P vs NP

Arno Pauly

April 19, 2021

Some questions

- We've talked about what we can and what we cannot compute – but what can we compute reasonably fast?
- What would fast even mean?
- Are non-deterministic Turing machines a thing?
- What kind of questions can theoretical computer scientists not solve?

Run-time

1. Fix a machine-style model of computation.
2. The time complexity of a given machine M is the function $T_M : \mathbb{N} \to \mathbb{N}$ where $T(n)$ is the maximum number of steps taken on an input of length n.
3. A machine runs in *polynomial time*, if there are natural numbers a, b, k such that $\forall n \in \mathbb{N}\ T_M(n) \leq an^k + b$.

Efficient Church-Turing thesis

For reasonable deterministic models of computation, the notion of polynomial time coincides.

The efficient Church Turing thesis

Efficient Church-Turing thesis

For reasonable deterministic models of computation, the notion of polynomial time coincides.

- So we don't need to specify whether we are talking 1-tape TMs, 2-tape TMs, Java programs, Python programs, register machines, etc, when talking about polynomial time.
- It is a convenient over-simplification to consider polynomial time computability to formalize what is practically computable.

Connecting to previous classes

- Every regular language is decidable in time $T(n) = n$.
- Context-free languages are decidable in time polynomial-time.
- It is unknown whether all context-sensitive languages are decidable in polynomial time (but the answer is expected to be **no**).

Non-deterministic Turing machines

A non-deterministic Turing machines can have multiple potential instructions applicable in a configuration (eg write 0 and move right or write 1 and move left).

Definition

A non-deterministic TM M decides a language L in time T iff

- For every word w, M never takes more than $T(|w|)$ steps, no matter how the non-deterministic choices are resolved.
- There is a way to resolve the non-deterministic choices on input w in a way to reach the yes-state iff $w \in L$.

Defining *P* and *NP*

Definition

Let P be the class of all languages decidable in polynomial time (by a deterministic TM). Let NP be the class of all languages decidable in polynomial time by a non-deterministic TM.

Question

Is $\mathrm{P} = \mathrm{NP}$*?*

- We don't know! (Almost everyone thinks the answer is **no**.
- We know of a lot of proof techniques that they don't work for this.
- (Oversimplified) If the answer is **yes**, cryptography doesn't work.

Karp-reduction

Definition

We say that a language L_1 is Karp-reducible to a language L_2, if there is a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that $w \in L_1 \Leftrightarrow f(w) \in L_2$. We write $L_1 \leq_p L_2$.

- If $L_2 \in \mathrm{P}$ and $L_1 \leq_p L_2$, then $L_1 \in \mathrm{P}$.
- If $L_2 \in \mathrm{NP}$ and $L_1 \leq_p L_2$, then $L_1 \in \mathrm{NP}$.
- If $\emptyset \neq L_2 \neq \Sigma^*$ and $L_1 \in \mathrm{P}$, then $L_1 \leq_p L_2$.

NP-completeness

Definition

We call a language L NP-complete, if $L \in \mathrm{NP}$ and for every $L_2 \in \mathrm{NP}$ it holds that $L_2 \leq_p L$.

- We know an insane amount of NP-complete languages.
- If we find a polynomial-time algorithm for a single NP-complete language, then $\mathrm{P} = \mathrm{NP}$.
- If we can prove for a single NP-complete language that it is not in P, then $\mathrm{P} \neq \mathrm{NP}$.

Examples of NP-complete languages I

Definition

A Hamiltonian cycle in a graph is a cycle visiting each vertex exactly once.

Proposition

The languages of all graphs having a Hamiltonian cycle is NP*-complete.*

Satisfiability

Definition

The instances of SAT are formulas built up from boolean variables $x_0, x_1, \ldots$, negation $\neg$ and *and* $\wedge$ and *or*. An instance is positive, if there is an assignment to the variables making the formula true.

Proposition

SAT *is* NP*-complete.*

Probabilistic and Quantum

- Besides non-determinism, we can also consider probabilistic or Quantum TMs.
- We can define what polynomial time means in these models.
- But we haven't fared any better in figuring out whether or not these models can actually do more in polynomial time than deterministic TMs.