What I know *after* taking CS 30

The document summarizes a subset of things which you should be knowing after taking CS 30. It is more a collection of facts, tricks and tidbits (rather than concepts).

- **The Arithmetic Sum Formula.**

$$\text{For any integer } n \geq 1, \quad \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

- **The Geometric Sum Formula.**

$$\text{For any integers } a > 1 \text{ and } n \geq 0, \quad \sum_{i=0}^{n} a^i = \frac{a^{(n+1)} - 1}{a - 1}$$

$$\text{For any integers } a < 1 \text{ and } n \geq 0, \quad \sum_{i=0}^{n} a^i = \frac{1 - a^{(n+1)}}{1 - a}$$

  And in particular,

$$\text{For any integer } a < 1, \quad \sum_{i=0}^{\infty} a^i = \frac{1}{1 - a}$$

- **Proofs by Contradiction.**

  The "Suppose not" idea. Extremely useful. Can prove propositions such as

  - $\sqrt{2}$ is irrational.
  - There are infinitely many primes.

- **Sets: Basic Definitions.**
  - *set*, *element*, $\in$, *subset* $\subset$, *superset* $\supset$, *empty set* $\emptyset$, *cardinality* $|\cdot|$.
  - *union* $A \cup B$, *intersection* $A \cap B$, *set difference* $A \setminus B$, *Cartesian product* $A \times B$.
- **Inclusion-Exclusion (baby version).** For any two *finite* sets A and B,

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

- **Functions: Basic Definitions.**
  - *valid function, domain, co-domain, range*.
  - *surjective (range = co-domain), injective (no collisions), bijective (both of the above)*.
- **Countable Sets.**

  A set S is countable if and only if there exists an *injective function* $f : S \to \mathbb{N}$ of natural numbers. Think of this as a *map/hash* from the elements of the set to positive integers which don't collide. Using this map, we can "print out" all the elements of the set in some order (and not necessarily ascending or descending order).

  Couple of tricks to prove a set is countable: (a) make space by multiplying by 2 (or some other number) (b) make space by making it exponent of primes. Using these we get:
  - *Finite sets are countable.*

- *Subset of a countable set is countable.*
- *Set of integers is countable.*
- *More generally, A, B countable means $A \cup B$ is countable.*
- *Therefore, for any* finite *collection $A_1, A_2, \ldots, A_n$, I know $A_1 \cup A_2 \cup \cdots \cup A_n$ is countable.*
- *Set of rationals is countable.*
- *Given countable sets $A_1, A_2, A_3, \cdots$, the* infinite *union $\cup_{n \in \mathbb{N}} A_n$ is countable.*
- *The set of all* strings *over an alphabet is countable.*

- **Uncountable Sets and Diagonalization.**

  A *classic* and *deep* "suppose-not" trick. If the set of reals were countable, then their "binary notation" can be written one on top of the other to create an infinite 2D table. Trick: look at the flip of the diagonal! Some uncountable sets:

  - *The Set of Reals*
  - *The Set of all Boolean functions $f : \mathbb{N} \to \{0, 1\}$*

- **Uncomputable Functions.**

  Since the set of all Boolean functions is uncountable, and the set of all Python programs is countable, there must exist a Boolean function f such that no Python program's I/O behavior mimics that of f. Such a function is *uncomputable*.

- **The Halting Problem.**

  Given a piece of code A and data I, both as strings, *there cannot exist any procedure* which takes (A, I) as input and decides in finite time whether $A(I)$ terminates in finite time or goes into infinite loop. This is also an example of the same diagonalization trick.

- **Modular Arithmetic: Definition.**

  $a \bmod n$ is the *unique* integer $r \in \{0, 1, 2, \ldots, n-1\}$ such that a divided by n leaves remainder r.

  $a \equiv b \bmod n$ or, more concisely, $a \equiv_n b$ if and only if $a \bmod n = b \bmod n$.

- **Operations in Modular Arithmetic.**

  - $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
  - $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
  - $a^b \bmod n = (a \bmod n)^b \bmod n$

- **Algebra in Modular Arithmetic.** Below, a, b, c are all integers, and n is a positive integer. The above operations imply that the following "natural" algebraic operations are kosher.

  - $a \equiv_n b \quad \Rightarrow \quad (a + c) \equiv_n (b + c)$.
  - $a \equiv_n b \quad \Rightarrow \quad a \cdot c \equiv_n b \cdot c$.
  - $a \equiv_n b \quad \Rightarrow \quad a^c \equiv_n b^c$ if $c > 0$.

  But **beware** that the last two implications go only in one direction.That is,

  $$a \cdot c \equiv_n b \cdot c \quad \text{doesn't necessarily imply} \quad a \equiv_n b$$

  So you can't "divide off" c from both sides. To see this, note $2 \cdot 4 \equiv_6 5 \cdot 4 \equiv_6 2$ but $2 \not\equiv_6 5$.

  Similarly,

  $$a^c \equiv_n b^c \quad \text{doesn't necessarily imply} \quad a \equiv_n b$$

  So you can't "take $1/c$th power. To see this, note $5^2 \equiv_8 3^2 \equiv_8 1$, but $5 \not\equiv_8 3$.

- **The Ring of Integers modulo n.**

  The set of possible remainders $\{0, 1, 2, \ldots, n-1\}$ is denoted as $\mathbb{Z}_n$ and is called the *ring* of integers modulo n. "Addition" of two numbers (modulo n) in the ring gives another number in the ring, and so does "multiplication" (modulo n).

- **Modular Exponentiation.** A pretty fast way to compute $a^b \bmod n$.

- **Greatest Common Divisor (GCD).**

  - $\gcd(a, n)$ *is the largest number dividing both* a *and* n.
  - *Euclid's recursive algorithm to find GCD of any two numbers.*
  - *Bezout's Theorem:* $\gcd(a, n) = g$ *implies the existence of two integers* x, y *such that* $xa + yn = g$.
  - *The above* (x, y) *can be found by Extended GCD algorithm.*
  - *In fact,* g *is the* smallest positive *integer which can be written as* $xa + yn$.

- **Co-prime or Relatively prime numbers.**

  Two numbers a, n are co-prime or relatively prime if and only if $\gcd(a, n) = 1$. Co-prime numbers have lots of nice properties. In particular, the following facts are useful (you should be able to prove all of them using Bezout's Theorem mentioned above).

  - *If* $\gcd(a, n) = 1$, *and* $ab \equiv_n 0$, *then* $b \equiv_n 0$. *As a consequence, we get*
  - *If* $\gcd(a, n) = 1$, *and* $a \cdot b \equiv_n a \cdot c$, *then* $b \equiv_n c$.
  - *If a* prime p *divides* a *and* p *divides* b, *then* p *divides* ab.
  - *If* $\gcd(a, n) = 1$ *and* $\gcd(b, n) = 1$, *then* $\gcd(ab, n) = 1$.

- **The Multiplicative Inverse.**

  Co-prime numbers have *inverses*; a supremely helpful fact. For any two pair of coprime numbers a and n, the *multiplicative inverse* of a in the ring $\mathbb{Z}_n$, also called the multiplicative inverse of a modulo n, is the *unique* element b in $\mathbb{Z}_n$ such that $ab \equiv_n 1$. We can use the Extended Euclid's GCD algorithm to compute the multiplicative inverses.

- **Fermat's Little Theorem**.

  For any prime p and number a such that $\gcd(a, p) = 1$, we have

  $$a^{p-1} = 1 \bmod p, \quad \text{or, more concisely,} \quad a^{p-1} \equiv_p 1$$

- **Public Key Cryptography.**

  A conceptual breakthrough due to Diffie and Hellman from 1976 which allowed secrets to be shared without the need for keys to be shared. Diffie-Hellman win Turing Award in 2015.

  - *Alice wants to send a message* m *to Bob.*
  - *Bob generates* **two** *keys: a* ***public*** *key* pk *which is told to all; a* ***secret*** *key* sk *which is only known to him.*
  - *Bob also publishes two algorithms* Enc *and* Dec.
  - *Alice uses* $\mathsf{Enc}(m, pk)$ *to get the encrypted cipher* c.
  - *Bob uses* $\mathsf{Dec}(c, sk, pk)$ *to decrypt the cipher.*
  - *Eve can't figure* m *out given* $\mathsf{Enc}(m, pk)$ *and* pk.

- **RSA protocol.**

  A fantastic algorithm implementing public key cryptography. Invented by Rivest, Shamir, Adleman in 1978. Rivest-Shamir-Adleman awarded Turing award in 2002.

  - *Bob picks two* ***large*** *primes* p, q. *Let* $N := pq$ *and* $\phi := (p-1)(q-1)$.
  - *Bob picks another number* e *such that* $\gcd(e, \phi) = 1$.

- *Bob figures out $d \equiv e^{-1} \bmod \phi$, that is, d is the multiplicative inverse of e in $\mathbb{Z}_\phi$.*
- *Bob's **public key** is (e, N). Bob's **secret** key is d.*
- *Encryption: Alice uses (e, N) to encrypt $m \mapsto m^e \bmod N$.*
- *Decryption: Bob uses (d, N) to decrypt cipher $c \mapsto c^d \bmod N$.*

- **Boolean Variables and Formulas, Propositional Logic.**
  - *Boolean variables take value* true *or* false.
  - *Using various operations ($\wedge, \vee, \Rightarrow$) we can get Boolean formulas from Boolean variables.*
  - *Every formula is defined by its **truth table** which specifies its value depending on all possible values of the variables.*
  - *A Boolean formula on n Boolean variables has 2^n rows in its truth table.*
  - *Two Boolean formulas are **equivalent** if and only if their truth tables are same.*
- **Important Equivalences in Propositional Logic.**
  - **(Negation of Negation.)** $\neg(\neg p) \equiv p$.
  - **(Operation with** true, false**.)** $p \wedge \mathsf{true} \equiv p$; $p \vee \mathsf{true} \equiv \mathsf{true}$; $p \wedge \mathsf{false} \equiv \mathsf{false}$; $p \vee \mathsf{false} \equiv p$.
  - **(Idempotence.)** $p \wedge p \equiv p$; $p \vee p \equiv p$.
  - **(Operation with Negation.)** $p \wedge \neg p \equiv \mathsf{false}$; $p \vee \neg p \equiv \mathsf{true}$.
  - **(Irrelevance.)** $p \vee (p \wedge q) \equiv p$; $p \wedge (p \vee q) \equiv p$.
  - **(Commutativity.)**
    - $p \vee q \equiv q \vee p$.
    - $p \wedge q \equiv q \vee p$.
  - **(Associativity.)**
    - $p \vee (q \vee r) \equiv (p \vee q) \vee r$.
    - $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$.
  - **(Distributivity.)**
    - $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$.
    - $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$.
  - **(Implications as an OR.)** $p \Rightarrow q \equiv \neg p \vee q$.
  - **(De Morgan's Law.)** $\neg(p \vee q) \equiv \neg p \wedge \neg q$; $\neg(p \wedge q) \equiv \neg p \vee \neg q$.
- **Predicate or First Order Logic.**
  - *A **predicate** P is a function from a domain (called the domain of discourse) to* {true, false}.
  - *Given predicates, we can use **quantifiers** ($\forall, \exists$) to define formulas in **predicate logic.***
  - *It's handy to think of $\forall x \in S : P(x)$ as a collection of $\wedge$'s.*
  - *It's handy to think of $\exists x \in S : P(x)$ as a collection of $\vee$'s.*
  - *Using this, we get $\neg\,(\forall x \in S : P(x)) = \exists x : \neg P(x)$, and $\neg\,(\exists x \in S : P(x)) = \forall x : \neg P(x)$*
  - *Quantifiers can be **nested**.*
  - ***Order is super important.** $\forall x, \exists y : P(x, y)$ and $\exists x, \forall y : P(x, y)$ are completely different.*
  - *Negation of a nested quantifier statement is another nested quantifier statement.*

- **Mathematical Induction.** A very simple and powerful method to prove theorems. The "strongest" form goes as follows: To prove a statement of the form $\forall n \in \mathbb{N} : P(n)$, it suffices to do the following.
  - **Base Case.** Check $P(1)$ is true. Sometimes, you need to check $P(1), P(2), \ldots, P(c)$ is true for some finite c.

- **Inductive Case.**
  - Fix $k \in \mathbb{N}$ such that $k \geq c$; c is the point to which you have checked base cases.
  - The *inductive hypothesis*(IH) is that $P(1) \wedge P(2) \wedge \cdots \wedge P(k)$ is true.
  - Use IH to prove $P(k+1)$ is true.

- **Kind of things we proved using induction.**
  - *Summations. Proving* $\sum_{i=1}^{n} i = n(n+1)/2$, $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$, *and so on.*
  - *Divisibility Facts.* 3 *divides* $n^3 - n$, 7 *divides* $3^{2n} - 2^n$.
  - *Integer Combinations. Any number* $n \geq 12$ *can be written as* $3x + 7y$ *for non-negative integers* x, y.
  - *Recurrences. You proved a bunch of facts about Fibonacci numbers.*

- **Strengthening the Induction Hypothesis.** With induction, sometimes non-intuitive things happen. Sometimes, it is *easier* to prove *harder* things.
  - *Recurrence Inequalities. Sometimes having a stronger RHS leads to inductive proofs. This will be very useful in analysis of algorithms.*
  - *The Bernoulli Inequality. For any real* $x \geq -1$ *and any number* n*, we have* $(1+x)^n \geq (1+nx)$*. We used this to establish* $(1 + 1/n)^n \geq 2$ *for all* n*. The latter is a bit painful to do with vanilla induction without the strengthening.*

- **Proving Programs Correct.** Induction *is* the way to prove correctness of recursive algorithms. This needed us to understand "what we induct on" since programs don't always take numbers as input, but take more interesting data types. The "size" is often the thing we induct on, but the right notion is key.

- **The Product Principle.** If we need to count a number of valid length k *sequences*, which satisfies the following property: the first character has N_1 choices, and for every choice of the first character, the second character has N_2 choices, and given any choices of the first two characters, there are N_3 choices for the third character, and so on and so forth, the kth character has N_k choices, then the number of valid sequences is $N_1 \cdot N_2 \cdots N_k$.

  Armed with this, we can count the

  - *Number of length* n *bit strings (Ans:* 2^n*)*
  - *Number of permutations of* $(1, 2, \ldots, n)$ *(Ans:* $n!$*)*
  - *Number of seven letter words with no vowels. (Ans: figure it out!)*
  - *Number of four digit number whose first two digits sum to* 8*. (Ans: figure it out!)*

- **The Sum Principle (I).** If S is the set of items we want to count (that is, we are trying to figure out $|S|$), and S can be partitioned into subsets $A_1, A_2, \ldots, A_k$ which are *pairwise disjoint*, where each A_i is easy to count, then we can count S using $|S| = |A_1| + |A_2| + \cdots + |A_k|$.

  Armed with this, we can count the

  - *Number of four digit number whose first two digits sum to* 8*.*
  - *Number of length* n *bit strings with exactly two ones.*

- **The Sum Principle (II): Inclusion-Exclusion.** At times, it is hard to partition the set S into disjoint subsets. Instead, suppose we can find sets A and B which are easy to count, and $A \cap B$ is also easy to count, and $S = A \cup B$, then we can figure out $|S|$ by using $|S| = |A| + |B| - |A \cap B|$. If we can find three sets A, B, C such that $S = A \cup B \cup C$, and A, B, C and all the intersections are easy to count, then we can figure out $|S|$ by the (toddler) version of inclusion-exclusion: $|S| = |A| + |B| + |C| - |A \cap B| -$

$|A \cap C| - |B \cap C| + |A \cap B \cap C|$. And why stop there; when S is the union of k sets $A_1, \ldots, A_k$ and all the intersections are easy to count, then indeed, $|S|$ can be found by the general inclusion-exclusion formula.

Armed with this, we can count

- *How many numbers between* 1 *and* 100 *are divisible by* 2 *or* 3*?*
- *How many numbers between* 1 *and* 100 *are divisible by* 2*,* 3*, or* 5*?*
- *How many length* n *bit strings start with* 3 *zeros or end with* 3 *ones?*

- **The Bijective Principle.** If we can find a bijection $f : A \to S$ from a set A to our set S, where A is easy to count, then we have counted S since $|S| = |A|$.

  Armed with this, we can count the

  - *Number of subsets of an* n *element universe. Ans : The bijection is from* n *length bit strings.*
  - *Number of odd subsets of an* n *element universe. Ans: The odd sets biject with the even sets.*

- **The Division Principle.** If we can find a mapping $f : A \to S$ from a set A, which is easy to count, to our set S in consideration, such that *for all* $s \in S$, we have $|\{a \in A : f(a) = s\}| = k$, then $|S| = |A|/k$.

  Armed with this, we can count the

  - *Number of anagrams of the word MASSACHUSETTS.*
  - *Number of* n*-length bit strings with exactly* k *ones.*

- **The Four Fold Formula.** If we have to choose k items out of (many copies of) n *distinct items*, how many ways can we do it? The answer depends on whether we are allowed to pick more than one copy of an item (repetition), and whether or not the order in which we pick the k items matters.

	Order Matters	Order Doesn't Matter
Repetition	$\mathbf{n^k}$	$\binom{\mathbf{n+k-1}}{\mathbf{k}}$
No Repetition	$\frac{\mathbf{n!}}{\mathbf{(n-k)!}}$	$\binom{\mathbf{n}}{\mathbf{k}} = \frac{\mathbf{n!}}{\mathbf{k!(n-k)!}}$

- **Combinatorial Identities.** A very useful way of proving identities (equations) is to show that the LHS and the RHS are just two different ways of counting the size of the same set. Using this idea, we can show the following

  - *(Pascal's Identity).* $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.
  - *(Binomial Expansion).* $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$.

  The Binomial Expansion, in turn, implies

  - $\sum_{k=0}^{n} \binom{n}{k} = 2^n$ *(set* $x = 1, y = 1$*).*
  - $\sum_{k=0}^{n} \binom{n}{k} (-1)^k = 0$ *(set* $x = -1, y = 1$*).*
  - $\sum_{k=0}^{n} \binom{n}{k} \frac{1}{2^k} = \left(\frac{3}{2}\right)^n$ *(set* $x = 1/2, y = 1$*).*

- **Experiments and Outcomes: Sample Space** Every time a probabilistic question is asked, figure out the *sample space*: that is, figure out what the unknown random experiment is, and what is the set of possible outcomes.

- **Events.** Figure out the subset of outcomes you are interested in. This subset is the *event* you are interested in.

- **The Probability Distribution.** Finally, we need to figure out the function or the *probability distribution* $\mathbf{Pr} : \Omega \to [0,1]$ such that $\sum_{\omega \in \Omega} \mathbf{Pr}[\omega] = 1$. Given this distribution, we can answer what the chance/likelihood/probability of an event $\mathcal{E}$ is: it is $\sum_{\omega \in \mathcal{E}} \mathbf{Pr}[\omega]$.

  At some level, *modelling assumptions* dictate the distribution. Make as few and as natural assumptions.

- **Tree Diagrams.** For small problem, the tree diagram which starts with our state of the world and goes through all possibilities is a sure-shot way of figuring out the probabilities of all outcomes. It gets unwieldy soon, but very useful for intuition.

- **Operations on Events.**

  - Given an event $\mathcal{E}$, the negation event $\neg\mathcal{E}$ is used to denote the event that $\mathcal{E}$ doesn't take place. That is, it is simply the subset $\neg\mathcal{E} = \Omega \setminus \mathcal{E}$. Sometimes, $\neg\mathcal{E}$ is denoted as $\overline{\mathcal{E}}$.

    $$\mathbf{Pr}[\mathcal{E}] + \mathbf{Pr}[\neg\mathcal{E}] = 1$$

  - Given two events $\mathcal{E}$ and $\mathcal{F}$, the notation $\mathcal{E} \cup \mathcal{F}$ is precisely the union of the subsets in the sample space. $\mathbf{Pr}[\mathcal{E} \cup \mathcal{F}]$ captures the likelihood that at least one of the events takes place.
  - Given two events $\mathcal{E}$ and $\mathcal{F}$, the notation $\mathcal{E} \cap \mathcal{F}$ is precisely the intersection of the subsets in the sample space. $\mathbf{Pr}[\mathcal{E} \cap \mathcal{F}]$ captures the likelihood that both the events takes place.
  - Two events $\mathcal{E}$ and $\mathcal{F}$ are *disjoint* or *exclusive* if $\mathcal{E} \cap \mathcal{F} = \emptyset$. That is, they both can't occur simultaneously. A collection of events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_k$ are *mutually exclusive* if $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ for $i \neq j$.
  - For mutually exclusive events,

    $$\mathbf{Pr}[\mathcal{E}_1 \cup \mathcal{E}_2 \cup \cdots \mathcal{E}_k] = \sum_{i=1}^{k} \mathbf{Pr}[\mathcal{E}_i]$$

  - The Inclusion-Exclusion formula (for two events, aka Baby version) tells us

    $$\mathbf{Pr}[\mathcal{E} \cup \mathcal{F}] = \mathbf{Pr}[\mathcal{E}] + \mathbf{Pr}[\mathcal{F}] - \mathbf{Pr}[\mathcal{E} \cap \mathcal{F}]$$

- **Conditional Probability.** For any two events $\mathcal{A}$ and $\mathcal{B}$, we have

  $$\mathbf{Pr}[\mathcal{A} \mid \mathcal{B}] = \frac{\mathbf{Pr}[\mathcal{A} \cap \mathcal{B}]}{\mathbf{Pr}[\mathcal{B}]}$$

- **Chain Rule.** For any set of events $\mathcal{A}_1, \mathcal{A}_2, \cdot, \mathcal{A}_k$,

  $$\mathbf{Pr}[\mathcal{A}_1 \cap \mathcal{A}_2 \cap \cdots \mathcal{A}_k] = \mathbf{Pr}[\mathcal{A}_1] \cdot \mathbf{Pr}[\mathcal{A}_2 \mid \mathcal{A}_1] \cdot \mathbf{Pr}[\mathcal{A}_3 \mid \mathcal{A}_1 \cap \mathcal{A}_2] \cdots \mathbf{Pr}[\mathcal{A}_k \mid \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{k-1}]$$

- **The Law of Total Probability.** For any two events $\mathcal{A}$ and $\mathcal{B}$, we have

  $$\mathbf{Pr}[\mathcal{A}] = \mathbf{Pr}[\mathcal{A} \mid \mathcal{B}] \cdot \mathbf{Pr}[\mathcal{B}] + \mathbf{Pr}[\mathcal{A} \mid \neg\mathcal{B}] \cdot \mathbf{Pr}[\neg\mathcal{B}] +$$

  More generally, if $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$ are k *mutually exclusive events* with $\sum_{i=1}^{k} \mathbf{Pr}[\mathcal{B}_i] = 1$, then

  $$\mathbf{Pr}[\mathcal{A}] = \sum_{i=1}^{k} \mathbf{Pr}[\mathcal{A} \mid \mathcal{B}_i] \cdot \mathbf{Pr}[\mathcal{B}_i]$$

- **Independence and Conditional Independence.** Two events $\mathcal{A}$ and $\mathcal{B}$ are *independent* if $\mathbf{Pr}[\mathcal{A} \cap \mathcal{B}] = \mathbf{Pr}[\mathcal{A}] \cdot \mathbf{Pr}[\mathcal{B}]$.

  Two events $\mathcal{A}$ and $\mathcal{B}$ are *conditionally independent given another event* $\mathcal{E}$ if $\mathbf{Pr}[\mathcal{A} \cap \mathcal{B} \mid \mathcal{E}] = \mathbf{Pr}[\mathcal{A} \mid \mathcal{E}] \cdot \mathbf{Pr}[\mathcal{B} \mid \mathcal{E}]$.

- **Bayes Rule.** For any two events $\mathcal{A}$ and $\mathcal{B}$, we have

$$\mathbf{Pr}[\mathcal{B} \mid \mathcal{A}] = \frac{\mathbf{Pr}[\mathcal{A} \mid \mathcal{B}] \cdot \mathbf{Pr}[\mathcal{B}]}{\mathbf{Pr}[\mathcal{A}]}$$

  The most common use is when $\mathcal{B}$ indicates a *hypothesis*, $\mathbf{Pr}[\mathcal{B}]$ is our *prior* belief about the hypothesis, $\mathcal{A}$ is an *outcome*, and $\mathbf{Pr}[\mathcal{B} \mid \mathcal{A}]$ is our *posterior* belief about the hypothesis given the outcome has occurred.

- **Random Variables.** A random variable is a *function/mapping* $X : \Omega \to$ Range from the set of outcomes to a range. Usually the range is the set of natural numbers, but it could be reals, integers, etc.

- **Expectation of a Random Variable.** The expectation of a random variable is an "weighted average" defined as

$$\mathbf{Exp}[X] := \sum_{\omega \in \Omega} X(\omega) \cdot \mathbf{Pr}[\omega]$$

- **Linearity of Expectation.** For any k random variables $X_1, X_2, \ldots, X_k$, we have

$$\mathbf{Exp}[\sum_{i=1}^{k} X_i] = \sum_{i=1}^{k} \mathbf{Exp}[X_i]$$

  One cannot understate the importance of this above fact.

- **Independent Random Variables.** Two random variables X and Y are *independent* if for any x, y in their ranges

$$\mathbf{Pr}[X = x, Y = y] = \mathbf{Pr}[X = x] \cdot \mathbf{Pr}[Y = y]$$

  k random variables $X_1, X_2, \ldots, X_k$ are *pairwise independent* if any two of them are independent. They are *mutually independent* if for any $x_1, x_2, \ldots, x_k$, we have

$$\mathbf{Pr}[X_1 = x_1, X_2 = x_2, \cdots, X_k = x_k] = \prod_{i=1}^{k} \mathbf{Pr}[X_i = x_i]$$

- **Expectation of Product of Mutually Independent Random Variables.** If $X_1, \ldots, X_k$ are mutually independent random variables, then

$$\mathbf{Exp}[\prod_{i=1}^{k} X_i] = \prod_{i=1}^{k} \mathbf{Exp}[X_i]$$

- **Variance of a Random Variable.** Given a random variable X, the variance $\mathbf{Var}[X]$ is defined as

$$\mathbf{Var}[X] := \mathbf{Exp}[(X - \mathbf{Exp}[X])^2] = \mathbf{Exp}[X^2] - (\mathbf{Exp}[X])^2$$

  The *standard deviation* is defined as

$$\sigma(X) := \sqrt{\mathbf{Var}[X]}$$

- **Linearity of Variance for Pairwise Independent Random Variables.** Given k *pairwise* independent random variables $X_1, \ldots, X_k$, we have

$$\mathbf{Var}[\sum_{i=1}^{k} X_i] = \sum_{i=1}^{k} \mathbf{Var}[X_i]$$

- **Concentration around the mean: Chebyshev's Inequality** For any random variable X and for any $t > 0$, we have

$$\mathbf{Pr}\big[\ |X - \mathbf{Exp}[X]|\ \ \geq\ \ t\big]\ \ \leq\ \ \frac{\mathbf{Var}[X]}{t^2}$$

  As a corollary we get that the probability X is ***not*** in the range $\big[\mathbf{Exp}[X] - c\sigma(X), \mathbf{Exp}[X] + c\sigma(X)\big]$ is at most $\frac{1}{c^2}$.

- **Graphs: Notations and Definitions.**
  - *Given an edge $e = (u, v)$, the vertices u and v are the **endpoints** of e. We say e **connects** u and v. We say that u and v are **incident** to e.*
  - *Two vertices $u, v \in V$ are **adjacent** or **neighbors** if and only if (u, v) is an edge.*
  - *The **incident edges** on v is denoted using the set $\partial(v)$. So,*

$$\partial_G(v) := \{(u, v) : (u, v) \in E\}$$

    We lose the subscript if the graph G is clear from context.
  - *Given a vertex v, the **neighborhood** of v is the set of neighbors of v. This is denoted sometimes as $N(v)$ or sometimes as $\Gamma(v)$. So,*

$$N_G(v) := |\{(u, v) : (u, v) \in E\}|$$

    if the graph G is clear from context.
  - *The cardinality of $N_G(v)$ is called the **degree** of vertex v. We denote it using $\deg_G(v)$. This counts the number of neighbors of v. Note that,*

$$\deg_G(v) = |N_G(v)| = |\partial_G(v)|$$

  - *A vertex v is **isolated** if its degree is 0. That is, it has no edges connected to it.*
  - *A graph $G = (V, E)$ is called **regular** if all degrees are equal, that is, $\deg_G(v) = \deg_G(u)$ for all u and v.*
  - *Given a graph $G = (V, E)$, we use $V(G)$ to denote V and $E(G)$ to denote E. This notation is useful when we are talking about multiple graphs.*

- **The Handshake Lemma.** In any graph $G = (V, E)$,

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

- **Perambulations in Graphs.** Fix $G = (V, E)$
  - *A **walk** w in G is an* alternating sequence *of vertices and edges*

$$w = (v_0, e_1, v_1, e_2, v_2, \ldots, e_k, v_k)$$

    such that the ith edge $e_i = (v_{i-1}, v_i)$ for $1 \leq i \leq k$. Both edges and vertices can repeat.
  - *A **trail** t in G is a walk with no edges repeating.*
  - *A **path** p in a graph G is a walk with no* vertices repeated.
  - *A **closed walk** is a walk whose origin and destination are the same vertex.*
  - *A **circuit** is a closed trail of length at least 1.*
  - *A **cycle** is a circuit with no vertex other than the source and destination repeating.*

- **Connectivity, Forests, and Trees.**

- u is ***reachable*** from v in G if there is a walk from u to v in G. A graph G is ***connected*** if any vertex is reachable from another vertex.
- Walk from u to v implies a path from u to v.
- A ***forest*** is a graph with no cycles.
- A ***tree*** is a forest which is connected.

- **Tree Theorem.**

  Let $G = (V, E)$ be a graph. The following are equivalent statements.

  1. *G is a tree.*
  2. *G has no cycles and adding any edge to G creates a cycle.*
  3. *Between any two vertices in G there is a unique path.*
  4. *G is connected, and deleting any edge from G disconnects the graph, and the resulting graph has exactly two connected components.*
  5. *G is connected and $|E| = |V| - 1$.*
  6. *G has no cycles and $|E| = |V| - 1$.*

- **Bipartite Graphs.**

  A graph $G = (V, E)$ is ***bipartite*** if the vertex set V can be partitioned into $V = L \cup R$ and $L \cap R = \emptyset$ such that *every* edge (x, y) has exactly one endpoint in L and the other endpoint in R.

  $$G \text{ is bipartite} \quad \Leftrightarrow \quad G \text{ has no cycles of } \textbf{\textit{odd}} \text{ length}$$

- **Matchings.**

  A ***matching*** $M \subseteq E$ is a subset of edges such that no two edges in M share an endpoint. In other words, a matching is a collection of *pairwise disjoint* set of edges.

- **Matchings in Bipartite Graphs: Hall's Theorem.**

  Let $G = (L \cup R, E)$ be a bipartite graph. A matching M is an L-matching if every vertex of L is an endpoint of some edge in M. *G has an L-matching if and only if*

  $$\text{For every subset } S \subseteq L, \ |N_G(S)| \geq |S|$$