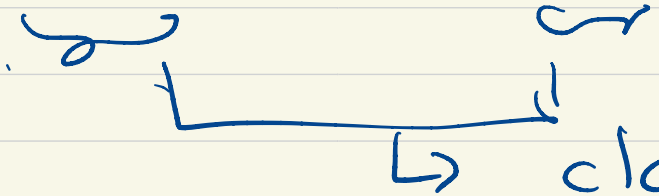
NP vs P

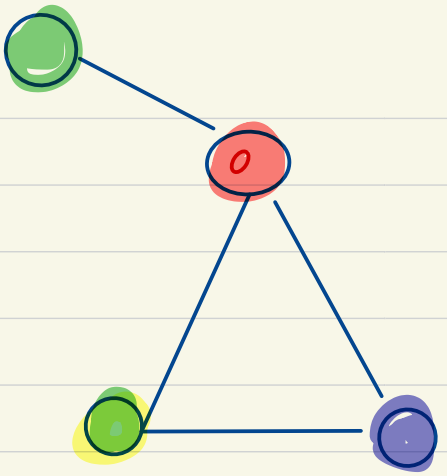


↳ classes of computational problems

P = problems for which we can FIND solutions in polynomial time $O(n^c)$ for some $c \in \mathbb{N}$

NP = problems for which we can VERIFY a given solution in polynomial time

3COLORING: $\in$



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

SOLUTION: Coloring $c: V \to \{R,B,G\}$

so that every edge (u,v)

$$c(u) \neq c(v)$$

#3COL

INPUT: $G=(V,E)$

SOL: Number of 3-colorings

Claim: 3COLORING $\in$ NP

Proof:

VERIFY (INPUT $G=(V,E)$, SOLUTION $c: V \to \{R,B,G\}$)

{ for every edge $(u,v) \in E$

Check if colors $c(u) \neq c(v)$.

}

MINIMUM SPANNING TREE $\in P$

INPUT: Graph $G = (V, E)$
weights $w: E \to \mathbb{R}$

SOLUTION: Spanning tree of smallest cost.

Claim: MST $\in$ NP

Proof: VERIFY (INPUT Graph G, Solution Tree T)

{ Run Kruskals to find MST T^*.

Check if cost (T^*) = cost (T).

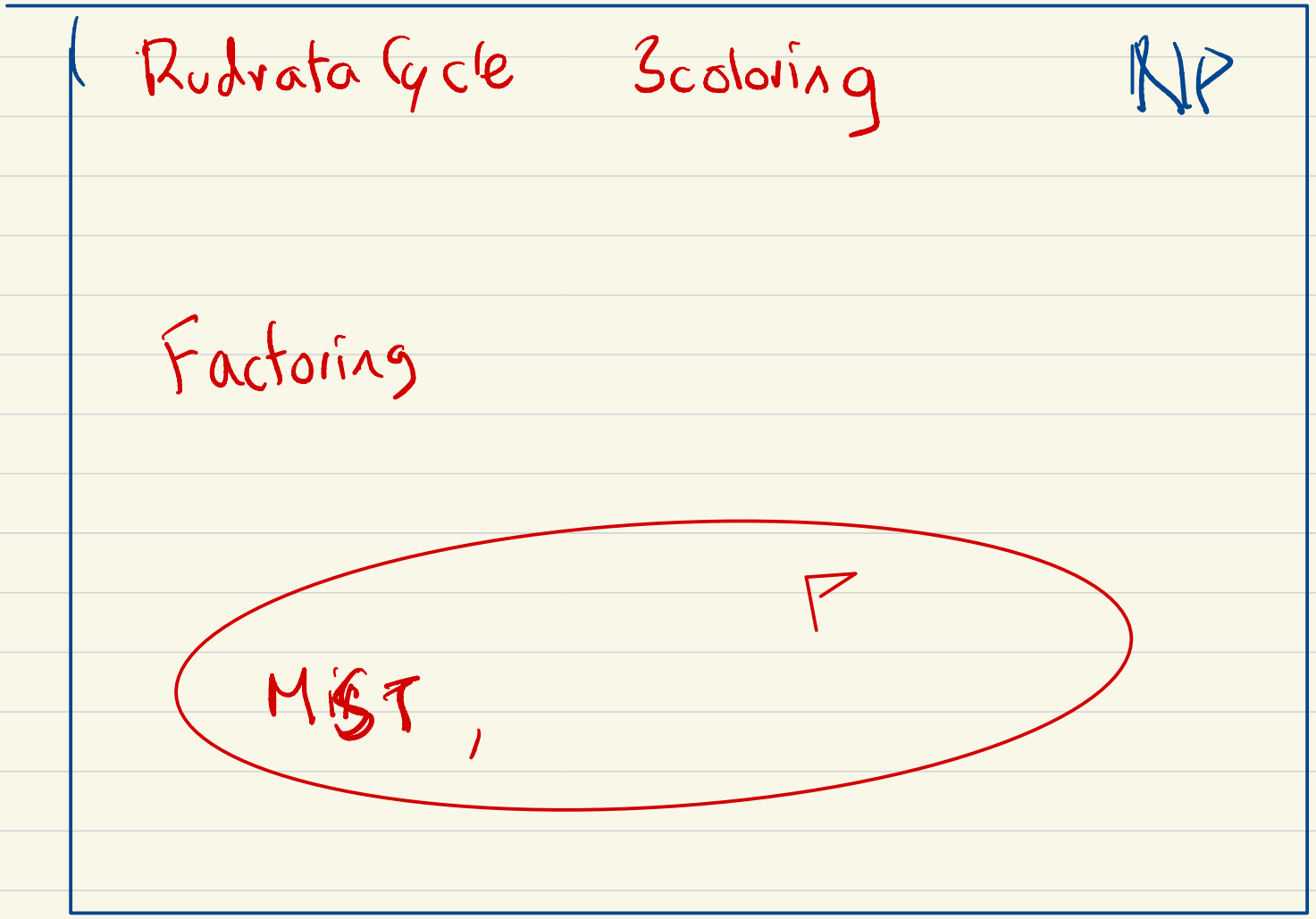
}

$P \subseteq NP$

can find solutions $\Rightarrow$ can verify them.

Counting Problems:

Games ?? Counting Problems



Is NP = P ??

FACTORING:

INPUT: A number N

SOLUTION: $p, q > 1$ such that

$$p \cdot q = N$$

$T210(((1001)))$

FACTORING $\in$ NP

Proof: VERIFY $\left(\begin{matrix} \text{INPUT} \\ N \end{matrix} , \begin{matrix} \text{Solution} \\ p, q \end{matrix} \right)$

{ Multiply p, q and check

} if $p \cdot q = N$.

BREAK RSA:

INPUT: Enc-Key, : K

EncryptedMessage : $E(m, K)$.

SOLUTION: Message m.

Break RSA $\in$ NP

Proof: VERIFY (INPUT Key K, $E(m, K)$, SOLUTION m^*)

{ Encrypt the solution m^*

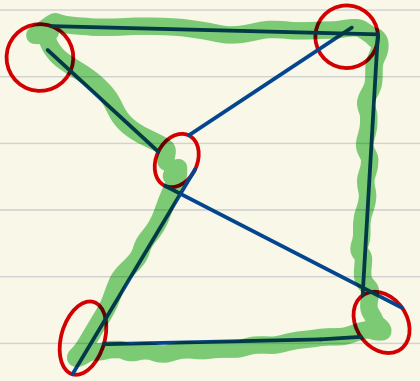
Compute $E(m^*, K)$

} Check if $E(m^*, k) = E(m, k)$

RUDRATA CYCLE: (Hamiltonian Cycle)

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

SOLUTION: A cycle that visits every node exactly once



Claim: RUDRATA CYCLE $\in$ NP

PROOF:

MIN-TSP (TRAVELLING SALESMAN PROBLEM)

INPUT: Weighted graph $G=(V,E)$

cost $w: E \to \mathbb{R}$.

Solution: The cycle (tours) of minimum total cost that contains all nodes exactly once.

BUDGET-TSP: $\in NP$

INPUT: Weighted graph $G=(V,E)$

cost $w: E \to \mathbb{R}$.

Budget B

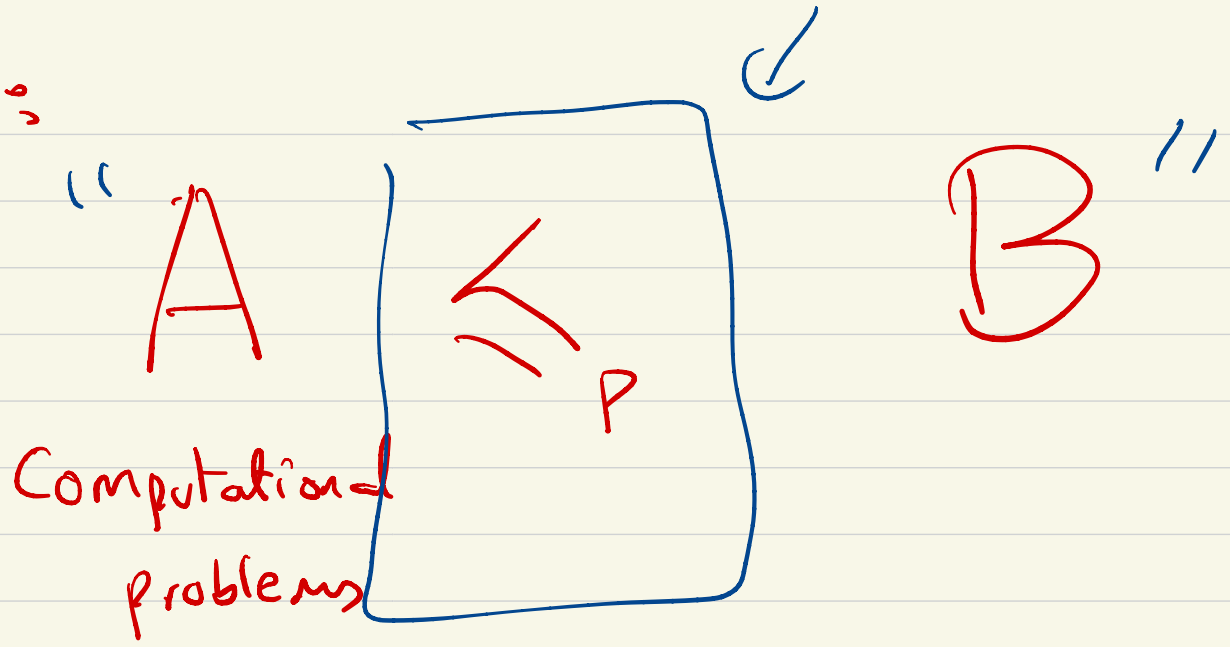
Solution: A TSP tour of cost $\leq B$.

Optimisation problem ⟶ Budget Version

- MinTSP
- ⋮

$\in NP$

REDUCTIONS:



DEF: A reduces in polytime to B

IF: USE an algorithm for B to solve A in polynomial time

$\Rightarrow$ "B is at least as hard as A."

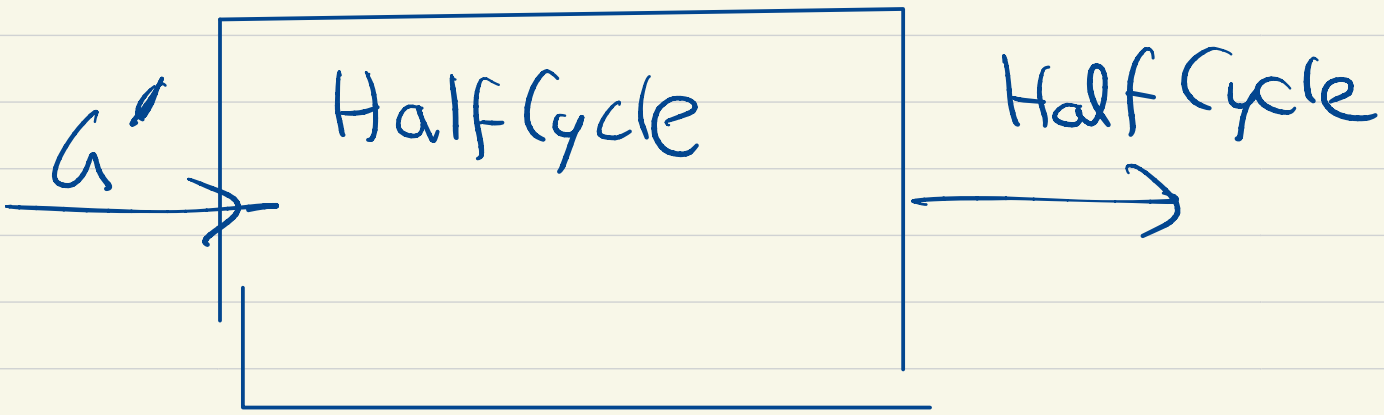
RUDRATA CYCLE
INPUT: Graph $G=(V,E)$
Solution: A cycle containing ALL vertices

$\leq_P$

HALF-CYCLE
INPUT: Graph $G=(V,E)$
Solution: A cycle containing $|V|/2$ vertices

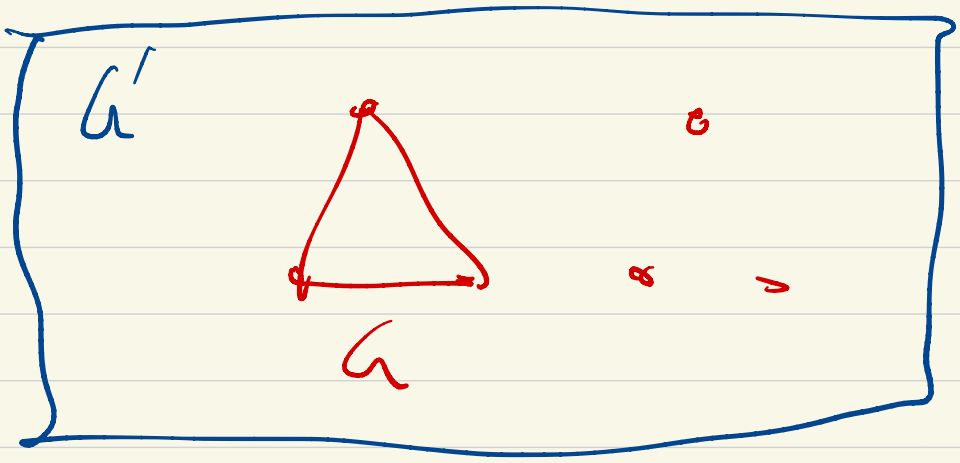
"USE an algorithm for HalfCycle to SOLVE Rudrata Cycle".

RUDRATA CYCLE INSTANCE $G=(V,E)$



Reduction Algo: From graph G

Construct $G' = G +$ |V| dummy vertices



→ Run HdfCycle (G') | G' has $2|V|$ vertices

→ a cycle C in G' with $\frac{2|V|}{2} = |V|$ vertices

cycle C is inside G

$\Rightarrow$ a cycle with $|V|$ vertices inside G

$\Rightarrow$ a Rodiata cycle of G.