What I know *after* taking CS 30

The document serves as a review of what we have covered so far.

1 Jargon

Sets

- **Sets: Basic Definitions.**
  - *roster notation, set builder notation*
  - *element* $\in$*, subset* $\subset$*, superset* $\supset$*, empty set* $\varnothing$*, 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

- $f : A \to B$, *domain*: A, *co-domain*: B, *range*: $\{f(a) : a \in A\}$.
- *surjective (range = co-domain), injective (no collisions), bijective (both of the above).*
- *How* to prove a function is surjective? injective? ("To this end, fix a ..." style)
- *composition* of functions: $f : A \to B, g : B \to C$ gives $(g \circ f) : A \to C$.

Logic

- **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*** *specifying its value on all possible settings of the variables.*
  - *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.**) $p \vee (p \wedge q) \equiv p$; $\quad p \wedge (p \vee q) \equiv p$.
- (**Commutativity.**) $p \vee q \equiv q \vee p$. $\quad p \wedge q \equiv q \wedge p$.
- (**Associativity.**) $p \vee (q \vee r) \equiv (p \vee q) \vee r \quad p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$
- (**Distributivity.**) $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r) \quad p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
- (**Implications as an OR.**) $p \Rightarrow q \equiv \neg p \vee q$.
- (**De Morgan's Law.**) $\neg(p \vee q) \equiv \neg p \wedge \neg q$; $\quad \neg(p \wedge q) \equiv \neg p \vee \neg q$.

- **Predicate or First Order Logic.**
  - *A **predicate** P is a function from a domain (called the domain of discourse) to* $\{\mathsf{true}, \mathsf{false}\}$.
  - *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 $\neg(\exists x \in S : P(x)) = \forall x : \neg P(x)$*
  - ***Caution:*** *$\forall$'s and $\vee$'s* do not distribute. *That is, $\forall x \in S : (P(x) \vee Q(x))$ is in general different from $(\forall x \in S : P(x)) \vee (\forall x \in S : Q(x))$.*
  - *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.*

2 Proofs

By Contradiction

- To prove a proposition p, one **assumes** the opposite/negation, and then **deduces** something absurd.

  If $\neg p$, then prove q occurs and *$\neg q$ occurs, for some possibly different proposition q.*

- Usual structure:
  - *Suppose not. That is $\neg p$ is true.*
  - *Then **interpret** what $\neg p$ means in English. That is, what does the **converse** mean.*
  - *Think why this can lead to q and $\neg q$ for some q*
- Examples seen in class
  - $\sqrt{2}$ is irrational.
    - *If not, there are two integers a and b such that $\sqrt{2} = \frac{a}{b}$.*
    - *Can assume $\gcd(a, b) = 1$.*
    - *Deduce, by taking squares, both a and b are even.*
  - Infinitely many primes.
    - *If not, there are finitely many primes. And so there is a largest prime q.*
    - *Trick: Look at $Q := q! + 1$.*
    - *Q is not a prime, and yet no prime divides it.*

By Induction

- Used to prove statements of the form

$$\forall n \in \mathbb{N} : P(n)$$

- Usual structure:
  - *First figure out what $P(n)$ is.*
  - ***Base Case:*** Establish $P(1)$ is true. Sometimes, one needs to do a bit more, that is, prove $P(2), P(3)$ is also true.
  - **Inductive Case:** Then try to show for ***any*** k greater than or equal to the last base case checked,

    Assuming $P(k)$ is true, prove that $P(k+1)$ is true.

    or, even better

    Assuming $P(1), P(2), \ldots, P(k)$ is true, prove that $P(k+1)$ is true.

The former is called weak induction, the latter is called strong induction. This is just jargon. Use strong induction whenever you can.
  - Since we have to show the inductive case for *every* k, this is done by first **fixing** a k.

- Induction can also be used for proving statements which don't obviously look of the form $\forall n \in \mathbb{N} : P(n)$. For example, induction was used to show the correctness of code, that is, statement of the form

  For every possible input, the code works

  One way to cast it into the $\forall n \in \mathbb{N} : P(n)$ format, one defines $P(n)$ as true if

  For every possible input **of size** n*, the code works*

  We will see this principle more when we talk about graphs.

General Proof Writing Principles

- Proofs are written in many drafts. The first draft has a vague collection of ideas. The second refines it. The third perfects it.

- Whenever you see a statement of the form

  "Prove for all/every/any object *foo* with property *bar*"

  Fix an arbitrary such object *foo* and call it k (or Sam, if it helps you think). Why? Because in the subsequent arguments you can keep referring to this particular k.

- Similarly, when you deduce

  "There exists an object *foo* with property *bar*"

  again, call it something – k or Mary, or whatever works.

- More generally, use names (variables) as much as possible. This *frees up mental space*.

- On that note, write every thought you have down on paper. Less stuff to keep in your head.

- A proof is nothing but a *story* – and just like stories, they get better with rewriting.

3 Combinatorics

- **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 exactly 5. (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.*

  Another way this principle is used when one wants to count the "negation" of the set. That is, if S is a subset of a certain "universe" U such that (a) U is easy to count, and (b) $U \setminus S$ is easy to count, then the sum principle shows that $|S| = |U| - |U \setminus S|$. Armed with this, we can count the

  - *Number of 8-bit strings with at least 2 ones.*
  - *Number of 5 digit numbers with at least one digit odd.*
  - *Number of 4 letter words with at least one consonant.*

- **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* 10 *bit strings start with* 3 *zeros or end with* 3 *zeros?*

- **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 $|S| = |A|$.

  Using this, we see that *Subsets of an* n*-sized set is in* ***one-to-one*** *correspondence with* n*-length bit strings.* Armed with this, we can count

  - *The cardinality of the power set, that is, the number of subsets of a* n *sized set.*
  - *The number of subsets containing a particular element* a*.*
  - *The number of odd-cardinality subsets.*

- **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 ways we can arrange* 5 *red balls,* 4 *orange balls, and* 3 *yellow balls in a line.*

  Indeed, we have the following formula

  > If we have n_1 objects of type 1, n_2 objects of type 2, and so on, till n_k objects of type k, then the number distinct ways we can arrange them in a line is
  >
  > $$\frac{(n_1 + n_2 + \cdots + n_k)!}{n_1! n_2! \cdots n_k!}$$

  In particular, if the object of type 1 is the number one and there are k of them, and the object of type 2 is the number zero and there are $(n - k)$ of them, for some two numbers n and k, then the number of ways of arranging k ones and $(n-k)$ zeros in a line is $\frac{n!}{k!(n-k)!}$. But this is ***precisely*** the number of n length bit strings with exactly k ones. Using the equivalence (see above) between subsets and bit strings, we see that

  > The number of subsets of size exactly k of an n sized set is precisely $\frac{n!}{k!(n-k)!}$

  This quantity is called $\binom{n}{k}$ pronounced "n choose k" — it is the number of ways one can choose (an unordered collection) of k objects from n objects.

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

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