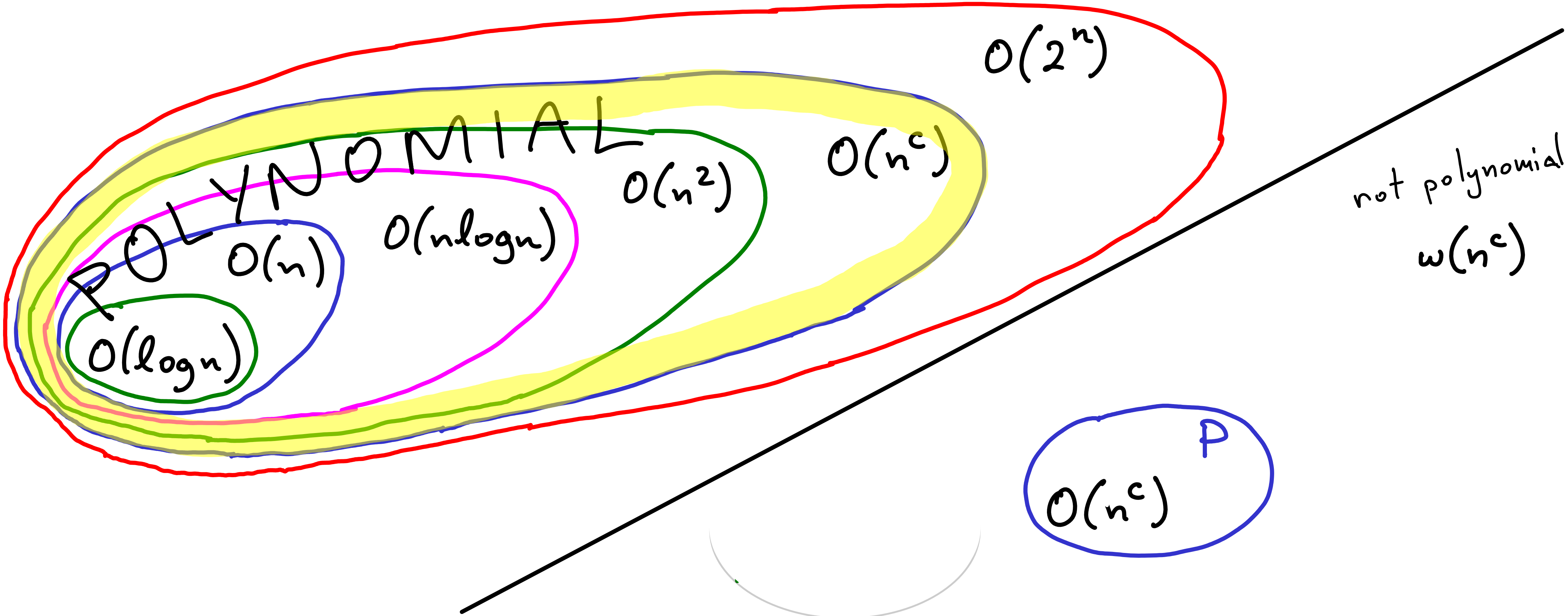
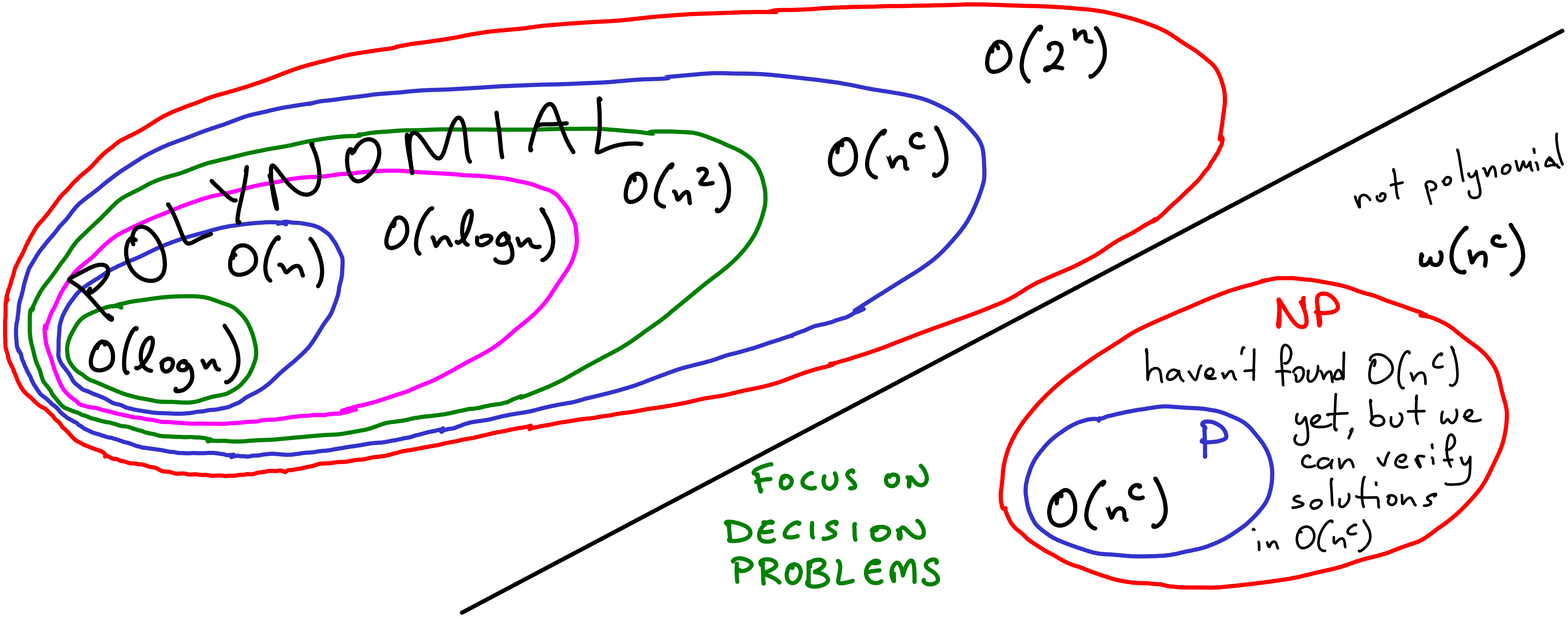
NP-COMPLETENESS: a brief informal introduction

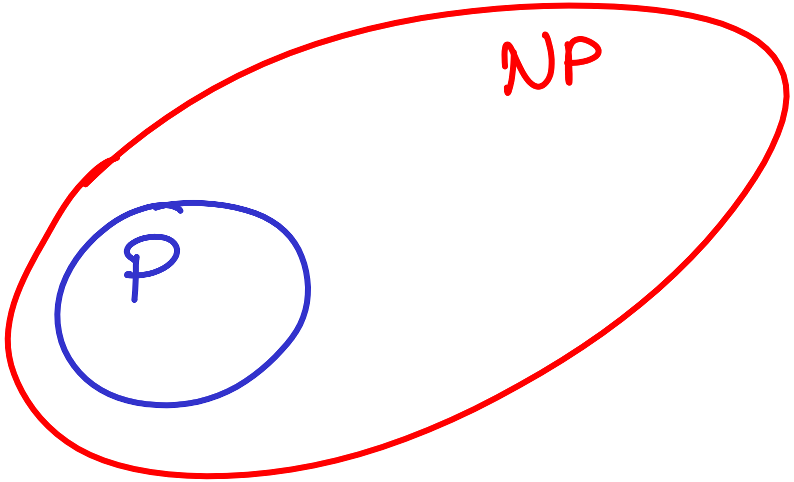
we've seen algorithms with several time complexities:



NP-COMPLETENESS: a brief informal introduction

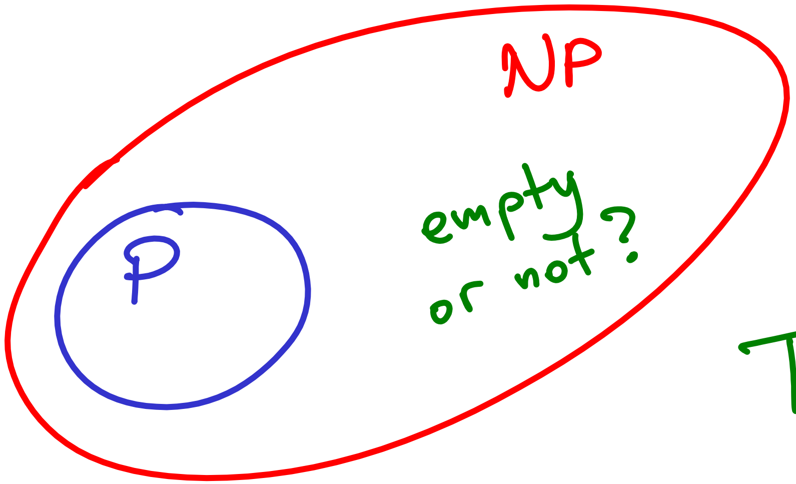
we've seen algorithms with several time complexities:





NP: nondeterministic polynomial

(NOT "non-polynomial")



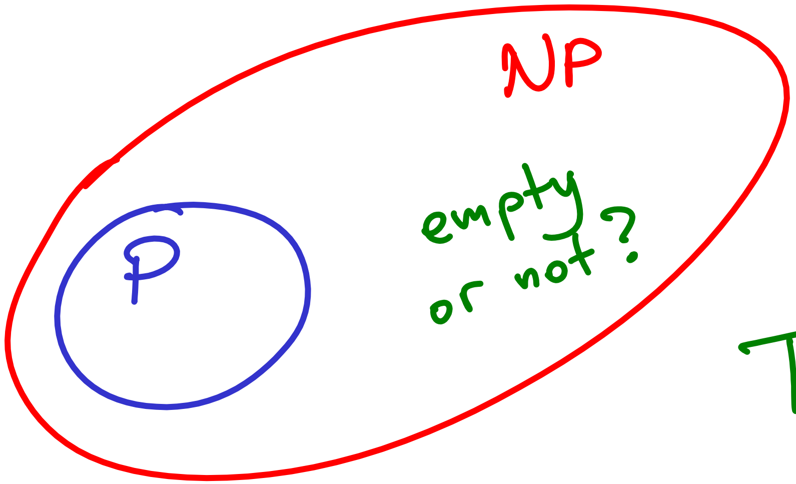
NP: nondeterministic polynomial

(NOT "non-polynomial")

The $1,000,000 question...

P v. NP : = or ≠ ?

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



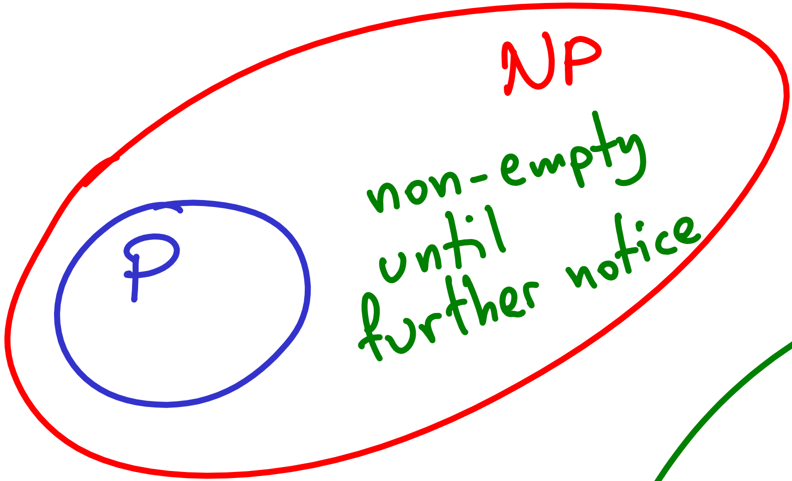
NP: nondeterministic polynomial
(NOT "non-polynomial")

The \$1,000,000 question...
P v. NP : = OR ≠ ?

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

$$T(\text{verify}) = o\left(n^c \cdot T(\text{solve})\right)$$
$$T(\text{solve}) = \omega\left(n^c \cdot T(\text{verify})\right) \quad ?$$

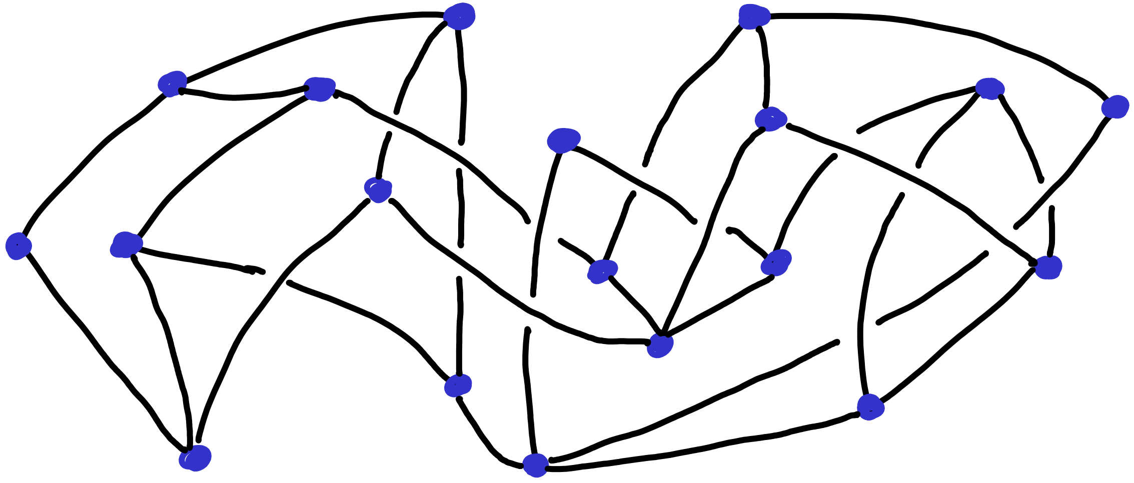


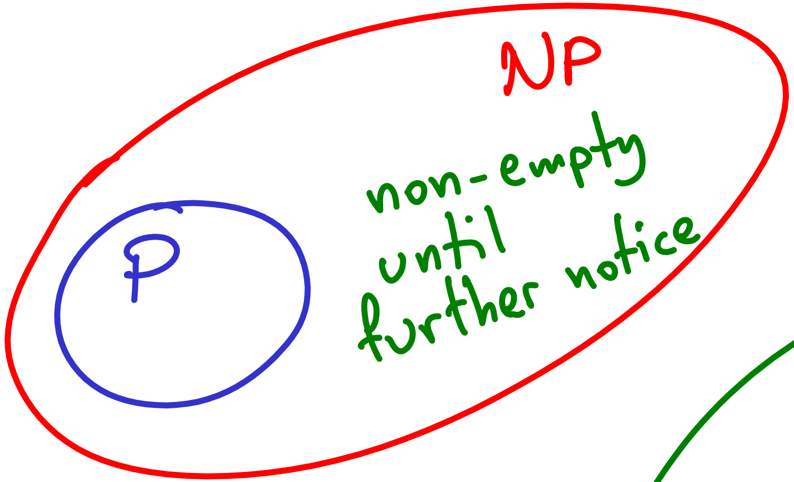
There are thousands of problems for which no known polynomial-time solution is known, yet we can verify proposed solutions in poly-time.

e.g. Hamiltonian cycle

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

↳ decide if one exists



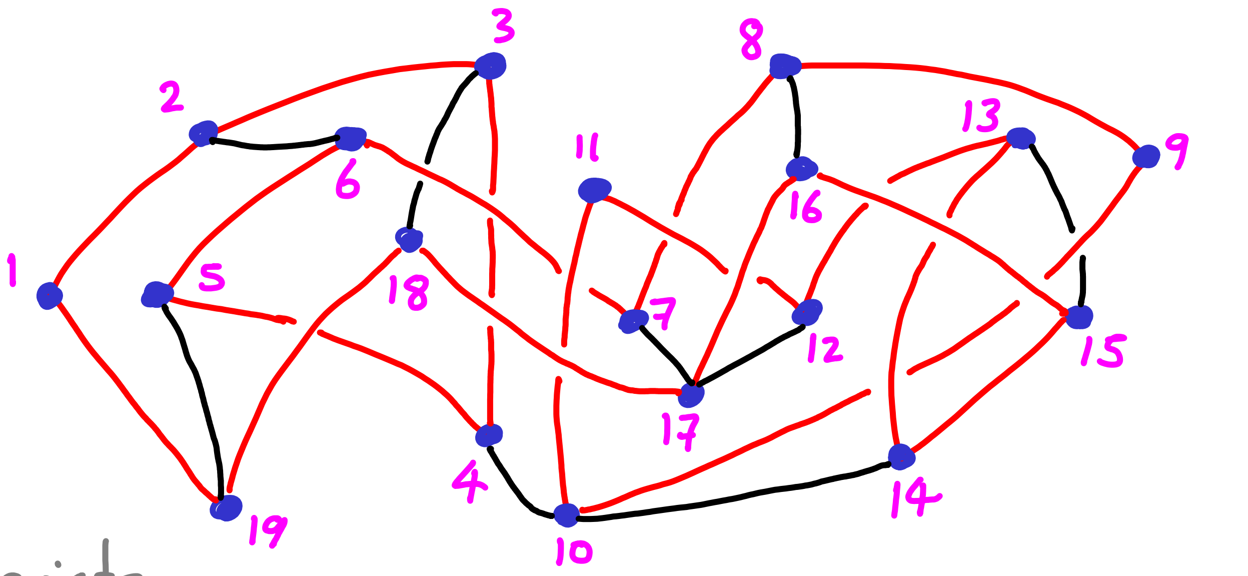


There are thousands of problems for which no known polynomial-time solution is known, yet we can verify proposed solutions in poly-time.

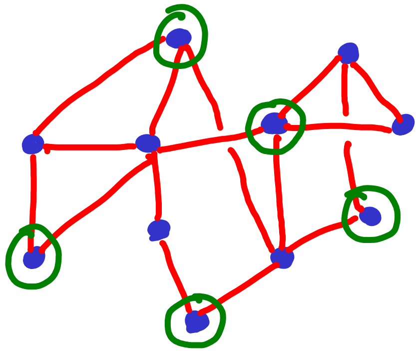
e.g. Hamiltonian cycle

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

↳ decide if one exists



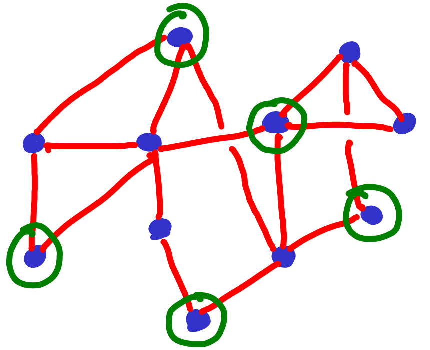
"is there a set of k independent vertices?"



(independent: no neighbors)

DECISION PROBLEM

"is there a set of k independent vertices?"



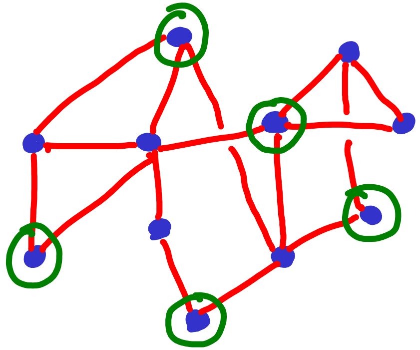
(independent: no neighbors)

OPTIMIZATION PROBLEM

"find the largest independent set"

DECISION PROBLEM

"is there a set of
k independent vertices?"



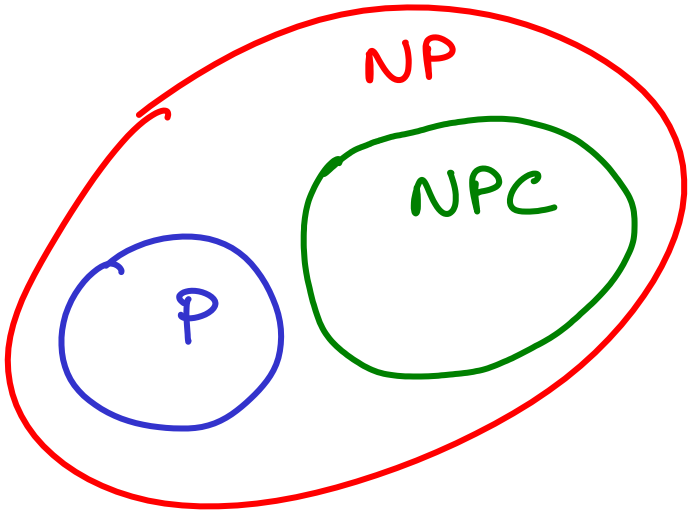
(independent: no neighbors)

OPTIMIZATION PROBLEM

"find the largest
independent set"

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

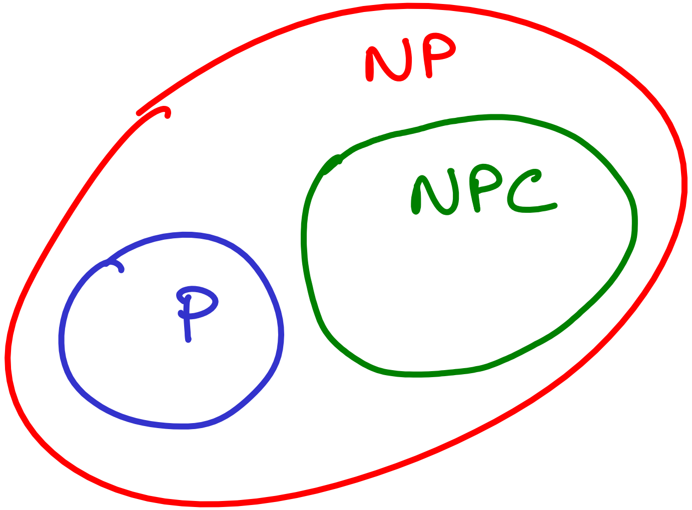
Often, optimization problems are not polynomially harder than decision.



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)

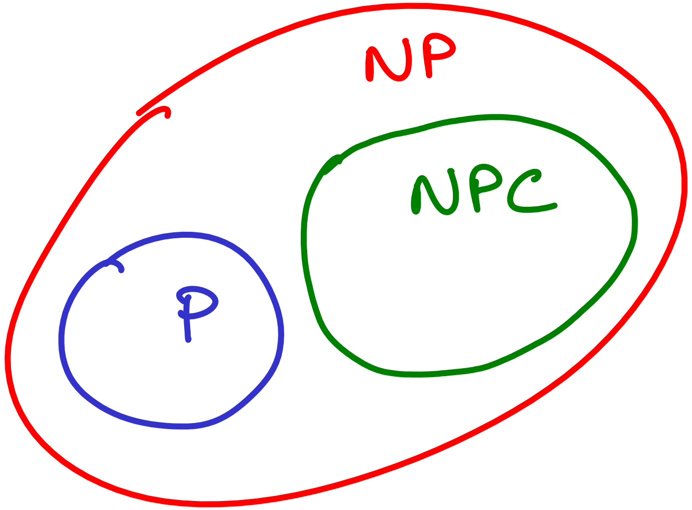


NP-COMPLETE PROBLEMS

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 <u>any</u> NPC problem, this implies the same for <u>every</u> problem in NP. → P = NP



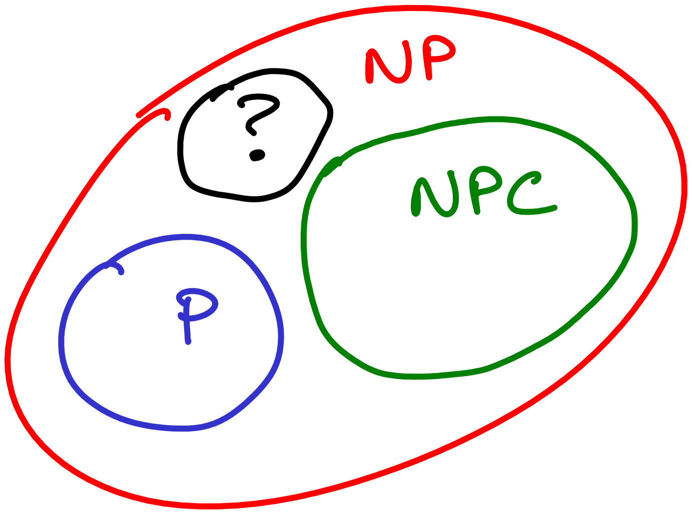
NP-COMPLETE PROBLEMS

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

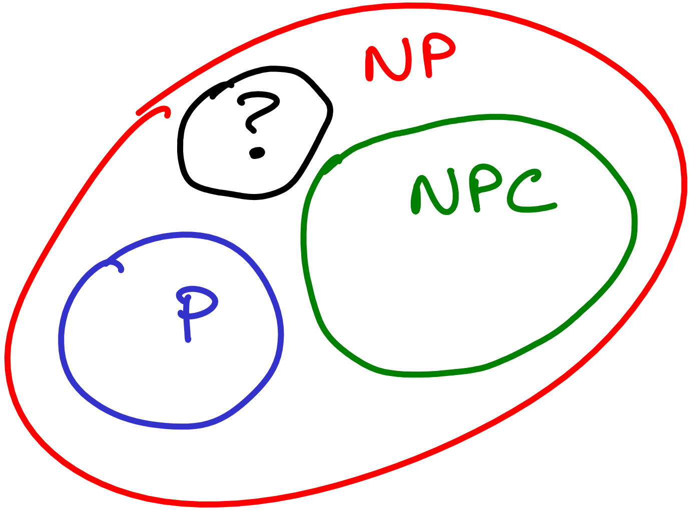
(decision problems with solutions that can be verified in poly-time, but for which no poly-time algo is known)

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

↳ if you ever prove that an NPC problem has no poly-time algo, then no NPC problem does → P ≠ NP

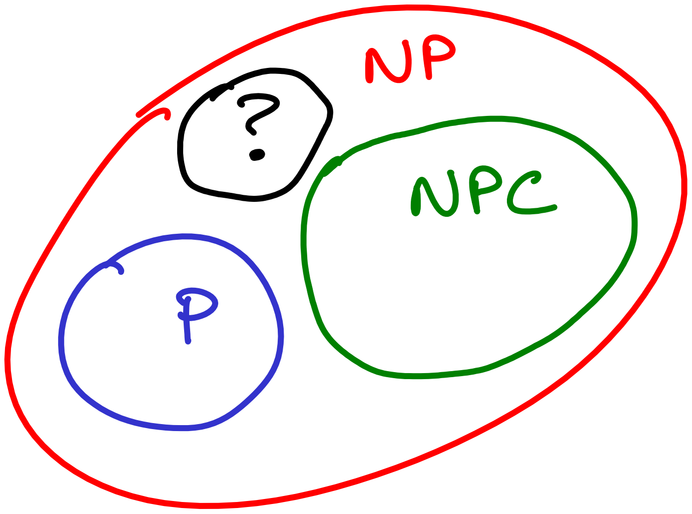


Are there other problems in NP but not in P or NPC?



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

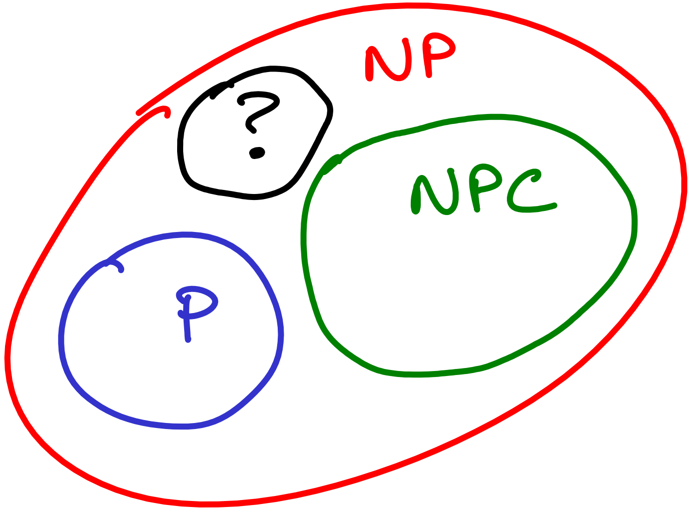
- if P=NP then N/A.



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

- if P=NP then N/A.
- if P≠NP then yes. [theorem]

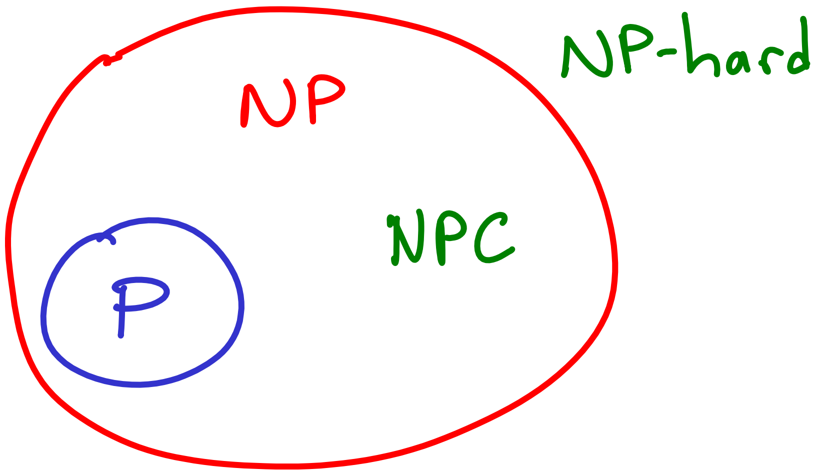
↳ few "natural" problems (almost everything in NP is P or NPC)



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

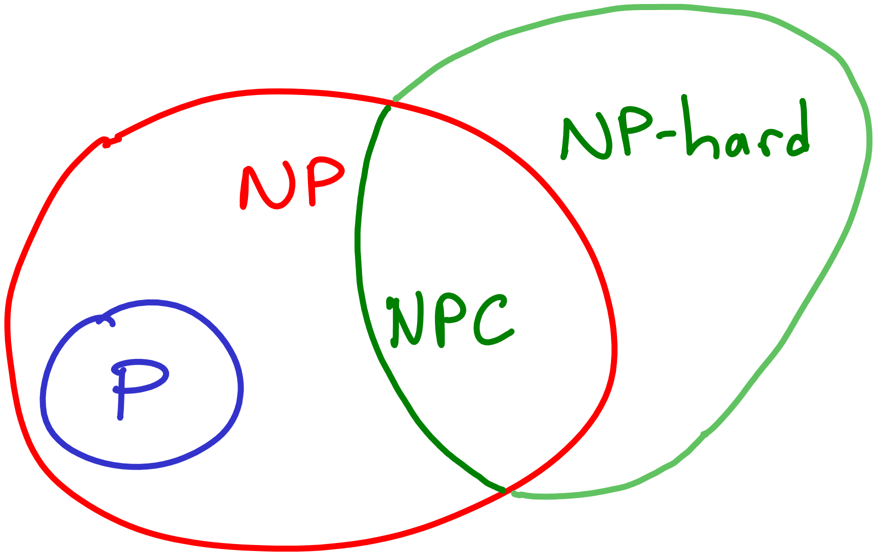
- if P=NP then N/A.
- if P≠NP then yes. [theorem]
  - ↳ few "natural" problems (almost everything in NP is P or NPC)

If we solved such a problem in poly-time, it would just move into P without dragging everything else along.



NP-hard problems

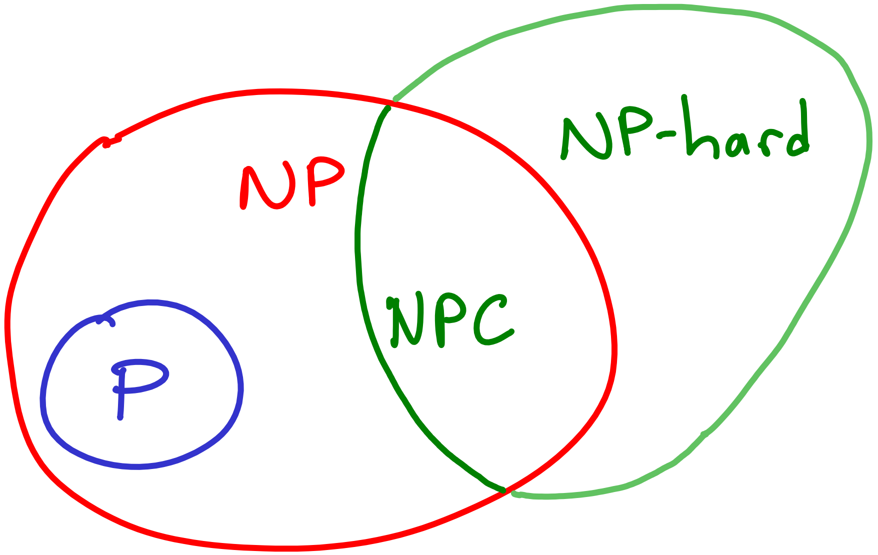
↳ as hard as any NP problem.



NP-hard problems

↳ as hard as any NP problem.

- NPC problems are NP-hard.

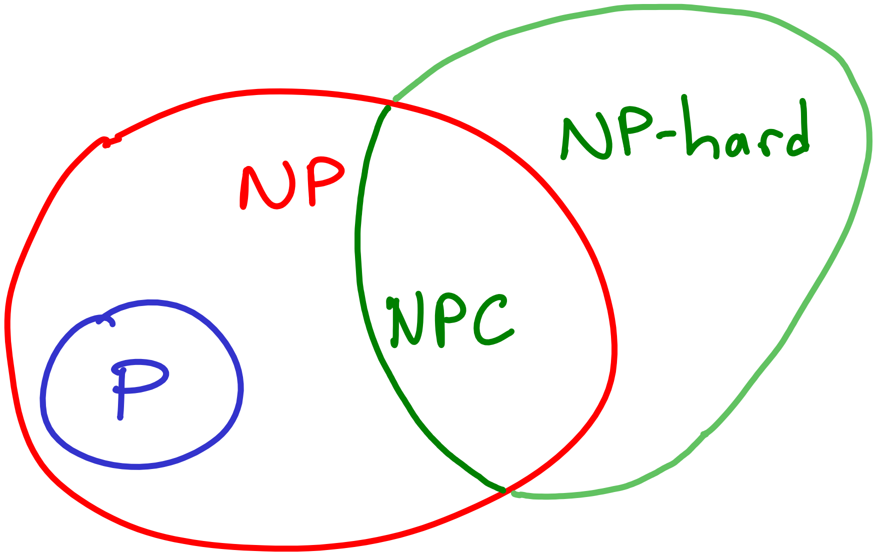


NP-hard problems

↳ as hard as any NP problem.

- NPC problems are NP-hard.
- NP-hard need not be NPC
  - ↳ might not be decision problems

independent set
- → decision ($\leq k$?) NPC
- → optimization (max k) NP-hard



NP-hard problems

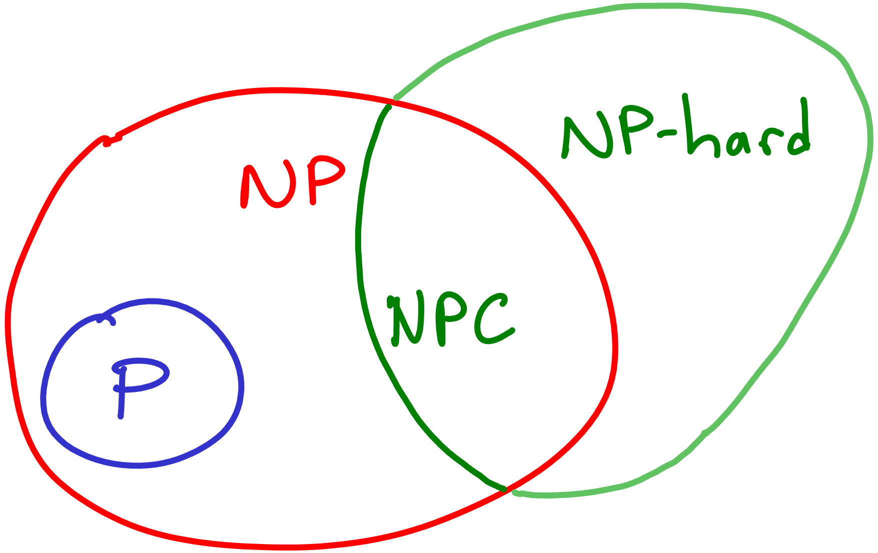
↳ as hard as any NP problem.

- NPC problems are NP-hard.
- NP-hard need not be NPC
  - ↳ might not be decision problems
  - ↳ or might not have poly-time verification.

not many of these

independent set → decision ($\leq k$?) NPC

independent set → optimization (max k) NP-hard



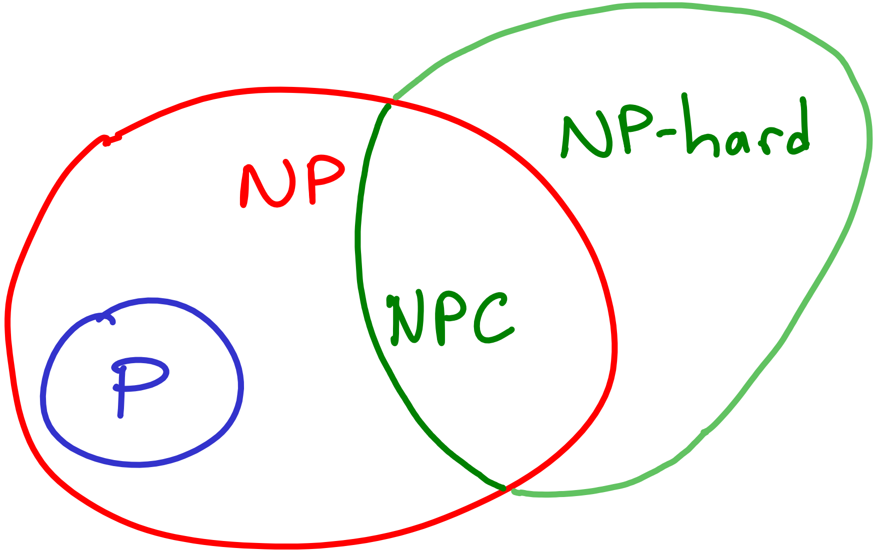
independent set
- decision (≤ k?)
  NPC
- optimization (max k)
  NP-hard

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.

  not many of these
- 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.

not many of these

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

independent set
- → decision ($\leq k$?) NPC
- → optimization (max k) NP-hard

NPC = NP-hard & in NP

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