Gems of TCS

Fine-grained Complexity

Sasha Golovnev

February 16, 2021

Fine-grained Complexity

- Efficient algorithms for important problems?

Fine-grained Complexity

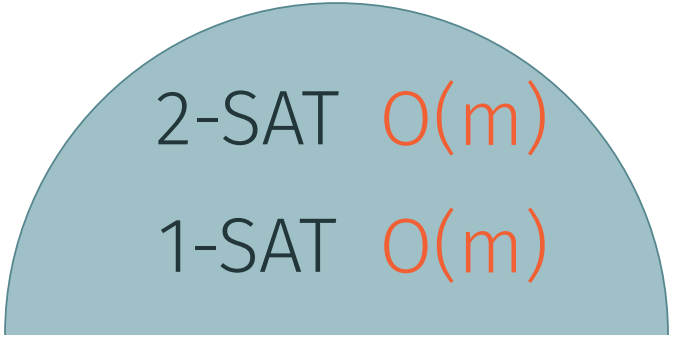
- Efficient algorithms for important problems?
- For many of them, we couldn't find better algorithms in decades

Fine-grained Complexity

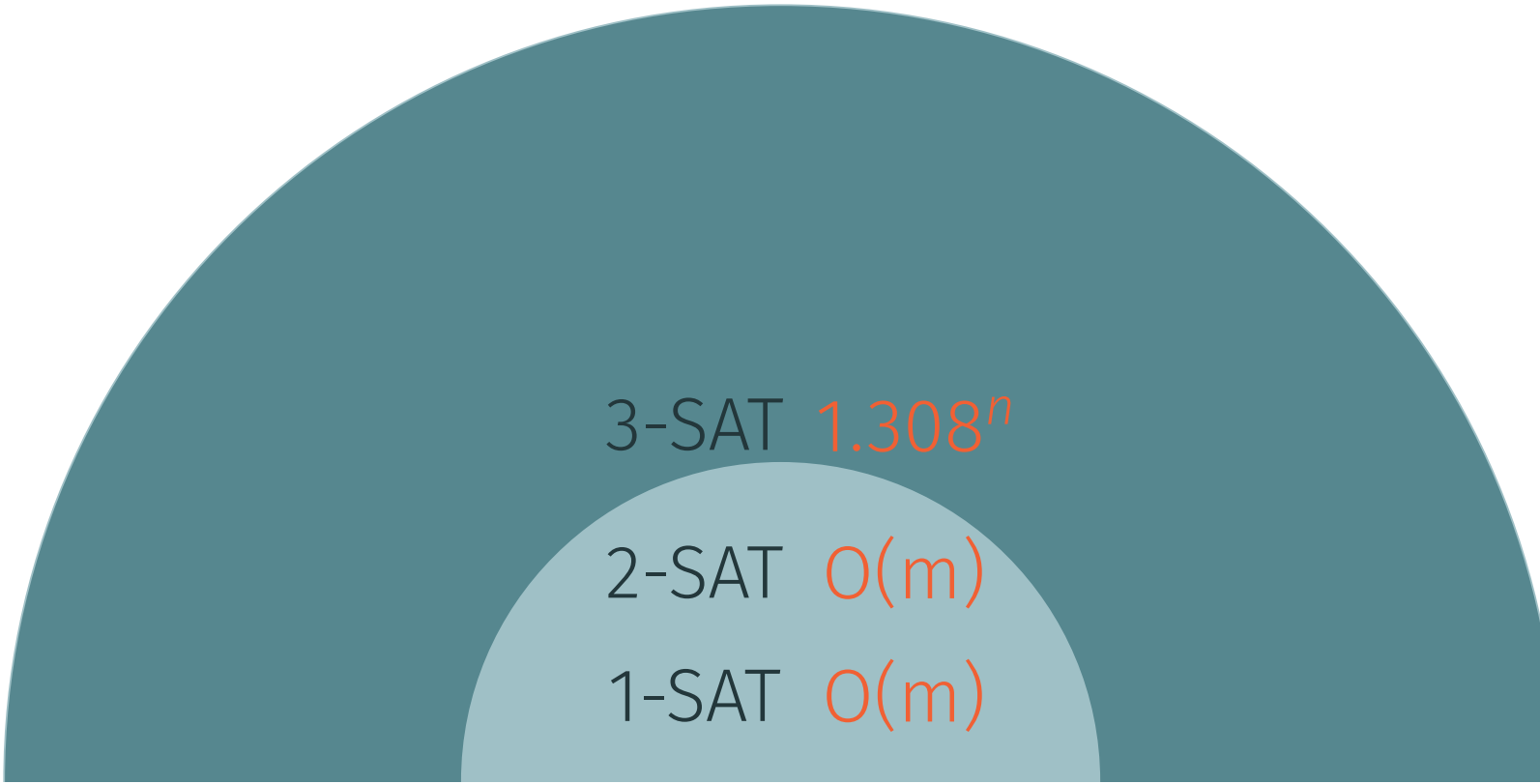
- Efficient algorithms for important problems?
- For many of them, we couldn't find better algorithms in decades
- Today: Identify reason why we're stuck

Algorithmic Complexity of SAT

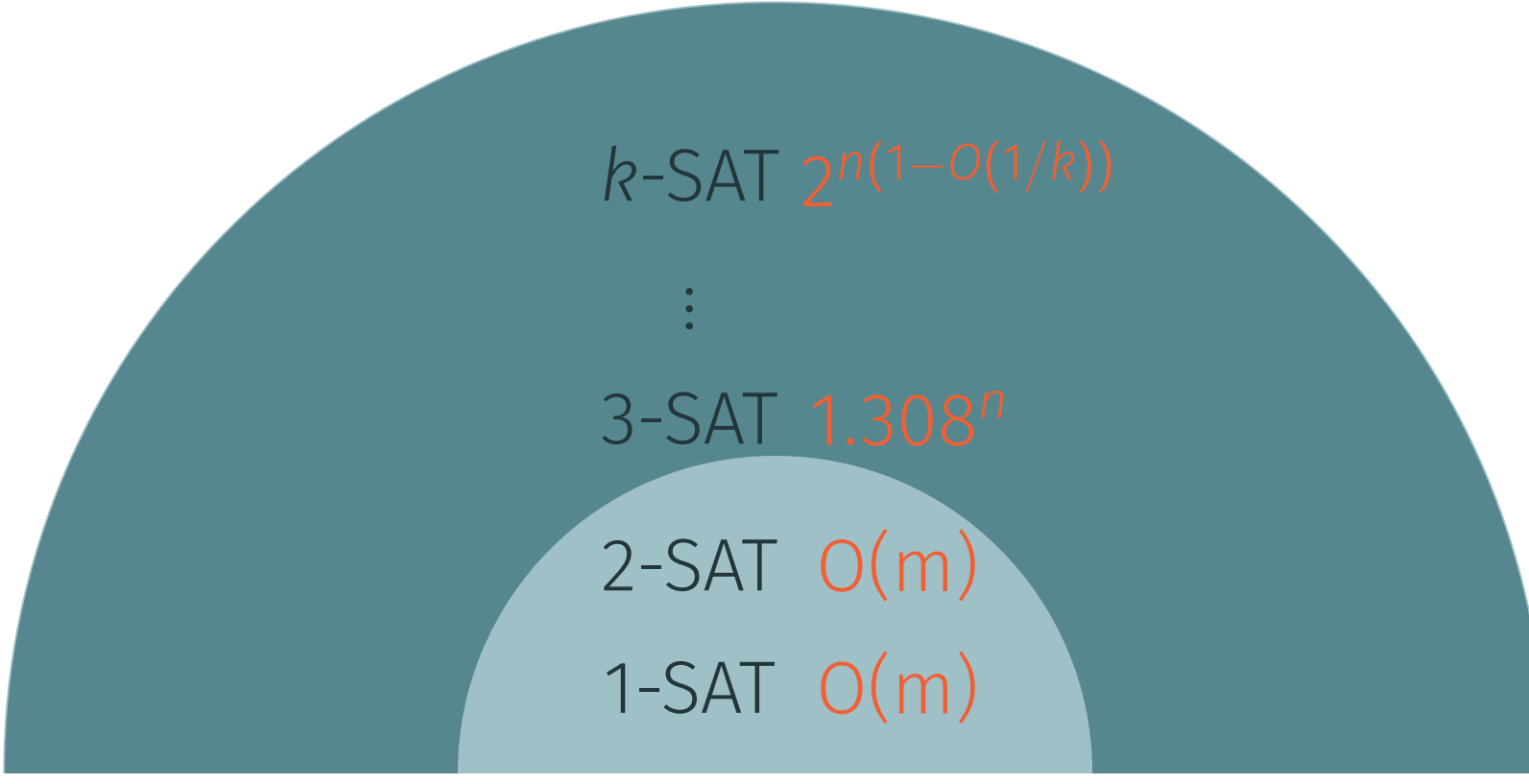
n variables $\in \{0,1\}$ 2^n assignments



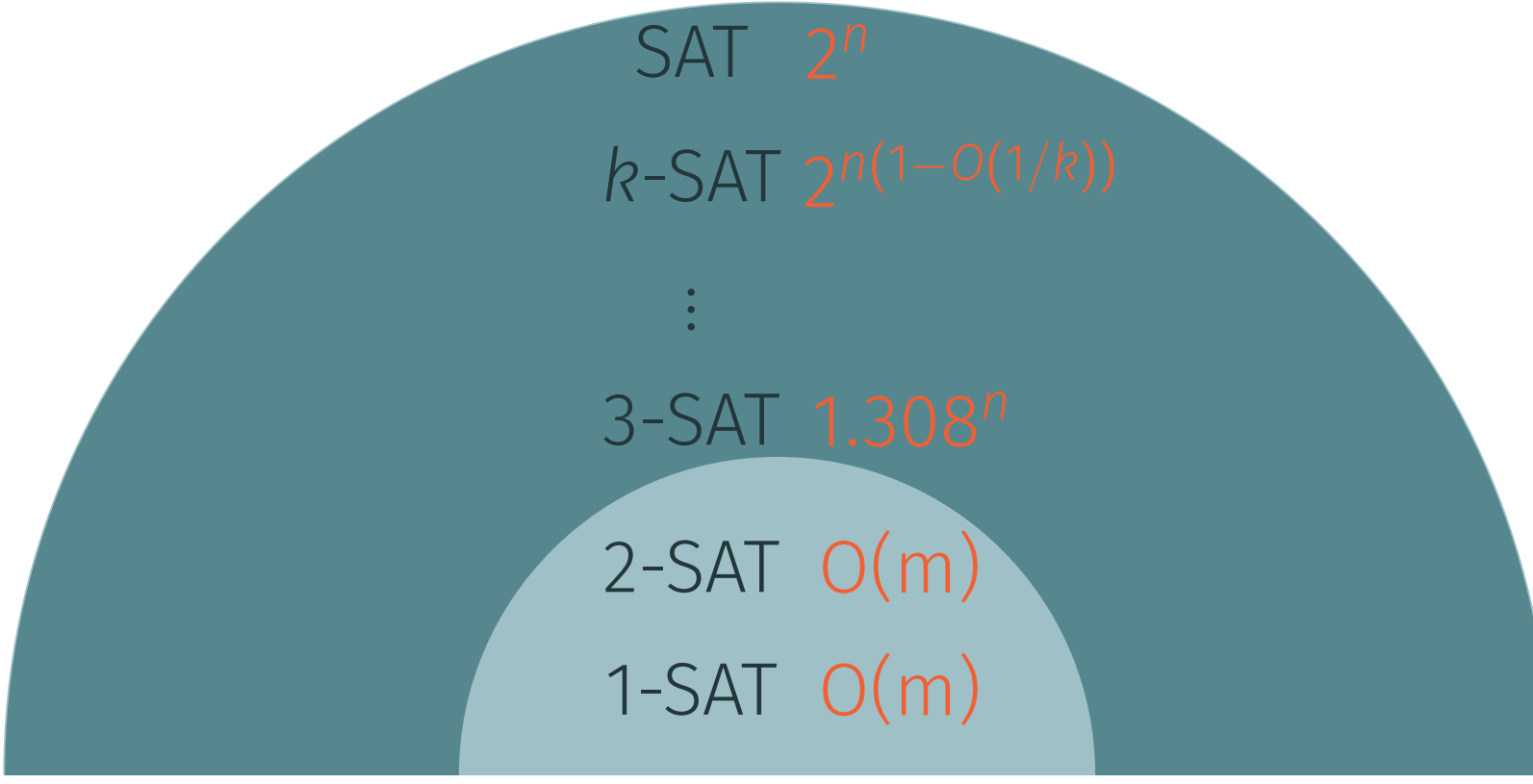
Algorithmic Complexity of SAT



Algorithmic Complexity of SAT



Algorithmic Complexity of SAT



Hardness of SAT

- SAT can be solved in time $2^n \, \mathbf{poly}(n)$
- We don't know how to solve SAT exponentially faster: in time 1.999^n

Hardness of SAT

- SAT can be solved in time $2^n \, \mathrm{poly}(n)$
- We don't know how to solve SAT exponentially faster: in time 1.999^n
- Strong Exponential Time Hypothesis (SETH)

SAT requires time 2^n

SAT cannot be solved in time $O((2-\varepsilon)^n)$ for any $\varepsilon > 0$

EQ: Any alg for SAT takes time $\tilde{\Omega}(2^n)$

Edit Distance

Edit Distance

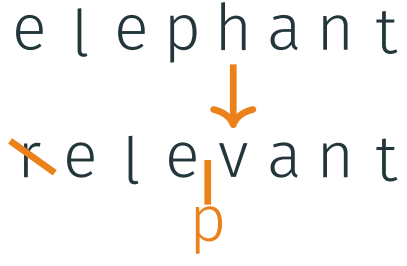
Edit Distance

e l e p h a n t

r e l e v a n t

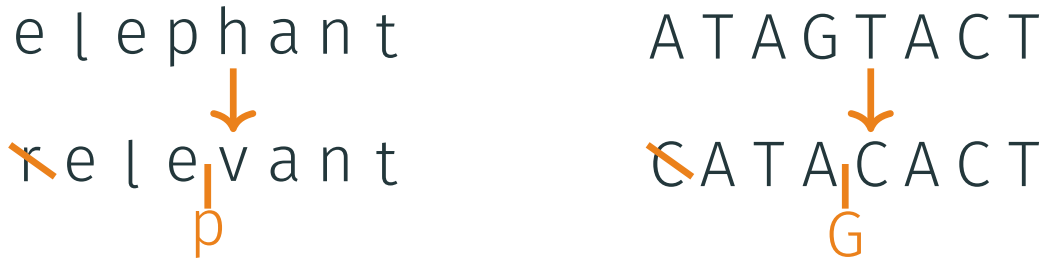
Edit Distance

Edit Distance



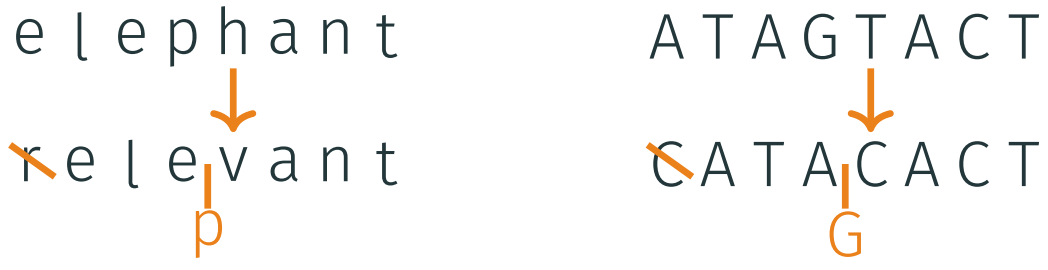
Edit Distance

Edit Distance



Edit Distance

Edit Distance



n = length of those strings

$\widetilde{O}(n^2)$

dynamic programming alg.

Other Problems

Longest Common Subsequence	Orthogonal Vectors	Longest Edit Distance
Hamming Closest Pair	All Pairs Max Flow	RNA-Folding
Regular Expression Matching	Graph Diameter	Subset Sum

Conjectured Hardness

- A conjecture for each problem?

Conjectured Hardness

- A conjecture for each problem?
- One conjecture to rule them all?

Conjectured Hardness

- A conjecture for each problem?
- One conjecture to rule them all?
- Fine-grained Complexity: Better-than-known algorithms for one problem would imply better-than-known algorithms for other problems

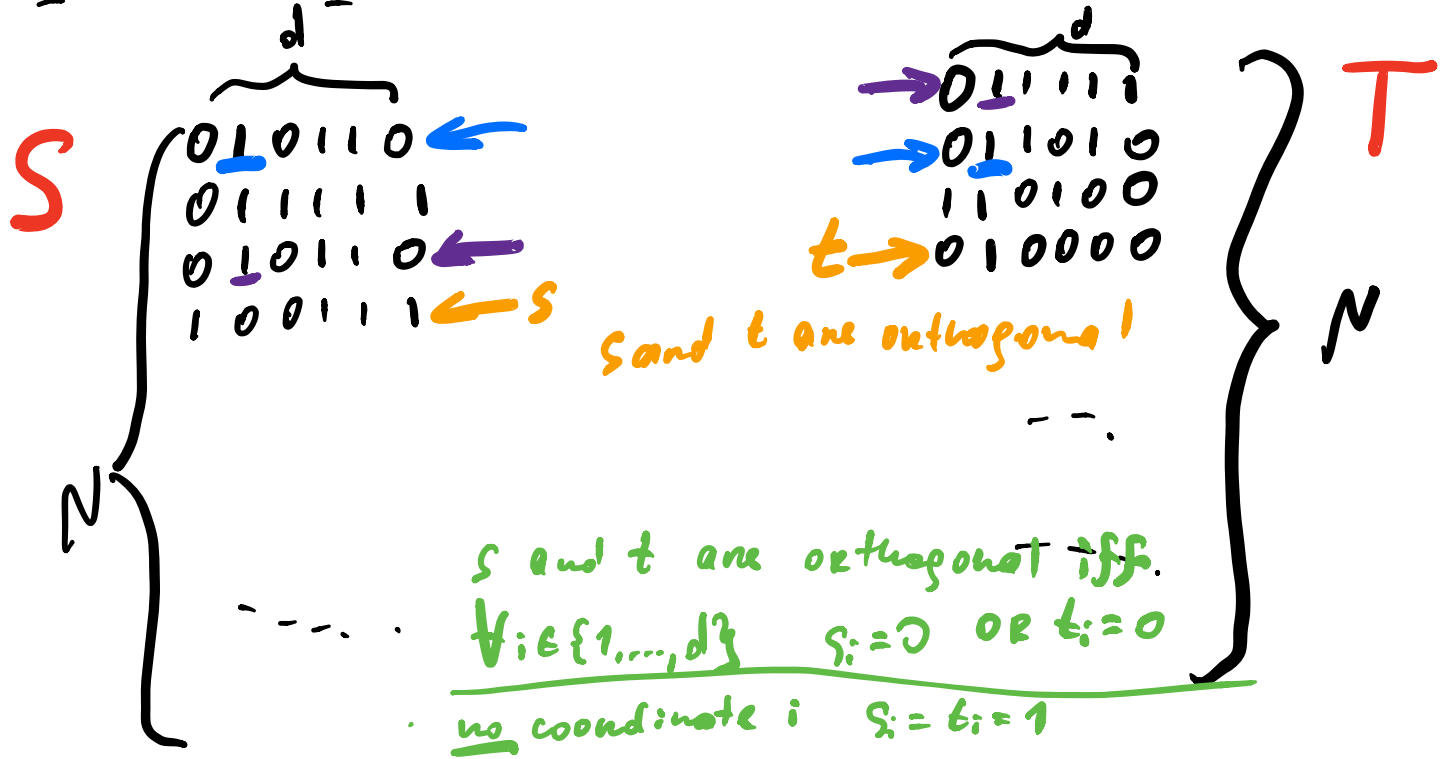
SAT

This problem cannot be solved any faster using known algorithmic techniques

Orthogonal Vectors (OV)

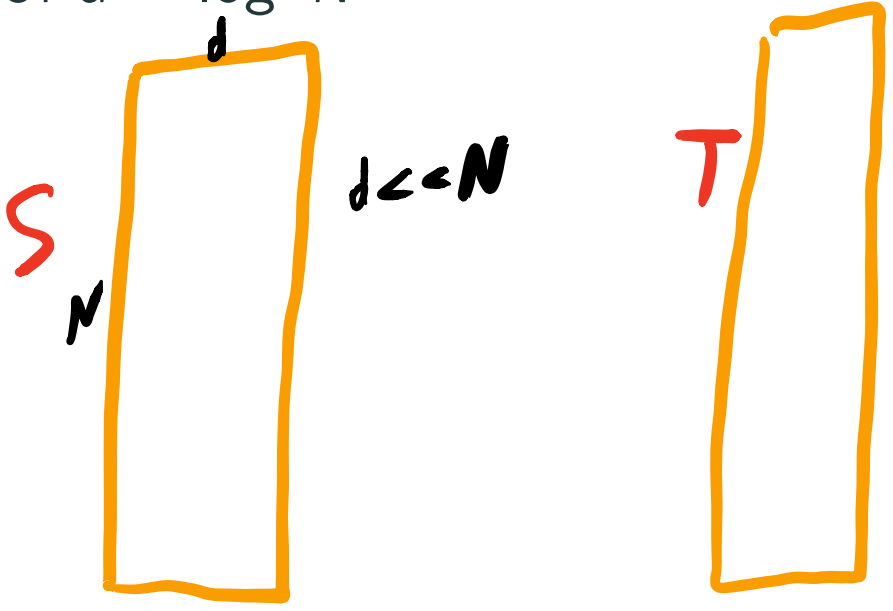
Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?



Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- Think of $d = \log^2 N$



Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- Think of $d = \log^2 N$
- Can solve in time $d \cdot N^2$

for $s \in S$
  for $t \in T$
    Check if s and t are orthogonal

N
N
d

Orthogonal Vectors Problem

- S, T are sets of N vectors from $\{0,1\}^d$. Are there $s \in S$ and $t \in T$ such that $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- Think of $d = \log^2 N$
- Can solve in time $d \cdot N^2$ $\approx N^2$
- SETH implies that OV cannot be solved in time $N^{1.99}$

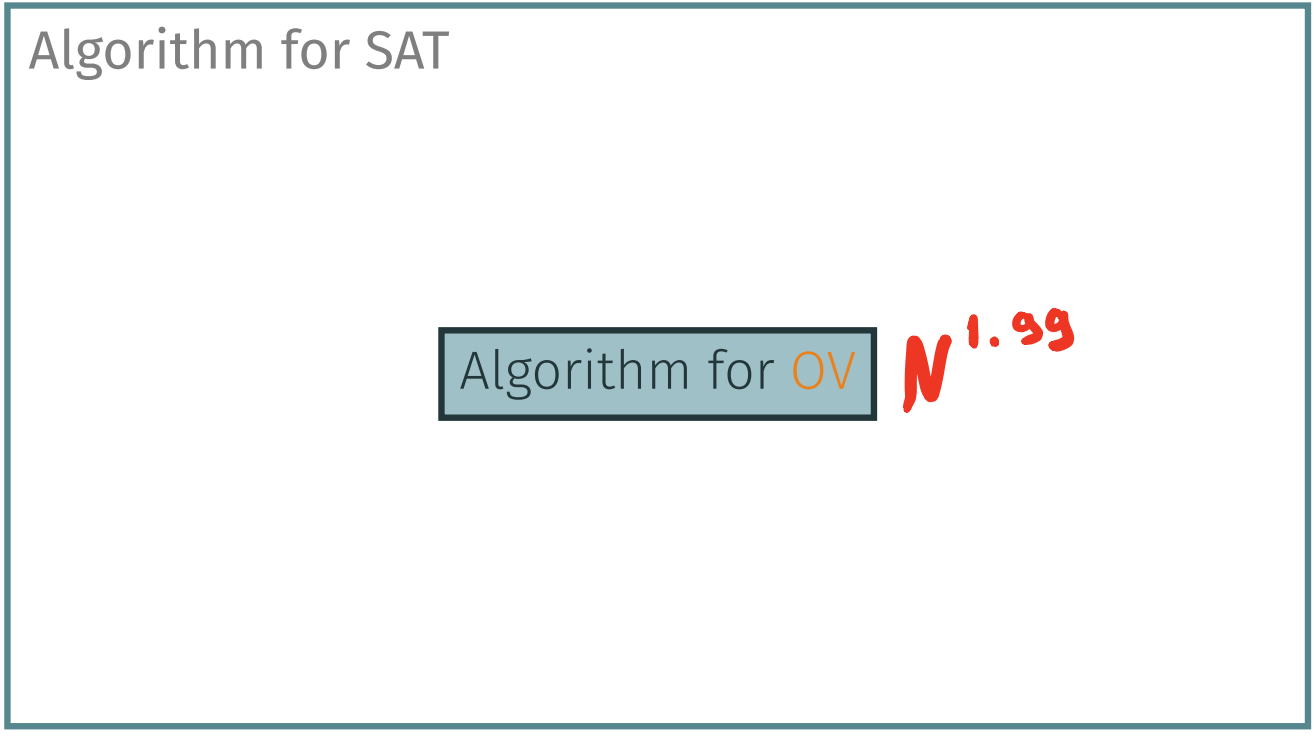
EQ: $N^{1.99}$-alg for OV $\Rightarrow$ 1.999^n for SAT

Fine-grained Reductions

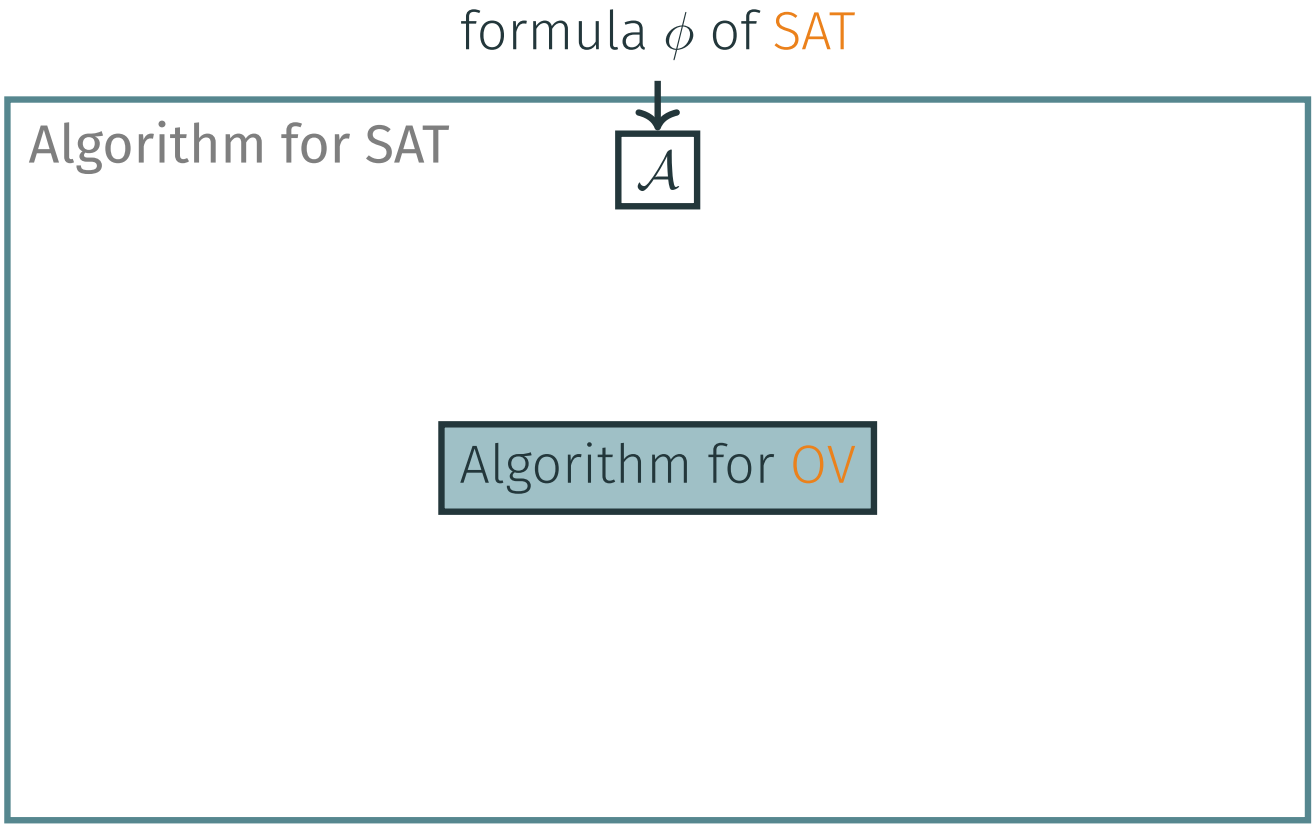
formula ϕ of SAT

Fine-grained Reductions

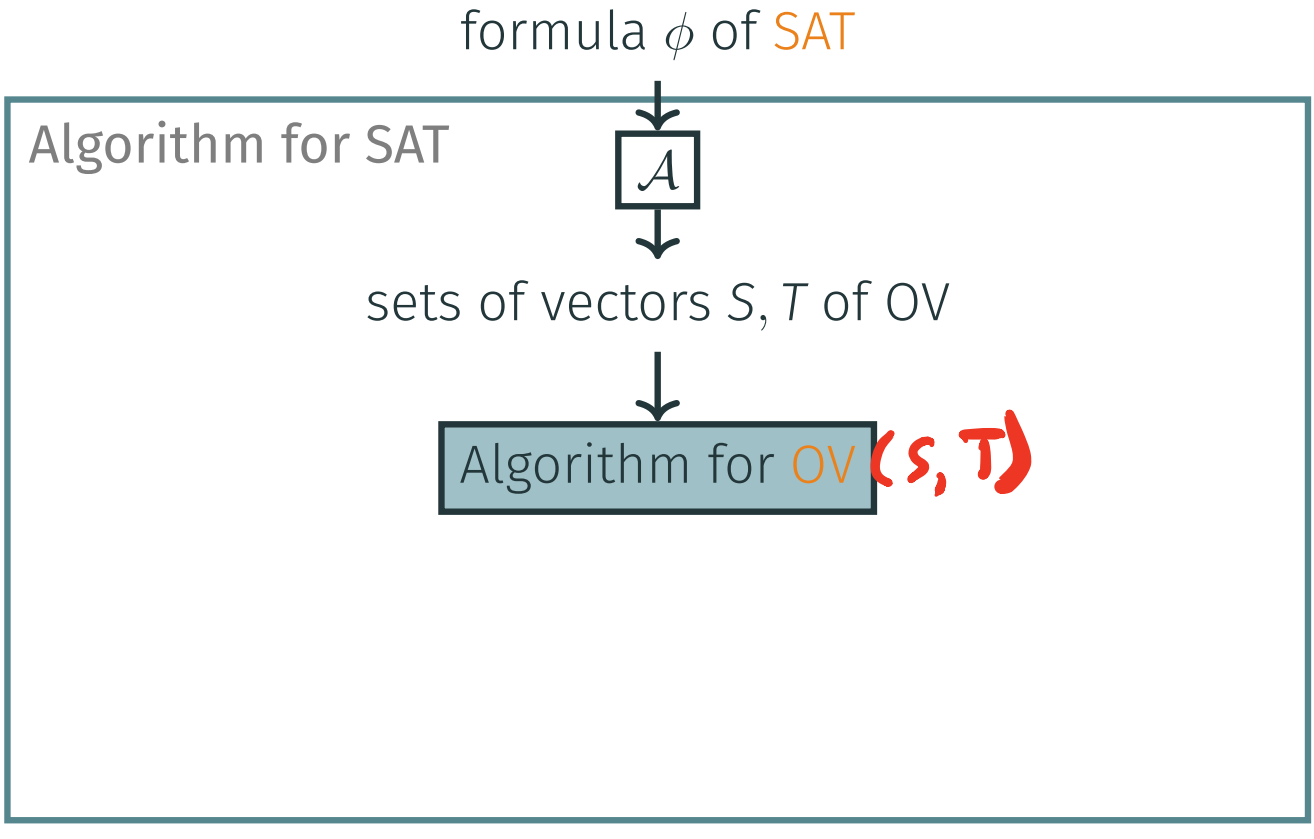
formula ϕ of SAT



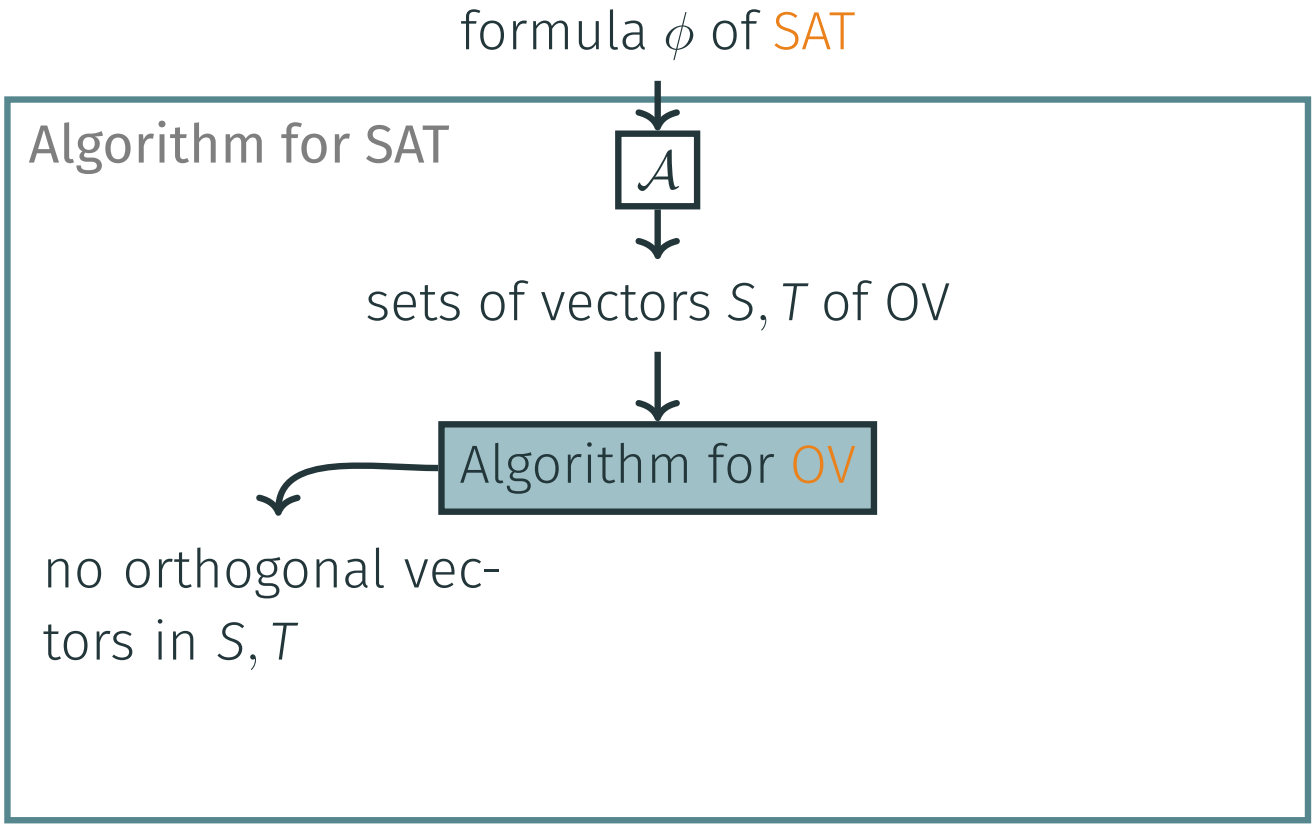
Fine-grained Reductions



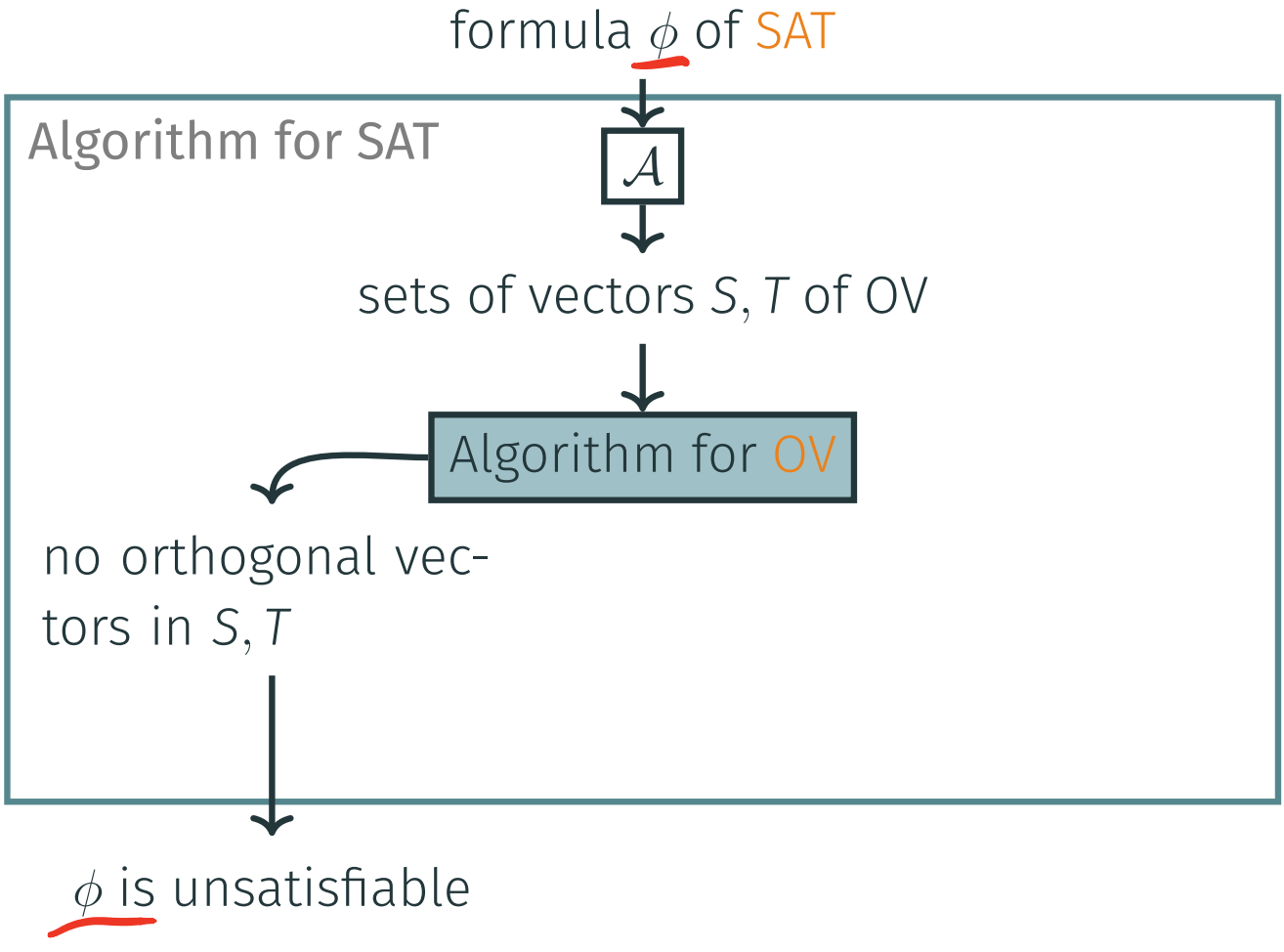
Fine-grained Reductions



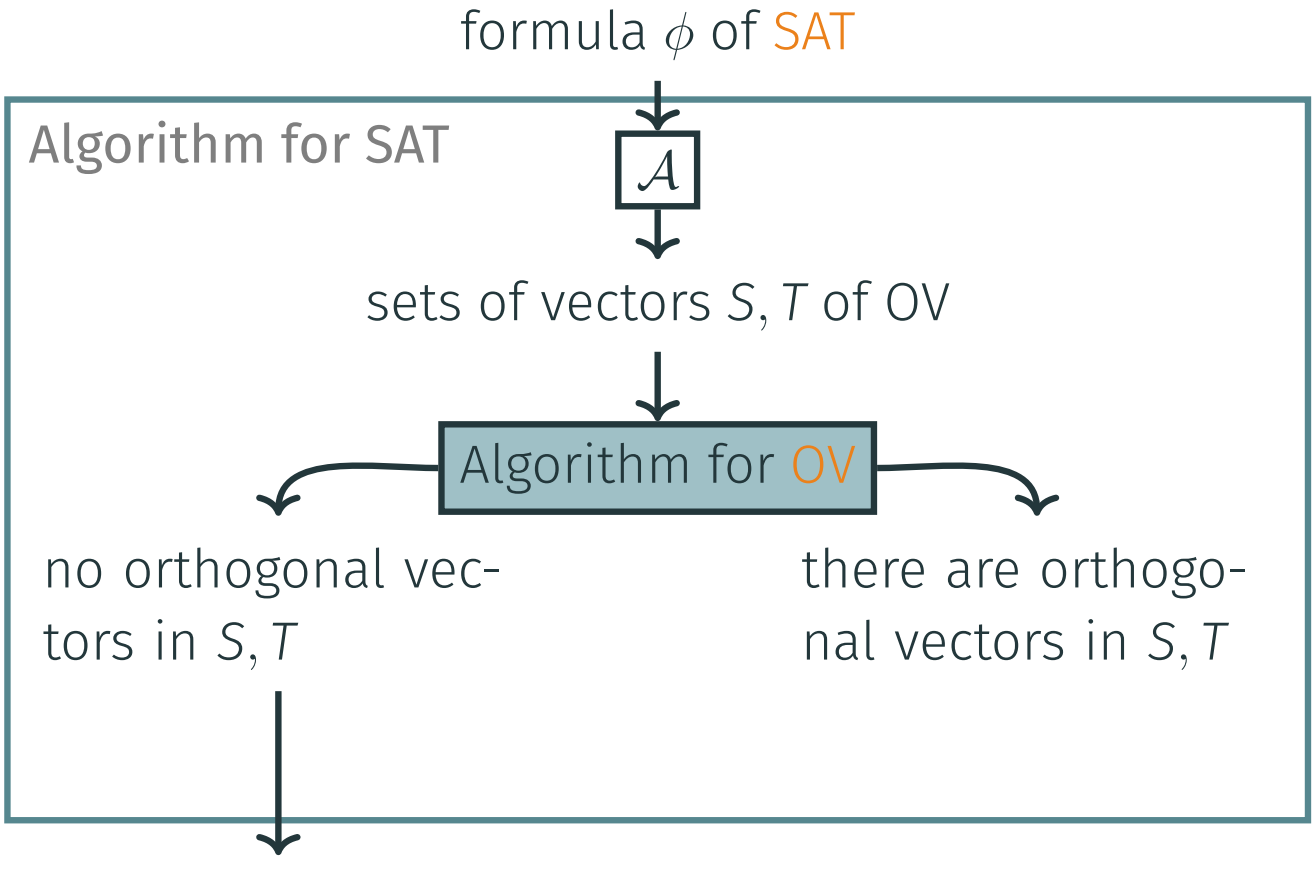
Fine-grained Reductions



Fine-grained Reductions

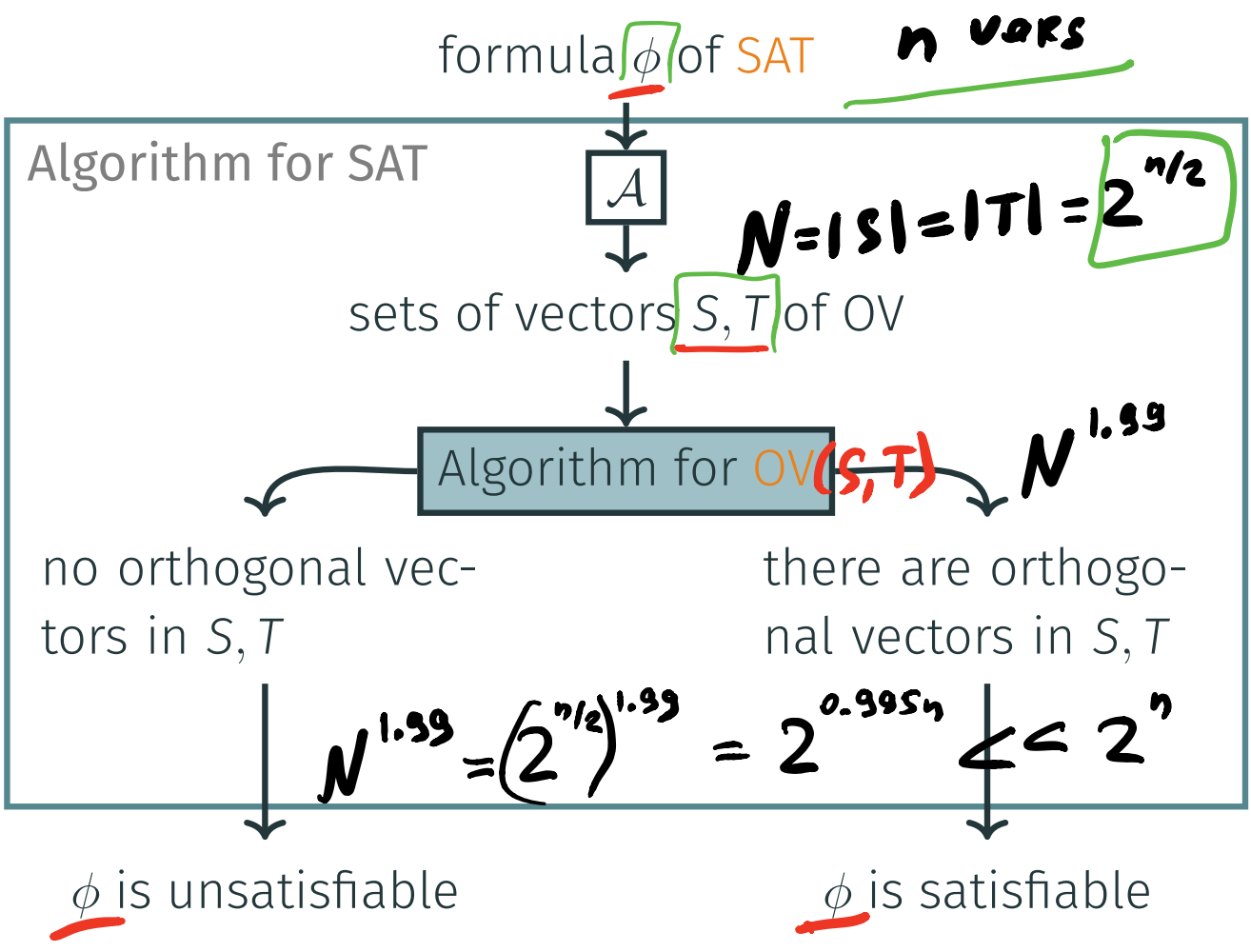


Fine-grained Reductions



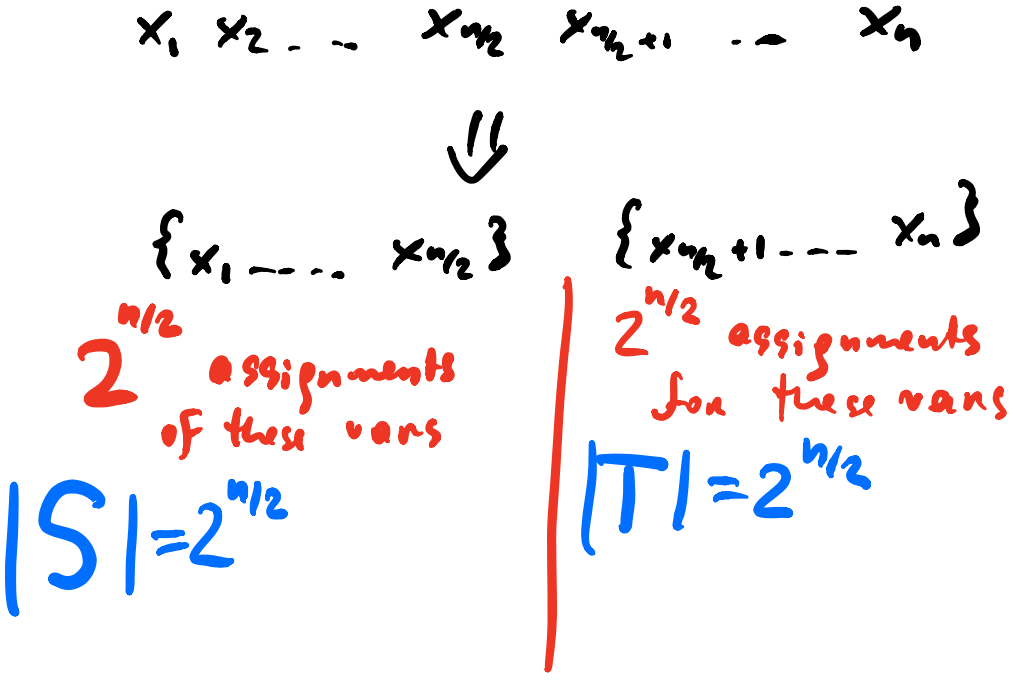
Fine-grained Reductions

Assume OV in time $N^{1.99}$



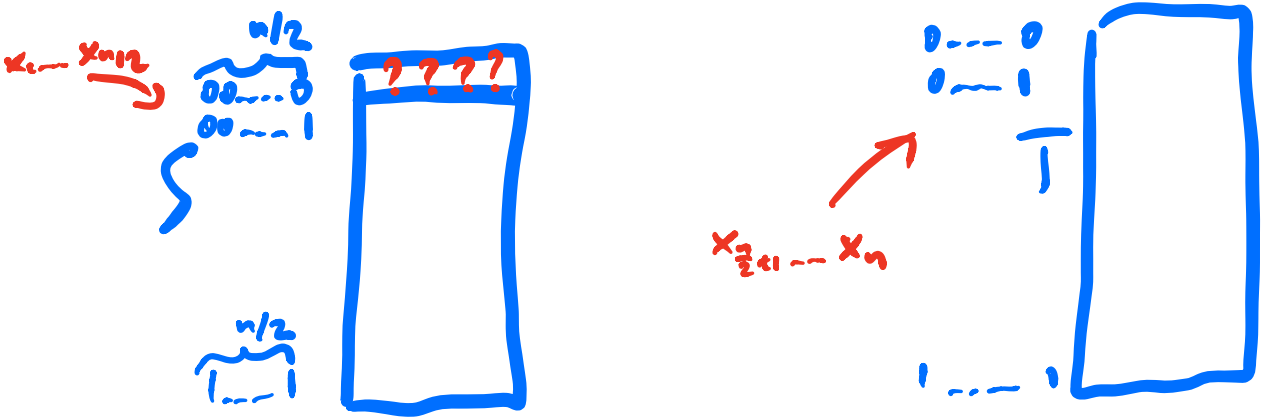
SETH $\Longrightarrow$ OV

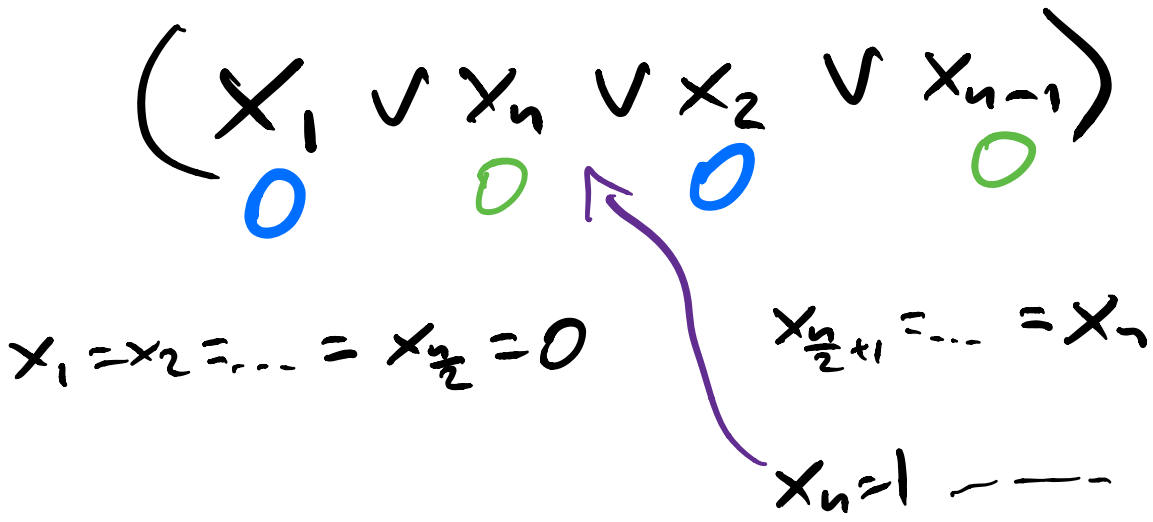
- Given a SAT formula ϕ, split its n input variables into two sets of size $n/2$



SETH $\Longrightarrow$ OV

- Given a SAT formula ϕ, split its n input variables into two sets of size $n/2$
- For each assignment to the first group — a vector in S, for each assignment to the second — a vector in T





a clause is satisfied iff it's satisfied by assignment of $x_1 \dots x_{\frac{n}{2}}$ or assignment of $x_{\frac{n}{2}+1} \dots x_n$

EQ: a clause is <u>not</u> satisfied iff it's not satisfied by $x_1 \dots x_{\frac{n}{2}}$ <u>and</u> not satisfied by $x_{\frac{n}{2}+1} \dots x_n$

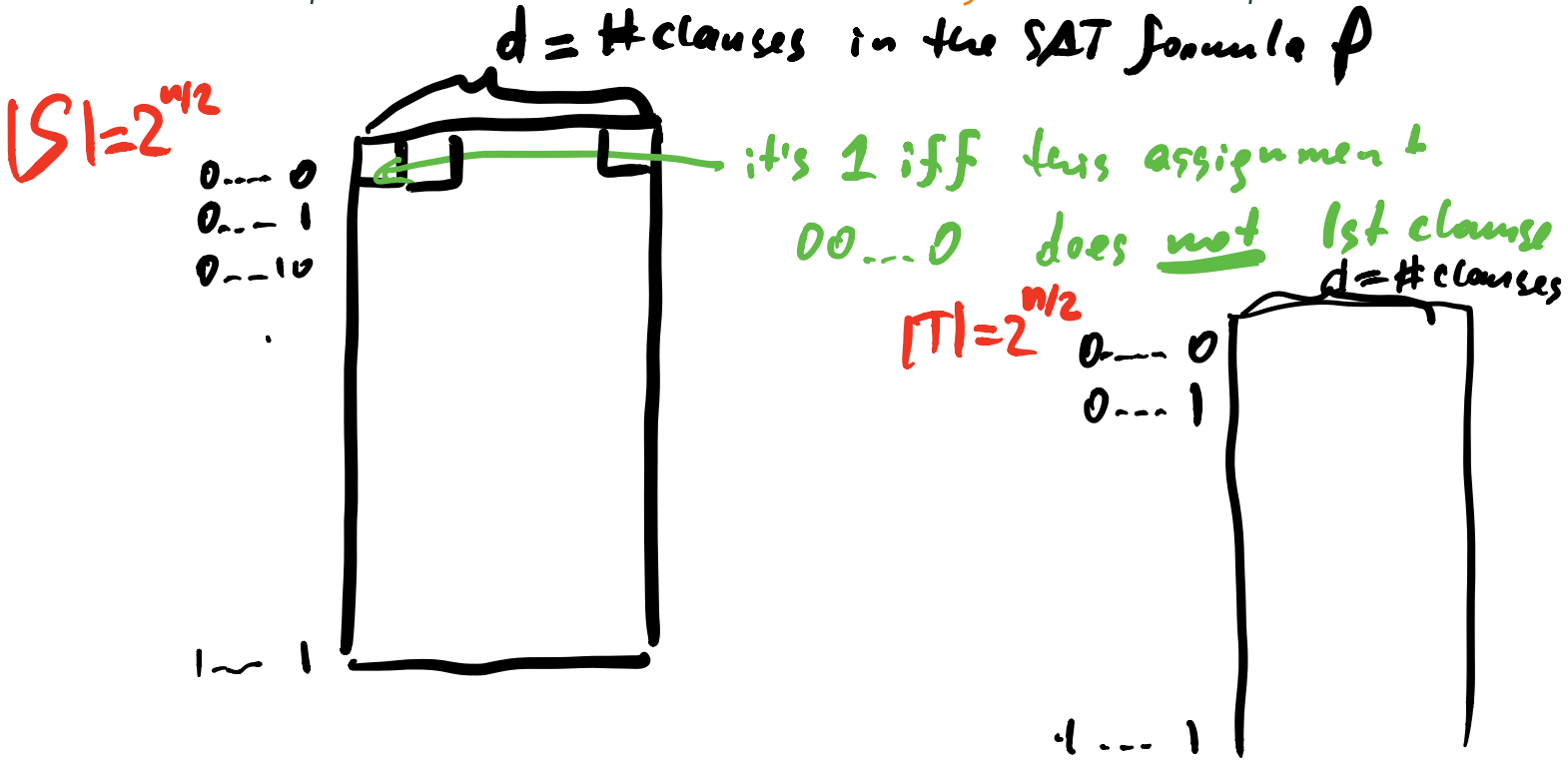
SETH $\Longrightarrow$ OV

- Given a SAT formula ϕ, split its n input variables into two sets of size $n/2$
- For each assignment to the first group — a vector in S, for each assignment to the second — a vector in T
- $N = 2^{n/2}$

SETH $\Longrightarrow$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$s_i = 1$ iff x doesn't satisfy clause C_i



SETH $\Longrightarrow$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$: (assignment to $x_1 \dots x_{n/2}$; assignment to $x_{\frac{n}{2}+1} \dots x_n$)

$$\forall i \in \{1, \dots, m\}: \; s_i \cdot t = 0$$

$(x_1 \vee x_n \vee x_2 \vee x_{n-1})$ is sat iff it's satisfiable

by $x_1 \dots x_{n/2}$ or $x_{\frac{n}{2}+1} \dots x_n$

0 0

it's not sat if both vectors have 1

SETH $\Longrightarrow$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in \{1, \ldots, m\}: \quad x_i \cdot y_i = 0$$

$N = 2^{n/2}$

- An $N^{1.99}$ algorithm for OV gives an algorithm for SAT with run time

$$N^{1.99}$$

SETH $\Longrightarrow$ OV

- For an assignment $x \in \{0,1\}^{n/2}$, add $s \in \{0,1\}^m$ to S:

$$s_i = 1 \text{ iff } x \text{ doesn't satisfy clause } C_i$$

- ϕ is SAT iff $\exists s \in S, t \in T$:

$$\forall i \in \{1,\ldots,m\}: \quad x_i \cdot y_i = 0$$

SETH => P ≠ NP

- An $N^{1.99}$ algorithm for OV gives an algorithm for SAT with run time

$$N^{1.99} = (2^{n/2})^{1.99} = 2^{0.995n}$$

=> refute SETH => you came up with a new algorithmic idea!!!

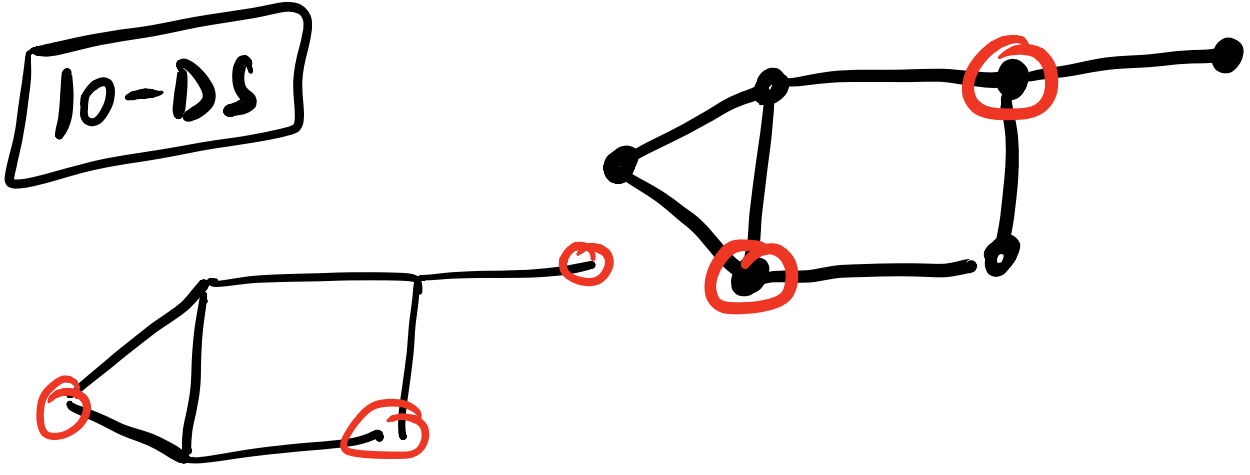
The Dominating Set Problem

Dominating Set

For every k,

- k-Dominating Set: Given $G = (V, E)$, $|V| = n$, find an $S \subseteq V$, $|S| = k$ such that

$$\forall v \in V\colon\ v \in S \text{ or } \exists u \in S\colon\ (v, u) \in E$$



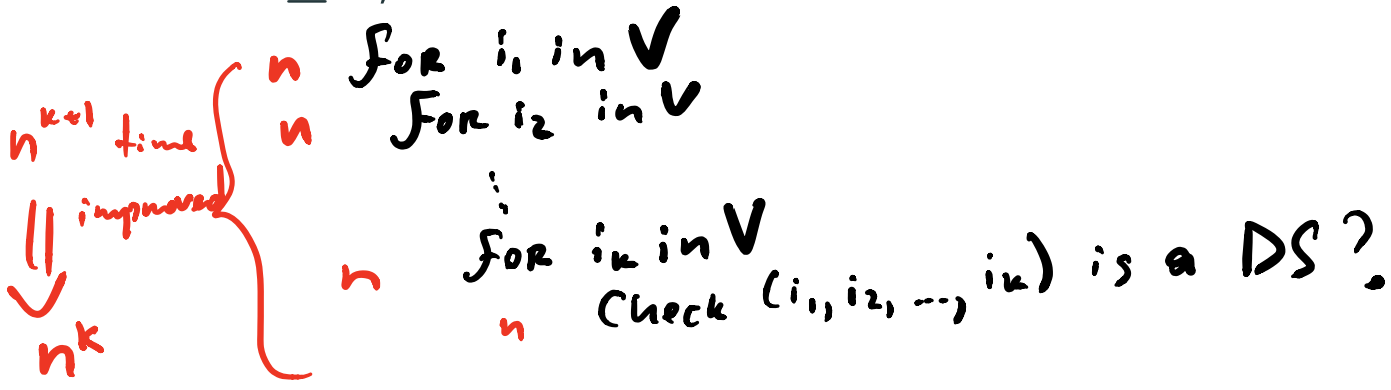
Dominating Set

- k-Dominating Set: Given $G = (V, E)$, $|V| = n$, find an $S \subseteq V$, $|S| = k$ such that

$$\forall v \in V\colon v \in S \text{ or } \exists u \in S\colon\ (v, u) \in E$$

10-DS, it can be solved in time n^{10}

- For $k \geq 7$, solvable in n^k



Dominating Set

- k-Dominating Set: Given $G = (V, E)$, $|V| = n$, find an $S \subseteq V$, $|S| = k$ such that

$$\forall v \in V\colon\ v \in S \text{ or } \exists u \in S\colon\ \ (v, u) \in E$$

- For $k \geq 7$, solvable in n^k
- SETH implies that k-DS cannot be solved in time $n^{k-0.01}$ for any k

Cannot solve 10-DS $n^{9.99}$

SETH $\Longrightarrow$ DS

SAT with n vars $x_1 \ldots x_n$ $k=10$

Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

Given SAT ϕ $\Rightarrow$ Graph G s.t.

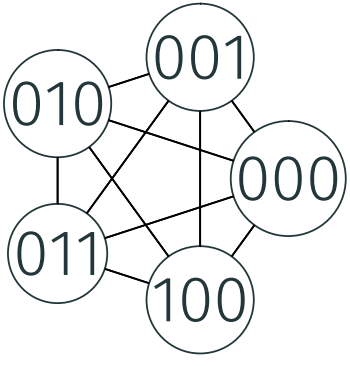
ϕ is satisfiable iff G has DS of size k

SETH $\Longrightarrow$ DS

Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

$2^{n/k}$ vertices

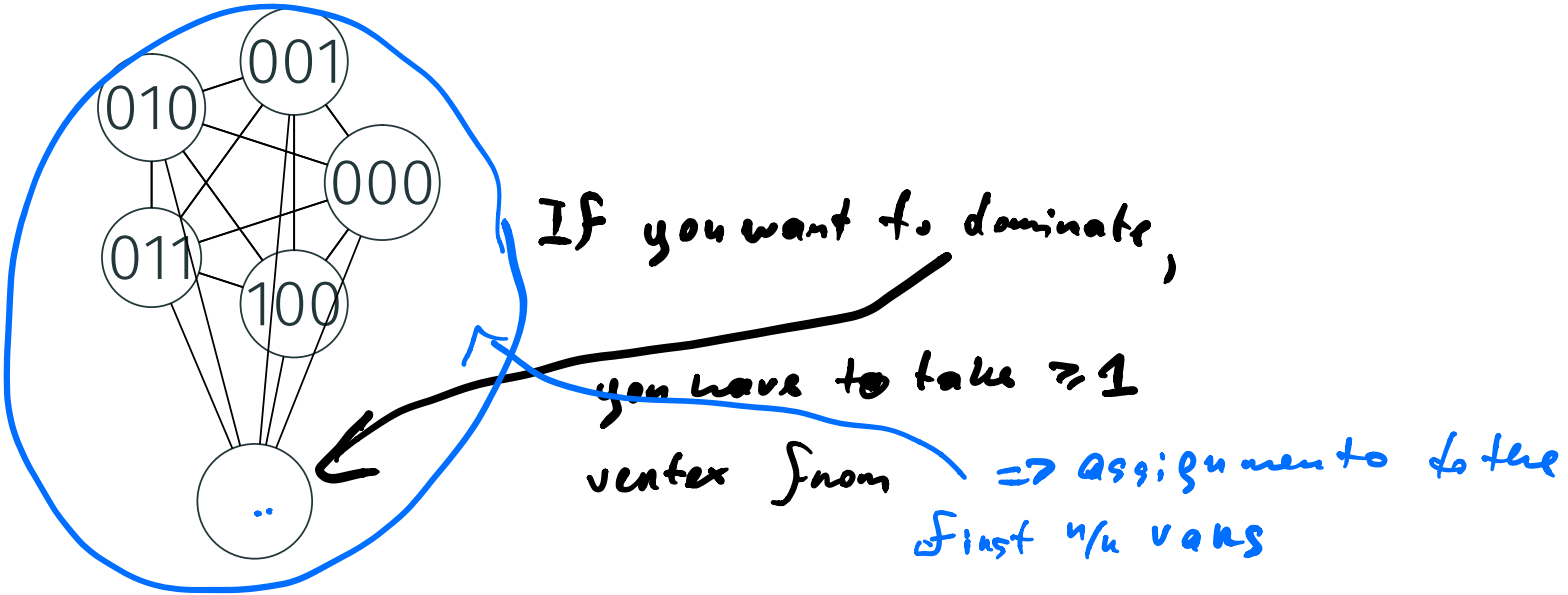


SETH $\Longrightarrow$ DS

Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}, |\mathrm{DS}| = k$

$2^{n/k}$ vertices



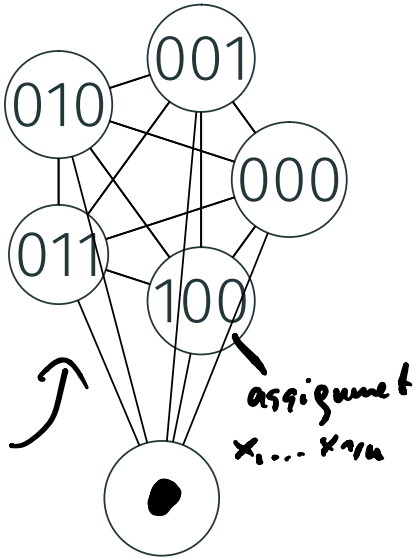
SETH $\Longrightarrow$ DS

Every DS of size k takes exactly one vertex from each of the k parts

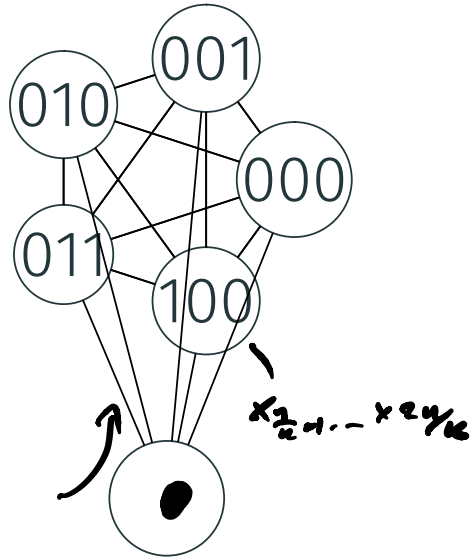
Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}$, $|\mathrm{DS}| = k$

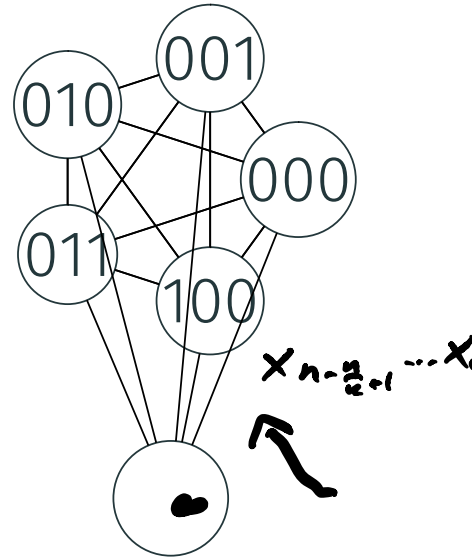
$2^{n/k}$ vertices



$2^{n/k}$ vertices



$2^{n/k}$ vertices



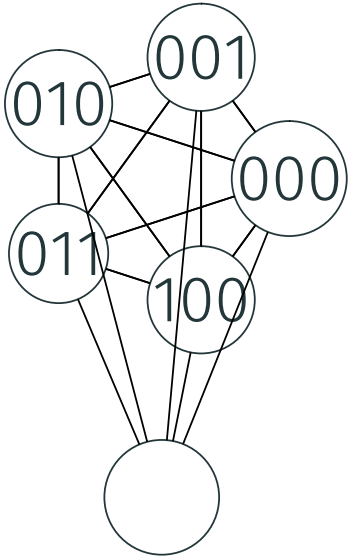
SETH $\Longrightarrow$ DS

Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}$, $|\mathrm{DS}| = k$

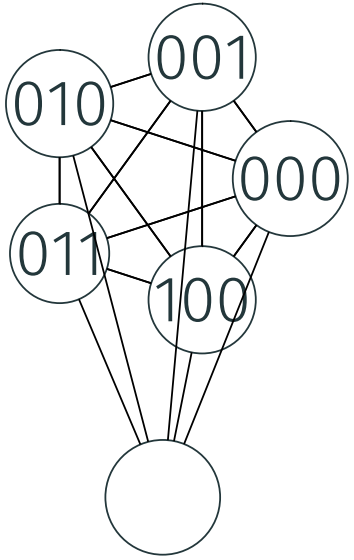
m vertices:

$2^{n/k}$ vertices



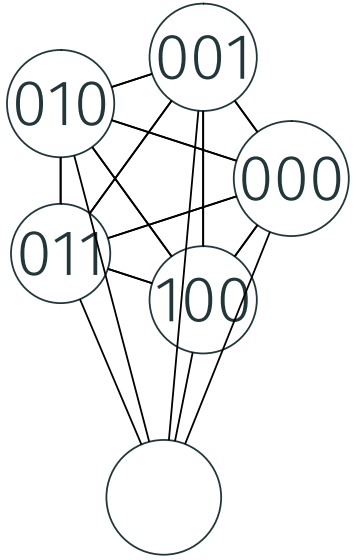
cl_1 cl_2

$2^{n/k}$ vertices



cl_3

$2^{n/k}$ vertices

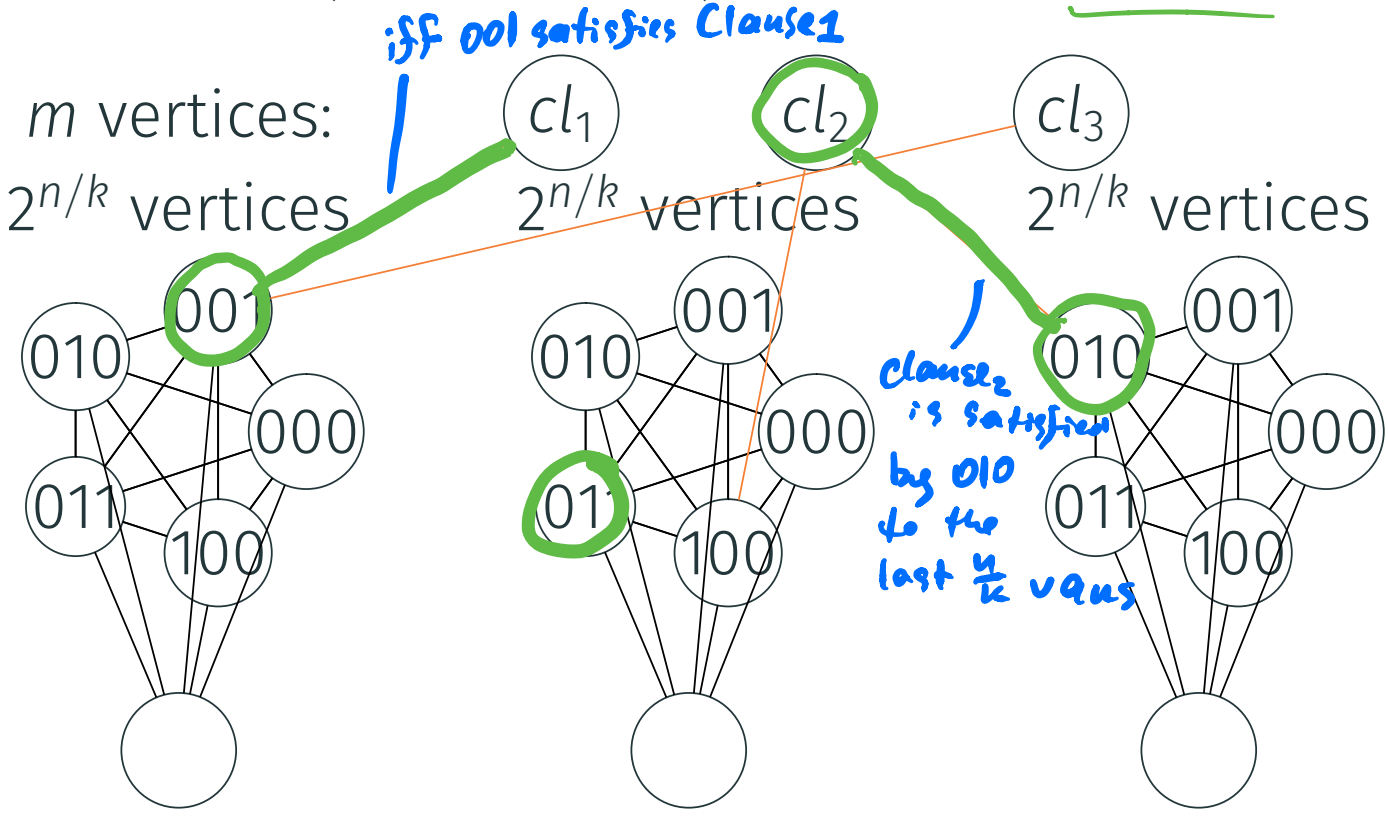


SETH $\Longrightarrow$ DS

|DS| = k must have 1 vertex from each part.

Partition vars in k groups:

$\{x_1, \ldots, x_{n/k}\}, \ldots, \{x_{n-n/k+1}, \ldots, x_n\}$, $|\mathrm{DS}| = k$



SETH $\Longrightarrow$ DS

- For every k, we reduce SAT on n vertices k-DS with

  $\approx 2^{n/k}$

  vertices

SETH $\Longrightarrow$ DS

- For every k, we reduce SAT on n vertices k-DS with

$$\approx 2^{n/k}$$

vertices

- If k-DS on N vertices can be solved in time $N^{k-0.1}$ then SAT can be solved in time

$$N^{k-0.1} = 2^{(n/k)(k-0.1)} = 2^{n-0.1n/k} << 2^n$$

=> refute SETH => congrats!! you found a new algorithmic idea!