 stands for different colors assigned to each vertex

For the graph G(V, E), vertices v(G) represent different courses, edges e(G) represent the clash of different courses

(ii) State the associated graph problem that you need to solve.

The graph problem is to give the max number of the vertices with same color

(b) Give the solution to the graph problem corresponding to this scenario; and solve Harry's problem.

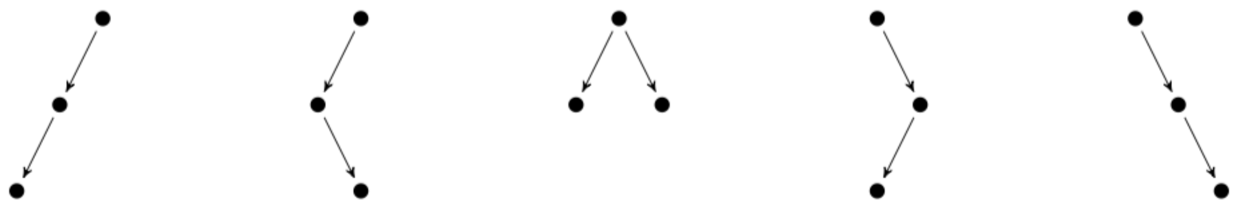
Assigning colors(1, 2, 3) to each vertex of the graph, then the max number of the vertices with same colors is 2 thus:

Harry can take 2 courses at most.

Problem 4

Recall from Assignment 2 the definition of a binary tree data structure: either an empty tree, or a node with two children that are trees.

Let T (n) denote the number of binary trees with n nodes. For example T (3) = 5 because there are five binary trees with three nodes:



(a) Using the recursive definition of a binary tree structure, or otherwise, derive a recurrence equation for T(n).

According to the definition of binary trees, for T(n) where n < 2(base case):

Since empty tree(n = 0) has no node and when n = 1, the tree has no children, then there is only one possibility:

$$T(1) = 1$$

$$T(0) = 1$$

Starting from the node at the top(the root node), placing the remained nodes(n - 1) in its two branches(left and right):(inductive case)

When n ⩾ 2, Giving a defining for T(l, r) where l+r = n-1 such that:

$$T(l,r) = T(l) \cdot T(r)$$

Which means a situation when put *l* nodes at the left branch of the top node and *r* nodes at the right branch of the top node. Then T(n):

$$T(n) = T(0, n-1) + T(1, n-2) + T(2, n-3) + \cdots + T(l,r) + \ldots + T(n-1, 0)$$

$$\mathrm{T(n)} = \sum_{i=0}^{n-1} \mathrm{T(i, n-i-1)}$$

$$\mathrm{T(n)} = \sum_{i=0}^{n-1} \mathrm{T(i)} \cdot \mathrm{T(\,n-i-1)}$$

A full binary tree is a non-empty binary tree where every node has either two non-empty children (i.e. is a fully-internal node) or two empty children (i.e. is a leaf).

(b) Using observations from Assignment 2, or otherwise, explain why a full binary tree must have an odd number of nodes.

Base case: when $n = 1$, according to the definition, a single node is a non-empty tree with two empty children. Thus it's a full binary tree.

Inductive case:

Assuming that a full binary tree has odd numbers $(2k + 1)$ of nodes. Then in order to create a new full binary tree, we need to add new nodes on it.

According to the definition of full binary tree, we can only add either $2j$ leaves or empty children on the previous tree.

Then the number of newly-added nodes is even $2j$.

Thus the total number of the nodes of new full binary tree is:

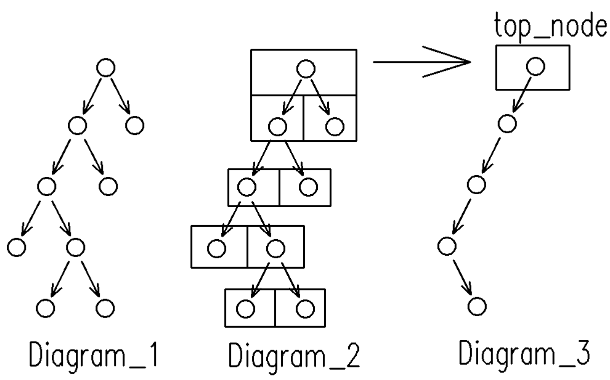
$$T(n) = 2j + 2k + 1$$
$$= 2(j + k) + 1 \ (j, k \geqslant 0),$$

which is odd. So a full binary tree has odd number of nodes

(c) Let B(n) denote the number of full binary trees with n nodes. Derive an expression for B(n), involving of T(n') where n' ≤ n. Hint: Relate the internal nodes of a full binary tree to T(n).

From (b), we have proved that a full binary tree must have odd number of nodes, so just focus on the cases where the number of nodes is odd.

Except the top node is single, every pair of children of a node can be regarded as a whole node, as the diagram shows below:



Here, when the number of nodes is 9, single out the top node and pair children of each node. Then each pair can be regarded as a new node. And the position connected with children correspond to the left child and right child of a common node. There are 5 of them together with the top node. Then the question becomes counting the number of binary tree when there are 5 nodes. And similarly, for common cases, here is the formula:

$$T(n') = B(2n+1)$$

A well-formed formula is in Negated normal form if it consists of just ∧, ∨, and literals (i.e. propositional variables or negations of propositional variables). That is, a formula that results after two steps of the process for transforming a formula into a logically equivalent one. For example, p ∨ (¬q ∧ ¬r) is in negated normal form; but p ∨ ¬(q ∨ r) is not.
Let F(n) denote the number of well-formed, negated normal form formulas there are that use precisely n propositional variables exactly one time each. So F (1) = 2, F (2) = 16, and F (4) = 15360.
(d) Using your answer for part (c), give an expression for F(n).

Recalling the definition of the parse tree and according to the specification, the parse tree of a well-formed formula in Negated normal form is a full binary tree with n leaves node(1).
According to Assignment 2 problem 3 (e):

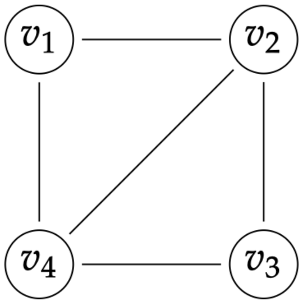
$$\text{leaves}(T) = \text{internal}(T) + 1$$

Then the number of internal nodes is (n -1)
Thus the total number of nodes is (2n - 1)
(1) From (c), we have proved that the number of binary is B(2n -1) = T(n-1)
(2) For each internal node, it represents either ∧ or ∨, then there are (2^{n-1}) posibilities
(3) Then foucusing on the leaves nodes, as there are n positions and n distinguishable variables, the possibilities is (n!)
(4) For each leaf, there may be a ‘ ¬ ’ before it or not, so there are 2^n possibilities.
combining (1) to (4) above:

$$F(n) = T(n-1) \cdot 2^{n-1} \cdot n! \cdot 2^n$$

Problem 5

Consider the following graph:



and consider the following process:

- **Initially, start at v_1.**
- **At each time step, choose one of the vertices adjacent to your current location uniformly at random, and move there.**

Let $p_1(n)$, $p_2(n)$, $p_3(n)$, $p_4(n)$ be the probability your location after n time steps is v_1, v_2 , v_3 , or v_4 respectively. So $P_1(0) = 1$ and $P_2(0) = P_3(0) = P_4(0) = 0$.

(a) Express $P_1(n + 1)$, $P_2(n + 1)$, $P_3(n + 1)$, and $P_4(n + 1)$ in terms of $P_1(n)$, $P_2(n)$, $P_3(n)$, and $P_4(n)$.

According to the specification and the graph, if the location after n+1 step is on v_1, then the location of the n step must be v_2 or v_4. In v_2, the possibility of choosing $v_2 – v_1$ as next step is 1/3. Similarly, in v_4, the possibility of choosing $v_4 – v_1$ as next step is 1/3. Then:

$$P_1(n + 1) = \frac{1}{3}P_2(n) + \frac{1}{3}P_4(n)$$

Similarly:

$$P_2(n + 1) = \frac{1}{3}P_4(n) + \frac{1}{2}P_1(n) + \frac{1}{2}P_3(n)$$

$$P_3(n + 1) = \frac{1}{3}P_2(n) + \frac{1}{3}P_4(n)$$

$$P_4(n + 1) = \frac{1}{3}P_2(n) + \frac{1}{2}P_1 1(n) + \frac{1}{2}P_3(n)$$

(b) As n gets larger, each $P_i(n)$ converges to a single value (called the steady state) which can be determined by setting $P_i(n + 1) = p_i(n)$ in the above equations. Determine the steady state probabilities for all vertices.

As $P_i(n)$ converge to a single value, the sum of them should be 1:

$$1 = P_1(n) + P_2(n) + P_3(n) + P_4(n)$$

Since $P_i(n+1) = P_i(n)$, combine with (a), we get the equations below

$$\begin{cases} 1 = P_1(n) + P_2(n) + P_3(n) + P_4(n) \\ P_1(n) = \frac{1}{3}P_2(n) + \frac{1}{3}P_4(n) \\ P_2(n) = \frac{1}{3}P_4(n) + \frac{1}{2}P_1(n) + \frac{1}{2}P_3(n) \\ P_3(n) = \frac{1}{3}P_2(n) + \frac{1}{3}P_4(n) \\ P_4(n) = \frac{1}{3}P_2(n) + \frac{1}{2}P_1(n) + \frac{1}{2}P_3(n) \end{cases}$$

Solving the equations then we get the values

$$\begin{cases} P_1(n) = \frac{1}{5} \\ P_2(n) = \frac{3}{10} \\ P_3(n) = \frac{1}{5} \\ P_4(n) = \frac{3}{10} \end{cases}$$

(c) The distance between any two vertices is the length of the shortest path between them. What is your expected distance from v_1 in the steady state?

Considering the distance between v_1 and other vertices:

Assuming that the length of each edge is 1, then the **length of the shortest path between v_1** and other vertices:

$$d(v_2) = 1, \quad d(v_3) = 2, \quad d(v_4) = 1$$

Then the expected distance:

$$d = d(v2) \cdot P2(n) + d(v3) \cdot P3(n) + d(v4) \cdot P4(n) = 1 \times \frac{3}{10} + 2 \times \frac{1}{5} + 1 \times \frac{3}{10} = 1$$