Algos: P vs NP

- Termination Analysis: examples like Yes, No, (?)
  - 1936: "$\exists$ program I can tell if another program(input) halts or thinks"
    - unbounded memory usage

- Tractable: done easily (e.g. in polynomial time)
  intractable: done reeeeally (e.g. in exp. time) as inputs grow in size
  - polynomial: with input size n, $O(n^k)$
    - $O(1)$, $O(\log n)$, $O(n)$, $O(n^2)$, $O(n^3)$...

- if $f(n)$ takes microseconds
  - $f(n) = \sqrt{n}$ ☺ 1 second $\rightarrow n = 1 \times 10^{12}$
  - $f(n) = \log n$ ☺ 1 second $\rightarrow n = 2^{1 \times 10^6}$

- "NP = (P) — polynomial/tractable problems

	Computational Challenge	Halting Problem
Polynomial	NP, Decidable	Undecidable

- NP = NON-deterministic polynomial
  - multiple decisions at once
  - set of ALL possible decisions from a given state

Hardest NP problems

- NP-complete: decision problems which are
  - NP
  - as "hard" as ANY NP problem
- (!) decision problems have binary output
  - Complexity: $O(2^n)$
  - (Circuit V Formula) Satisfiability - program equivalence checking
- if problem is NP-complete, then it is intractable UNLESS: $P = NP$ (i.e. any NP problem solved in polynomial time)
- Shortest simple path: Dijkstra algo
  - Polynomial
- LONGEST: NP-Complete
- Euler tour: visiting all edges of graph exactly once
- Hamiltonian cycle: each vertex visited exactly once
  - Solution: since $\exists$ (node!) possible permutations, O(node!) runtime
- $(\neg A \vee B) \wedge A \wedge \neg B$ → unsatisfiable
  - Enumeration: one approach to check
  - Deduction

$$\frac{A \rightarrow B \quad A}{B} \qquad \frac{B \quad \neg B}{\bot}$$

- IF a THEN b ELSE c → $(a \rightarrow b) \vee (\neg a \rightarrow c)$

conjuctive normal form!

- 3-CNF: AND of clauses, each clause has 3 distinct literals
- reductibility: