

P

and

NP

So far ...	Integer multiplication	$O(n \log n)$
	Minimum Spanning Trees	$O((n+m) \log(n))$
	All pairs shortest paths	$O(n^3)$
	...	

Def: A problem is **efficiently solvable**
 if it can be solved in **polynomial time** $= O(n^k)$
 (in theory; in practice, efficient can even mean $O(n)$ only)

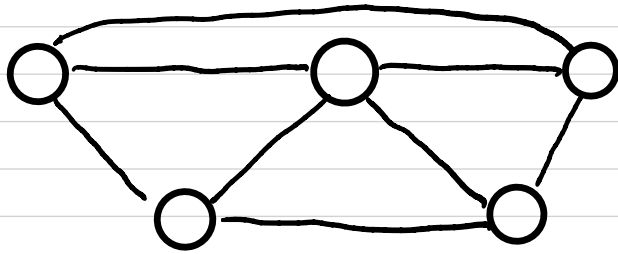
Def: **P** = "complexity class" of all problems
 which are efficiently solvable

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

3-coloring

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

Solution: A coloring $c: V \rightarrow \{\text{Red, Green, Blue}\}$
s.t. each edge receives 2 different colors



Factorization

Input: n -bit number N

Solution: two numbers $p, q > 1$ s.t. $p \cdot q = N$

Rudrata Cycle aka Hamiltonian Cycle

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

Solution: **tour** = cycle $v_1, v_2, \dots, v_n, v_1$ visiting every node 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 ($\begin{matrix} \text{Input} & \text{Solution} \\ G & \text{tour } v_1, v_2, \dots, v_n, v_1 \end{matrix} \end{matrix})$: Check that

- visits each vertex x once
- $\forall i, (v_i, v_{i+1}) \in E$

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??? $\leftarrow$ Probably not!

Search version: **Search-TSP**

Find a tour w/ total weight $\leq B \leftarrow$ "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**

Super formally

Def: A **binary relation** is a subset $R \subseteq \{0,1\}^* \times \{0,1\}^*$
If $(I, S) \in \{0,1\}^* \times \{0,1\}^*$, $I = \text{Instance}$, $S = \text{Solution}$

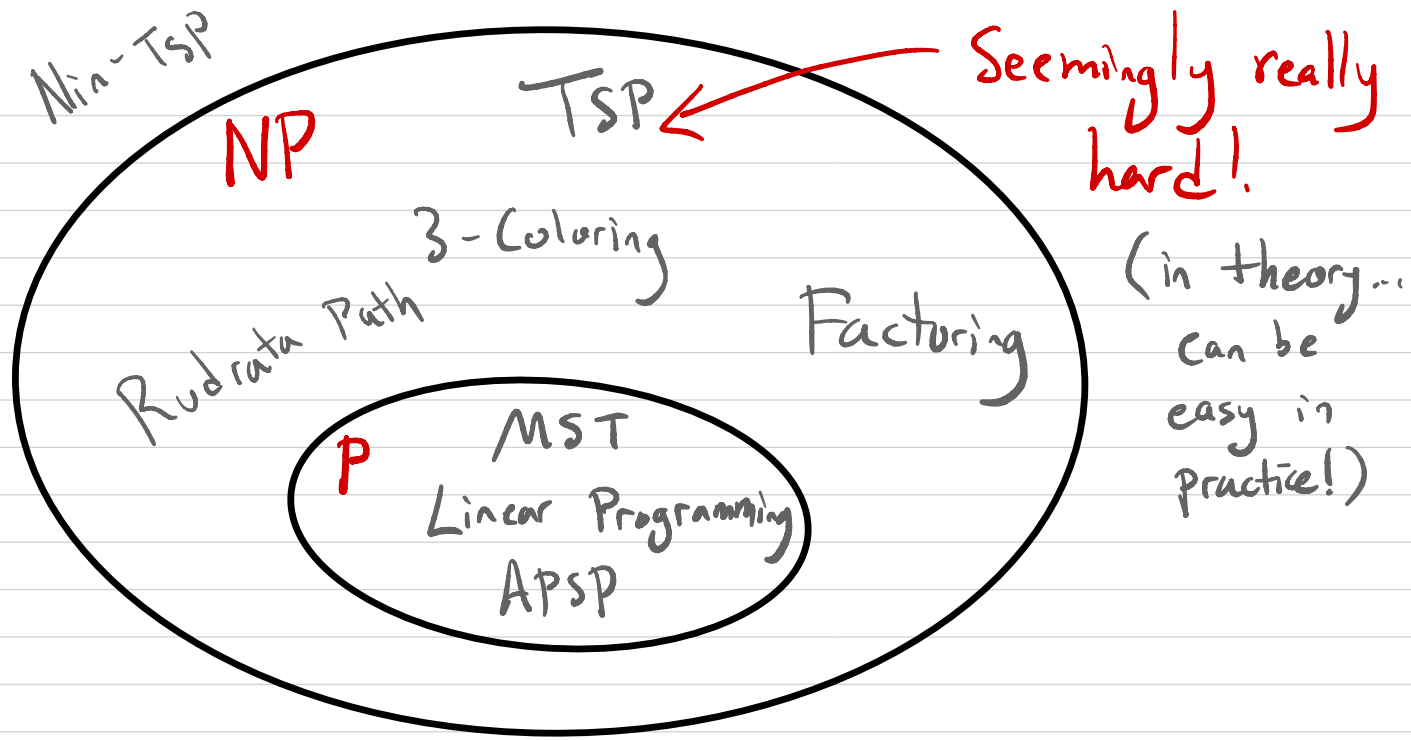
Problems:

- **Verify**(R): given (I, S) , is $(I, S) \in R$?
- **Search**(R): given I , return S s.t. $(I, S) \in R$
or "no solution" if none exists
- **Decide**(R): given I , output "yes" if exists solution
"no" if none exists

Def: **NP** = all R which can be verified in time $\text{poly}(n)$, $n = |I|$
P = all R in NP for which **Search**(R) can be solved in time $\text{poly}(n)$

Disclaimer: These are **nonstandard** definitions of P and NP.
In the real world, these are sometimes called **FP** and **FNP**.
P and NP typically defined for **decision** problems. (CS172)

Things not in NP: 1. Optimization versions of some problems
believed to be 2. Really, really hard problems (**Halting**)



Biggest open problem in TCS: **Is $P = NP$?**

Q: How to compare difficulty of solving problems? A: **Reductions**