# P, NP, and Reductions

# Admin Corner

# Last time

Def: P = "complexity class" of all problems which are efficiently solvable

Def: NP = class of problems whose solutions can be verified efficiently

Thm:
- 3-Coloring $\in$ NP
- Factoring $\in$ NP

# Rudrata Cycle aka Hamiltonian Cycle

Input: Graph $G=(V,E)$

Solution: tour = cycle visiting each vertex exactly once

Trivial alg: Try all $n!$ ways of ordering vertices

Best known alg: Time $O(1.657^n)$

Not known to be in P!

Fact: Hamiltonian Cycle $\in$ NP

Pf: Verify( Input $G$, Solution tour ): Check that:
- visits each vertex once
- only uses edges in graph

Eulerian cycle: find cycle visiting each edge exactly once

↑ in P

# Traveling Salesperson Problem (TSP)

Input: Graph $G=(V,E)$ w/ edge weights
Solution: tour w/ low total weight

Optimization version: Min-TSP
Find the tour w/ min total weight

Best known alg: time $O(n^2 2^n)$

Min-TSP $\in$ NP ??? ← Probably not!

Search version: Search-TSP
Find a tour w/ total weight $\leq B$ ← "Budget" (part of input)

Fact: Search-TSP $\in$ NP

Decision version: Decision-TSP
Does there exist a tour w/ weight $\leq B$? (Yes/No answer)

Fact: Dec-TSP $\in$ NP

Things not in NP: (believed to be)
1. Some optimization versions of problems
2. Counting versions of problems
   "How many 3-colorings are there?"
3. Really, really hard problems (Halting)

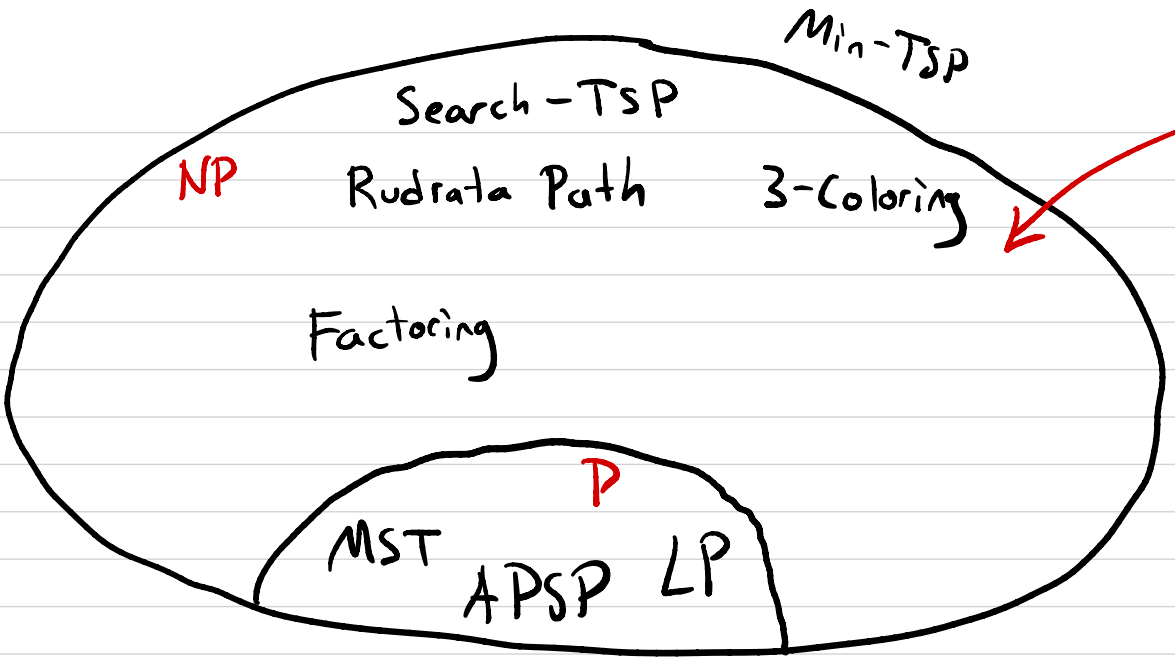
Super, duper formally:
- NP typically defined for decision problems (in CS 172)
- we'll allow search problems too

Thm: $P \subseteq NP$

E.g. Minimum Spanning tree (in P)
Input: $G = (V, E)$ w/ weights on edges
Output: MST $T$

Claim: $MST \in NP$
"pf":

Verify(Input $G$, Solution $T$):
1. Run Kruskal($G$). Let $T^*$ = its output.
2. Check cost($T$) = cost($T^*$).




Seems really hard!

Biggest open problem in TCS: Is P=NP?

Q. How to compare difficulty of solving problems?

A. Reductions.

# Reductions

3 min break

Close the doors!

**Def:** "Problem A reduces in polynomial time to Problem B"
if "you can use any efficient alg for B to efficiently solve A"
aka: "$A \leq_p B$" (A's difficulty $\leq$ B's difficulty)
(A is at most as hard as B)

**Two-player Zero sum game**
Input: Payoff matrix M  Solution: Row player's optimal strategy

**Thm:** Zero Sum $\leq_p$ Linear Programming

Zero Sum input

$M = \begin{array}{|c|c|}\hline 3 & 1 \\ \hline 2 & 4 \\ \hline \end{array}$

Reduction alg →

LP input

$$\max \Sigma$$
$$\text{s.t. } \Sigma \leq 3p_1 + 2p_2$$
$$\Sigma \leq p_1 + 4p_2$$
$$p_1 + p_2 = 1$$
$$p_1, p_2 \geq 0$$

LP alg →

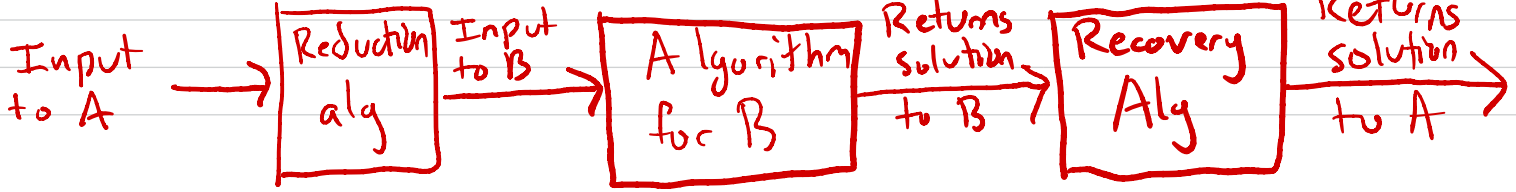
LP solution

$\Sigma = 2.5$
$p_1 = 1/2$
$p_2 = 1/2$

Recovery alg →

Zero Sum solution

$p_1 = 1/2$
$p_2 = 1/2$



## Rudrata / Hamiltonian Cycle

Input: Graph $G = (V, E)$

Solution: tour of $G$

(= cycle visiting each vertex exactly once)

## Min-TSP

Input: cities $1, \ldots, n$

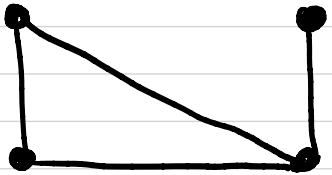
pairwise distances $d_{ij}$, $\forall i \neq j$

Solution: tour of minimum distance

---

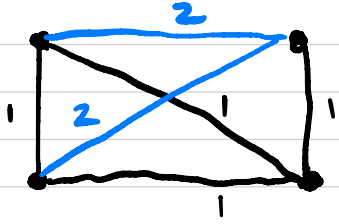
Thm: Rudrata Cycle $\leq_p$ Min-TSP (neither known to be in P!)

Rudrata instance $G$

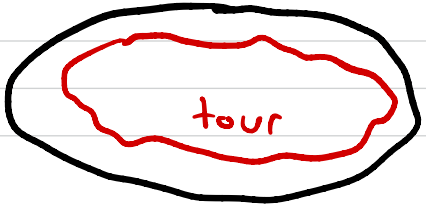


Reduction →

Min-TSP instance




Min-TSP solution




Recovery →

- Output tour if total weight $= n$
- Output "no tour" otherwise

# Reduction notes

- Reduction & Recovery algs must be efficient (poly-time)
- $A \leq_p B$ and $B \leq_p C \Rightarrow A \leq_p C$
- Can prove $A \leq_p B$, even if $A$ & $B$ not known to be efficient
- "$A \leq_p B$" and "A reduces to B" most confusing thing in Algos

**#1 Most Common Mistake:**

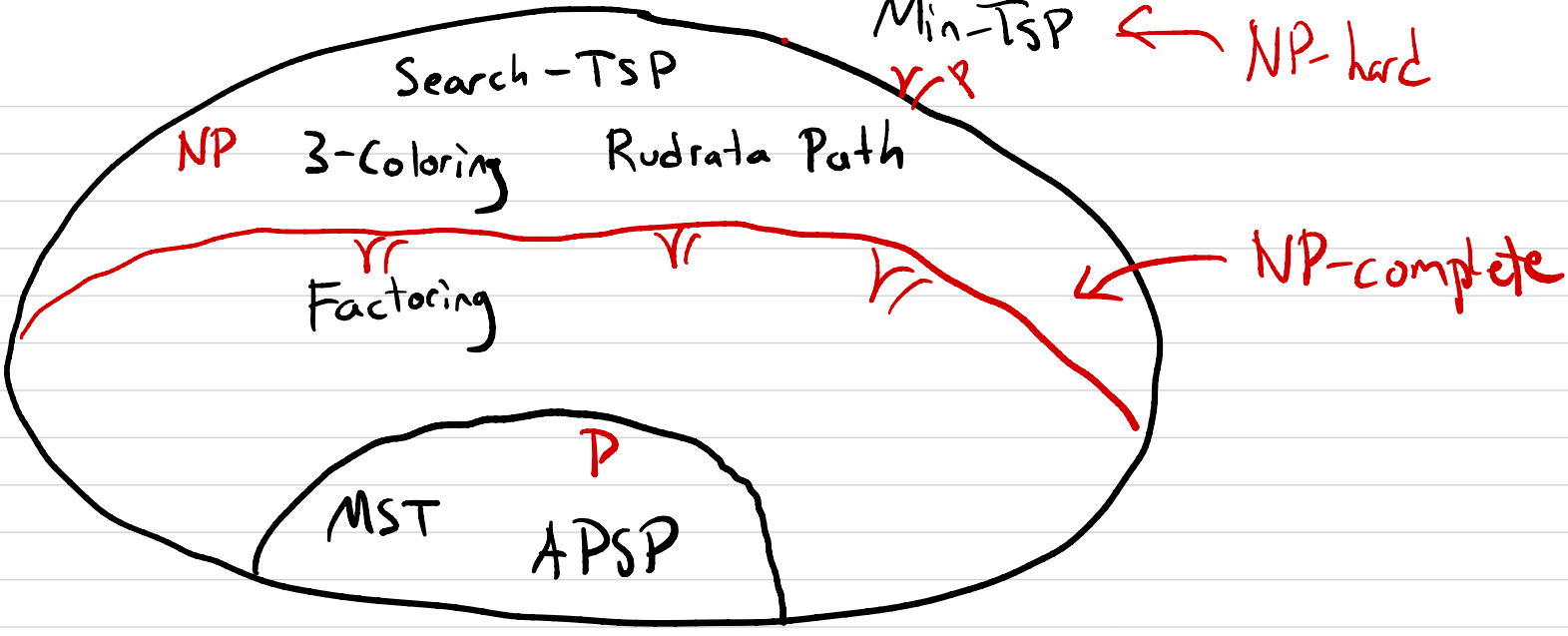
"Prove" $A \leq_p B$ by taking Instance of B $\xrightarrow{\text{reduction}}$ Instance of A ✗✗

Quiz: We have seen $A \leq_p B$ $B \leq_p A$

A = LP, B = Max Flow

A = Max Flow, B = Bipartite matching

✗ ✗



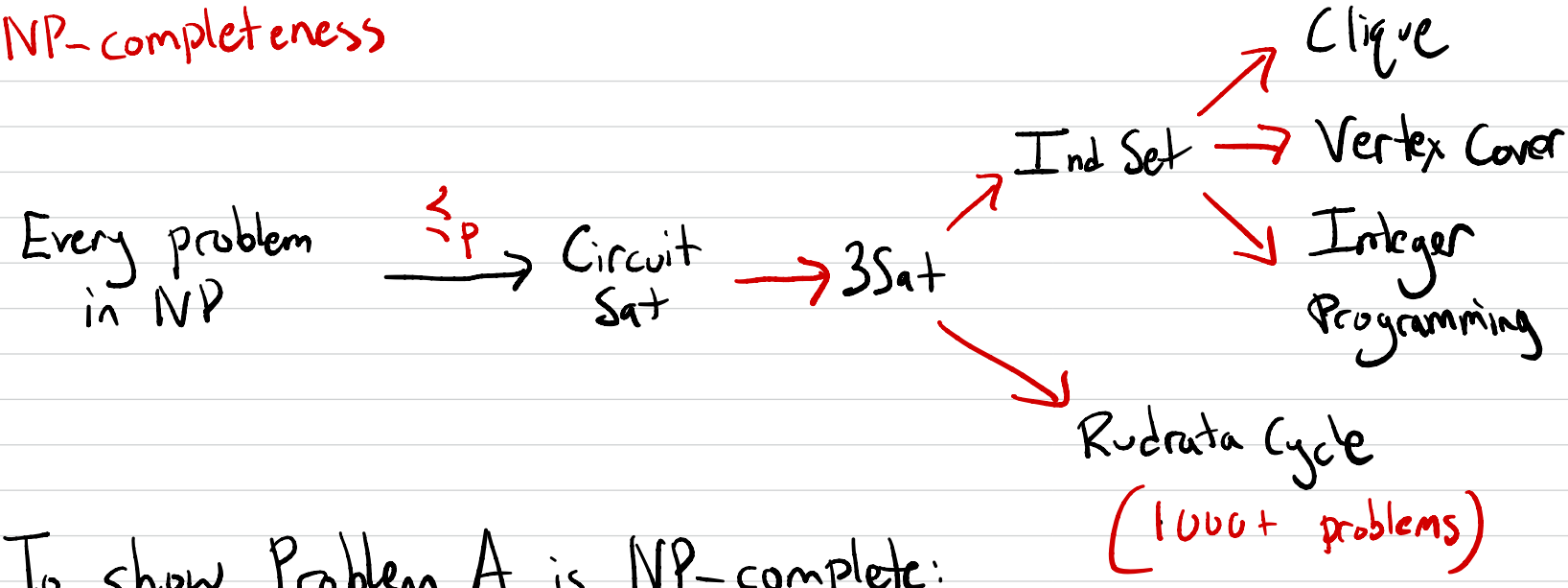

Def: A problem A is NP-hard if every problem $B \in NP$ reduces to A ($B \leq_P A$)

Def: A is NP-complete if $A \in NP$ & A is NP-hard.

Fact: If A & B are NP-complete, $A \leq_P B$ and $B \leq_P A$.

Fact: $\exists$ poly-time alg for NP-complete problem $\Rightarrow$ P=NP.

# NP-completeness




To show Problem A is NP-complete:

1.) $A \in NP$ (show verification algorithm)

2.) Pick some (usually well-known) NP-complete problem B.
and show $B \leq_P A$. (Example: $3Sat \leq_P A$)

## Independent set

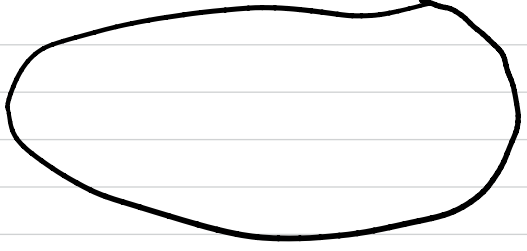
Input: Graph $G=(V,E)$, $k=\{0,1,2,3,\dots\}$

Solution: Independent set of size $k$
= set $S \subseteq V$ with $|S|=k$
s.t. $\forall u,v \in S$, $(u,v) \notin E$.

## Integer programming

Input: A linear program

Solution: An integer solution to linear program




reduction →

$$x_i = \begin{cases} 1 & \text{if } i \in \text{Independent Set} \\ 0 & \text{o.w.} \end{cases}$$

IP: $\max \; x_1 + \cdots + x_n$

s.t. $0 \le x_i \le 1$

$\forall (i,j) \in E \quad x_i + x_j \le 1$

To prove:

1.) G has independent set of size $k \Rightarrow \exists$ solution to IP w/ val $\geq k$

2.) $\exists$ solution to IP w/ val $\geq k \Rightarrow$ G has independent set of size $k$.