NP-COMPLETENESS: a brief informal introduction

We've seen algorithms with several time complexities:
NP-COMPLETENESS: a brief informal introduction

We've seen algorithms with several time complexities:

Focus on decision problems.

NP: haven't found \( O(n^c) \) yet, but we can verify solutions in \( O(n^c) \).

P: \( O(n^c) \).

Not polynomial: \( w(n^c) \).
NP: non-deterministic polynomial
(NOT "non-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?
NP: non-deterministic polynomial (NOT "non-polynomial")

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

$P \lor NP : = \not= ?$

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

within a polynomial factor?

considered ~ equivalent

\[ T(verify) = o(n^c \cdot T(solve)) \]
\[ T(solve) = \omega(n^c \cdot T(verify)) \]
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.

\( \Rightarrow \) decide if one exists
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.

⇒ decide if one exists
"is there a set of $k$ independent vertices?"

(independent: no neighbors)
**Decision Problem**

"is there a set of $k$ independent vertices?"

(independent: no neighbors)

**Optimization Problem**

"find the largest independent set"
**Decision Problem**

"is there a set of k independent vertices?"

**Optimization Problem**

"find the largest independent set"

(independent: no neighbors)

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)
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.  \( \rightarrow P = NP \)
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. $\Rightarrow P = NP$

$\Leftarrow$ 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?
Are there other problems in NP but not in P or NPC?

- If P=NP then N/A.
Are there other problems in NP but not in P or NPC?

- If $P=NP$ then N/A.
- If $P \neq NP$ then yes. \[\text{[theorem]}\]

Few "natural" problems (almost everything in NP is P or NPC)
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 are as hard as any NP problem.
NP-hard problems

- As hard as any NP problem.
- NPC problems are NP-hard.
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
NP-hard problems
\(\Rightarrow\) as hard as any NP problem.
- NPC problems are NP-hard.
- NP-hard need not be NPC
  \(\Rightarrow\) might not be decision problems
  \(\Rightarrow\) might not have poly-time verification.

independent set
\(
\rightarrow\)

NP-hard problems
\(\Rightarrow\) as hard as any NP problem.
- NPC problems are NP-hard.
- NP-hard need not be NPC
  \(\Rightarrow\) might not be decision problems
  \(\Rightarrow\) or might not have poly-time verification.

independent set
\(\leftarrow\)

NP-hard problems
\(\Rightarrow\) as hard as any NP problem.
- NPC problems are NP-hard.
- NP-hard need not be NPC
  \(\Rightarrow\) might not be decision problems
  \(\Rightarrow\) or might not have poly-time verification.

independent set
\(\leftarrow\)

NP-hard problems
\(\Rightarrow\) as hard as any NP problem.
- NPC problems are NP-hard.
- NP-hard need not be NPC
  \(\Rightarrow\) might not be decision problems
  \(\Rightarrow\) or might not have poly-time verification.

independent set
\(\leftarrow\)

NP-hard problems
\(\Rightarrow\) as hard as any NP problem.
- NPC problems are NP-hard.
- NP-hard need not be NPC
  \(\Rightarrow\) might not be decision problems
  \(\Rightarrow\) or might not have poly-time verification.

independent set
\(\leftarrow\)
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 \(\Rightarrow\) same for all NP
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 → same for all NP

NPC = NP-hard & in NP

independent set

decision (≤k?)

NPC

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