Complexity Theory (in 10 minutes)

1. Representations
2. Decision problems (Languages)
3. Runtime & computation model
4. Complexity classes

Representations

Key idea: General computational problems can be phrased as functions

$$f: \mathbb{N} \to \mathbb{N}$$

by encoding inputs/outputs as natural numbers.

Concretely, can use encoding:

input or output $\longrightarrow$ binary string $\longrightarrow$ natural number.

After picking one such encoding, can consider problems as functions $\mathbb{N} \to \mathbb{N}$!

Important point: Different choices of encodings can lead to slightly different input/output sizes, and thus slightly different runtime.

BUT different "reasonable" encodings will differ by a polynomial factor.
↳ E.g. adjacency list vs. adjacency matrix

So definition of "solvable in poly time" will be well-defined relative to choice of encoding.

Decision problems & languages

A language is a set $L \subseteq \mathbb{N}$.

A decision problem is a function $\mathbb{N} \to \{0,1\}$.

Observation: Languages & decision problems are the same thing. Consider the correspondence

$$\{\text{Languages}\} \longrightarrow \left\{\begin{matrix}\text{Decision} \\ \text{problems}\end{matrix}\right\}$$

$$L \longmapsto \left(x \longmapsto \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases} \right).$$

χ_L, characteristic function

Intuition: A language is a subset $L \subseteq \mathbb{N}$. The corresponding decision problem is "Tell me whether $x \in \mathbb{N}$ is an element of L"

key point: Arbitrary computational problems $\mathbb{N} \to \mathbb{N}$ can be turned into decision problems by "thresholding" on an additional argument.

cutoff

$$x \longmapsto f(x)$$

turns into

$$\langle x, k \rangle \longmapsto \begin{cases} 1 & f(x) \geq k, \\ 0 & f(x) < k. \end{cases}$$

This makes life easier when comparing different problems!

Don't need to translate between "units" of different outputs.

Runtime & computational model

The runtime of an algorithm is measured by worst-case on each input size, i.e.,

$$\boxed{T(n) = \max_{|x|=n} s(A,x)},$$

where $s(A,x)$ = # steps of computation used by algo A on input x.

But what exactly is a "step of computation"?

Requires formally defining computational model.

One of Alan Turing's greatest contributions:
the Turing Machine model.

Intuition:

- Infinitely long tape with input, followed by blank cells

0	1	1	0	1	⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔	⊔

- Turing Machine can read/write to tape, move left/right, and change its internal state (among finitely many states).

Other computational models (e.g. RAM) are slightly different, but only by polynomial factor in runtime.

↳ I.e., TM simulates RAM with poly overhead, and vice versa.

Complexity Classes

Now comes the fun part!

P = the set of decision problems solvable in polynomial time

Notice: Complexity classes contain problems, not algorithms!

↳ "Max-flow is in P" ✓

↳ "Edmonds-Karp is in P" ✗

EXP = decision problems solvable in exponential time
$O(2^{poly(|x|)})$.

NP = decision problems that can be verified in polynomial time.

Definition — Let $P : \mathbb{N} \to \{0,1\}$ be a computational problem and denote $L = \{x \in N : P(x) = 1\}$. We say that P is in **NP** if there exists an algorithm $A(\cdot,\cdot)$ such that:

- If $x \in L$, there exists a string y such that $A(x,y) = 1$;
- If $x \notin L$, then for every y, $A(x,y) = 0$;
- For every x and y, the running time of $A(x,y)$ is polynomial in $|x|$ and $|y|$.

Lemma: $NP \subseteq EXP$

Proof idea: Try all certificates!

Lemma: $P \subseteq NP$.

Proof: To verify a candidate solution, simply solve the problem outright!

Conjecture: $P \subsetneq NP$.

$\Uparrow$

Biggest open problem in all of CS, maybe all of math!