

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

2-SAT $O(m)$

1-SAT $O(m)$

ALGORITHMIC COMPLEXITY OF SAT

3-SAT 1.308^n

2-SAT $O(m)$

1-SAT $O(m)$

ALGORITHMIC COMPLEXITY OF SAT

k -SAT $2^{n(1-O(1/k))}$

⋮

3-SAT 1.308^n

2-SAT $O(m)$

1-SAT $O(m)$

ALGORITHMIC COMPLEXITY OF SAT

SAT 2^n

k -SAT $2^{n(1-O(1/k))}$

⋮

3-SAT 1.308^n

2-SAT $O(m)$

1-SAT $O(m)$

HARDNESS OF SAT

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

HARDNESS OF SAT

- SAT can be solved in time $2^n \text{ poly}(n)$
- 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

$O(2 - \epsilon)^n$ for any $\epsilon > 0$

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

EDIT DISTANCE

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

e l e p h a n t
r e l e v a n t
p

The diagram shows the words 'elephant' and 'relevant' aligned. An orange arrow points from the 'p' in 'elephant' to the 'v' in 'relevant'. An orange slash is over the 'r' in 'relevant'. An orange 'p' is below the 'v' in 'relevant'.

EDIT DISTANCE

Edit Distance

e l e p h a n t
↓
~~r~~ e l e v a n t
p

A T A G T A C T
↓
~~C~~ A T A C A C T
G

EDIT DISTANCE

Edit Distance

e l e p h a n t
↓
~~r~~ e l e v a n t
p

A T A G T A C T
↓
~~C~~ A T A C A C T
G

$n = \text{length of these strings}$
 $\tilde{O}(n^2)$

dynamic programming alg.

OTHER PROBLEMS

Longest Common Subsequence

$O(N^2)$
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 **SAT** problems

↙ 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 $s \cdot t = \sum_{i=1}^d s_i \cdot t_i = 0$?

S

| | | |
|--|------------------------------|-------|
| | $\overbrace{\hspace{2cm}}^d$ | |
| | 0 1 0 1 1 0 | ← |
| | 0 1 1 1 1 1 | |
| | 0 1 0 1 1 0 | ← |
| | 1 0 0 1 1 1 | ← S |

T

| | | |
|--|------------------------------|-------|
| | $\overbrace{\hspace{2cm}}^d$ | |
| | 0 1 1 1 1 | → |
| | 0 1 1 0 1 0 | → |
| | 1 1 0 1 0 0 | |
| | 0 1 0 0 0 0 | → t |

S and t are orthogonal!

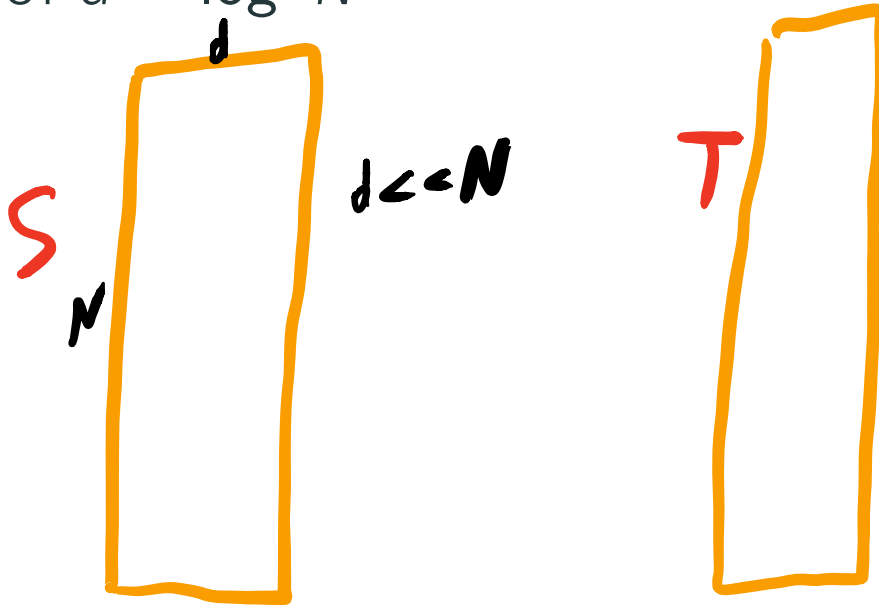
s and t are orthogonal iff.

$\forall i \in \{1, \dots, d\} \quad s_i = 0 \text{ OR } t_i = 0$

no coordinate i $s_i = t_i = 1$

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 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

Algorithm for SAT

Algorithm for OV

$N^{1.99}$

FINE-GRAINED REDUCTIONS

formula ϕ of SAT

Algorithm for SAT



Algorithm for OV

FINE-GRAINED REDUCTIONS

formula ϕ of SAT

Algorithm for SAT



sets of vectors S, T of OV

Algorithm for OV (S, T)

FINE-GRAINED REDUCTIONS

formula ϕ of SAT

Algorithm for SAT

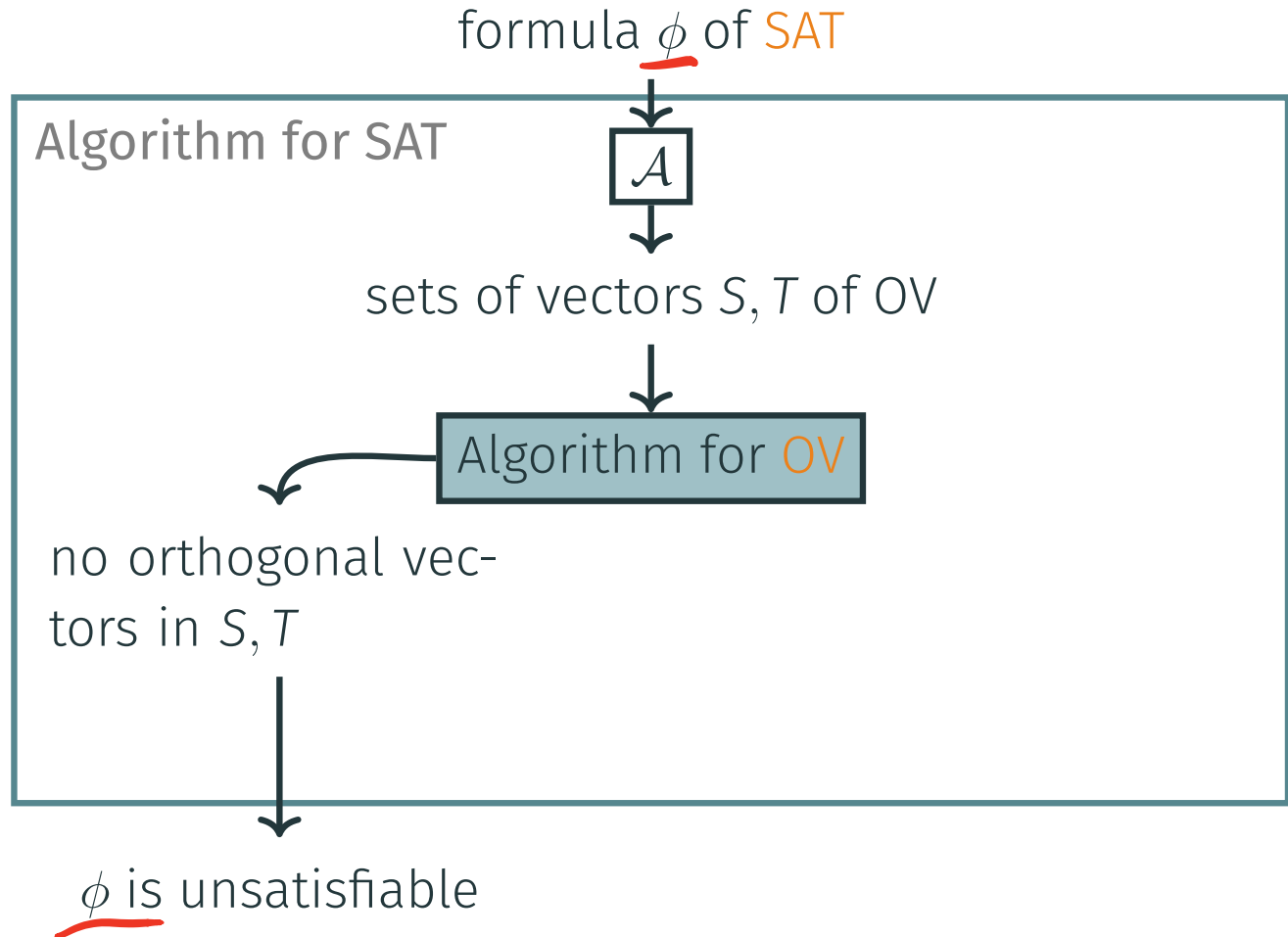


sets of vectors S, T of OV

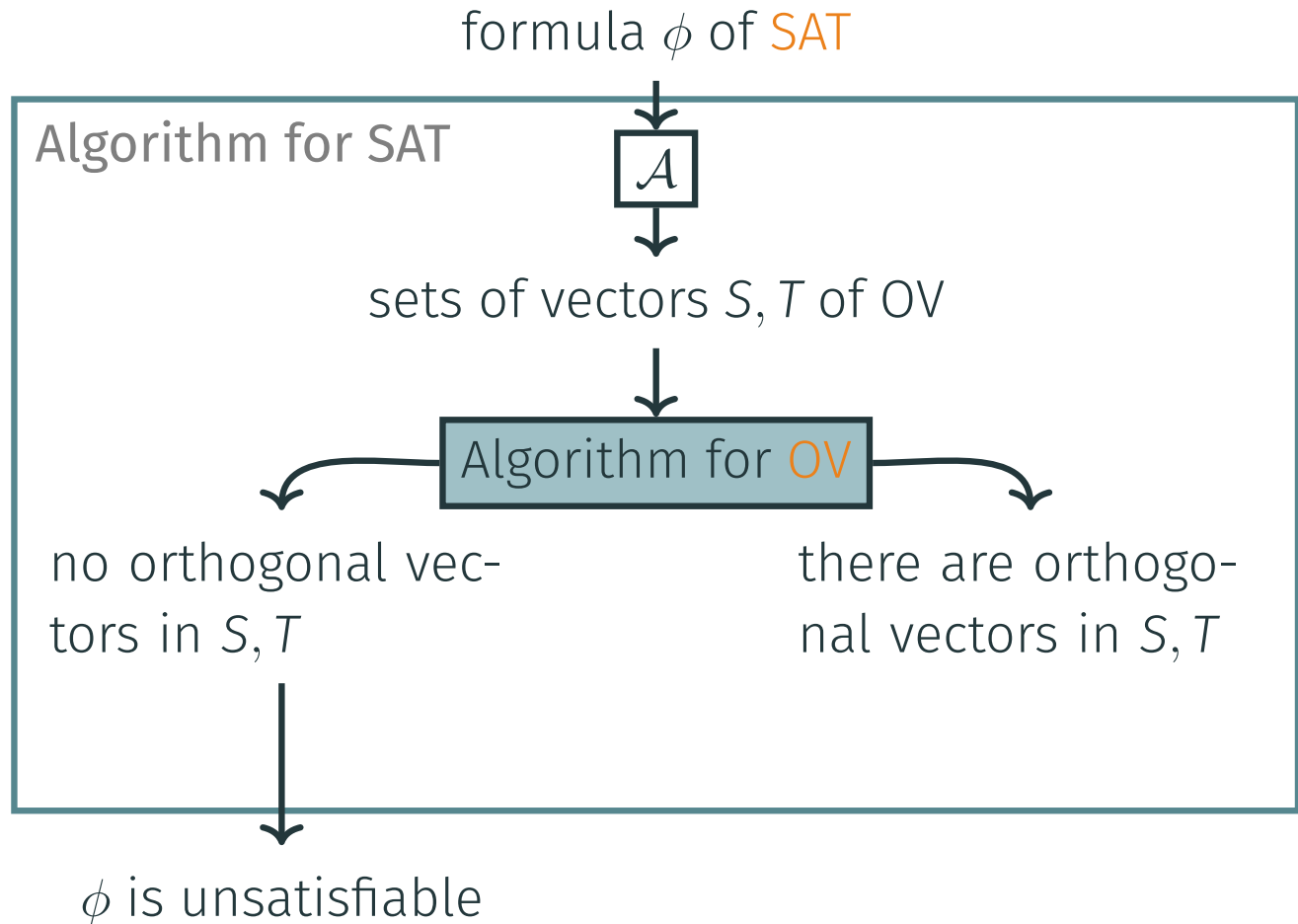
Algorithm for OV

no orthogonal vec-
tors in S, T

FINE-GRAINED REDUCTIONS



FINE-GRAINED REDUCTIONS



FINE-GRAINED REDUCTIONS

Assume OV in time $N^{1.99}$

formula ϕ of SAT n vars

Algorithm for SAT



sets of vectors S, T of OV

$$N = |S| = |T| = 2^{n/2}$$

Algorithm for OV(S, T)

$$N^{1.99}$$

no orthogonal vectors in S, T

there are orthogonal vectors in S, T

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

ϕ is unsatisfiable

ϕ is satisfiable

SETH \implies OV

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

$x_1 \ x_2 \ \dots \ x_{n/2} \ x_{n/2+1} \ \dots \ x_n$



$\{x_1 \ \dots \ x_{n/2}\}$

$2^{n/2}$ assignments
of these vars

$$|S| = 2^{n/2}$$

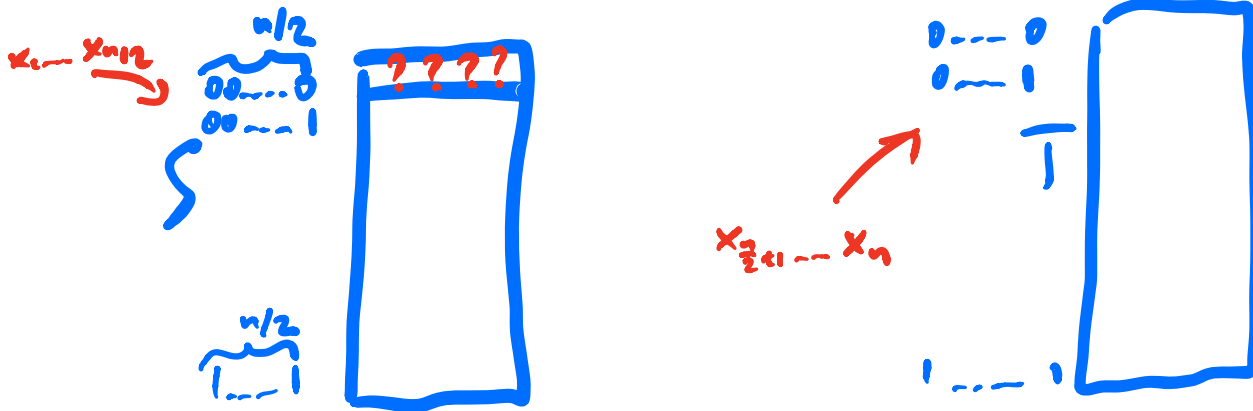
$\{x_{n/2+1} \ \dots \ x_n\}$

$2^{n/2}$ assignments
for these vars

$$|T| = 2^{n/2}$$

SETH \implies 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



$$(x_1 \vee x_n \vee x_2 \vee x_{n-1})$$

$$x_1 = x_2 = \dots = x_{\frac{n}{2}} = 0$$

$$x_{\frac{n}{2}+1} = \dots = x_n$$

$$x_n = 1 \text{ ---}$$

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 not satisfied
 iff it's not satisfied by
 $x_1, \dots, x_{\frac{n}{2}}$ and not satisfied
 by $x_{\frac{n}{2}+1}, \dots, x_n$

SETH \implies 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
- $N = 2^{n/2}$

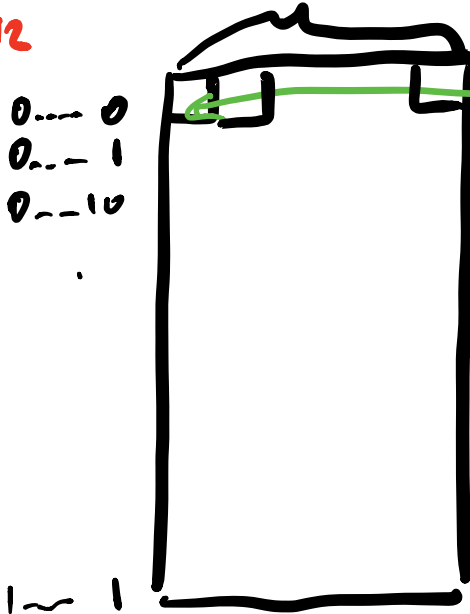
SETH \implies OV

- 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

$d = \# \text{clauses in the SAT formula } \phi$

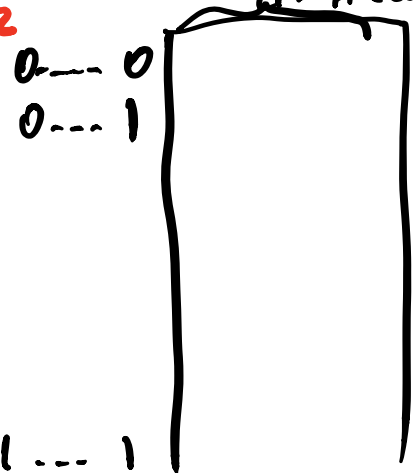
$|S| = 2^{n/2}$



it's 1 iff this assignment

00...0 does not 1st clause

$|T| = 2^{n/2}$



SETH \implies 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

- ϕ is SAT iff $\exists s \in S, t \in T$:
 $\underbrace{\hspace{10em}}$ **assignment to $x_1 \dots x_{n/2}$**
 $\underbrace{\hspace{10em}}$ **assignment to $x_{n/2+1} \dots x_n$**

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

$(x_1 \vee x_n \vee x_2 \vee x_{n-1})$ is sat iff $\underbrace{\hspace{10em}}$ **it's satisfiable**
by $x_1 \dots x_{n/2}$ OR $x_{n/2+1} \dots x_n$

$\underbrace{\hspace{10em}}$ **it's not sat** if both vectors have 1

SETH \implies 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 $\exists s \in S, t \in T$:

$$\forall i \in \{1, \dots, m\}: x_i \cdot y_i = 0 \quad N = 2^{n/2}$$

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

$$N^{1.99}$$

SETH \implies **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

- ϕ is SAT iff $\exists s \in S, t \in T$: SETH \implies **$P \neq NP$**

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

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

\implies refute **SETH** \implies you came up with a new algorithmic idea!!!

The Dominating Set Problem

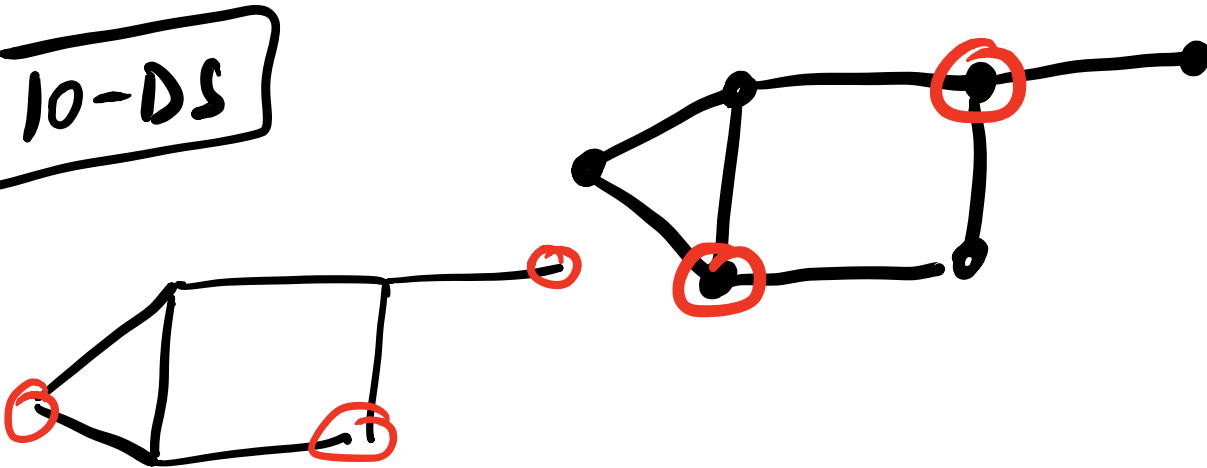
DOMINATING SET

For every k ,

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

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

10-DS



DOMINATING SET

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

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

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

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

n^{k+1} time
↓ improved
 n^k

n for i_1 in V
 n for i_2 in V
 \vdots
 n for i_k in V
 n Check (i_1, i_2, \dots, i_k) is a DS?

DOMINATING SET

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

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

- 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 \implies DS

SAT with n vars x_1, \dots, x_n $k=10$

Partition vars in k groups:

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

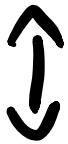
Given SAT $\phi \implies$ Graph G s.t.

ϕ is satisfiable iff G has DS of size k

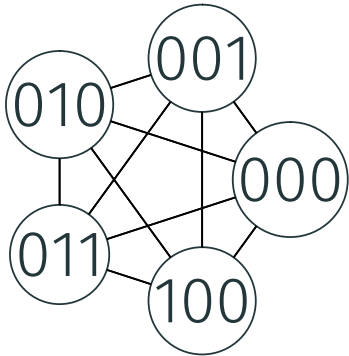
SETH \implies DS

Partition vars in k groups:

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



$2^{n/k}$ vertices

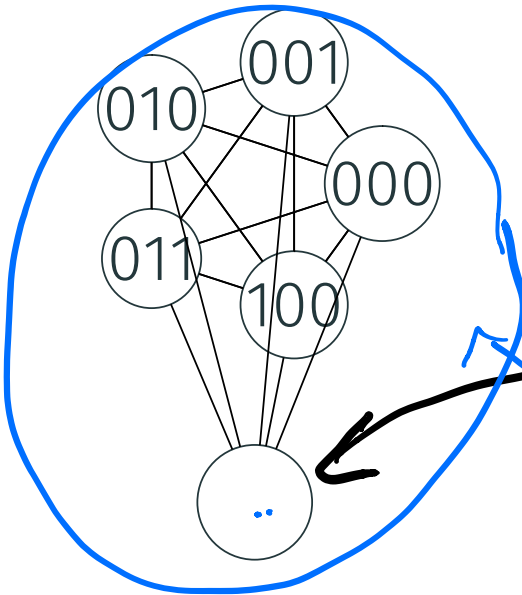


SETH \implies DS

Partition vars in k groups:

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

$2^{n/k}$ vertices



If you want to dominate,
you have to take ≥ 1
vertex from \implies assignments to the
first n/k vars

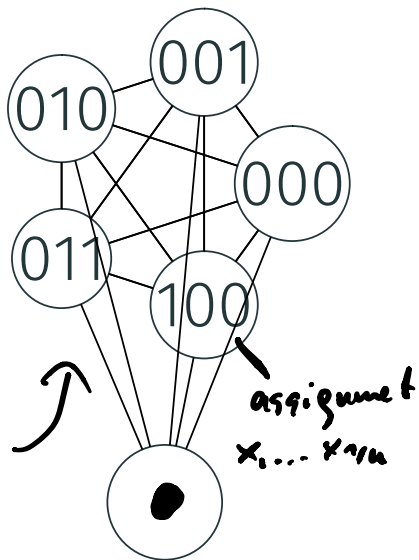
SETH \implies DS

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

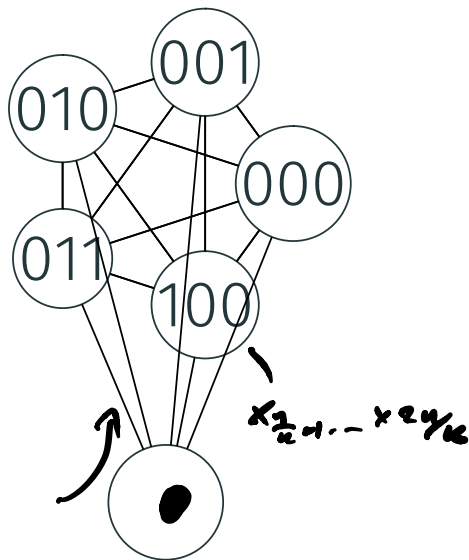
Partition vars in k groups:

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

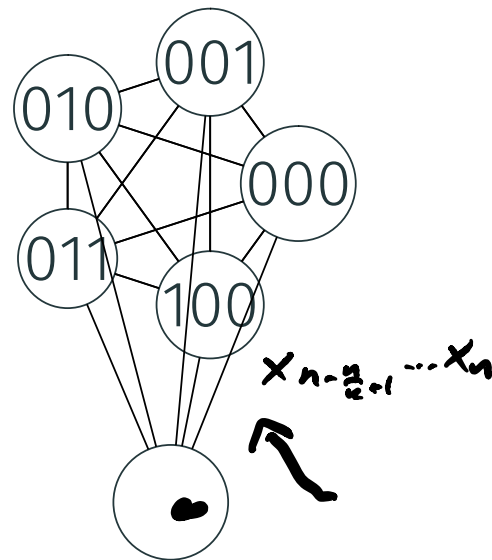
$2^{n/k}$ vertices



$2^{n/k}$ vertices



$2^{n/k}$ vertices



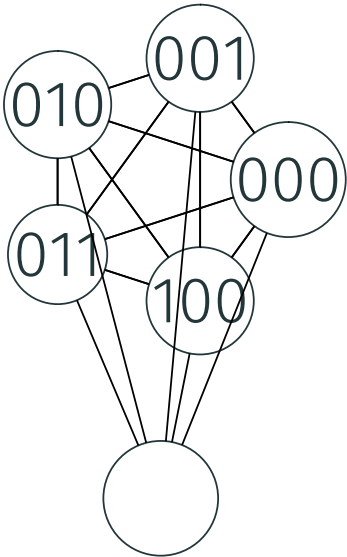
SETH \implies DS

Partition vars in k groups:

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

m vertices:

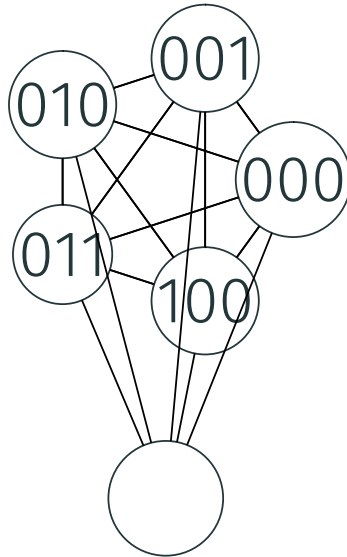
$2^{n/k}$ vertices



cl_1

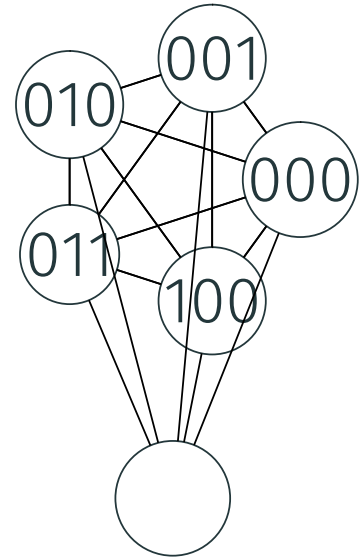
cl_2

$2^{n/k}$ vertices



cl_3

$2^{n/k}$ vertices

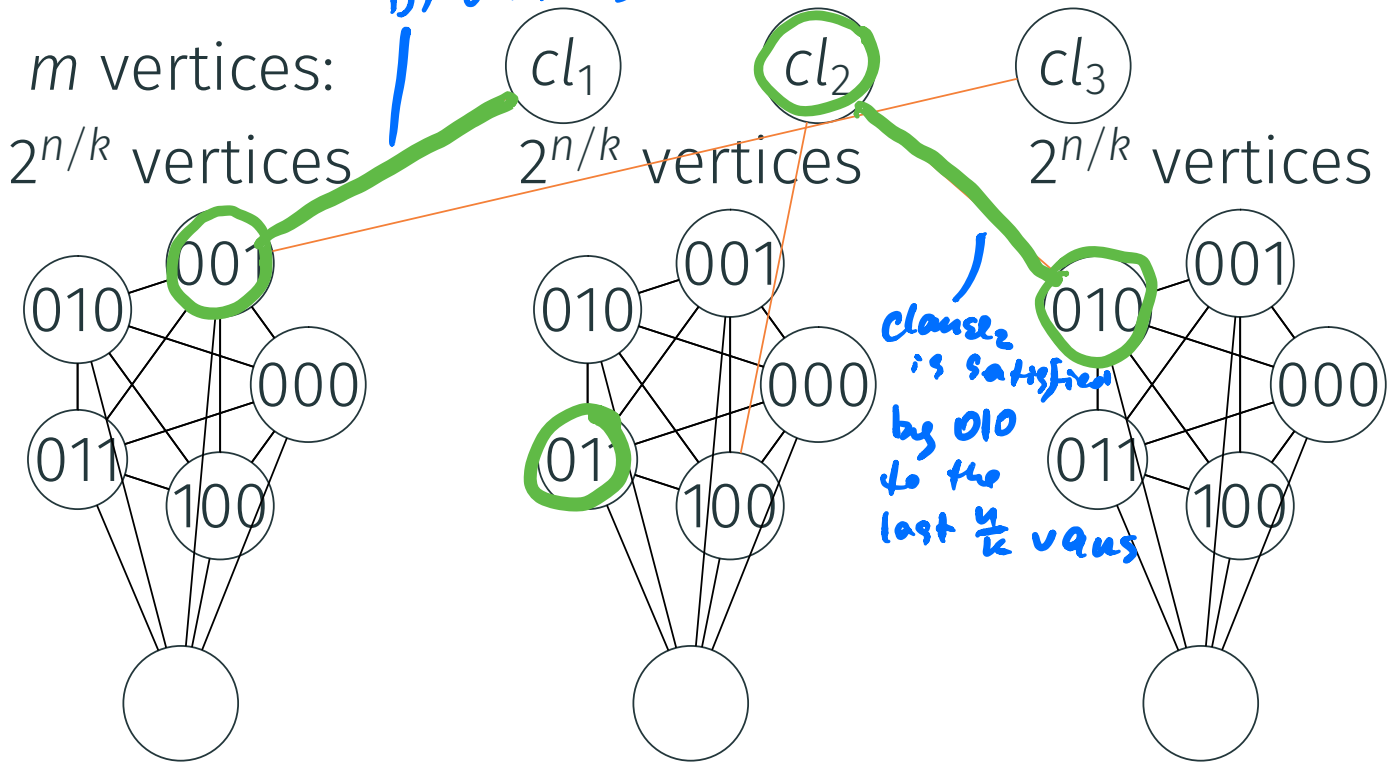


SETH \implies DS $|DS|=k$ must leave
 1 vertex from each part.

Partition vars in k groups:

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

iff 001 satisfies Clause 1



SETH \implies DS

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

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

vertices

SETH \implies 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

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

\implies refute SETH \implies congrats!! you found a new algorithmic idea!