richardwu.ca

Math 239 Final Exam Guide

Introduction to Combinatorics

Kevin Purbhoo • Spring 2017 • University of Waterloo

Last Revision: January 9, 2018

Table of Contents

Abstract

These notes are intended as a resource for myself; past, present, or future students of this course, and anyone interested in the material. The goal is to provide an end-to-end resource that covers all material discussed in the course displayed in an organized manner. If you spot any errors or would like to contribute, please contact me directly.

1 Combinatorial Analysis

1.1 Union, Catersian Product and Power

Union $A \cup B$ If A and B are disjoint, then

$$|A \cup B| = |A| + |B|$$

Cartesian product $A \times B$ Note that

$$A \times B = \{(a, b) : a \in A, b \in B\}$$

and the following holds

$$|A \times B| = |A||B|$$

Cartesian power A^k As an extension to the Cartesian product

$$A^k = \{(a_1, \ldots, a_k) : a_i \in A, \forall i\}$$

the following holds

$$|A^k| = |A|^k$$

Note $A^0 = \{()\}$ (set with one empty tuple).

1.2 Binomial Coefficient

Given n objects, you want to choose/select k objects without replacement. Order does not matter. The number of ways to do this is

$$\binom{n}{k} = \frac{n(n-1)\ldots(n-k+1)}{k!} = \binom{n}{n-k}$$

1.3 Bijections

A bijection $f : S \to T$ holds if

f **is injective** Also called **1-1**, every element in S can only map to one element in T (vertical line test). Formally

$$f(x_1) = f(x_2) \iff x_1 = x_2$$

f **is surjective** Also called **onto**, every element in T is mapped to by an element in S (the codomain T is equivalent to the range of f). Formally

$$\forall t \in T, \exists s \in S : f(s) = t$$

The trick to show a bijection between two sets is to show for $f : S \to T$ a unique way of mapping each element $s \in S$ to an element $t \in T$, and similar $f^{-1} : T \to S$ (show **1-1** both ways).

1.4 Combinatorial Proofs

To show that an equation with combinatorial arguments holds, one can formulate a set of objects and show two ways of counting: each way of counting corresponds to one side of each equation.
One can proof combinatorial equalities by algebraic manipulation if permitted.

1.5 Series Identities

The Binomial Series identities can be derived using Taylor Series expansion about $x = 0$

Binomial Series

$$(x+y)^n = \sum_{i=0}^{n} \binom{n}{i} x^i y^{n-i}$$

where $n \in \mathbb{N}$

Negative Binomial Series

$$(1-x)^{-n} = \sum_{i=0}^{n} \binom{n+i-1}{i} x^i$$

Geometric Series identities are useful in simplifying generating series

Finite Geometric Series

$$a + ar + ar^2 + \ldots + ar^{n-1} = \frac{a(1-r^n)}{1-r}$$

Infinite Geometric Series

$$a + ar + ar^2 + \ldots = a\sum_{i \geq 0} r^i = \frac{a}{1-r}$$

1.6 Compositions

A composition of n is an *ordered* sequence $(\alpha_1, \ldots, \alpha_k)$ of k positive integers where

$$\alpha_1 + \ldots + \alpha_k = n$$

1.7 Generating Series

Generating series lets us abstract counting the number of ways to choose a **configuration** $\sigma \in S$ into a nice, algebraic form.
A generating series for a set of configurations S follows the form

$$\Phi_S(x) = \sum_{\sigma \in S} x^{w(\sigma)}$$

where $w(\sigma)$ is the **weight function** for each configuration σ. Think of the weight function as a way of quantifying the value we're interested in each configuration For a composition of k parts (σ), we are interested in the composition value or the sum of its parts. Thus if $\sigma = (\alpha_1, \ldots, \alpha_k)$

$$w(\sigma) = \alpha_1 + \ldots + \alpha_k$$

Some useful identities arise when we set $x = 1$:

$\Phi_S(1) = |S|$

$$\Phi_S(1) = \sum_{\sigma \in S} = 1^{w(\sigma)} = |S|$$

$\Phi'_S(1) =$ **sum of weights**

$$\Phi'_S(1) = \sum_{\sigma \in S} = w(\sigma) 1^{w(\sigma)-1}$$

$\Phi'_S(1)/\Phi_S(1)$ **average weight** Follows from above.

1.8 Sum and Product Lemma (Generating Series)

Sum Lemma Suppose $S = A \cup B$, A and B are disjoint. Then

$$\Phi_S(x) = \Phi_A(x) + \Phi_B(x)$$

Product Lemma Suppose $S = A \times B$. Then

$$\Phi_S(x) = \Phi_A(x) \cdot \Phi_B(x)$$

1.9 11 Steps to Generating Series Problems

To solve a generating series problem, we follow 11 steps:

Example 1.1. How many compositions of n of k parts?

1. Do you even need generating series?

   Example 1.2. Yes.

2. Identify parameters in problem and constants to be treated as parameters.

   Example 1.3. n and k

3. Define the set of configurations S by removing one of the parameters.

   Example 1.4. Remove n, focus on k parts. Let S be compositions of k parts.

4. Define S in terms of simpler unions and Cartesian products.

   Example 1.5. $S = (\mathbb{N}_{\geq 1})^k$, where $\mathbb{N}_{\geq 1} = \{1, 2, 3, \ldots\}$

5. Reintroduce removed parameter as weight function $w(\sigma)$ on S.

   Example 1.6. n: compositions that have weight n. Thus for $\sigma = (\alpha_1, \ldots, \alpha_k)$ we define $w(\sigma) = \alpha_1 + \ldots + \alpha_k$ or the sum of its parts.

6. Define weight function on simpler sets.

   Example 1.7. $\alpha \in \mathbb{N}_{\geq 1}$ is a positive integer. Thus $w(\alpha) = \alpha$.

7. Check weight functions behave correctly for product lemma.

Example 1.8. Note for any $\sigma = (\alpha_1, \ldots, \alpha_k)$, the following holds

$$w(\sigma) = w_{\mathbb{N}_{\geq 1}}(\alpha_1) + \ldots + w_{\mathbb{N}_{\geq 1}}(\alpha_k)$$

8. Compute generating series of simpler sets.

   Example 1.9.
   $$\Phi_{\mathbb{N}_{\geq 1}}(x) = \sum_{\alpha \in \mathbb{N}_{\geq 1}} x^{w_{\mathbb{N}_{\geq 1}}(\alpha)} = \sum_{\alpha \in \mathbb{N}_{\geq 1}} x^{\alpha} = x + x^2 + x^3 + \ldots$$

9. Use Sum and Product Lemma to get formula for $\Phi_S(x)$.

   Example 1.10.
   $$\Phi_S(x) = \sum_{\sigma \in S} (\Phi_{\mathbb{N}_{\geq 1}}(x))^k = (x + x^2 + x^3 + \ldots)^k$$

10. Simplify $\Phi_S(x)$.

    Example 1.11.
    $$\Phi_S(x) = (x + x^2 + x^3 + \ldots)^k = (\frac{x}{1-x})^k = x^k(1-x)^{-k}$$
    by the infinite geometric series.

11. Answer is the coefficient of x^n or $[x^n]\Phi_S(x)$ where n is your removed parameter.

    Example 1.12.
    $$[x^n]\Phi_S(x) = [x^n]x^k(1-x)^{-k} = [x^{n-k}](1-x)^{-k}$$
    Using the Negative Binomial series, $(1-x)^{-k} = \sum_{i=1} \binom{k+i-1}{k-1} x^i$. Note $i = n - k$ thus the coefficient is $\binom{k+(n-k)-1}{k-1} = \binom{n-1}{k-1}$ ways.

1.10 Formal Power Series (FPS)

We can **add/subtract** and **multiply** power series $A(x) = a_0 + a_1 x + \ldots$ and $B(x) = b_0 + b_1 x + \ldots$ for rational numbers a_i, b_i.

Addition
$$A(x) + B(x) = \sum_{i \geq 0} (a_i + b_i) x^i$$

Multiplication
$$A(x)B(x) = \sum_{j \geq 0} (\sum_{i=0}^{j} a_i b_{j-i}) x^j$$

Note both FPS $A(x), B(x)$ must start at the same index. One can use variable substitution to align them.

2 Counting Binary Strings

A binary string is a string consisting of only **0's and 1's**. In general, we are interested in all binary strings that match a given description of length n.
We will eventually use generating series to count binary strings.

2.1 Concatenations

For two given sets of binary strings A, B, we define the concatenation as

$$AB = \{ab : a \in A, b \in B\}$$

Example 2.1. So for $A = \{10, 0\}, B = \{01, 1\}$

$$AB = \{1001, 101, 001, 01\}$$

There is also the * operator on a set of binary strings A where

$$\begin{aligned} A^* &= \{\epsilon\} \cup A \cup AA \cup AAA \cup \dots \\ &= \{\epsilon\} \cup A \cup A^2 \cup A^3 \cup \dots \end{aligned}$$

or it is the concatenation of any number of strings in A. ϵ is the empty string.
Note $\{0, 1\}^*$ is the set of all binary strings.

2.2 Unambiguous Expressions

With concantenation of sets AB we may end up with duplicate elements from two different configurations.

Example 2.2. With $A = \{0, 01\}, B = \{11, 1\}$, note

$$AB = \{011, 01, 0111\}$$

where 011 could come from either $0 \in A, 11 \in B$ or $01 \in A, 1 \in B$.

The above expression is **ambiguous**. Formally, an expression is **unambiguous** *if and only if*

$$|AB| = |A \times B|$$

In general, we prefer unambiguous expressions since we can apply the Sum and Product Lemmas to them for generating series.

2.3 0 and 1-Decompositions

For 0-decompositions, we express our set of binary strings S by fixing our 0 characters in the string. Similarly we can do the same for 1 characters with 1-decompositions.

Example 2.3. To express all binary strings, we imagine fixing every 0 character ($\{0\}$). 0 or more 1 characters can occur before each 0 ($\{1\}^*$). There may be 0 or more of these 0 preceded by 1s blocks (($\{1\}^*\{0\}$)*). Since we delimited the expression with the 0 always at the end of string, we need to account for binary strings that end with 1s ($\{1\}^*$). Thus we get the final exprssion

$$(\{1\}^*\{0\})^*\{1\}^*$$

This is called the 0-decomposition. Similarly for 1-decomposition

$$(\{0\}^*\{1\})^*\{0\}^*$$

These two expressions are unambiguous by construction.

2.4 Block Decompositions

A **block** is a maximal substring of 0s or 1s. That is, they are the longest contiguous sections composed of the same character of the binary strings. For the string 000110001000, we separate it out into its blocks

$$000|11|000|1|000$$

Similar to 0 and 1-decompositions, we can decompose binary strings by fixing the blocks of 0s (and similarly 1s). A block of 0s can be expressed as $\{0\}\{0\}^*$ (or $\{0\}^*\{0\}$).

Example 2.4. To express the set of all binary strings, we fix every block of 0s. If we follow the same procedure as we did for 0-decompositions, every block of 0s can be preceded by 1s. Note however that the following is ambiguous

$$(\{1\}^*\{0\}\{0\}^*)^*$$

since we need to deliminate the blocks of 0s with a non-empty character (inside the parentheses). For example, 000 is ambiguously constructed in more than 1 way: $(\{\epsilon\}\{0\})(\{\epsilon\}\{00\})$ or $(\{\epsilon\}\{00\})(\{\epsilon\}\{0\})$.
Instead of just 0 or more 1s, we associate each block of 0s by preceding them with a block of 1s. Thus we get the final unambiguous expression for block decomposition

$$\{0\}^*(\{1\}\{1\}^*\{0\}\{0\}^*)^*\{1\}^*$$

Note that we added an extra $\{0\}^*$ and $\{1\}^*$ before and after the expression, respectively. This is necessary since we mandated that the expression in the parenthese begins with a 1-block and end with a 0-block, whereas binary strings can start and end with 0s and 1s, respectively.
Note we can swap the 1s and 0s to get another expression for block decomposition.

2.5 Subsets of Decompositions

Note expressions that are either subsets of the 0 or 1-decompositions or the block decompositions are unambiguous. That is

$$(\{1, 11, 111\}\{0\})^*\{1\}^*$$

is unambiguous since it is a subset of the 0-decomposition where $\{1, 11, 111\} \subset \{1\}^*$.

2.6 Generating Series for Binary Strings

In general, the weight function $w(\sigma)$ of a binary string σ can defined as its length. For example, $w(010) = 3$.
Given that a binary string expression for a set of binary strings S is **unambiguous**, we can apply the Sum and Product Lemmas to find the generating series $\Phi_S(x)$:

Sum Lemma For $S = A \cup B$ (e.g. $\{1, 11\}$ where $A = \{1\}$ and $B = \{11\}$), A and B are disjoint, we have

$$\Phi_S(x) = \Phi_A(x) + \Phi_B(x)$$

Product Lemma For $S = AB$ (i.e concatenation) and given the expression is unambiguous, we have

$$\Phi_S(x) = \Phi_A(x)\Phi_B(x)$$

"Power" Lemma For $S = A^*$, we have

$$\Phi_S(x) = (1 - \Phi_A(x))^{-1}$$

a direct result of the Sum Lemma and the infinite geometric series.

Example 2.5. Express and simplify the generating series for the block decomposition expression $S = \{1\}^*(\{0\}\{0\}^*\{1\}\{1\}^*)^*$. Note the expression is unambiguous (it is the block decomposition of all binary strings). Use the Sum, Product, and "Power" Lemmas:

$$\begin{aligned}
\Phi_S(x) &= \Phi_{\{1\}^*(\{0\}\{0\}^*\{1\}\{1\}^*)^*\{0\}^*}(x) \\
&= \Phi_{\{1\}^*}(x) \cdot \Phi_{(\{0\}\{0\}^*\{1\}\{1\}^*)^*}(x) \cdot \Phi_{\{0\}^*}(x) \\
&= (1 - \Phi_{\{1\}}(x))^{-1} \cdot (1 - \Phi_{\{0\}\{0\}^*\{1\}\{1\}^*}(x))^{-1} \cdot (1 - \Phi_{\{0\}}(x))^{-1} \\
&= (1-x)^{-1} \cdot (1 - \frac{\Phi_{\{0\}}(x)\Phi_{\{1\}}(x)}{(1 - \Phi_{\{0\}}(x))(1 - \Phi_{\{1\}}(x))})^{-1} \cdot (1-x)^{-1} \\
&= (1-x)^{-2} \cdot (1 - \frac{x^2}{(1-x)^2})^{-1} \\
&= \frac{1}{(1-x)^2} \cdot \frac{1}{1 - \frac{x^2}{(1-x)^2}} \\
&= \frac{1}{(1-x)^2 - x^2} \\
&= \frac{1}{1-2x}
\end{aligned}$$

For a quick sanity check, by the negative binomial theorem

$$\begin{aligned}
(1-2x)^{-1} &= \sum_{i \geq 0} \binom{1+i-1}{i} (2x)^i \\
&= \sum_{i \geq 0} 2^i x^i \\
\therefore [x^n](1-2x)^{-1} &= 2^n
\end{aligned}$$

which aligns with our intuition about the number of binary strings of length n.

3 Coefficients of Recurrence Relations

Note for a rational function $C(x)$, we can represent it as a power series

$$C(x) = \frac{f(x)}{g(x)} = \sum_{n=0} a_n x^n$$

Note the coefficients a_n can be represented using just $g(x)$ given $deg(f) < deg(g)$. For

$$g(x) = 1 + q_1 x^1 + \ldots + q_k x^k$$

the recurrence relation $C(x)$ satisfies is

$$a_n + q_1 a_{n-1} + \ldots + q_k a_{n-k} = 0$$

3.1 Coefficients of Rational Functions

Previously we expressed the generating series as a rational function. In order to find the coefficient $[x^n]$, we can use partial fractions.
In general, for a rational function $\frac{f(x)}{g(x)}$ where $deg(f) < deg(g)$ and

$$g(x) = \Pi_i(1 - \theta_i x)^{m_i}$$

that is we factor $g(x)$ into its linear roots with multiplicities m_i, we have

$$[x^n]\frac{f(x)}{g(x)} = \sum_i P_i(n)\theta_i^n$$

such that $deg(P_i) < m_i$ (that is $P_i(n)$ is a polynomial with degree 1 or more less than m_i).

Example 3.1. For the rational function $\frac{1}{(1-2x)(1-3x)}$, we have

$$\begin{aligned}[x^n]\frac{1}{(1-2x)(1-3x)} &= [x^n]\frac{-2}{(1-2x)} + [x^n]\frac{3}{(1-3x)} \\ &= (-2)[x^n]\frac{1}{(1-2x)} + 3[x^n]\frac{1}{(1-3x)} \\ &= -2^{n+1} + 3^{n+1}\end{aligned}$$

by the partial fraction decomposition and the negative binomial theorem.

3.2 Coefficients of Recurrence Relations

Let $C(x) = \sum_{n\geq 0} c_n x^n$ where the coefficients c_n satisfy the recurrence relation

$$c_n + q_1 c_{n-1} + \ldots + q_k c_{n-k} = 0$$

We define the characteristic polynomial $g(x)$ as:

$$g(x) = x^k + q_1 x^{k-1} + \ldots + q_{k-1}x + q_k$$

with roots β_i and multiplicity m_i for $i = 1, \ldots, j$ (j roots).
For this recurrence we have the general solution for c_n (the coefficient at x^n, that is $[x^n]C(x)$)

$$c_n = P_1(n)\beta_1^n + \ldots + P_j(n)\beta_j^n$$

where $deg(P_i) < m_i$ and β_i are the roots of the characteristic polynomial $g(x)$.
We can solve for the coefficients of $P_i(n)$ by using the initial conditions and the remainder factor theorem on our general solution.

Example 3.2. Find c_n explicitly where

$$c_n - 4c_{n-1} + 5c_{n-2} - 2c_{n-3} = 0$$

for $n \geq 3$ with initial conditions $c_0 = 1, c_1 = 1, c_2 = 2$.
Note the characteristic polynomial for this recurrence is

$$g(x) = x^3 - 4x^2 + 5x - 2 = (x-1)^2(x-2)$$

Therefore the general solution is

$$c_n = (A + Bn)1^n + C \cdot 2^n$$

By the remainder factor theorem

$$c_0 = 1 = (A + B(0))1^0 + C \cdot 2^0 = A + C$$
$$c_1 = 1 = (A + B(1))1^1 + C \cdot 2^1 = A + B + 2C$$
$$c_2 = 2 = (A + B(2))1^2 + C \cdot 2^2 = A + 2B + 4C$$

Solving the system of equations we get $A = 0, B = -1, C = 1$. Thus we have

$$c_n = -n1^n + 2^n = 2^n - n$$

for $n \geq 0$.

4 Graph Theory Basics

A graph consists of a **non-empty** set of **labelled** vertices $V(G)$ and a (possibly empty) set of edges $E(G)$. Note we do not allow:

- Directed edges
- Multiple edges between two vertices
- Loops
- Infinite number of vertices (and edges)

4.1 Isomorphism

Two graphs G_1, G_2 are isomorphic if there exists a bijection $f : V(G_1) \to V(G_2)$ such that

$$u, v \in G_1 \text{ adajcent} \iff f(u), f(v) \in G_2 \text{ adjacent}$$

4.2 Degrees and Neighbors

The neighbors $N(v)$ of a given vertex v is the set of vertices adjacent to v. The degree $deg(v) = |N(v)|$. Since every edge "generates" two degrees, we have

Theorem 4.1. The **Handshaking Lemma** states that

$$\sum_{v \in V(G)} deg(v) = |E(G)|$$

4.3 Types of Graphs

k-regular Every vertex has degree k (and thus $|E(G)| = \frac{k|V(G)|}{2}$).

Complete K_p All p vertices are adjacent to every other $p-1$ vertices (and $|E(G)| = \binom{p}{2}$). Thus K_p is $(p-1)$-regular.

Bipartite Vertices $V(G)$ may be bipartitioned into two sets A, B such that every edge is incident with one $a \in A$ and one $b \in B$.

Complete bipartite $K_{m,n}$ Bipartite where $|A| = m, |B| = n$ and every vertex $a \in A$ is adjacent with every vertex $b \in B$ (thus $|E(G)| = m \cdot n$).

n**-cycle** C_n A graph with n vertices $V(G) = \{v_1, \ldots, v_n\}$ and n edges $E(G) = \{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}, \{v_n, v_1\}\}$.

n**-cube** Q_n $V(G)$ is the set of all binary strings with length n (thus $|V(G)| = 2^n$). There exists an edge joining $u, v \in V(G)$ if strings u, v differ in one position (thus $deg(v) = n, \forall v \in V(G)$, so $|E(G)| = \frac{n \cdot 2^n}{2} = n2^{n-1}$).

Planar Graph has a drawing with no edges crossing.

4.4 Petersen Graph

A special graph with $|V(G)| = 10, |E(G)| = 15$ and is:

- non-planar
- 3-regular
- contains K_5 and $K_{3,3}$ subdivisions as subgraph
-

4.5 Subgraphs

A **subgraph** H of graph G satisfies: $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. Obviously $E(H)$ must only contain vertices from $V(H)$.
A **proper subgraph** is where $H \neq G$.
A **spanning subgraph** is where $V(H) = V(G)$.

4.6 Walks and Paths

A $v_0, v - n$-**walk** is an alternating sequence of vertices and edges

$$v_0, e_1, v_1, \ldots, e_n, v_n$$

where the length is the number of edges n. It is **closed** if $v_0 = v_n$.
A $v_0, v - n$-**path** is a walk with distinct vertices (and by extension distinct edges).
Walks and paths are transitive and symmetric relations ($v_0, v_n \to v_n, v_0$-path/walk and x, y-path/walk and y, z-path/walk implies x, z-path/walk). Only walks are reflexive too.

4.7 Cycles

As previously defined, a **cycle** is a connected graph with n vertices and n edges where every vertex has degree 2.
There are $2n$ closed walks of length n in an n-cycle.
A cycle must have at least length 3 where length is defined as the number of edges (or vertices).
The **girth** of a graph is the length of the *shortest cycle*. If there are no cycles, the girth is infinite. One can use the BFS algorithm to find the shortest cycle.
A spanning cycle is called a **Hamilton cycle**.

4.8 Connected Graphs

A graph is **connected** if for every vertex $x \neq y \in V(G)$ there is a path from x to y.
The **Hub Model** states that if there is a path from a given vertex v to every other vertex $x \in V(G)$, then the graph is connected.
A **component** C is a maximal connected subgraph of a graph G (no connected subgraph of G properly contains C).
The **cut** induced by a set of vertices $X \subseteq V(G)$ is the set of edges that have exactly one edge in X.

4.9 Eulerian Circuit

An **Eulerian circuit** of a (connected) graph G is a closed walk that contains every edge of G exactly once.
There exists an Eulerian circuit in a connected graph G *if and only if* every vertex has even degrees.

4.10 Bridges

If $e \in E(G)$, $G - e$ is the subgraph where $V(G - e) = V(G)$ and $E(G - e) = E(G) \setminus \{e\}$.
A **bridge** e is an edge where $G - e$ has more components than G.
Removing a bridge e in a connected graph G results in $G - e$ having exactly 2 components.
An edge e is a bridge *if and only if* it is not contained in any cycle of G.

5 Trees

A **tree** T is a *connected* graph with no cycles. Therefore, there is a unique path joining every vertex u and v.
Since there are no cycles, every edge in T is a bridge.
A tree with p vertices also has exactly $p - 1$ edges (by mathematical induction).

5.1 Vertices of Degree One

A tree has at least two vertices of degree one. In fact, there are at least r degree one vertices where r is the maximum degree of a vertex.
A vertex of degree 1 in a tree is called a **leaf**.

5.2 Spanning Trees

A **spanning tree** is a spanning subgraph that is also a tree.
Note that a graph G is connected *if and only if* it has a spanning tree.

5.3 Breadth-First Search (BFS)

Breadth-First Search involves constructing a spanning tree (if it exists) by successively adding neighboring vertices to the tree.
The **predecessor/parent** function $pr(x)$ of a vertex x is the neighbor of x that x was joined to in the BFS tree.
The Breadth-First Search Algorithm is as follows:

1. Select a root vertex $r \in V(G)$ as the initial subgraph D, with $pr(r) = \varnothing$. Add r into queue
2. Take the next vertex x in the queue (**active**) (the first iteration $x = r$)
3. Join all neighbors $n \in N(x)$ not in D to the tree D. Update $pr(n) = x$.

4. Push all the neighbors $N(x)$ into the queue.

The algorithm terminates with either $V(D) = V(G)$ (there is a spanning tree) or $|V(D)| < |V(G)|$ (no spanning tree and G is not connected).
The **level** of x is defined recursively as $level(x) = level(pr(x)) + 1$ where $level(r) = 0$.

6 Planar Graphs

Remember a graph is **planar** if there exists a drawing of the graph where no edges cross (**planar embedding**).
Note a graph is planar if and only if all its components are planar (this is useful for showing a graph is not planar by showing that one of its component is non-planar).
A planar embedding partitions the plane into regions called **faces**.
The vertices and edges in (incident to) a face forms the **boundary**.
Two faces are **adjacent** if they are incident with a common edge.
A **boundary walk** of a face f is the closed walk constructed by moving around the perimeter of the face.
The **degree of a face** is the length of the boundary walk. Note that a bridge is incident with only 1 face (and a non-bridge incident with 2 faces), thus bridges contribute 2 degrees to 1 face and non-bridges contribute 1 degree each to 2 faces.
The **Faceshaking Lemma** states for a graph with faces $f_1, \ldots, f_s$

$$\sum_{i=1}^{s} deg(f_i) = 2|E(G)|$$

6.1 Euler's Formula and Non-Planarity

See **Graph Strategies** section.

6.2 Kuratowski's Theorem

Note that we can show K_5 and $K_{3,3}$ are non-planar (by using our non-planarity inequalities and/or Euler's formula).
An **edge subdivision** of a graph is when you replace an edge by a path of 1 or more. Subdividing does not change planarity (and similarly, reverse subdividing does not).
A graph is non-planar if and only if there are no non-planar edge subdivisions.
Kuratowski's Theorem states that a graph is non-planar if and only if it has a subgraph that is an edge subdivision of K_5 or $K_{3,3}$.

6.3 k-Colouring

A **k-colouring** of a graph is a function from $V(G)$ to a set of size k (our colours) such that adjacent vertices always have have different colours. A graph with a k-colouring is k-colourable.
Note if a graph is k-colourable, it is t-colourable where $t \geq k$.
It is trivial to show that all planar graphs are 6-colourable (see **Graph Strategies**).
5-colourable uses **edge contraction**, which is intuitively reducing an edge $e = \{x, y\}$ down to length 0 and combining the two vertices x, y into one vertex z. All former incident edges to x and y are now incident to the vertex z. The graph with e contracted is denoted as G/e.
Note G/e is planar if G is planar, but the converse is not necessarily true.

7 Matchings

A **matching** of a graph G is a set (may be empty) of edges $M \subseteq E(G)$ where any vertex $v \in V(G)$ are incident to at most one edge in M.
All graphs contain at least one matching: the empty set.
A vertex v is **M-saturated** if it is incident with an edge in M (and inversely, M-unsaturated).
The largest matching in G is called a **maximum matching**.
A **maximal matching** is a given matching M in G of which you cannot create a larger matching by adding an edge $E(G) \setminus M$.
Maximum matchings are maximal but maximal matchings are not necessarily maximum.
A **perfect matching** M occurs when every $v \in V(G)$ is M-saturated, that is $|M| = |V(G)|/2$.

7.1 M-Alternating and M-Augmenting Paths

A **M-alternating path** is a path $v_0, \ldots, v_n$ with edges that alternate between being in M and not in M, that is:

1. $\{v_i, v_{i+1}\} \in M$ if i is even; $\{v_i, v_{i+1}\} \notin M$ if i is odd, OR
2. $\{v_i, v_{i+1}\} \notin M$ if i is even; $\{v_i, v_{i+1}\} \in M$ if i is odd

An **M-augmenting path** is an M-alternating path between two M-unsaturated vertices (v_0, v_n are not incident to any edges in M).
If M has an M-augmenting path, it is not maximum (flip the set memberships of the edges in the M-agumenting path and union it with the other edges in M).

7.2 Covers

A cover C is a set of vertices where every every edge $e \in E(G)$ is incident to at least one vertex in C.
Note $V(G)$ is a cover for G.
A **minimum cover** is the smallest cover of a graph G.
Matchings are bounded by all covers, that is $|M| \leq |C|$.
Furthermore, if M is a matching, C is a cover, and $|M| = |C|$, then M is maximum and C is minimum (the converse is not true, e.g K_3).

7.3 Konig's Theorem

Konig's Theorem states that the converse is true for bipartite graphs: the maximum matching has the same size as the minimum cover.
See **course notes** for the full example.
A Bipartite Matching Algorithm can be used to determine the maximum matching (and minimum cover) for any bipartite graphs (see **Graph Strategies**).

7.4 Hall's Theorem

In a bipartite graph with bipartition (A, B), it is clear that to saturate every vertex in A, $|B| \geq |A|$.
For a given bipartite graph G with bipartition A, B, there is a matching saturating every vertex in A *if and only if* for every subset $D \subseteq A$

$$|N(D)| \geq |D|$$

7.5 Bipartite Perfect Matching

A bipartite graph has a perfect matching *if and only if* $|A| = |B|$ and every subset $D \subseteq A$ satisfies

$$|N(D)| \geq |D|$$

If G is a k-regular bipartite graph with $k \geq 1$, then G has a perfect matching.

8 Graph Strategies

8.1 Miscellaneous Theorems/Lemmas

Handshaking Lemma

$$\sum_{v \in V(G)} deg(v) = 2|E(G)|$$

Odd-Degree Vertices There must exist an even number of odd degree vertices (by Handshaking Lemma).

Average Degrees The average degrees for vertices in a graph is

$$\frac{2|E(G)|}{V(G)}$$

Similarly for average degrees for s faces

$$\frac{2|E(G)|}{s}$$

Eulerian Circuits In a connected graph G there exists an Eulerian circuit *if and only if* every vertex has even degrees.

8.2 Isomorphism

Isomorphic Find mapping between vertices. If none...

Non-isomorphic Find a substructure that does not exist in the other (e.g. length of given cycle, degree of vertices, length of longest path).

Note the degrees of faces of isomorphic graphs *need not* be the same (only their sum, by Faceshaking Lemma, are invariantly the same). One can come up with a counter-example.

8.3 k-regular Graphs

Generally it is not wise to use induction on k in k-regular graphs since it is unclear how to deduce a $k-1$-regular graph.

8.4 Bipartition

Bipartite Show there exists two sets of vertices A, B where all edges are incident with one $a \in A$ and one $b \in B$.

Not Bipartite Show that there exists an odd cycle.

Some useful theorems about bipartite graphs:

Odd Cycles $\iff$ Not Bipartite A graph is bipartite if and only if it has no odd cycles.

8.5 Connectedness

Connected Show that there is a path from a given vertex v to every other vertex $x \in V(G)$.

Non-Connected Show there exists a proper non-empty vertex subset $X \subset V(G)$ that induces an empty cut/set.

Some useful theorems for connectedness:

Walk $\iff$ Path There is a walk from x to y if and only if then there is a path from x to y.

Transitivity of Paths If there is a path from x to y and y to z, then there is a path from x to z.

Non-Empty Induced Sets A graph G is connected if and only if every proper subset $X \subset V(G)$ induces a non-empty cut/set.

Bridge in Connected Graph If e is a bridge in a connected graph G, then $G - e$ has exactly two components.

Bridge $\iff$ Not Cycle An edge e is a bridge *if and only if* it is not contained in any cycle of G.

Two Distinct Paths $\rightarrow$ Cycle If there exists two distinct paths from u to v in G, then G contains a cycle (proof: you cannot just concatenate the two paths since they may have common vertices!).

No Cycle $\rightarrow$ Most One Path If there is not a cycle, then there is at most one path between every vertex pair.

8.6 Trees

Many proofs for trees can be done using induction on p vertices, since there will always be at least one vertex of degree one (and one can remove the vertex to produce a $p-1$ connected that is still a tree (induction hypothesis). Some useful theorems for trees:

Tree Exactly One Path There is exactly one (unique) path between every pair of vertices in a tree.

Every Edge is a Bridge Every edge in a tree is a bridge (follows from every edge not in a cycle is a bridge).

At Least Two Degree One Vertices In a tree with at least two vertices there exists at least two vertices of degree 1.

Number of Degree One Vertices Note that for a tree with $p \geq 2$ vertices, and n_i be the number of vertices of i degree. Then $n_1 = 2 + n_3 + 2n_4 + \ldots + (p-2)n_p$.

That is there are as many vertices of degree one as the degree of the vertex with the maximum degree p.

p Vertices and $p-1$ Edges A tree with p vertices has exactly $p-1$ edges (shown by induction).

Trees are Bipartite All trees are bipartite (shown by induction).

8.7 Spanning Trees

For a spanning subtree, we have:

Connected $\iff$ Spanning Tree A graph is connected *if and only if* there exists a spanning tree.

Connected Graph with $|V(G)|-1$ edges A connected graph that has $|E(G)| = |V(G)| - 1$ is a tree. This follows from the fact that all connected graphs have a spanning tree T which also has the same vertices (spans) and edges as G.

Adding then Removing an Edge Given a spanning tree T in G and $e \in E(G), e \notin E(T)$, $T+e$ contains exactly one cycle C. Given any edge e' on C, $T+e-e'$ is also a spanning tree of G.

Removing then Adding an Edge If T is a spanning tree of G and $e \in E(T)$, then $T-e$ has 2 components. If e' is an edge induced by the cut of one of the components, then $T-e+e'$ is also a spanning tree of G.

Bipartite and Levels A graph is bipartite *if and only if* there are no edges joining the same level.

8.8 Breadth-First Search (BFS)

There are a number of important properties of BFS trees:

BFS Algorithm Terminates The BFS Algorithm terminates with either $|V(D)| = |V(G)|$ (D is a spanning tree of G) or $|V(D)| < |V(G)|$ then G is not connected and D is not a spanning tree.

Non-Decreasing Levels The vertices enter a breadth-first search tree in non-decreasing order of level.

Non-Tree Edges For a connected graph with a BFST, each non-tree edge in the graph joins vertices of that are at most one level apart in the BFST (and all tree edges of course join vertices exactly 1 level apart).

8.9 Applications of BFS

Some applications of the BFS algorithm:

Shortest Path To find the shortest path from x to y, construct a BFST rooted at x. The level of y is the length of the shortest path.

Shortest Cycle (Girth) For every vertex $v \in V(G)$, construct BFST T_v rooted at v. For every edge $e \in E(G) \setminus E(T_v)$, consider $T_v + e$. From previous lemma, $T_v + e$ must contain a cycle, thus out of all the $T_v + e$ graphs there exists a shortest cycle.

8.10 Planarity

One can use induction for certain problems on planar graphs with the vertex that has at most degree 5.

Planar Produce a planar embedding of the graph.

Tip: Look for the longest cycle. The rest of the vertices/edges must either go inside or outside the cycle.

Not Planar We can check our planar invariant equations and see if any of them are violated (we can check for any component, since if a component is non-planar then the graph is non-planar):

1. For connected graphs with p vertices, q edges, s faces, does Euler's formula

$$p - q + s = 2$$

hold for every component? If no, then the graph is non-planar.

2. For connected bipartite graphs where $p \geq 3$ vertices and q edges, does

$$q \leq 2p - 4$$

hold? If no, the graph is non-planar.

3. For all connected graphs where $p \geq 3$ vertices and q edges, does

$$q \leq 3p - 6$$

hold? If no, the graph is non-planar.

4. By Kuratowski's Theorem, can you find a subgraph that is a subdivision of K_5 or $K_{3,3}$? If yes, then the graph is non-planar.

If there are not 5 vertices with at least degree 4, then look for $K_{3,3}$ (similarly there needs to be at least 6 vertices with at least degree 3).

To find a subgraph of subdivision $K_{3,3}$:

(a) Look for a cycle C of length at least 6. In the end, 6 of these vertices will be the 6 vertices in your $K_{3,3}$.

(b) Find 3 chords P_1, P_2, P_3 such that any two of them cross.
A chord P_1 is a path P that has its ends u_1, v_1 as vertices on the cycle but is otherwise disjoint (the path is not within the cycle).
Two chords cross P_1, P_2 cross if their paths overlap.

(c) The 6 vertices (after reverse-subdiving the cycle down) that are part of the chords and the chords themselves (after reverse-subdividing) are the edges of $K_{3,3}$.

Some other useful theorems for planarity:

Cycles and Faces If a graph G is connected and not a tree (therefore there is a cycle), the boundary of each face in the the planar embedding of G contains a cycle.

No Cycles If the boundary of any face f does not contain a cycle, then the graph is a tree.

Vertices vs. Edges Note that for a connected graph with faces of at least degree d^*, p vertices, and q edges

$$q(d^* - 2) \leq d^*(p - 2)$$

(shown by Faceshaking Lemma where $2q \geq sd^*$ (s faces) and Euler's formula).

Degree Five Vertex A planar graph (connected or non-connected) has at least one vertex of at most degree 5 (shown using $q \leq 3p - 6$).

8.11 k-colouring

To show a graph is k-colourable on p vertices, most of the time one can use induction on a graph G' with $p > k$ vertices. Show there is a way to reduce G' down to $p - 1$ vertices while maintaining the properties and invariants of the graph type.
Some useful properties:

Bipartite $\iff$ 2-Colourable A graph is bipartite *if and only if* it is 2-colourable.

4-Colourable All planar graphs are 4-colourable.

5-Colourable and 6-Colourable Follows from 4-colourable. 6-Colourable proof uses induction and the fact that there is a vertex of at most degree 5.

5-Colourable proof is similar to 6-Colourable proof but uses edge contraction and Kuratowski's Theorem to show there cannot be a K_5 subgraph when inducting with the vertex of degree 5.

K_p graphs A K_p graph is p-colourable but not $(p - 1)$-colouarble.

8.12 Matchings

Find Maximum Matching in Bipartite Graphs Use the Bipartite Matching Algorithm. The objective is to find M-augmenting paths:

1. Keep track of a queue of current vertices.
2. Start with all unsaturated vertices in A (this is X_0) and create trees for each one. Push into queue.
3. For each active vertex u in the queue, if $u \in A$, then find an edge $e = \{u, v\} \notin M$ where v has never been added before. Append the new edge e and vertex v onto tree and push v into queue. If $u \in B$, do the same but find an edge $e \in M$. That is when moving from A to B, use unsaturated edges, and when moving from B to A, use saturated edges.
4. Once you run out of vertices, you will have a forest (collection of trees). One may also construct the Konig's Theorem diagram to look for unsaturated vertices in Y.
5. If there is an M-augmenting path P, set $M = M \Delta P$ (or $M = M \cup P \setminus (M \cap P)$. That is invert the set membership of all edges in the augmenting path and take the new set of matching edges M. With the new (larger) matching, repeat these steps until one cannot find an M-augmenting path.

Show K-Perfect Matching Use proof by induction on k.

Show Any Graph has Perfect Matching Proof by contradiction by contradicting the definition of a perfect matching.

Use Hall's theorem for bipartite graphs.

Some useful theorems:

Bipartite and Lower Bound on Matching If G is a bipartite graph with bipartitions (A, B) where $|A| = |B| = n$ and q edges, then G has a matching of size at least q/n. Prove using Konig's Theorem and bound on minimum cover to show bound on maximum matching size.

Hall's Theorem A bipartite graph has a matching saturating every vertex of A *if and only if* for every subset $D \subseteq A$

$$|N(D)| \geq |D|$$

Bipartite Perfect Matching A bipartite graph has a perfect matching *if and only if* $|A| = |B|$ and for every subset $D \subseteq A$

$$|N(D)| \geq |D|$$

K-Regular Bipartite If a bipartite graph is k-regular, $k \geq 1$ then it has a perfect matching.