P vs NP, Reductions, NP-Completeness, Cook-Levin Theorem, Bandersnatches – Lecture 17

James Marshall

P vs NP

There is clearly some relationship between the complexity classes **P** and **NP**

Languages that can be decided in polynomial time can also have certificates checked in polynomial time

or *?*

$$\mathrm{NP} \subseteq \mathrm{EXPTIME} = \bigcup_k \mathrm{TIME}\left(2^{n^k}\right)$$

Reductions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is *polynomial time computable* if some deterministic TM exists that runs in polynomial time, and halts with only *f(w)* on its tape when started on any input *w*.

Definition

Some language A is *polynomial time mapping reducible* (or *polynomial time reducible*) to language B (written $A \leq_P B$), if a polynomial time computable function f exists where, for every w

$$w \in A \Leftrightarrow f(w) \in B.$$

f is the *polynomial time reduction* of A to B.

Theorem

If $A \leq_P B$ and $B \in \mathbf{P}$, then $A \in \mathbf{P}$

Proof (sketch)

B can be decided in polynomial time, A can be reduced to B in polynomial time, hence A can be decided in polynomial time.

Satisfiability

$$SAT = \{\langle \phi \rangle | \phi \text{ is a satisfiable Boolean formula}\}$$

Examples

$$3SAT = \{\langle \phi \rangle | \phi \text{ is a satisfiable } 3\text{-cnf formula}\}$$

Examples

Definition

A language *B* is ***NP-complete*** if:

1. *B* is in **NP**
2. every *A* in **NP** is polynomial time reducible to *B*

Theorem (Cook-Levin)

3SAT is NP-complete

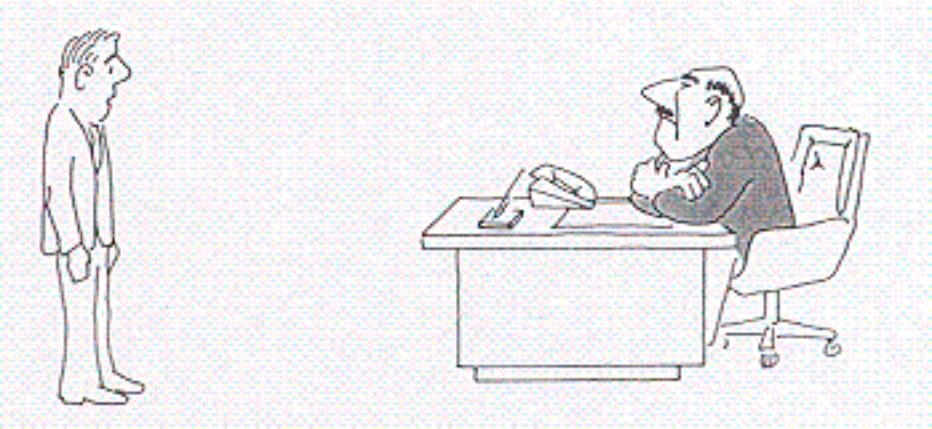
Proof (sketch)

1. *3SAT* is obviously in **NP**, as we can verify a possible satisfying assignment in polynomial time.

2. We can construct a polynomial time reduction to *3SAT* for any language *A*, by first considering a non-deterministic TM for deciding *A*, called *M*. Then the reduction for *A* takes a string *w* and writes a Boolean formula ϕ that simulates *M*'s operation on it, such that if *M* accepts *w* then there is a satisfying assignment for ϕ, and if *M* doesn't accept *w* then there is no satisfying assignment for ϕ. A Boolean formula can simulate the operation of another machine, as the Boolean operators are analogous to the logic gates used in electronic circuitry to construct 'real-world' computers.

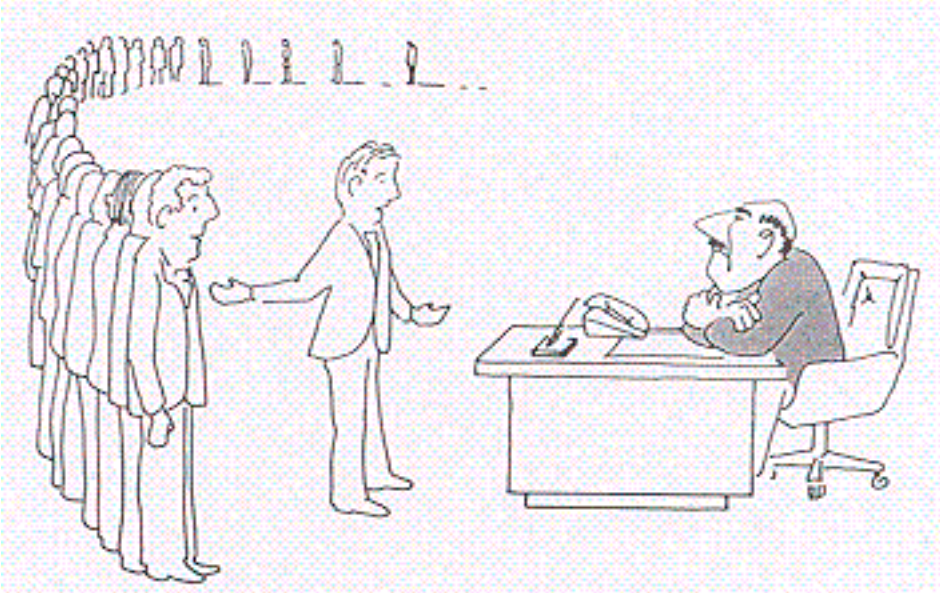
Corollary

$SAT \in \mathbf{P}$ iff $\mathbf{P} = \mathbf{NP}$



'I can't find an efficient algorithm, I guess I'm just too dumb.'

'I can't find an efficient algorithm, because no such algorithm is possible.'



'I can't find an efficient algorithm, but neither can all these famous people.'

(©1979 Garey & Johnson, from 'Computers and Intractability: A Guide to the Theory of NP-Completeness')