Welcome to Codebreaking at Cal! These notes will serve as a theoretical resource to supplement the material in the slides and labs. Credit to Alec Li and CS 70 for LaTeXtemplates and style inspiration.

Notation

As in standard in many mathematical subjects, there is quite a lot of notation which may be confusing at first glance. This note seeks to outline the notation commonly used in math and cryptography. It is not intended to be a lesson in propositional logic, set theory, or other math – please visit resources like CS 70's website for that.

Propositional Logic

- $\forall$ = For all, $(\forall x \in \mathbb{N})(1 \mid x)$ states that for all natural numbers x, 1 divides x.
- $\exists$ = Exists, $(\exists x \in \mathbb{N})(x = 7)$ states that there exists some x in the natural numbers such that x equals 7.
- $\neg$ = Not, $\neg\ (x = 5)$ states that "x=5" is not true.
- $\implies$ = Implies, $(x \mid 8 \implies x \mid 4)$ states that if x divides 8, then it must also divide 4.
- $\iff$ = If and only if (alternatively equal to "equivalent"), means $\implies$ from both directions.

Sets

- $\in$ = Within ($5 \in \{5, 8\}$)
- $\cap$ = Intersection (only keep what both sets have)
- $\cup$ = Union (combine both sets)
- $\setminus$ = Set difference (remove all elements of B from A)
- $\emptyset$ = Empty set (set with no elements)
- $\mathbb{N}$ = Natural numbers (1, 2, ...)
- $\mathbb{Z}$ = Integers (-2, -1, 0, 1, 2 ...)
- $\mathbb{Q}$ = Rationals (3.5, -5.8, ..), can be represented as $\frac{a}{b}$ for $a, b \in \mathbb{Z}$
- $\mathbb{R}$ = Reals ($\sqrt{2}$, 3, ..), can be represented as the union of rational and irrational numbers

Bit Operations

- & - AND
  - 1 & 1 = 1
  - 1 & 0 = 0
  - 0 & 1 = 0
  - 0 & 0 = 0
- $\|$ - OR
  - 1 | 1 = 1
  - 1 | 0 = 1
  - 0 | 1 = 1
  - 0 | 0 = 0
- $\sim$ - NOT
  - $\sim$ 1 = 0
  - $\sim$ 0 = 1
- $\oplus$ - XOR
  - 1 $\oplus$ 1 = 0
  - 1 $\oplus$ 0 = 1
  - 0 $\oplus$ 1 = 1
  - 0 $\oplus$ 0 = 0

Asymptotic Notation

When we discuss algorithms, we will need a way to describe their behavior "objectively". Big-O notation represents the complexity of an algorithm as its input approaches infinity.

NOTE: These are quite informal definitions of asymptotic complexity notation. If you already know what they are, feel free to skip.

- O: The "upper-bound" set of algorithms. An algorithm is $\in O(X)$ if it runs in at most X time. For example, an algorithm to read each member of the list takes N time for N elements, and is thus $O(n)$. Note, however, that is technically $O(n^2)$, $O(n^3)$, etc for all $X \geq n$, since these are always larger than n. For the purposes of this class, we will only consider the tightest upper bound.
- Ω: The lower-bound set of algorithms. An algorithm is $\in \Omega(X)$ if it runs in at least X time. Our earlier example is also $\Omega(n)$ as it must check every element regardless. Similarly, we only care about the tightest lower bound.

- Θ: The "exact" set of algorithms. An algorithm is $\in \Theta(X)$ if it is both $O(X)$ and $\Omega(X)$. Therefore our list-reading algorithm is $\Theta(N)$.

When analyzing algorithms, we drop all constant and lower-order terms, as they are insignificant as the input size approaches infinity.

For example, $O(2n^2+5) = O(n^2)$.