14 March 2025

Boolean Satisfiability, P, and NP

Plan

* Boolean Satisfiability Problem
* Announcements
* Defining P vs. NP.

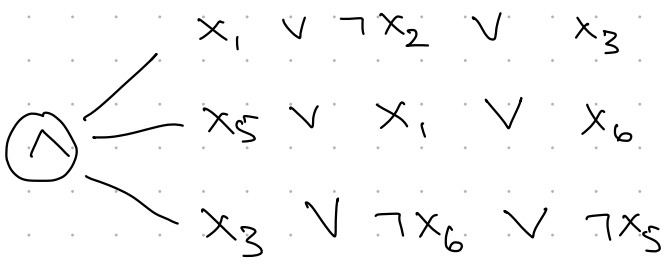
Systems of Boolean Equations

XOR-SAT

$$X_1 \oplus \neg X_2 \oplus X_3$$
$$X_5 \oplus X_1 \oplus X_6$$
$$X_3 \oplus \neg X_6 \oplus \neg X_5$$

} Does there exist an assignment $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. every expression is 1?

CNF-SAT (i.e. AND-of-OR SAT)



} Does there exist an assignment $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. every clause is satisfied?

Systems of Boolean Equations

XOR-SAT ← Polynomial-time solver! (via Gaussian Elimination)

$$x_1 \oplus \neg x_2 \oplus x_3$$
$$x_5 \oplus x_1 \oplus x_6$$
$$x_3 \oplus \neg x_6 \oplus \neg x_5$$

} Does there exist an assignment $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. every expression is 1?

CNF-SAT (i.e. AND-of-OR SAT)

$\wedge$
- $x_1 \vee \neg x_2 \vee x_3$
- $x_5 \vee x_1 \vee x_6$
- $x_3 \vee \neg x_6 \vee \neg x_5$

} Does there exist an assignment $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. every clause is satisfied?

Systems of Boolean Equations

XOR-SAT ← Polynomial-time solver! (via Gaussian Elimination)

$$x_1 \oplus \neg x_2 \oplus x_3$$
$$x_5 \oplus x_1 \oplus x_6$$
$$x_3 \oplus \neg x_6 \oplus \neg x_5$$

} Does there exist an assignment $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. every expression is 1?

CNF-SAT (i.e. AND-of-OR SAT)

$$\bigwedge \begin{cases} x_1 \vee \neg x_2 \vee x_3 \\ x_5 \vee x_1 \vee x_6 \\ x_3 \vee \neg x_6 \vee \neg x_5 \end{cases}$$

} Does there exist an assignment $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. every clause is satisfied?

No known efficient algorithm!

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

n variables

$x_1, x_2, \ldots, x_n$

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

literal: variable or its negation

n variables $x_1, x_2, \ldots, x_n$

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

literal: variable or its negation

m clauses: a "disjunction" (i.e. OR) of a collection of literals

n variables $x_1, x_2, \cdots, x_n$

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

Formula: "Conjunction" (i.e. AND) of clauses

literal: variable or its negation

m clauses: a "disjunction" (i.e. OR) of a collection of literals

n variables $x_1, x_2, \dots, x_n$

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

Formula: "Conjunction" (i.e. AND) of clauses

literal: variable or its negation

m clauses: a "disjunction" (i.e. OR) of a collection of literals

n variables $x_1, x_2, \ldots, x_n$

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

Boolean (CNF) Formula

$$\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_5 \vee x_1 \vee x_6) \wedge (x_3 \vee \neg x_6 \vee \neg x_5)$$

Formula: "Conjunction" (i.e. AND) of clauses

literal: variable or its negation

m clauses: a "disjunction" (i.e. OR) of a collection of literals

n variables $x_1, x_2, \ldots, x_n$

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

That is, $\vec{x} = \vec{a} \in \{0,1\}^n$ s.t. $\phi(\vec{a}) = 1$

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_1) \wedge (x_3 \vee x_1)$$

Is ϕ satisfiable?

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_1) \wedge (x_3 \vee x_1)$$

Is ϕ satisfiable?

x_1	x_2	x_3	C_1	C_2	C_3	ϕ
0	0	0				
1	0	0				
0	1	0				
0	0	1				
1	1	0				
1	0	1				
0	1	1				
1	1	1				

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_1) \wedge (x_3 \vee x_1)$$

Is ϕ satisfiable?

x_1	x_2	x_3	C_1	C_2	C_3	ϕ
0	0	0	1	1	0	
1	0	0	1	1	1	
0	1	0	1	1	0	
0	0	1	0	1	1	
1	1	0	1	0	1	
1	0	1	1	1	1	
0	1	1	1	1	1	
1	1	1	1	0	1	

$$\phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_1) \wedge (x_3 \vee x_1)$$

Is ϕ satisfiable?

x_1	x_2	x_3	C_1	C_2	C_3	ϕ
0	0	0	1	1	0	0
1	0	0	1	1	1	1
0	1	0	1	1	0	0
0	0	1	0	1	1	0
1	1	0	1	0	1	0
1	0	1	1	1	1	1
0	1	1	1	1	1	1
1	1	1	1	0	1	0

Announcements

* HW5 ongoing

* Prelim 2 Conflict Survey on Ed.

* Recitation on Saturday
  ↳ Mathematical Algorithms

SAT. Given a boolean formula ϕ,
does there exist a satisfying assignment?

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

Observation. Given an assignment $\vec{x} = \vec{a}$ there is an efficient algorithm to verify

$$\phi(\vec{a}) = 0 \quad \text{or} \quad \phi(\vec{a}) = 1$$

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

Observation. Given an assignment $\vec{x} = \vec{a}$ there is an efficient algorithm to verify

$$\phi(\vec{a}) = 0 \quad \text{or} \quad \phi(\vec{a}) = 1$$

For each clause $C = (\ell_1 \vee \ell_2 \vee \cdots \ell_k) \in \phi$.

| Assign literals $\ell_i \in \{x_i, \neg x_i\}$ according to a_i
| if $C(a) = 0$, Return False

Return True

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

Observation. Given an assignment $\vec{x} = \vec{a}$ there is an efficient algorithm to verify $\phi(\vec{a}) = 0$ or $\phi(\vec{a}) = 1$

SOLVING SAT

⇓

Brute Force SAT Algorithm

For all assignments $\vec{a} \in \{000\cdots0, 000\cdots1, \cdots, 111\cdots1\}$

  Verify $(\phi, \vec{a})$

  If $\phi(\vec{a}) = 1$: Return True

Return False

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

Observation. Given an assignment $\vec{x} = \vec{a}$ there is an efficient algorithm to verify $\phi(\vec{a}) = 0$ or $\phi(\vec{a}) = 1$

SOLVING SAT ⇓

Brute Force SAT Algorithm

For all assignments $\vec{a} \in \{000\cdots0, 000\cdots1, \ldots, 111\cdots1\}$

  Verify $(\phi, \vec{a})$

  If $\phi(\vec{a}) = 1$: Return True

Return False

} $O(n\,m)$

- Set each literal in each clause
- Determine if there is a 1 in each clause.

SAT. Given a boolean formula ϕ, does there exist a satisfying assignment?

Observation. Given an assignment $\vec{x} = \vec{a}$ there is an efficient algorithm to verify $\phi(\vec{a}) = 0$ or $\phi(\vec{a}) = 1$

SOLVING SAT

⇓

Brute Force SAT Algorithm

For all assignments $\vec{a} \in \{000\cdots0, 000\cdots1, \ldots, 111\cdots1\}$ ($= 2^n$)

  Verify $(\phi, \vec{a})$ } $O(nm)$

  If $\phi(\vec{a}) = 1$: Return True

Return False

- Set each literal in each clause
- Determine if there is a 1 in each clause.

$\Rightarrow \Theta(2^n \cdot \text{poly}(n, m))$

Theorem. There exists a $O(2^n \cdot \text{poly}(n,m))$ - time algorithm for CNF-SAT.

Better Algorithms?

Not yet ---

Theorem. There exists a $O(2^n \cdot \text{poly}(n,m))$ - time algorithm for CNF-SAT.

Better Algorithms?

Not yet ---

No known $O(1.999^n)$ time algorithm for SAT

But why?

Complexity Theory

Categorize problems based on how EASY / HARD they are to solve.

Complexity Theory

Categorize problems based on how EASY / HARD they are to solve.

P ≡ Problems that can be solved in polynomial time.

Complexity Theory

Categorize problems based on how EASY / HARD they are to solve.

P ≡ Problems that can be solved in polynomial time.

NP ≡ Problems that can be verified in polynomial time.

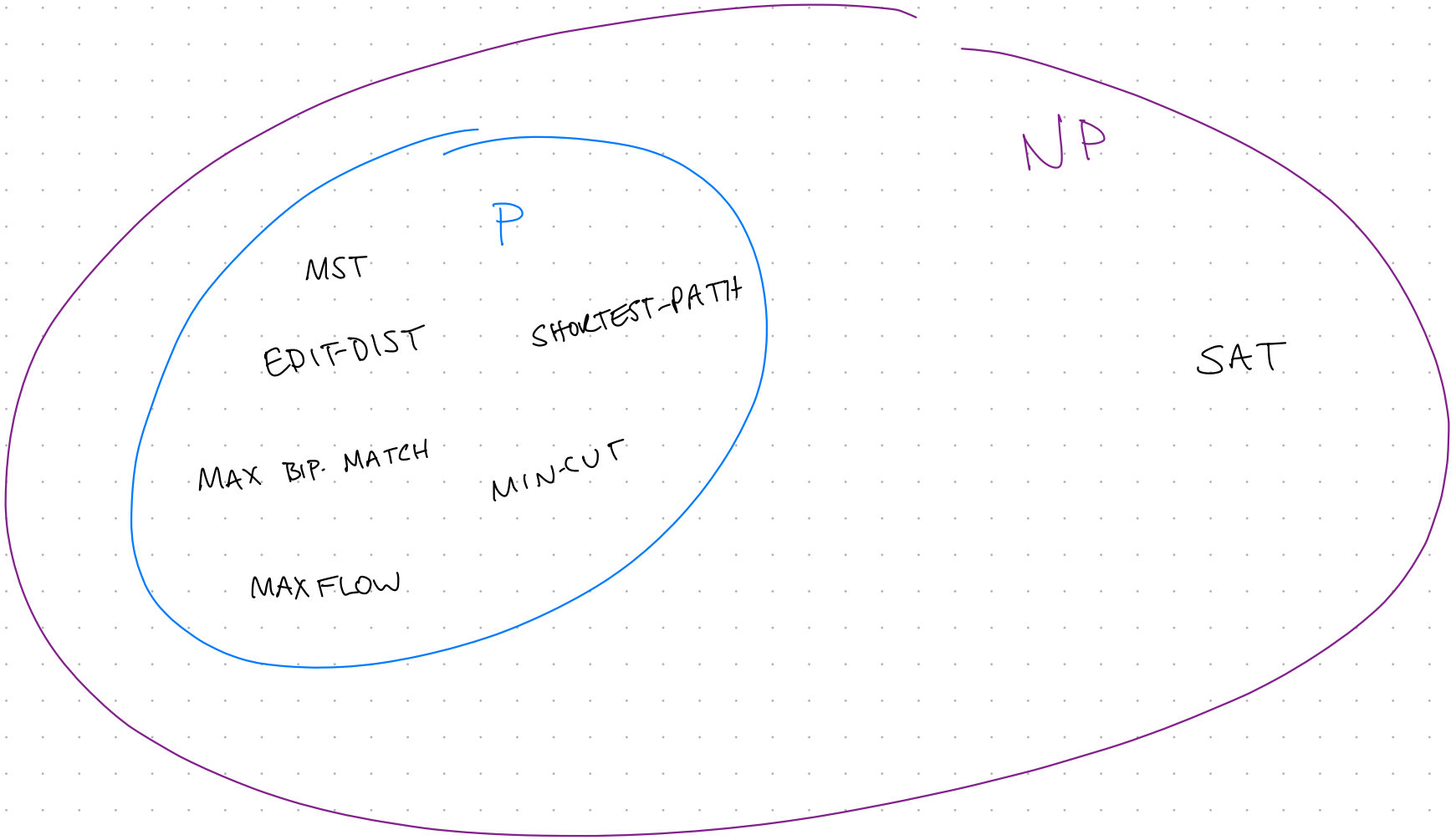
Complexity Theory

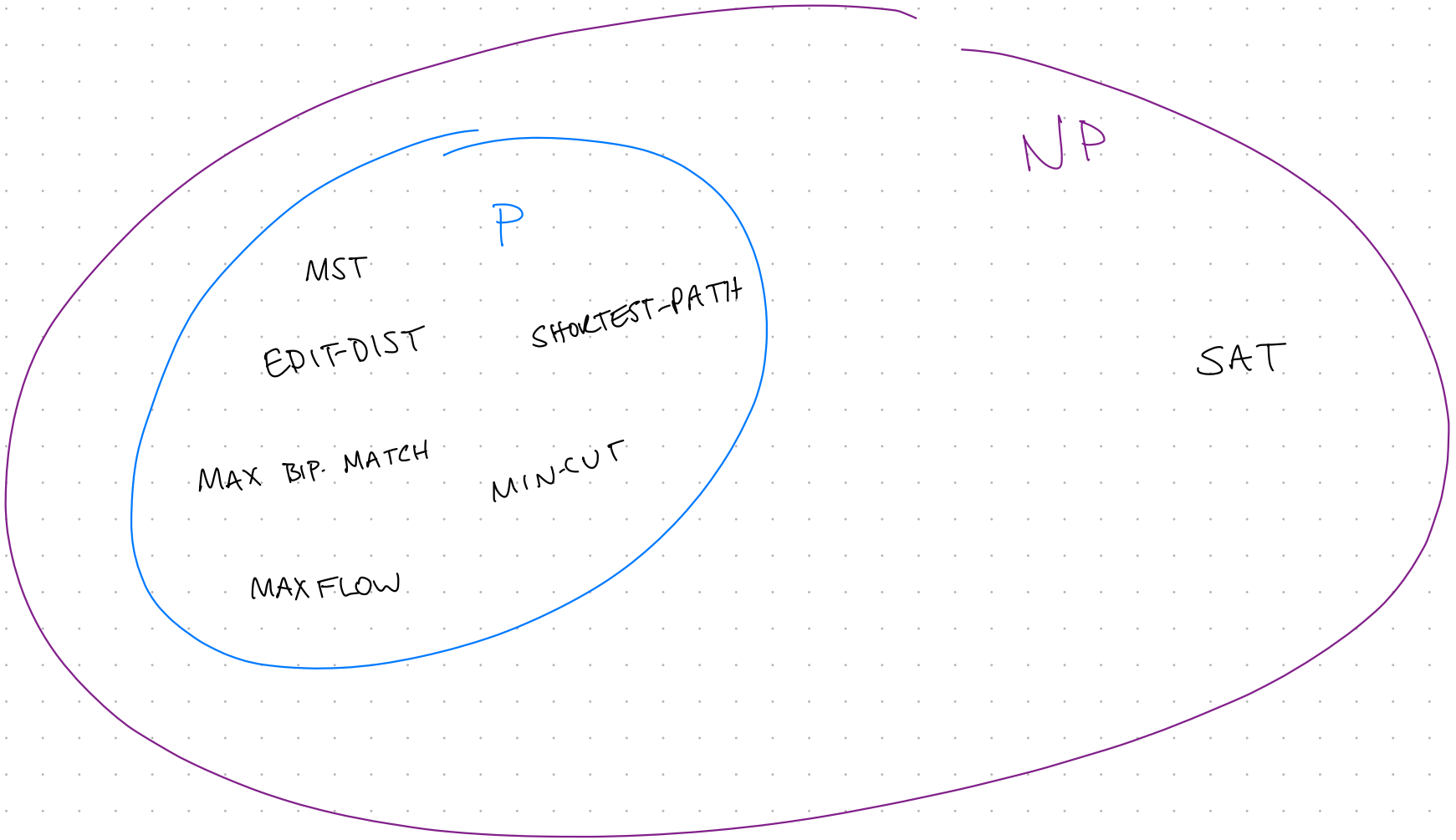
Categorize problems based on how EASY / HARD they are to solve.

P ≡ Problems that can be solved in polynomial time.

NP ≡ Problems that can be verified in polynomial time.

P vs. NP. Does efficient verification imply efficient solving?

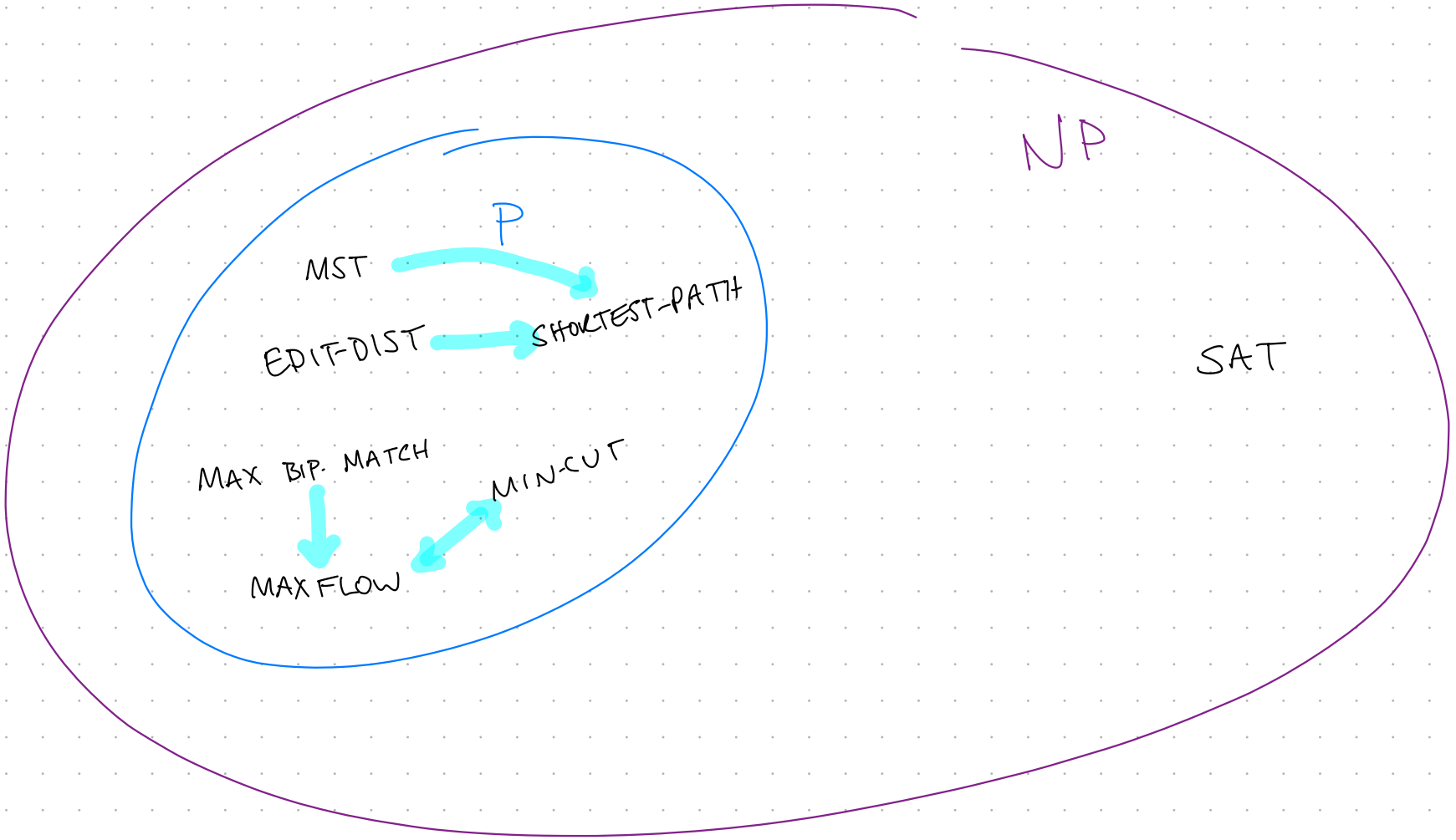

NP
P
MST
EDIT-DIST
SHORTEST-PATH
SAT
MAX BIP. MATCH
MIN-CUT
MAX FLOW



Reductions as a measure of hardness

Suppose Q reduces to R via poly-time reduction.

$$Q \leq_p R$$

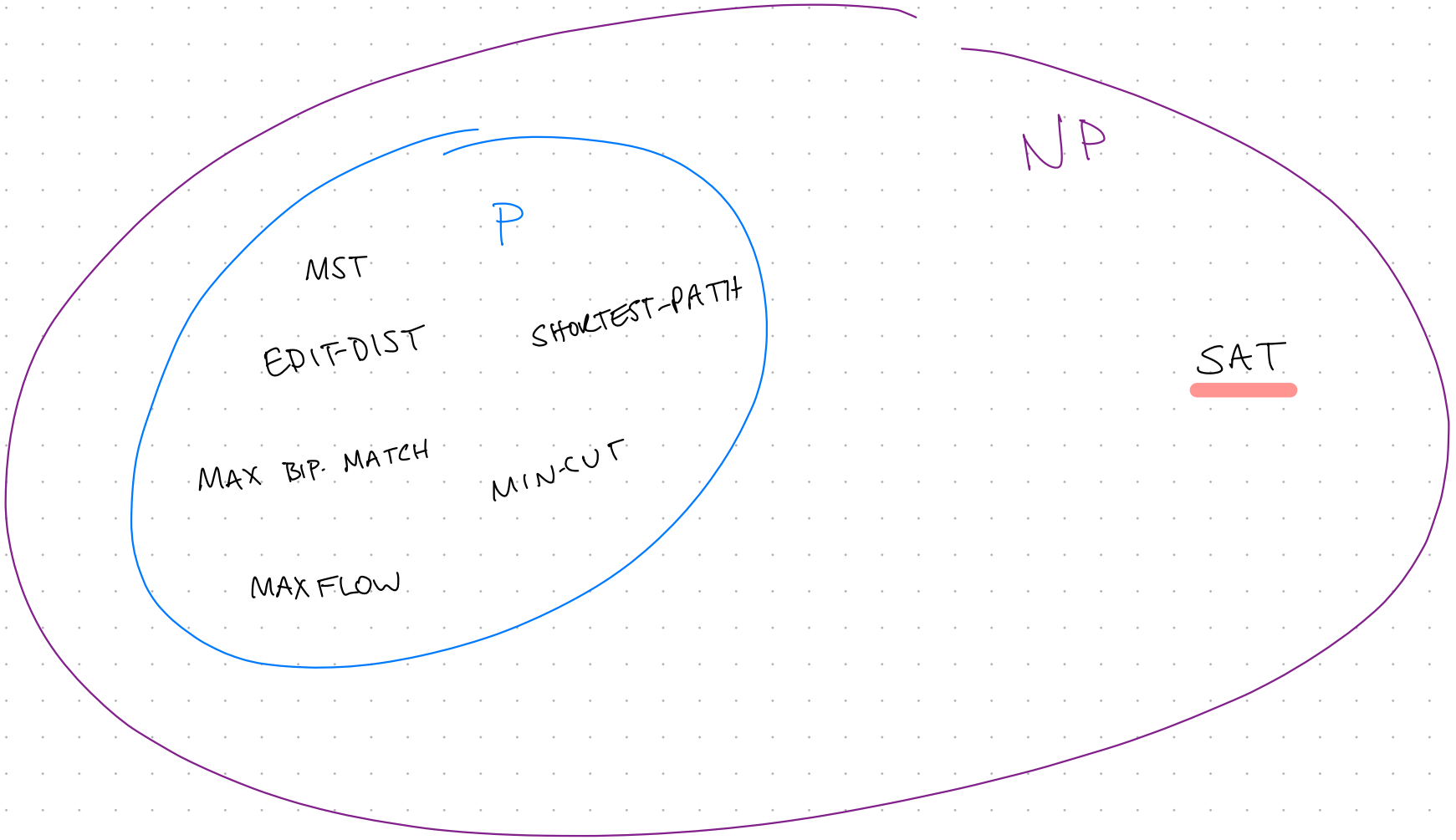


Reductions as a measure of hardness

Suppose Q reduces to R via poly-time reduction.

$Q \leq_p R$

If R is easy, so is Q!



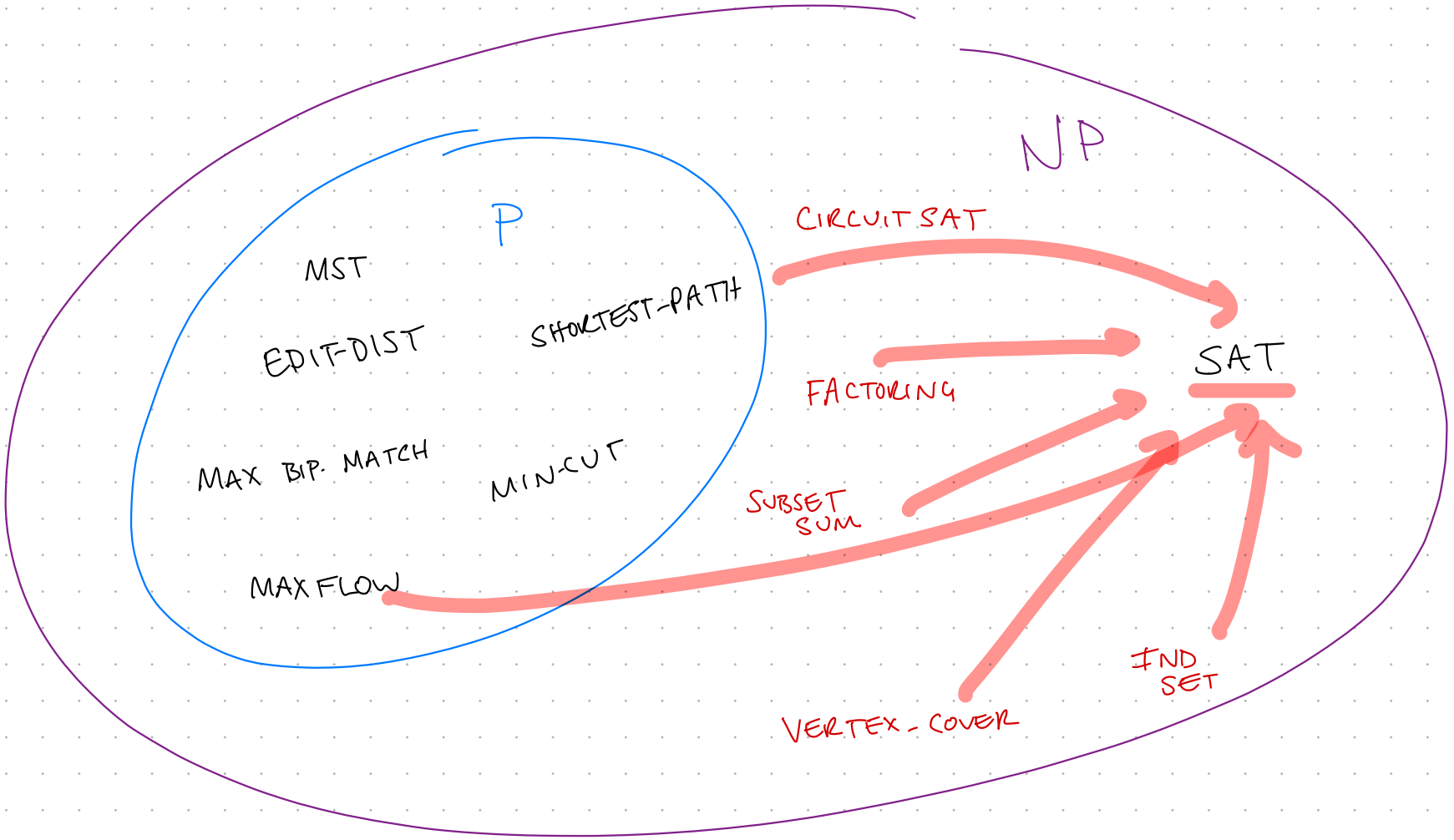
Reductions as a measure of hardness

Suppose Q reduces to R via poly-time reduction.

$Q \leq_p R$

If R is easy, so is Q!

If Q is Hard, so is R!



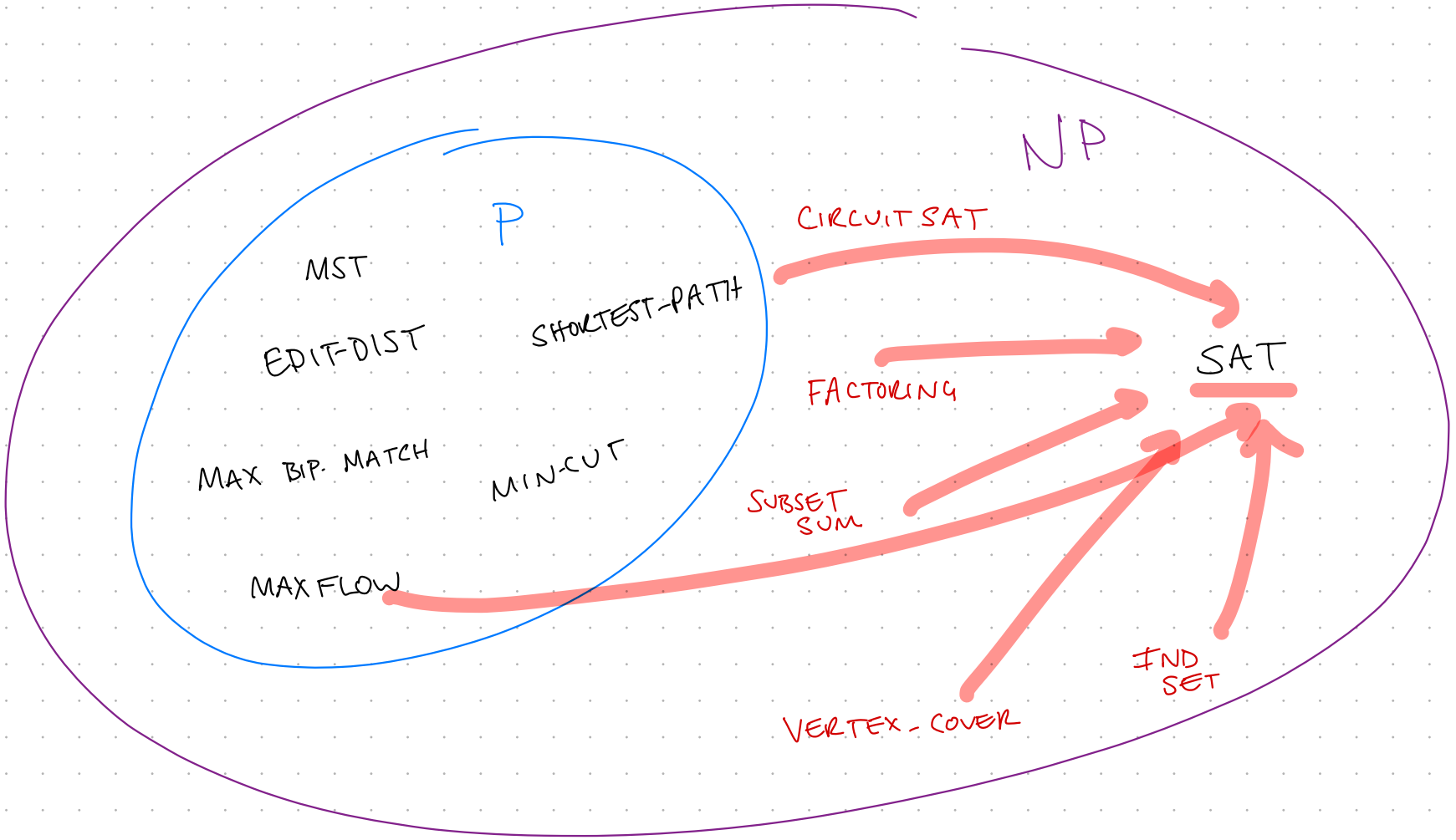
Reductions as a measure of hardness

Suppose Q reduces to R via poly-time reduction.

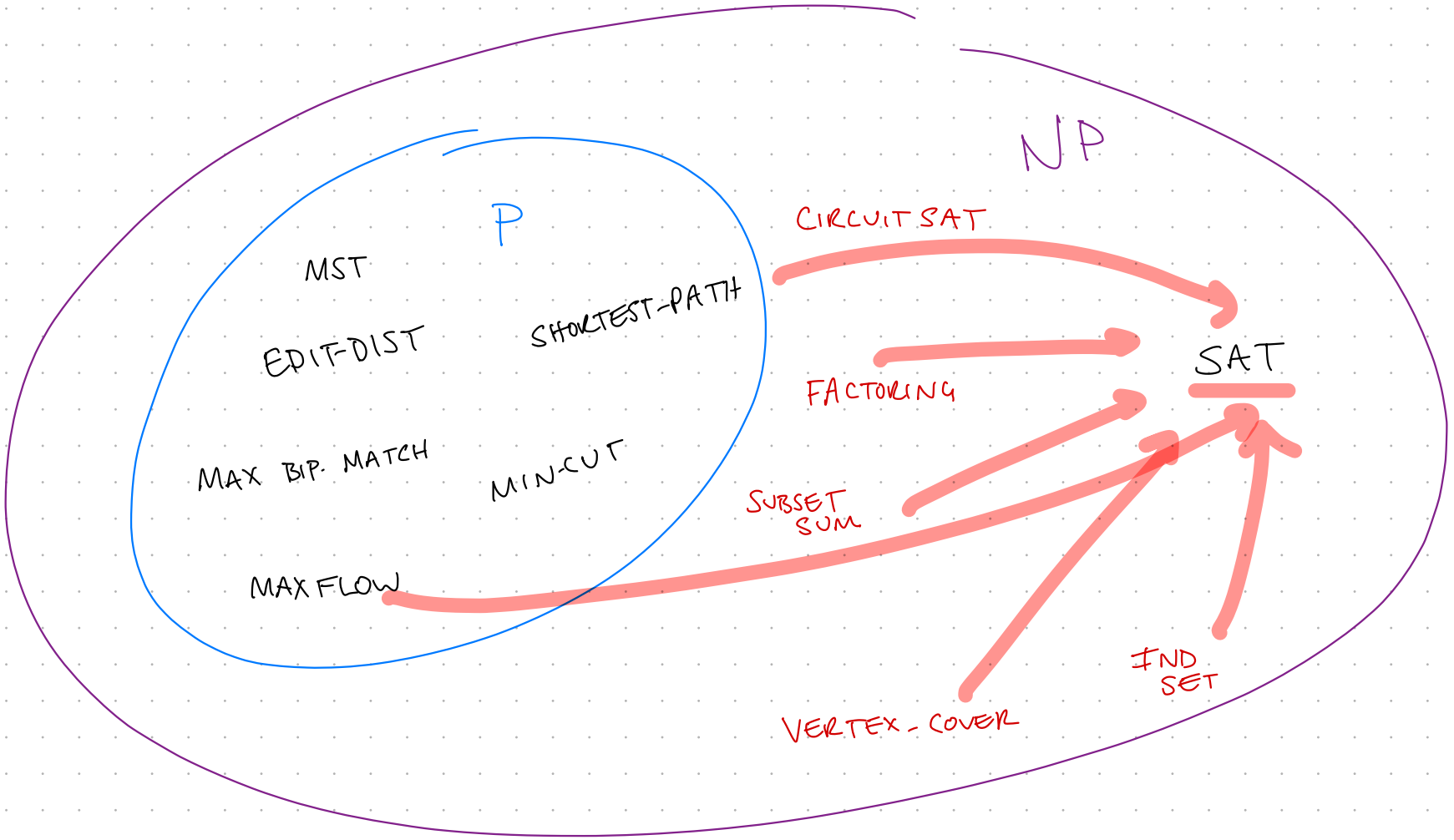
$Q \leq_p R$

If R is easy, so is Q!

If Q is Hard, so is R!



Theorem. Every problem $Q \in NP$ reduces to SAT!



Theorem. Every problem $Q \in NP$ reduces to SAT!

If any efficiently-verifiable problem is hard to solve, then SAT is hard to solve.

Complexity Theory

Categorize problems based on how EASY / HARD they are to solve.

P ≡ Problems that can be solved in polynomial time.

NP ≡ Problems that can be verified in polynomial time.

NP-Hard ≡ Q is NP-Hard if every problem in NP reduces to Q.

NP-Complete

Q is NP-Complete if.

* $Q \in NP$
* Q is NP-Hard

The Cook-Levin Theorem SAT is NP-Complete.