6.120 Exam 2 Review

Gwyn Tangog (gwynt@mit.edu) - 1,3,4,8
Andrea Leang (akleang@mit.edu) - 2,5,6,7

April 12, 2025

1 Number Theory

1.1 Notes, Rules, Formulae

(a) **Divisibility Rules.**

D1. If $a \mid b$, then $a \mid bc$ for all c.

D2. If $a \mid b$ and $b \mid c$, then $a \mid c$.

D3. If $a \mid b$ and $a \mid c$, then $a \mid sb + tc$ for all integers s and t.

D4. For all $c \neq 0$, $a \mid b$ if and only if $ca \mid cb$.

(b) **GCD Rules.** Note: G6 is the backbone of the Euclidean algorithm and the Pulverizer

G1. Every common divisor of a and b divides $\gcd(a, b)$.

G2. $\gcd(ka, kb) = k \cdot \gcd(a, b)$ for all $k > 0$.

G3. If $\gcd(a, b) = 1$ and $\gcd(a, c) = 1$, then $\gcd(a, bc) = 1$.

G4. If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.

G5. If $m \mid a$ and $m \mid b$, then $m \mid \gcd(a, b)$.

G6. $\gcd(a, b) = \gcd(b, \text{rem}(a, b))$.

(c) **Euclidean Algorithm.** This provides the gcd of a and b.

$$\gcd(a, b) = \gcd(b, \text{rem}(a, b)).$$

For example, we can compute the GCD of 259 and 70 as follows:

$$\begin{aligned}
\gcd(259, 70) &= \gcd(70, 49) && \text{since rem}(259, 70) = 49 \\
&= \gcd(49, 21) && \text{since rem}(70, 49) = 21 \\
&= \gcd(21, 7) && \text{since rem}(49, 21) = 7 \\
&= \gcd(7, 0) && \text{since rem}(21, 7) = 0 \\
&= 7.
\end{aligned}$$

(d) **The Pulverizer.** This provides linear combination of a and b that make $gcd(a, b)$. This also provides us with information on modular inverses. In particular, it gives us the modular inverse of x modulo y, and the modular inverse of y modulo x.

x	y	$(\text{rem}(x, y))$	$=$	$x - q \cdot y$
259	70	49	$=$	$259 - 3 \cdot 70$
70	49	21	$=$	$70 - 1 \cdot 49$
			$=$	$70 - 1 \cdot (259 - 3 \cdot 70)$
			$=$	$-1 \cdot 259 + 4 \cdot 70$
49	21	7	$=$	$49 - 2 \cdot 21$
			$=$	$(259 - 3 \cdot 70) - 2 \cdot (-1 \cdot 259 + 4 \cdot 70)$
			$=$	$\boxed{3 \cdot 259 - 11 \cdot 70}$
21	7	0		

(e) **Modulo Rules.**

M1. $a \equiv a \pmod{n}$

M2. $a \equiv b \pmod{n}$ implies $b \equiv a \pmod{n}$

M3. $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$ implies $a \equiv c \pmod{n}$

M4. $a \equiv b \pmod{n}$ implies $a + c \equiv b + c \pmod{n}$

M5. $a \equiv b \pmod{n}$ implies $ac \equiv bc \pmod{n}$

M6. $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$ imply $a + c \equiv b + d \pmod{n}$

M7. $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$ imply $ac \equiv bd \pmod{n}$

(f) **Fermat's Little Theorem**. If p is prime and a is not congruent to $0 \mod p$, then $a^{p-1} \equiv 1 \mod p$

1.2 Practice and Techniques

(a) Find the last digit of 6^{100}. [Skill: Pattern Finding]

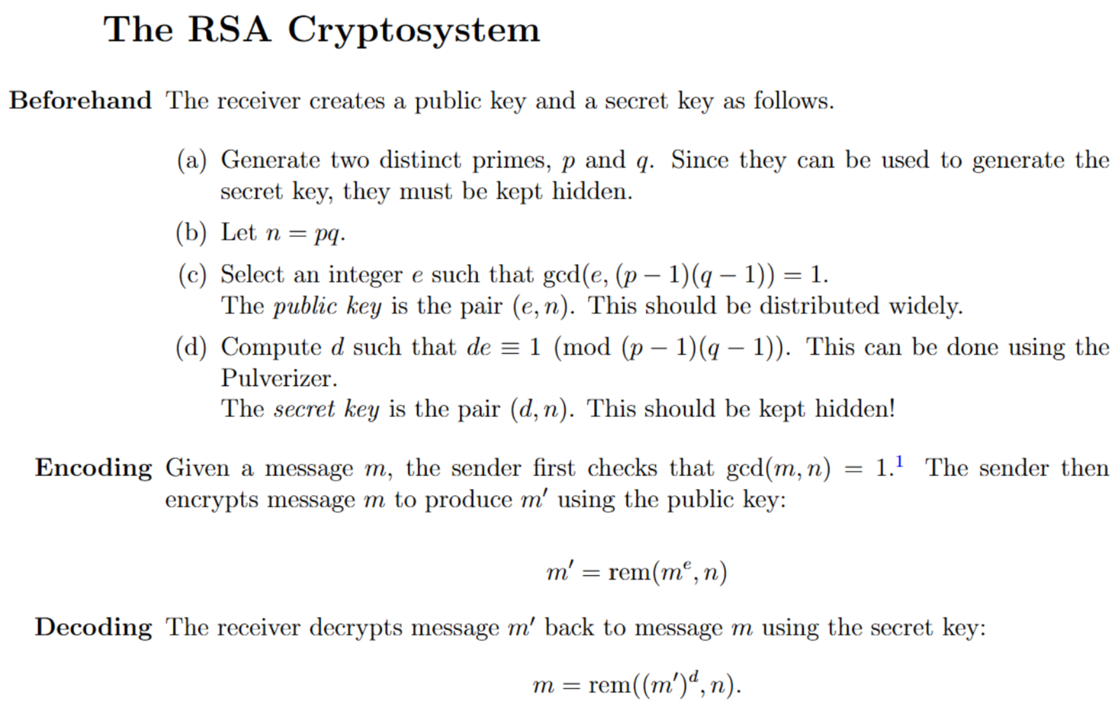
(b) Find the last digit of 12^{202} [Skill: Simplifying + Pattern Finding]

(c) Evaluate $123^{57} \mod 124$. [Skill: Simplifying (Using negative modulo)]

(d) Show how the Pulverizer can be used to find the modular inverse of two numbers. [Skill: General Proofs, Pulverizer]

(e) Find the modular inverse of 17 mod 19. [Skill: Modular Inverse, Applying the Pulverizer]

1.3 Cryptography



The RSA Cryptosystem

Beforehand The receiver creates a public key and a secret key as follows.

(a) Generate two distinct primes, p and q. Since they can be used to generate the secret key, they must be kept hidden.

(b) Let $n = pq$.

(c) Select an integer e such that $\gcd(e, (p-1)(q-1)) = 1$.
The *public key* is the pair (e, n). This should be distributed widely.

(d) Compute d such that $de \equiv 1 \pmod{(p-1)(q-1)}$. This can be done using the Pulverizer.
The *secret key* is the pair (d, n). This should be kept hidden!

Encoding Given a message m, the sender first checks that $\gcd(m, n) = 1$.[1] The sender then encrypts message m to produce m' using the public key:

$$m' = \text{rem}(m^e, n)$$

Decoding The receiver decrypts message m' back to message m using the secret key:

$$m = \text{rem}((m')^d, n).$$

Key notes:

(a) If we know what the factors of n are, we can hack the RSA system.

(b) RSA takes longer to hack with larger n.

2 Graphs, Coloring, and Graph Induction

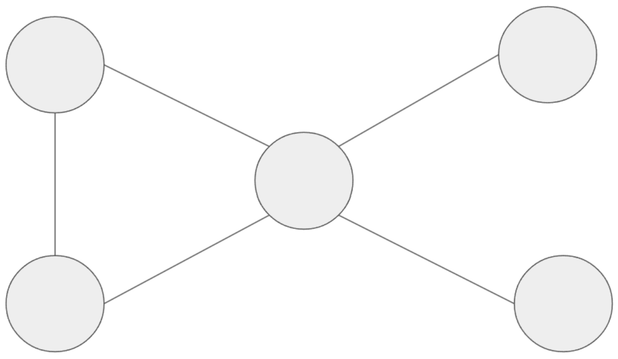
Key notes:

(a) To prove that a graph is at least k-colorable, propose a k-coloring.

(b) To prove that a graph is at most k-colorable, explain why the graph can't be colored with k-1 colors. One way is looking for a complete k-1 subgraph (this forces at least a k-coloring).

(c) Handshake Lemma: $\sum_{v \in V} \text{degree}(v) = 2|E|$

Coloring example:



Graph Induction:

(a) P(n) should include 1. what graphs we're interested in (i.e. trees, planar graphs, etc.) and 2. the property we want to prove about the graphs.

(b) Perform induction as we normally do. However, to prevent build-up error, we must "examine" an arbitrary graph of interest with $n + 1$ nodes. Then take away a special node that is convenient. (i.e. a leaf for trees). Make sure the resulting subgraph is a "graph of interest"!!!

(c) Remember that by the induction hypothesis, we know nothing about the graph with $n+1$ nodes. :(However, after taking away a node, we suddenly know many things. :D

(d) Build-up error happens when we go from n to $n + 1$ and not all cases are covered.

(e) Counter-example tools: Adding nodes and complete graphs!

(f) BE CAREFUL: Induction is bogus/erroneous when:

   (a) If there is build-up error.

   (b) If we take away a node from an $n+1$ graph but a key property is not maintained. (An example of this is the claim "All graphs where all nodes have degree ≥ 1 are connected.")

Induction example 1: We know that every planar graph has a vertex with degree at most 5. Prove by induction that any planar graph can be colored in at most 6 colors. (Rec 11)

3 Matching

Key notes:

(a) Please remember the stable matching algorithm.

(b) The stable matching algorithm is optimal for the asker and is pessimal for the one being asked.

(c) A matching is stable if there are no rogue couples.

(d) A rogue couple appears when two people prefer each other over their current partners.

(e) Unfortunately, one-sided unrequited love is still stable. A.k.a. “I don’t feel the same” is still stable. :(

Example:

Preferences of humans:

Gwyn	Beagle	Corgi	Dachshund
Zach	Corgi	Beagle	Dachshund
Ronitt	Corgi	Beagle	Dachshund

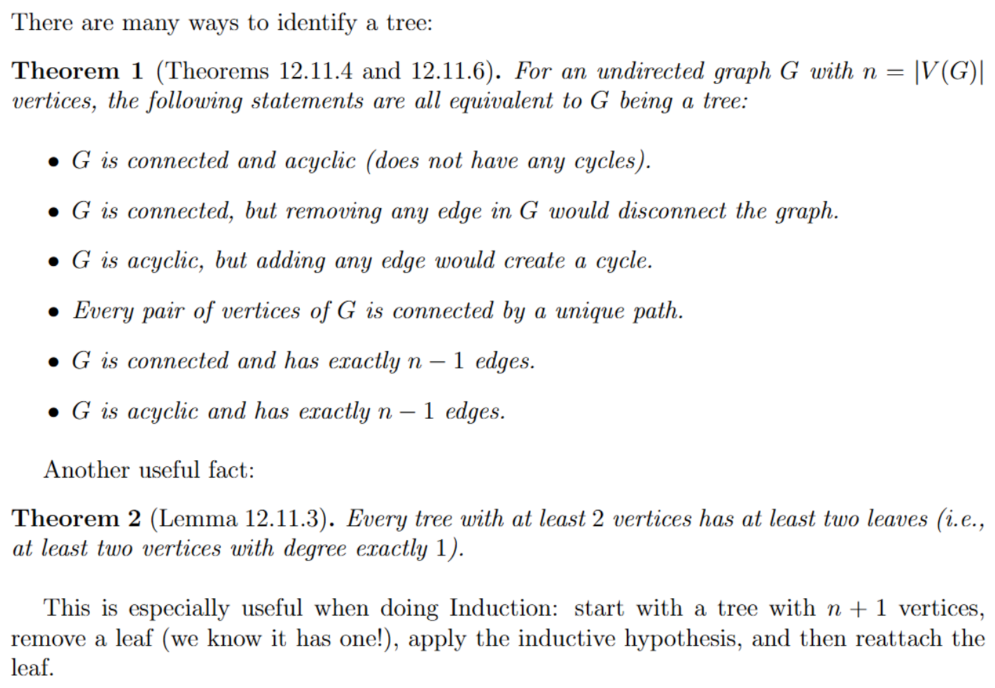
Preferences of pets:

Beagle	Gwyn	Ronitt	Zach
Corgi	Ronitt	Zach	Gwyn
Dachsund	Gwyn	Ronitt	Zach

We will examine human asking and pets asking!

4 DAGs and Trees

4.1 Vocabulary and Notes


There are many ways to identify a tree:

Theorem 1 (Theorems 12.11.4 and 12.11.6). *For an undirected graph G with $n = |V(G)|$ vertices, the following statements are all equivalent to G being a tree:*

- *G is connected and acyclic (does not have any cycles).*
- *G is connected, but removing any edge in G would disconnect the graph.*
- *G is acyclic, but adding any edge would create a cycle.*
- *Every pair of vertices of G is connected by a unique path.*
- *G is connected and has exactly $n - 1$ edges.*
- *G is acyclic and has exactly $n - 1$ edges.*

Another useful fact:

Theorem 2 (Lemma 12.11.3). *Every tree with at least 2 vertices has at least two leaves (i.e., at least two vertices with degree exactly 1).*

This is especially useful when doing Induction: start with a tree with $n + 1$ vertices, remove a leaf (we know it has one!), apply the inductive hypothesis, and then reattach the leaf.

Note that for induction: We assume $P(1)$ up to $P(n)$ then we want to prove $P(n+1)$. We usually try to remove a special node, making sure the conditions are preserved, so that our remaining graph falls under the inductive hypothesis. Afterwards, we put our special node back in and prove that the inductive hypothesis is preserved. Theorem 2 is the reason we usually remove a leaf during induction!

4.2 Practice

(a) Prove that if there exists any closed walk through u in G, the shortest positive length closed walk through u is a cycle. [Skill: Graph proof by contradiction]

(b) Prove that every forest with at least one edge must have a leaf. [Skill: Direct graph proof].

(c) Call a graph mild when every vertex has degree at most 10. Prove by strong induction that every mild forest contains a matching that uses at least 10 percent of its edges. [Skill: Graph proof by induction]

5 Connectivity and Digraphs

5.1 Vocabulary

Undirected Graphs

(a) A pair of nodes (a, b) are **connected nodes** if there is a sequence of edges from a to b.

(b) A **walk** from (a, b) is a sequence of edges from a to b. Vertices may be repeated.

(c) A **path** from (a, b) is a sequence of edges from a to b. Vertices may NOT be repeated.

(d) A **closed walk** is a walk from a to b where $a = b$. A.K.A. we start and end at the same node.

(e) A **cycle** is a closed walk where no vertices are repeated except for the first and last node.

(f) A **connected graph** is a graph where all nodes are connected to each other.

Directed Graphs

(a) A directed graph is **strongly connected** if for all pairs of vertices (u, v) there is a path from u to v and a path from v to u.

(b) The rest of the terms remain the same.

5.2 Practice

Example: Prove that if G has an odd-length closed walk, then it has an odd-length cycle. (Recitstion 13)

6 Scheduling

Key notes:

(a) Know how to make a Hasse Diagram! Include only the covering edges (the edges that are the only path between its endpoints).

(b) Two nodes a and b in a Hasse Diagram are **comparable** if a can reach b or b can reach a.

(c) A **chain** is a subset of nodes where every pair of nodes are comparable.

(d) A **critical path** is a maximum-length chain.

(e) An **antichain** is a subset of nodes where every pair are not comparable.

(f) A shortest parallel schedule is equal to the length of the longest chain.

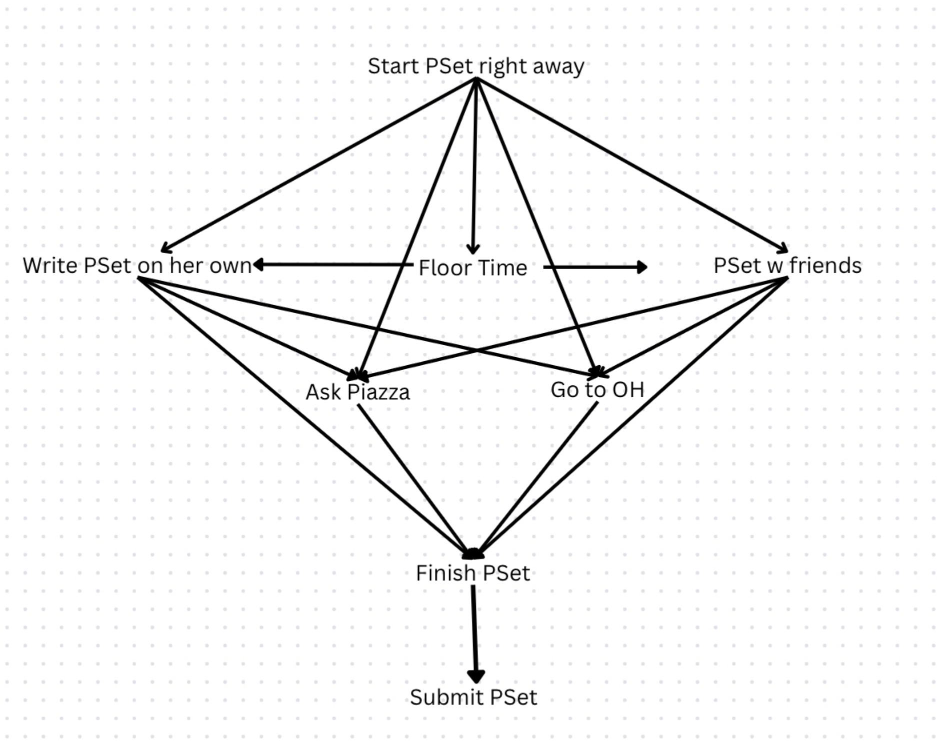
(g) A **topological order** of a DAG is a list of all nodes such that every node appears earlier in the list from every other node reachable from it.

Example: Andrea's General PSet completion Process: Let's help Andrea get her pset done. She starts right away. During her pset writing process, she can write the pset on her own, go to OH for help, ask questions on Piazza, lie on the floor waiting for the answers to come to her, or pset with friends. She can always go to OH or ask questions on Piazza again after writing a bit of the pset. She needs to write Once she finishes her pset, she submits it to gradescope.

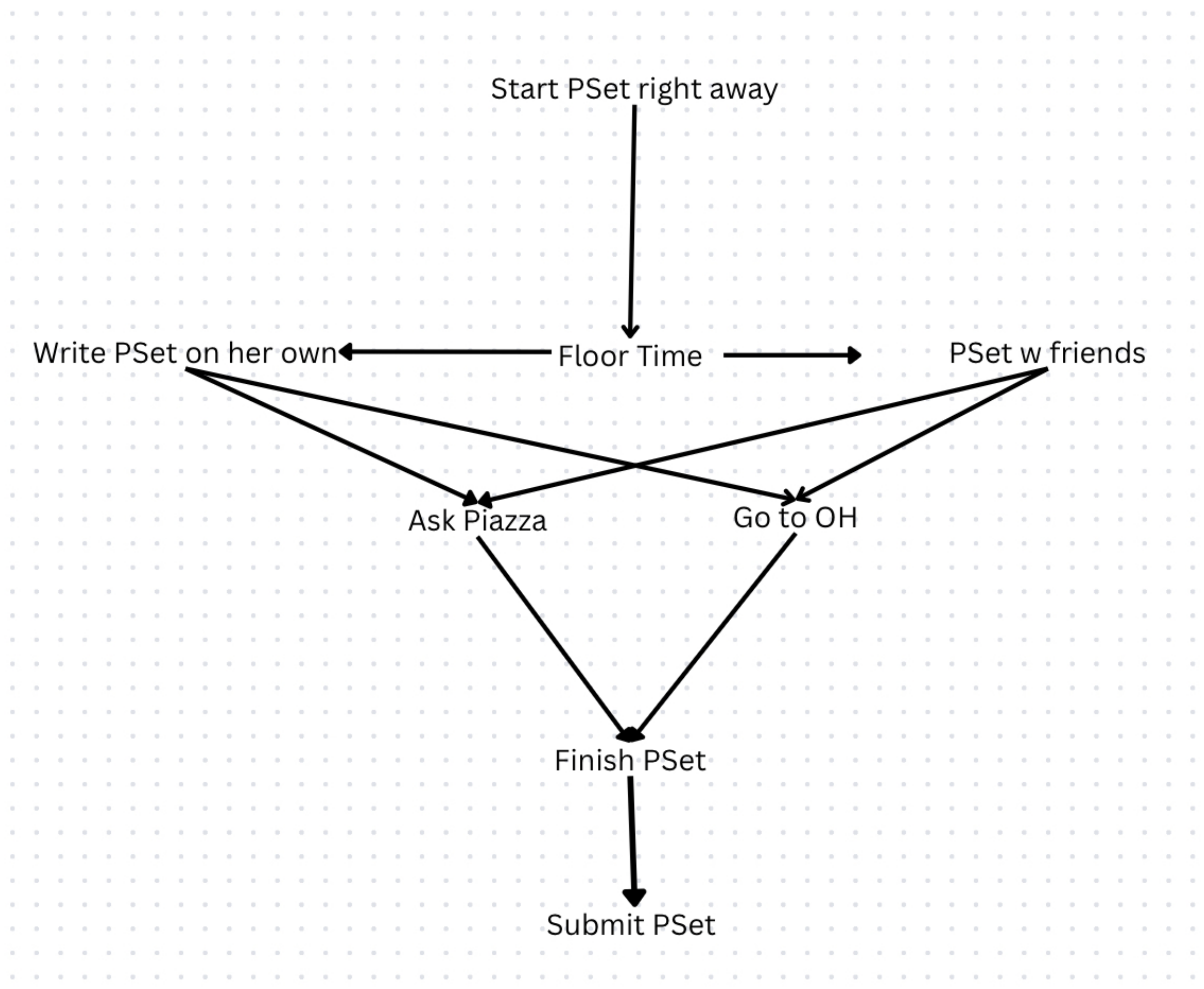
Tasks relations:

Start PSet right away $\implies$ write PSet on her own	write PSet on her own $\implies$ Finish PSet
write PSet on her own $\implies$ go to OH	write PSet on her own $\implies$ ask on Piazza
go to OH $\implies$ finish PSet	start PSet right away $\implies$ go to OH
Start PSet right away $\implies$ ask on Piazza	ask on Piazza $\implies$ finish PSet
Start PSet right away $\implies$ Floor Time	Floor Time $\implies$ write PSet on her own
Floor Time $\implies$ PSet w friends	PSet w friends $\implies$ finish PSet
Stat PSet right away $\implies$ PSet w friends	PSet w friends $\implies$ go to OH
PSet w friends $\implies$ ask on Piazza	finish PSet $\implies$ submit on Gradescope

Full Diagram:



Hasse Diagram:



7 Relations

7.1 Definitions

- A relation $R \subseteq A \times B$ has a domain A, codomain B, and a subset R of ordered pairs. Notation aRB and $R(a, b)$ means $(a, b) \in R$.
- A relation R is a **function** iff there is at most one arrow out of every $a \in A$.
- A relation R is **total** if there is at least one arrow out of every $a \in A$.
- A **total function** has exactly one arrow out of every $a \in A$.
- A relation is **injective** iff there is at most one arrow in to every $b \in B$. Alternatively, $f(a) = f(a') \implies a = a'$.
- A relation is **surjective** iff there is at least one arrow in to every $b \in B$. Alternatively, for all $b \in B$, $b = f(a)$ for some $a \in A$.

- A total function that is both injective and surjective is called a **bijection**.

7.2 Equivalence Relations

- Reflexive: aRa for all $a \in A$
- Symmetric: aRb iff bRa for all $a, b \in A$
- Antisymmetric: aRb and bRa implies $a = b$
- Transitive: aRb and bRc implies aRc
- A relation R is an equivalence relation if it is reflexive, symmetric, and transitive.
- A relation R is a weak partial order if it is reflexive, antisymmetric, and transitive.
- A WPO is a total ordering if every pair of elements is comparable (aRb or bRa).

7.3 Examples

- The linear function $f(x) = mx + b$ is a bijection.
- The function $f : \mathbb{Z} \to \{0, 1, 2\}$ where $x \mapsto x \bmod 3$ is a total surjection.
- The relation "has the same numbers as prime factors" on the integers is an equivalence relation.
- The "divides" relation on the nonnegative integers is a WPO.

8 Counting

8.1 Binomial Coefficient

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

- Read as "n choose k"; this is the number of ways to choose a subset of k items from a set of n items.

8.2 Bijection Rule

If there is a bijection between sets A and B, then $|A| = |B|$.

- Strategy: if we are given a set that is difficult to count, often we can define a bijection to another set that is easier to count, and then count that set instead.
- Example: counting book arrangements with no adjacent books (see recitation 15)

8.3 Product Rule

For finite sets $A_1, \dots, A_n$, $|A_1 \times \cdots \times A_n| = |A_1| \cdots |A_n|$.

- Example: the number of binary strings of length n is 2^n, as we have two choices for each of the n digits. In other words, $|A_i| = 2$ for the ith digit.
- If we are making multiple choices and taking the "and" of all of them, then use the product rule.

8.4 Sum Rule

For pairwise disjoint finite sets $A_1, \dots, A_n$, $|A_1 \cup \cdots \cup A_n| = |A_1| + \cdots + |A_n|$.

- Example: suppose you are at a restaurant that serves four types of sandwiches and six types of pasta dishes. Then you have ten choices for a meal.
- If we are making multiple choices and taking the "or" of all of them, then use the sum rule.

8.5 Generalized Product Rule

If we are making k choices to count a set A, and there are n_i options for choice i, where $i \in \{1, \dots, k\}$, and the number of options for each choice does not depend on the results of previous choices, then $|A| = n_1 \cdot n_2 \cdots n_k$.

- Strategy: construct a "recipe" where we make one choice at a time.
- Example: Suppose we're playing a game with $n \geq 15$ cards and we begin by dealing five-card hands to three players. We have $\binom{n}{5}$ possible hands for the first player, $\binom{n-5}{5}$ possible hands for the second player, and $\binom{n-10}{5}$ possible hands for the third player. By the generalized product rule, there are $\binom{n}{5}\binom{n-5}{5}\binom{n-10}{5}$ ways to deal cards to the three players.
- Note that while the set of possible options for the second choice depends on the first choice, the number of options is the same regardless of what option we choose first.
- This is often useful when we are making choices without replacement.

8.6 Division Rule

If there is a k-to-1 correspondence between sets A and B (every $a \in A$ has one arrow out and every $b \in B$ has k arrows in), then $|A| = k \cdot |B|$.

- Example: Suppose we are seating five guests around a circular table. If they were in a line, there would be 5! ways to arrange the guests, but we must divide by 5 because there is a 5-to-1 correspondence between permutations of the guests in a line and arrangements of the guests around the table.

8.7 Bookkeeper Method

There are $\frac{10!}{2! \cdot 2! \cdot 3!}$ to rearrange the letters in the word "bookkeeper."

More generally, if we have an n-letter word with $a_1, \ldots, a_k$ occurrences of letter $l_1, \ldots, l_k$, there are $\frac{n!}{a_1! \cdots a_k!}$ ways to rearrange the letters in the word.

8.8 Donuts and Dividers

If we want to assemble a box of n donuts where there are k different flavors of donuts (assume donuts of the same flavor are indistinguishable), there are $\binom{n+k-1}{k-1}$ ways to do so.

We can biject this to a binary string with n zeros and $k-1$ ones, where the zeros represent the donuts and the ones represent the dividers. Then, we can count the number of ways to place the ones in the string.