NP-COMPLETENESS: a brief informal introduction

we've seen algorithms with several time complexities:

- POLYNOMIAL:
  - $O(n)$
  - $O(n \log n)$
  - $O(n^2)$
  - $O(n^c)$
- $O(2^n)$
- $O(w(n^c))$

Focus on decision problems

- $O(n^c)$ NP
  - haven't found $O(n^c)$ yet, but we can verify solutions in $O(n^c)$

Not polynomial
NP: non-deterministic polynomial
(NOT "non-polynomial")

The $1,000,000$ question...

$P \text{ v. } NP : = \text{ or } \neq ?$

Is it ever "much" harder to solve a decision problem than it is to verify a solution, if the verification takes poly-time?

$T(\text{verify}) = o(n^c \cdot T(\text{solve}))$

$T(\text{solve}) = \omega(n^c \cdot T(\text{verify}))$

within a polynomial factor?

considered ~equivalent}
There are thousands of problems for which no known polynomial-time solution is known, yet we can verify proposed solutions in poly-time.

e.g. Hamiltonian cycle

given a graph, find a cycle that visits each vertex exactly once.

\( \Rightarrow \text{decide if one exists} \)
Decision Problem

"is there a set of k independent vertices?"

Optimization Problem

"find the largest independent set"

(binary search on k: 0...|V|)

Often, optimization problems are not polynomially harder than decision.
**NP-COMPLETE PROBLEMS**

1) in \( NP \), & not known to be in \( P \)

(decision problems with solutions that can be verified in poly-time, but for which no poly-time algo is known)

2) if you ever find a polynomial-time solution for any \( NPC \) problem, this implies the same for every problem in \( NP \). \( \rightarrow P = NP \)

\( \downarrow \) if you ever prove that an \( NPC \) problem has no poly-time algo, then no \( NPC \) problem does \( \rightarrow P \neq NP \)
Are there other problems in \( NP \) but not in \( P \) or \( NPC \)?

- if \( P = NP \) then N/A.
- if \( P \neq NP \) then yes. [theorem]

few "natural" problems
(almost everything in \( NP \) is \( P \) or \( NPC \))

If we solved such a problem in poly-time, it would just move into \( P \) without dragging everything else along.
NP-hard problems

- As hard as any NP problem.
- NPC problems are NP-hard.
- NP-hard need not be NPC
  - Might not be decision problems
  - Or might not have poly-time verification.
- Like NPC, solving an NP-hard problem quickly means same for all NP

NPC = NP-hard & in NP

Independent set

Decision (≤k?)

Optimization (max k)

NP-hard
AN IMPORTANT DETAIL just mentioned here

For NPC problems, we measure input in terms of a finite alphabet [e.g. binary: represent k with $\Theta(\log_2 k)$ bits]

—unlike our treatment of constants so far