NP-COMPLETENESS: a brief informal introduction
NP-COMPLETENESS: a brief informal introduction

We've seen algorithms with several time complexities:

- \(O(n)\)
- \(O(\log n)\)
- \(O(n)\log n\)
- \(O(n^2)\)
- \(O(n^c)\)
- \(O(2^n)\)

Focus on decision problems.

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

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

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

$P \text{ v. } NP : = \text{ or } \neq ?$
NP: non-deterministic polynomial
(NOT "non-polynomial")

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

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

Is it ever "much" harder to solve a 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...

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

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

not within a polynomial factor:

$T(\text{verify}) = o(n^c \cdot T(\text{solve}))$
$T(\text{solve}) = \omega(n^c \cdot T(\text{verify}))$
There are thousands of problems for which no
known polynomial-time solution is known, yet we
can verify proposed solutions in poly-time.

For which no non-empty further notice
until empty P
NP
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.

\( \text{\textbf{NP}} \) non-empty until further notice

\( \text{P} \)

\( \implies \text{decide if one exists} \)
NP non-empty until further notice

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" (size)
**Decision Problem**

"Is there a set of k independent vertices?"

- (Independent: no neighbors)

**Optimization Problem**

"Find the largest independent set" (size)

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. $\Rightarrow P = NP$

$\Rightarrow$ 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. [*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≠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

\[ \triangleright \text{as hard as any NP problem.} \]

- NPC problems are NP-hard.
- NP-hard need not be NPC

\[ \triangleright \text{might not be decision problems} \]

\[ \triangleright \text{or might not have poly-time verification.} \]

\[ \text{independent set} \]

\[ \text{decison (} \leq k? \text{)} \]

\[ \text{NPC} \]

\[ \text{optimization (max k)} \]

\[ \text{NP-hard} \]
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
  - or might not have poly-time verification.

- like NPC, solving an NP-hard problem quickly → 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 implies same for all NP

**NP** = NP-hard & in NP

Independent set
- Decision (\( \leq k \)?)
- NPC
- Optimization (\( \text{max } k \))
- NP-hard
AN IMPORTANT DETAIL just mentioned here

For NPC problems, we measure input in terms of a finite alphabet (e.g. binary 1 = 1 bit; k = Θ(log k) bits)