EE360C ALGORITHMS - FALL 2018 - November 29 - Morning

$\boxed{P}$: Decision problem that can be efficiently computed.

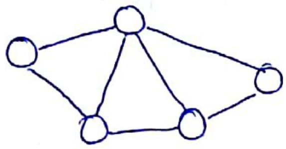
Problem $X \in P$ if there is poly-time alg A s.t.:

- s is a yes-instance of $X \iff A(s) = \text{yes}$
- s is a no-instance of $X \iff A(s) = \text{no}$

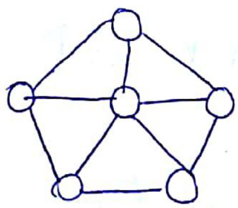
aka input

- Is this undirected graph 3-colorable?

can assign 3 colors to nodes s.t. no two adjacent nodes are colored the same



yes instance



no instance

non-deterministic-polynomial time

$\boxed{NP}$ Decision problems that can be efficiently verified

for "yes" answer

Problem $X \in NP$ if there is a poly-time alg A taking arguments s and t such that:

instances/input — certificate/witness (in example above, assignment of colors)

- s is a yes instance of $X \iff \exists t$ s.t. $A(s,t) = \text{yes}$
- s is a no-instance of $X \iff \forall t$ have $A(s,t) = \text{no}$.

$P \overset{?}{=} NP$ the most important problem in CS.

We can't prove $P \neq NP$, but at least we can identify the problems most likely to not be in P.

Identify the "hardest problems" in NP $\rightarrow$ $\boxed{\text{NP-Complete}}$

Specific Sense: If we could solve any of these problems in poly-time, then we can solve any problem in NP in poly-time.

If $X \in$ NP-Complete and $X \in P$, then $P = NP$.

$\boxed{\text{Def}}$: Polynomial time reduction

For problems X and Y we write $X \leq_p Y$
(x is polynomial time reducible to Y)

there is a poly-time alg R transforming instances of X to instances of Y s.t.

- S_x is a yes-instance of $X \Leftrightarrow R(S_x)$ is a yes-instance of Y
- S_x is a no-instance of $X \Leftrightarrow R(S_x)$ is a no-instance of Y

Claim: If $X \leq_p Y$ and $Y \in P$ then $X \in P$

Pf: If A is a poly-time algo for Y, then $A(R(\cdot))$ is a poly-time algo for X.

Def A problem X is <u>NP-complete</u> if

1. $X \in NP$
2. $\forall Z \in NP: \; Z \leq_p X$

<u>Levin & Cook 1971</u>: first NP-Complete problem.

Circuit SAT: "Given a Boolean circuit represented as a graph, is there a way to set inputs so that the circuit outputs 1"?

Intuition: Any algo can be "represented" as a logic circuit.

To show your problem X is NP-Complete, it's enough:

1. $X \in NP$
2. $\underbrace{\text{[some Known NP-Complete Problem]}}_{Y} \leq_p X$

transitive