Online Orthogonal Vectors Revisited

Karthik Gajulapalli
Alexander Golovnev
Samuel King
Sidhant Saraogi

Columbia Theory Lunch, 2026

Alexander Golovnev

Samuel King

Sidhant Saraogi

Outline

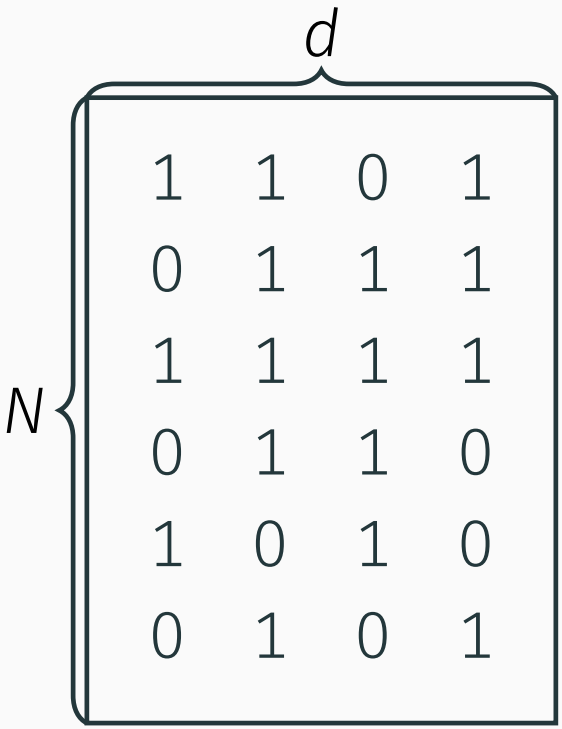
- Online Orthogonal Vectors
- Algorithms
- Lower Bounds

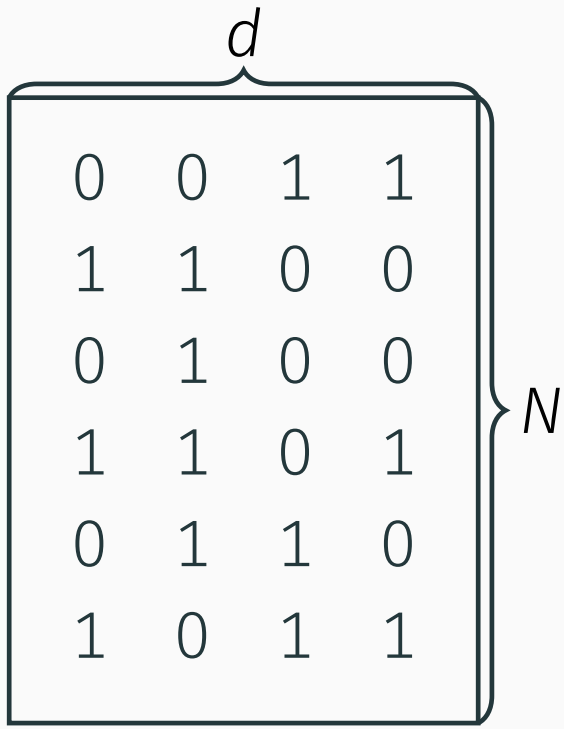
Outline

- Online Orthogonal Vectors
- Algorithms
- Lower Bounds

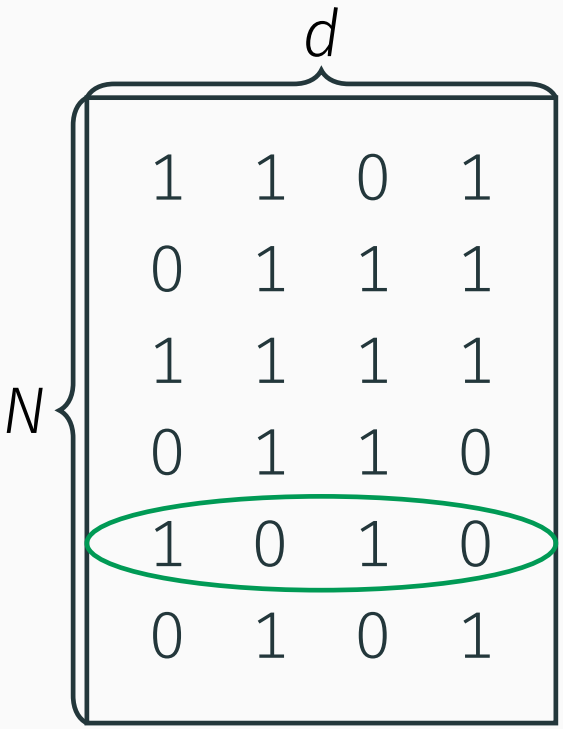
(Offline) Orthogonal Vectors

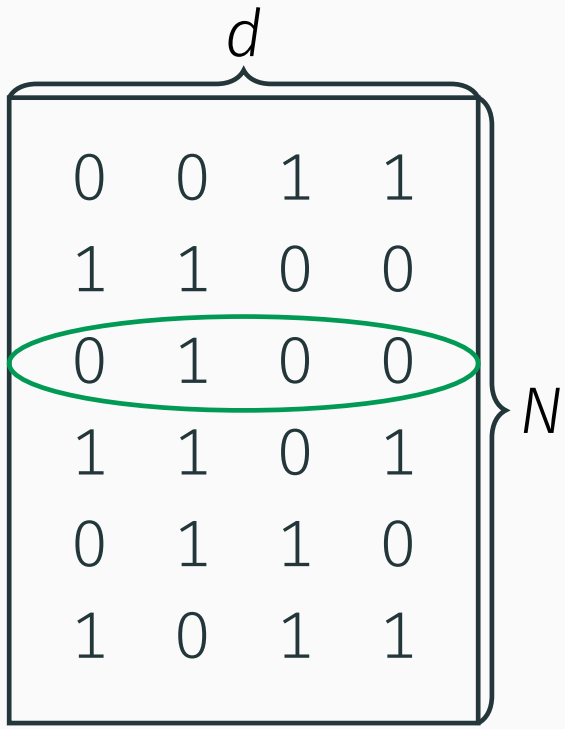
Orthogonal Vectors Problem





Orthogonal Vectors Problem





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 $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- d is small compared to N, say, $d = O(\log N)$

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 $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- d is small compared to N, say, $d = O(\log N)$
- Can solve in time $N^2 \cdot d$

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 $s \cdot t = \sum_{i=1}^{d} s_i \cdot t_i = 0$?
- d is small compared to N, say, $d = O(\log N)$
- Can solve in time $N^2 \cdot d$

Is there a $N^{1.9999999}$ algorithm for OV?

Unconditional Lower bounds

Unconditional Lower bounds

- Complexity Theory Dream: Prove Unconditional Lower bounds

Unconditional Lower bounds

- Complexity Theory Dream: Prove Unconditional Lower bounds
- We don’t really have a good toolkit yet to argue about lower bounds.

Boolean Satisfiability (SAT)

SAT

- $\varphi(x_1, \ldots, x_n) = (x_1 \vee \overline{x_2} \vee \ldots \vee x_7) \wedge$
$\ldots \wedge$
$(x_2 \vee \overline{x_3} \vee \ldots \vee x_8)$

SAT

- $\varphi(x_1, \ldots, x_n) = (x_1 \vee \overline{x_2} \vee \ldots \vee x_7) \wedge \ldots \wedge (x_2 \vee \overline{x_3} \vee \ldots \vee x_8)$
- φ is satisfiable if $\exists x \in \{0,1\}^n : \varphi(x) = 1$. Otherwise, φ is unsatisfiable

SAT

- $\varphi(x_1, \ldots, x_n) = (x_1 \vee \overline{x_2} \vee \ldots \vee x_7) \wedge \ldots \wedge (x_2 \vee \overline{x_3} \vee \ldots \vee x_8)$
- φ is satisfiable if $\exists x \in \{0,1\}^n : \varphi(x) = 1$. Otherwise, φ is unsatisfiable
- n Boolean vars, m clauses

SAT

- $\varphi(x_1, \ldots, x_n) = (x_1 \vee \overline{x_2} \vee \ldots \vee x_7) \wedge \ldots \wedge (x_2 \vee \overline{x_3} \vee \ldots \vee x_8)$
- φ is satisfiable if $\exists x \in \{0,1\}^n : \varphi(x) = 1$. Otherwise, φ is unsatisfiable
- n Boolean vars, m clauses
- $m = O(n)$

SAT

- $\varphi(x_1, \ldots, x_n) = (x_1 \vee \overline{x_2} \vee \ldots \vee x_7) \wedge \ldots \wedge (x_2 \vee \overline{x_3} \vee \ldots \vee x_8)$
- φ is satisfiable if $\exists x \in \{0,1\}^n : \varphi(x) = 1$. Otherwise, φ is unsatisfiable
- n Boolean vars, m clauses
- $m = O(n)$
- k-SAT is SAT where clause length $\leq k$

SAT from the perspective of lower bounds

- SAT is NP-hard

SAT from the perspective of lower bounds

- SAT is NP-hard
- If we believe $P \neq NP$, then we can view NP-hardness as a conditional lower bound

SAT from the perspective of lower bounds

- SAT is NP-hard
- If we believe $P \neq NP$, then we can view NP-hardness as a conditional lower bound

Assumption: $P \neq NP$

There is no polynomial time algorithm for SAT.

Corollary:

Under $P \neq NP$, there is not polynomial time algorithm for YFP.

SAT from the perspective of lower bounds

- SAT is NP-hard
- If we believe $P \neq NP$, then we can view NP-hardness as a conditional lower bound

Assumption: $P \neq NP$

There is no polynomial time algorithm for SAT.

Corollary:

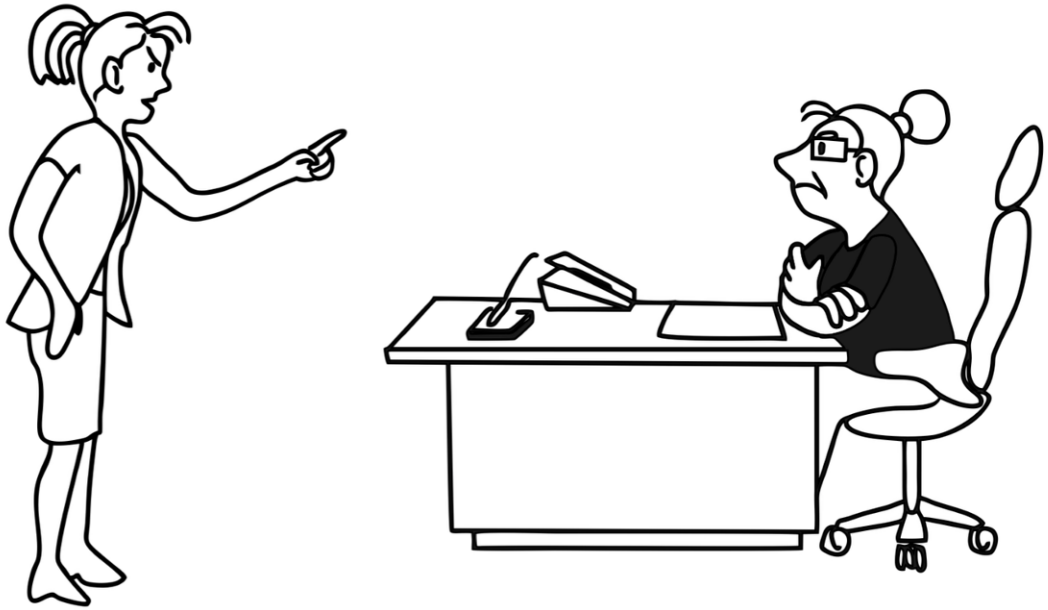
Under $P \neq NP$, there is not polynomial time algorithm for YFP.

- YFP $=$ { Hamiltonian Path, Traveling Salesman, Independent Set, Clique, 3-Coloring, . . .}

Proof by Human Effort

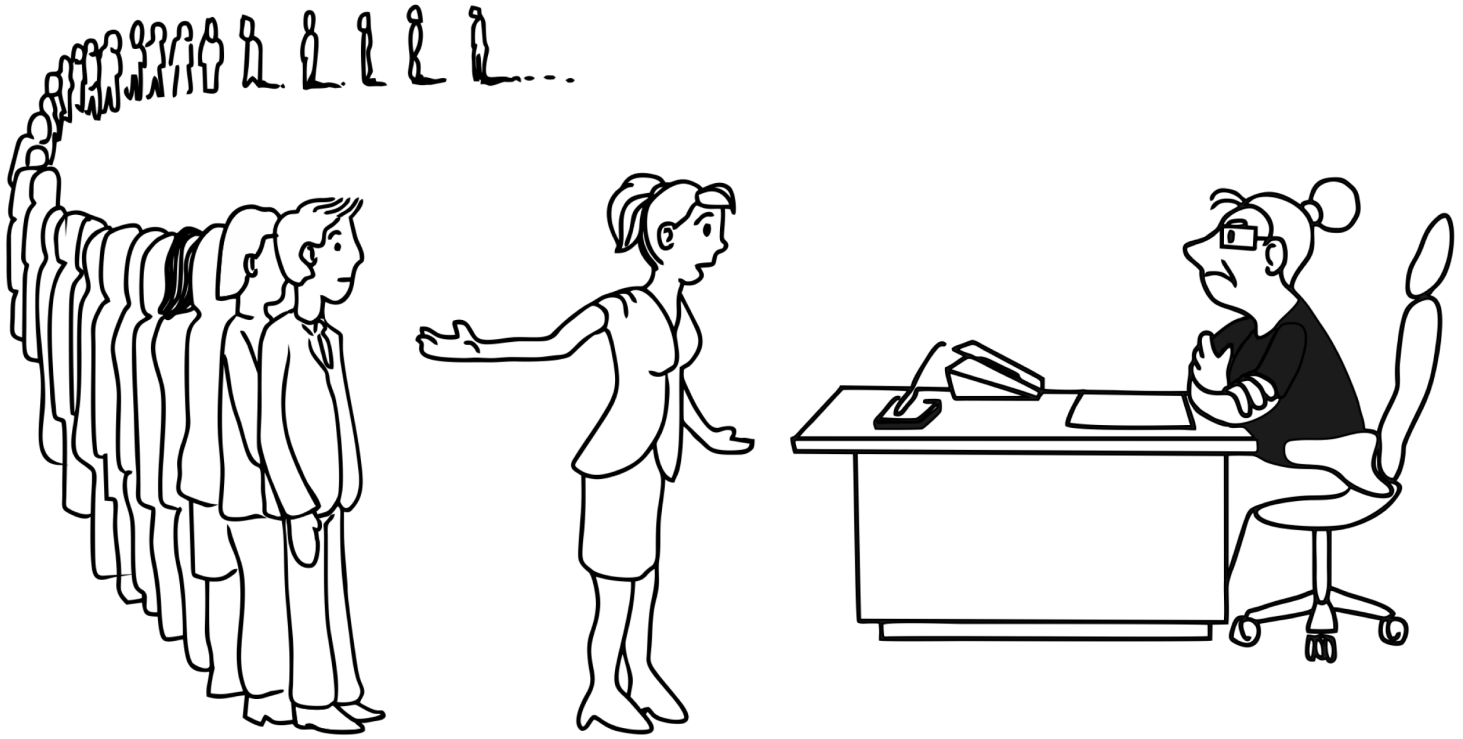
“I can’t find an efficient algorithm, I guess I’m just too dumb.”

Proof by Human Effort



“I can’t find an efficient algorithm, because no such algorithm is possible!”

Proof by Human Effort



“I can’t find an efficient algorithm, but neither can all these famous people.”

More conditional lower bounds?

More conditional lower bounds?

- SAT can be solved in time $O(2^n \cdot \mathsf{poly}(|\varphi|))$

More conditional lower bounds?

- SAT can be solved in time $O(2^n \cdot \mathbf{poly}(|\varphi|))$
- We don't know how to solve SAT exponentially faster: in time $2^{0.9999n}$ even when $m = O(n)$

More conditional lower bounds?

- SAT can be solved in time $O(2^n \cdot \mathbf{poly}(|\varphi|))$
- We don't know how to solve SAT exponentially faster: in time $2^{0.9999n}$ even when $m = O(n)$
- Can we prove conditional lower bounds on polynomial time problems?

Strong Exponential Time Hypotheses

Strong Exponential Time Hypothesis (SETH)

For every $\varepsilon > 0$, there exists k such that no algorithm solves k-SAT on formulas with n variables in time $2^{(1-\varepsilon)n}$

Strong Exponential Time Hypotheses

Strong Exponential Time Hypothesis (SETH)

For every $\varepsilon > 0$, there exists k such that no algorithm solves k-SAT on formulas with n variables in time $2^{(1-\varepsilon)n}$

- Without loss of generality, assume $m = cn$ for a large constant c [IPZ01]

Strong Exponential Time Hypotheses

Strong Exponential Time Hypothesis (SETH)

For every $\varepsilon > 0$, there exists k such that no algorithm solves k-SAT on formulas with n variables in time $2^{(1-\varepsilon)n}$

- Without loss of generality, assume $m = cn$ for a large constant c [IPZ01]
- SETH as a limit of current algorithmic techniques

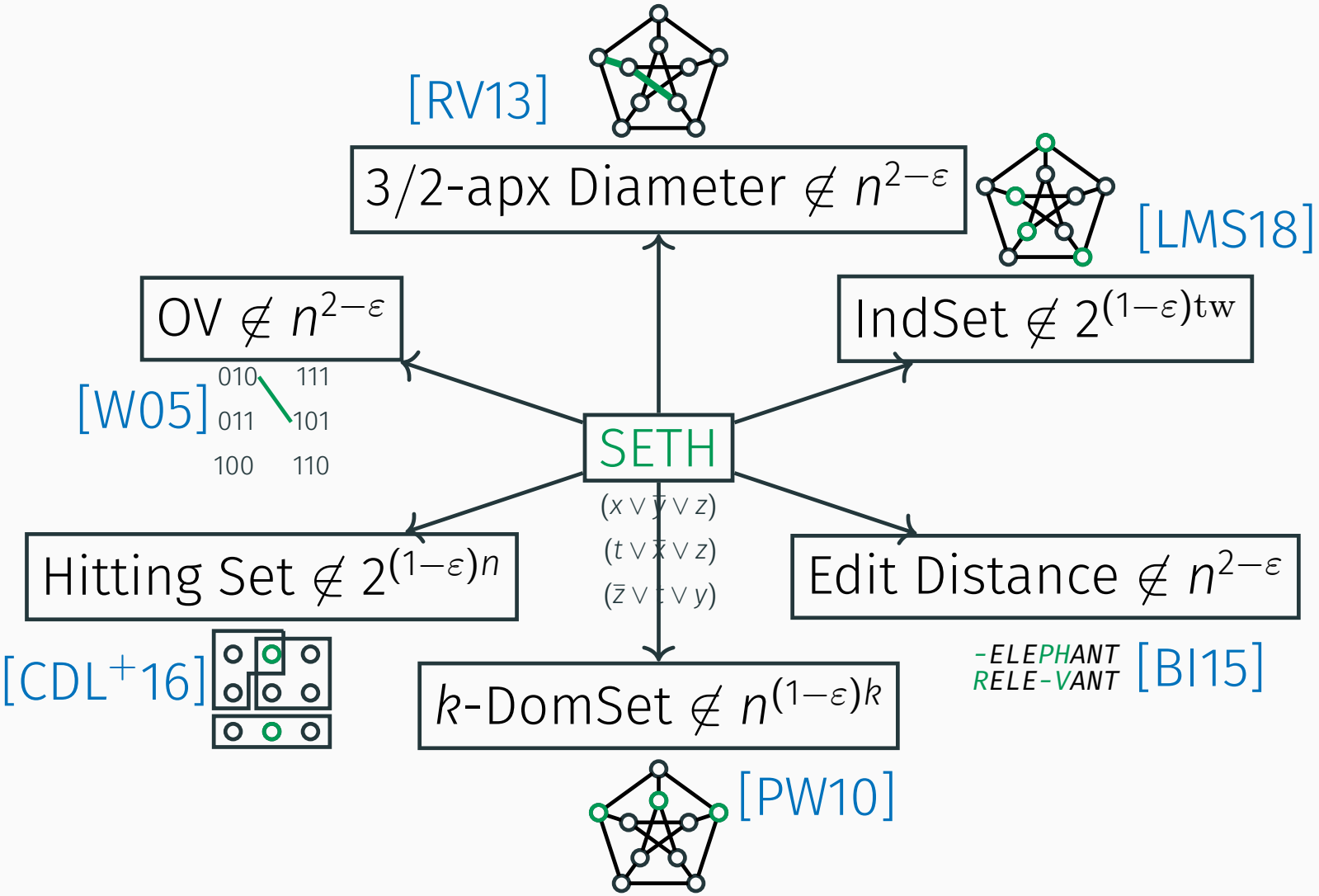
Strong Exponential Time Hypotheses

Strong Exponential Time Hypothesis (SETH)

For every $\varepsilon > 0$, there exists k such that no algorithm solves k-SAT on formulas with n variables in time $2^{(1-\varepsilon)n}$

- Without loss of generality, assume $m = cn$ for a large constant c [IPZ01]
- SETH as a limit of current algorithmic techniques
- Refuting SETH implies super-linear circuit lower bounds

Fine-Grained Reductions



SETH $\Longrightarrow$ OV

- Under SETH, OV cannot be solved in time $N^{2-\varepsilon}$ [W05]

SETH $\Longrightarrow$ OV

- Under SETH, OV cannot be solved in time $N^{2-\varepsilon}$ [W05]
- Given a CNF φ, split its n input variables into two sets of size $n/2$

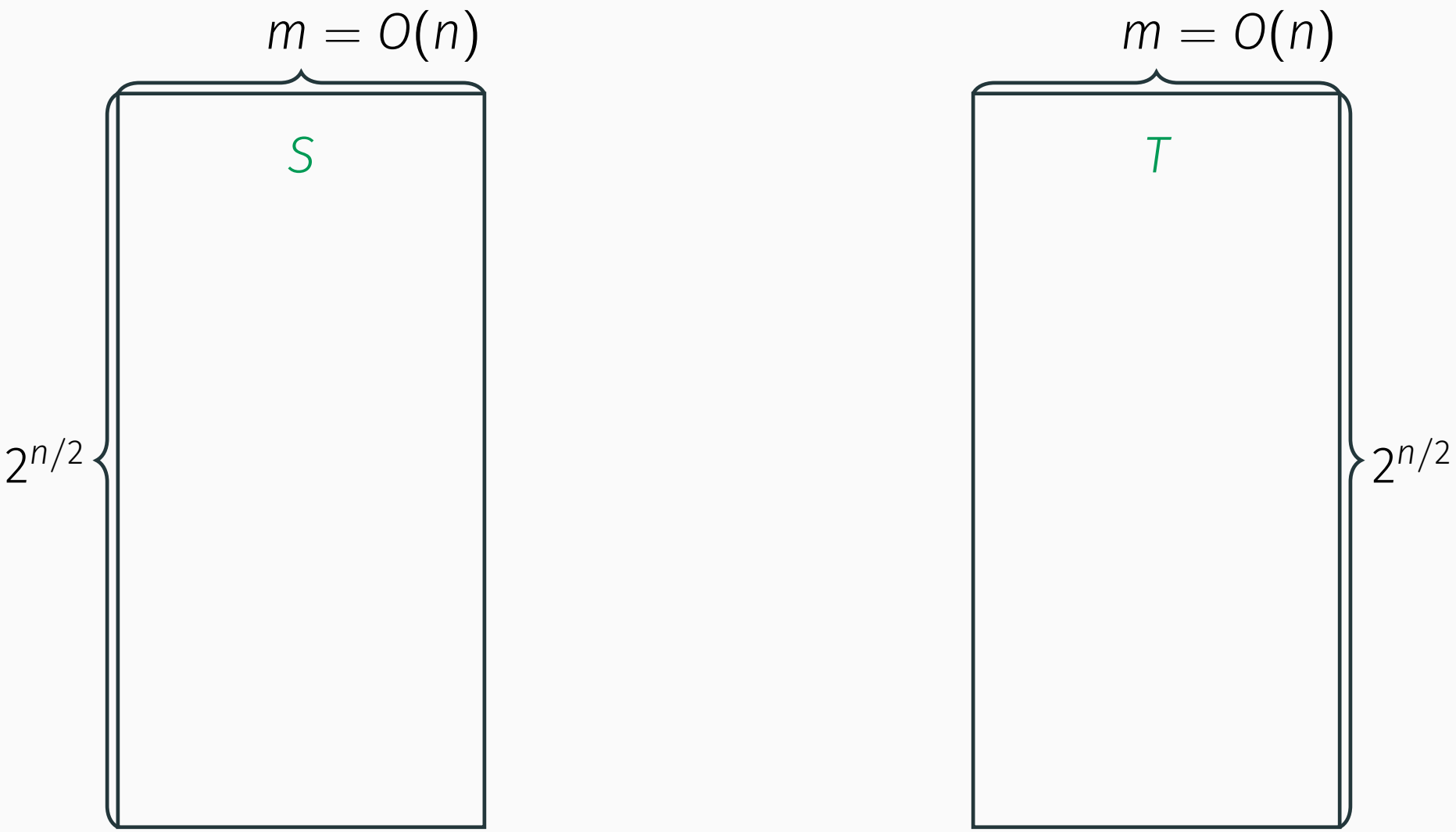
SETH $\Longrightarrow$ OV

- Under SETH, OV cannot be solved in time $N^{2-\varepsilon}$ [W05]
- Given a CNF 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

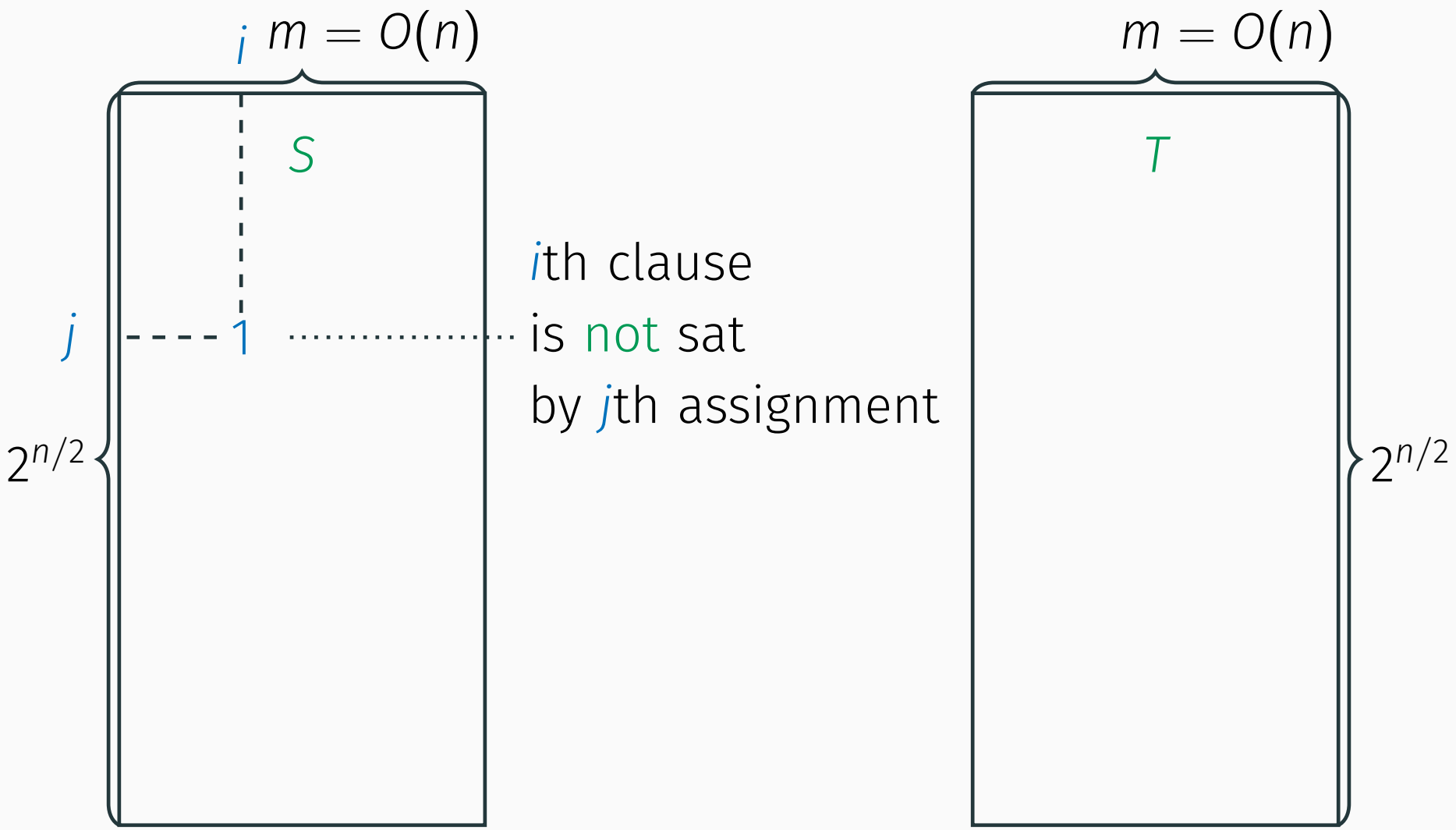
SETH $\Longrightarrow$ OV

- Under SETH, OV cannot be solved in time $N^{2-\varepsilon}$ [W05]
- Given a CNF 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



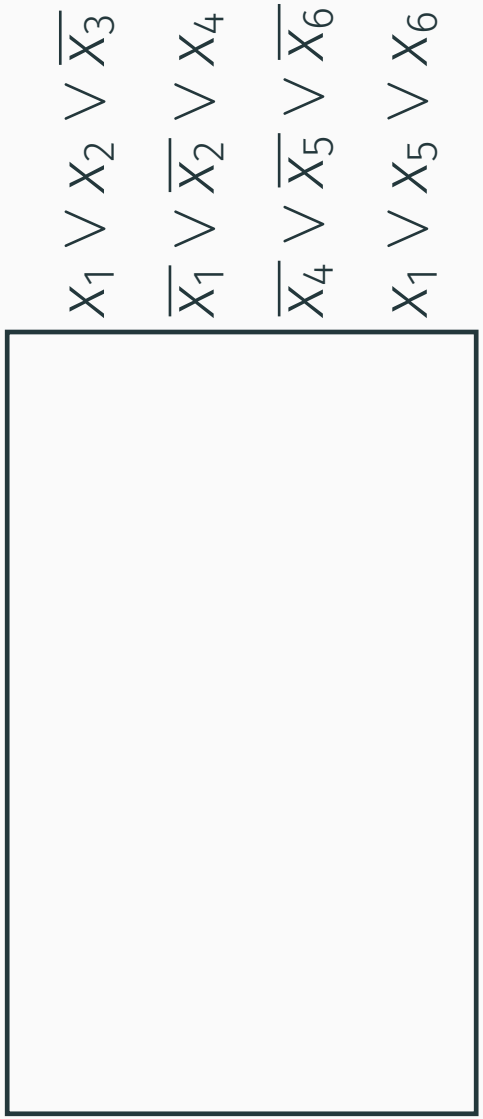
SETH $\Longrightarrow$ OV

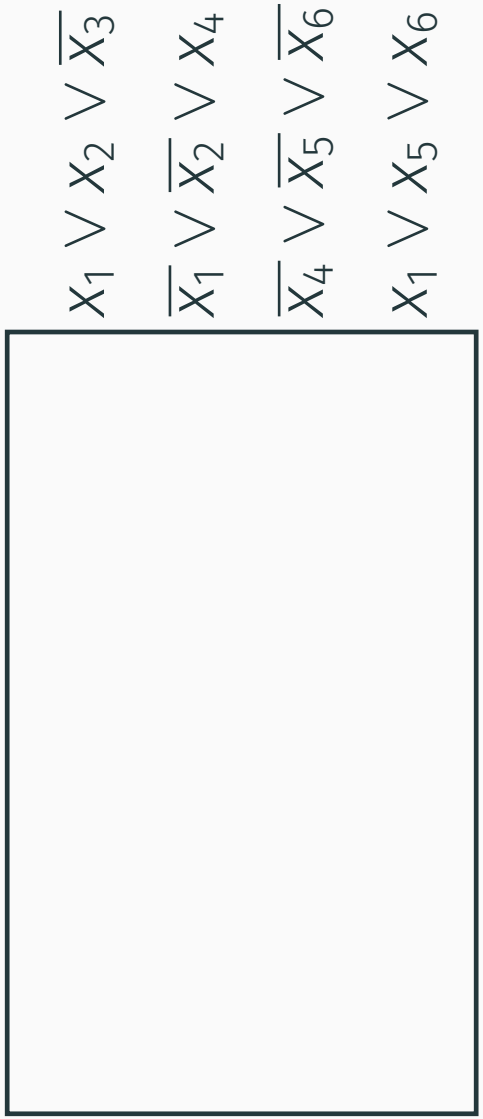


SETH $\Longrightarrow$ OV. Example

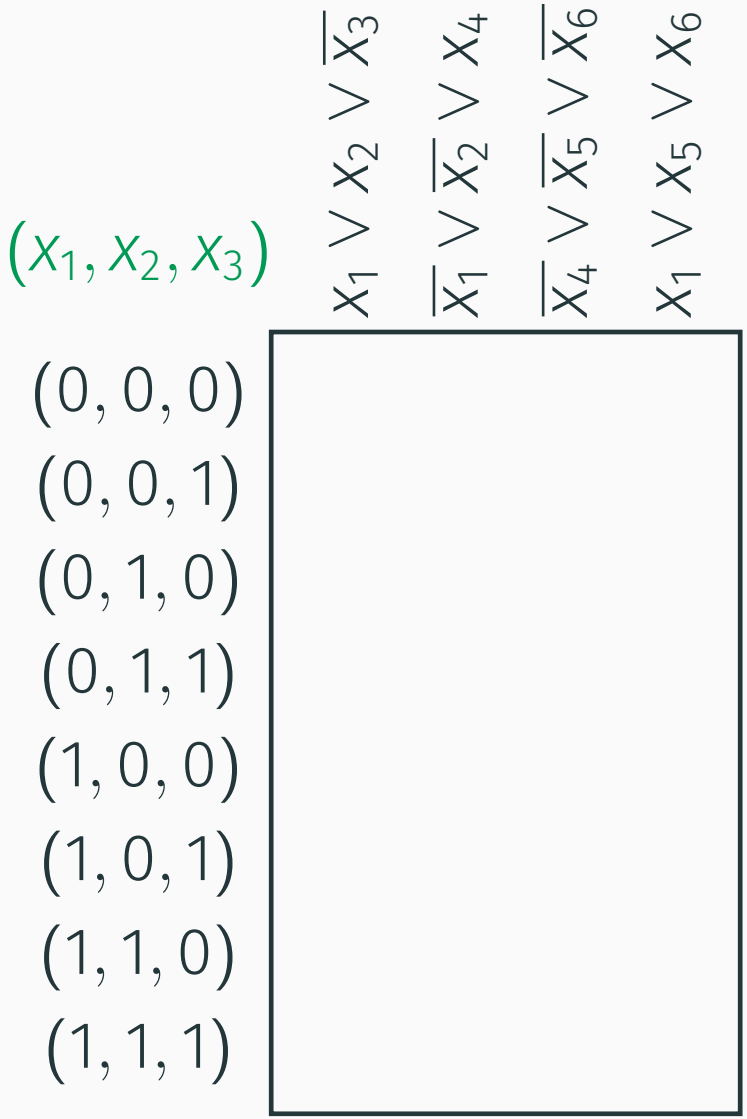
$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}) \wedge (x_1 \vee x_5 \vee x_6)$$

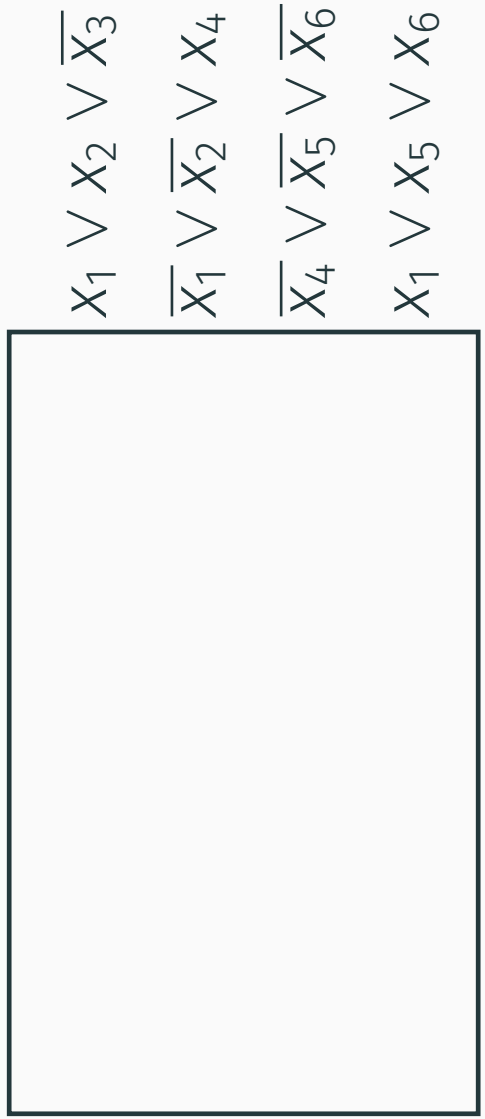
SETH $\Longrightarrow$ OV. Example



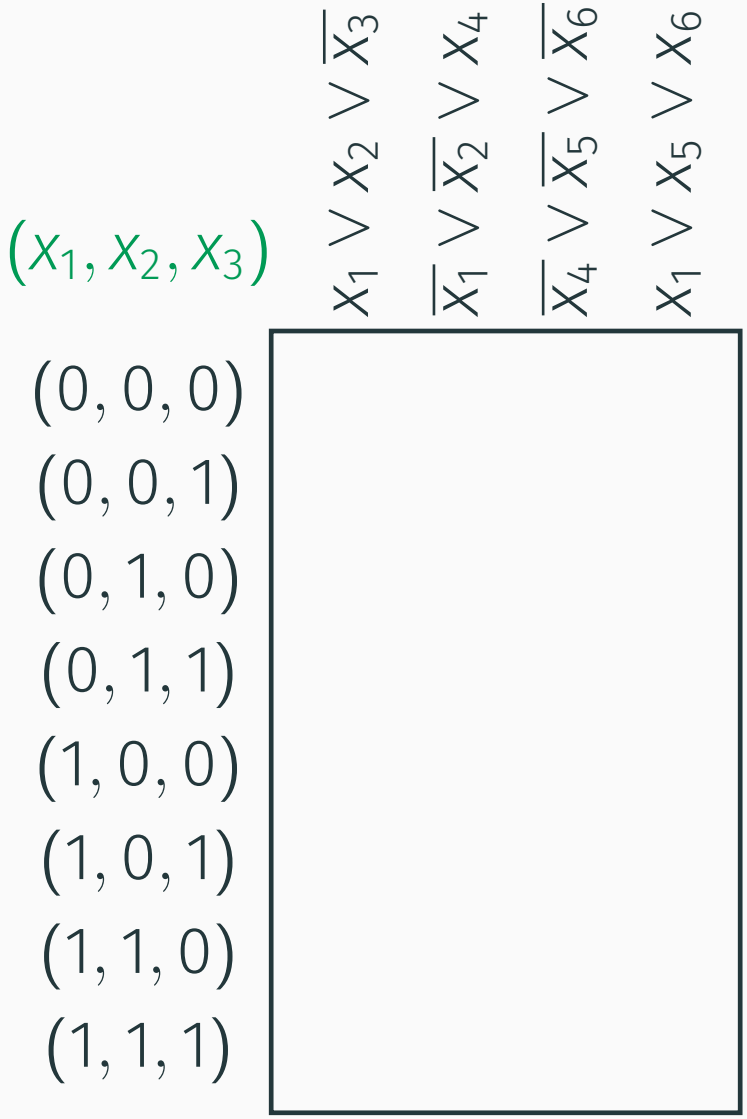


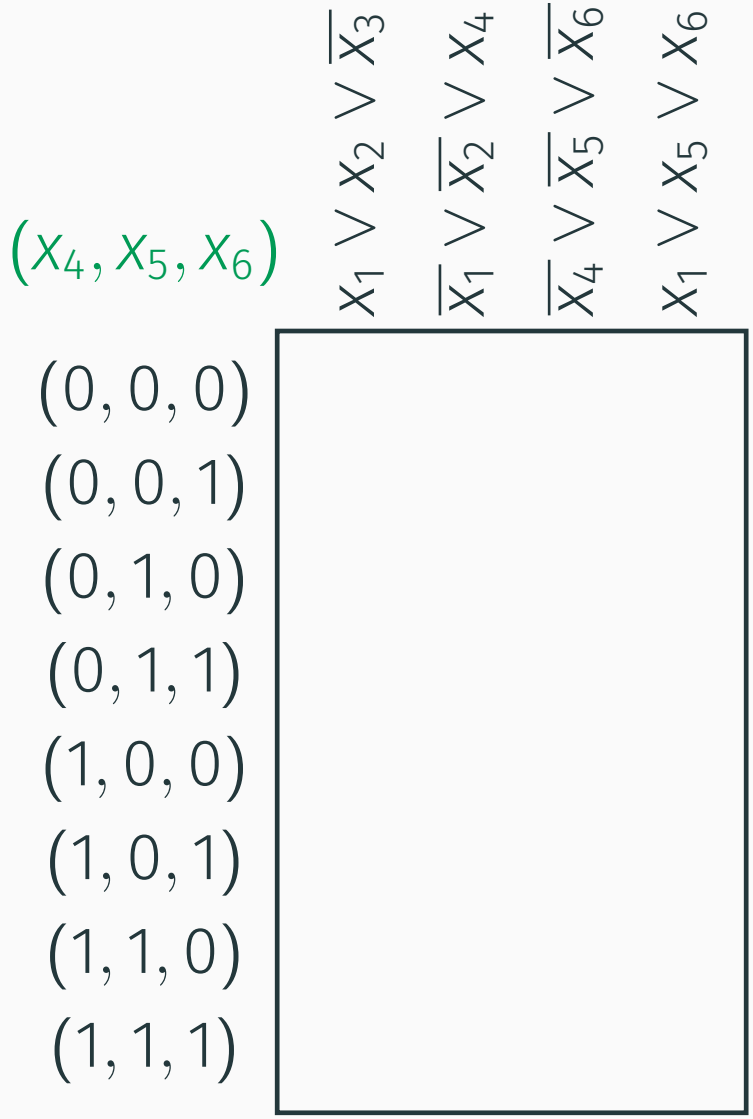
SETH $\Longrightarrow$ OV. Example



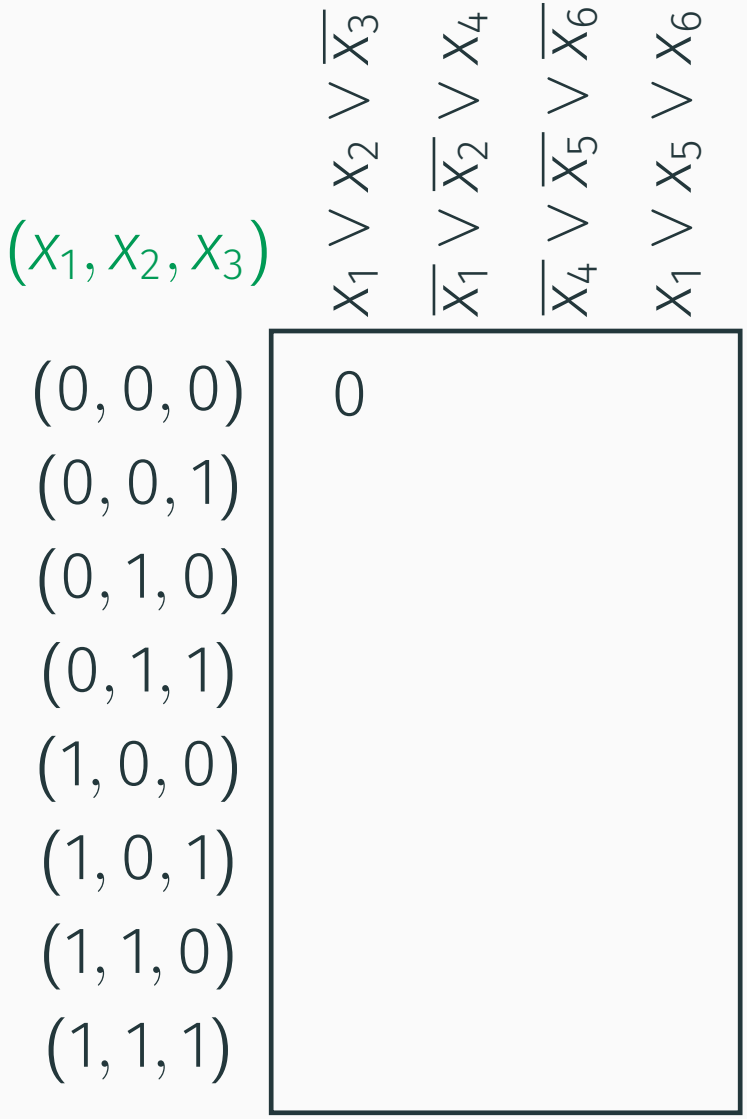


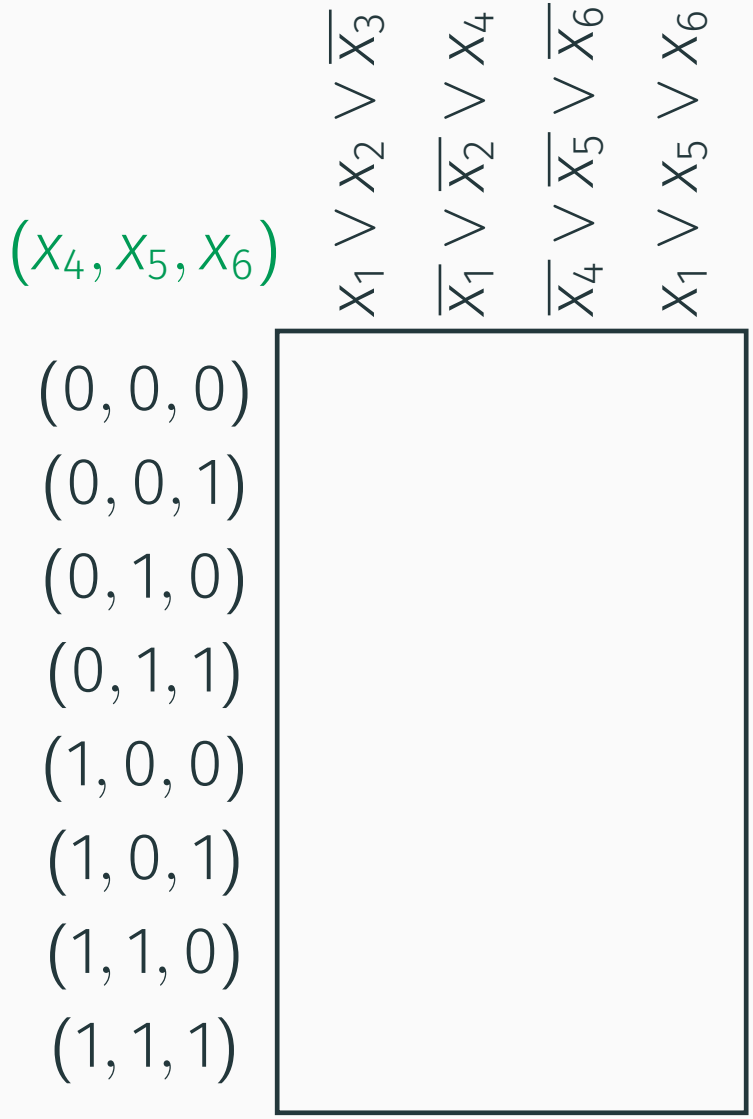
SETH $\Longrightarrow$ OV. Example





SETH $\Longrightarrow$ OV. Example





SETH $\Longrightarrow$ OV. Example

(x_1, x_2, x_3)	$x_1 \vee x_2 \vee \overline{x_3}$	$\overline{x_1} \vee \overline{x_2} \vee x_4$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}$	$x_1 \vee x_5 \vee x_6$
$(0,0,0)$	0			
$(0,0,1)$	1			
$(0,1,0)$				
$(0,1,1)$				
$(1,0,0)$				
$(1,0,1)$				
$(1,1,0)$				
$(1,1,1)$				

(x_4, x_5, x_6)	$x_1 \vee x_2 \vee \overline{x_3}$	$\overline{x_1} \vee \overline{x_2} \vee x_4$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}$	$x_1 \vee x_5 \vee x_6$
$(0,0,0)$				
$(0,0,1)$				
$(0,1,0)$				
$(0,1,1)$				
$(1,0,0)$				
$(1,0,1)$				
$(1,1,0)$				
$(1,1,1)$				

SETH $\Longrightarrow$ OV. Example

(x_1, x_2, x_3)	$x_1 \vee x_2 \vee \overline{x_3}$	$\overline{x_1} \vee \overline{x_2} \vee x_4$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}$	$x_1 \vee x_5 \vee x_6$
$(0,0,0)$	0	0	1	1
$(0,0,1)$	1	0	1	1
$(0,1,0)$	0	0	1	1
$(0,1,1)$	0	0	1	1
$(1,0,0)$	0	0	1	0
$(1,0,1)$	0	0	1	0
$(1,1,0)$	0	1	1	0
$(1,1,1)$	0	1	1	0

(x_4, x_5, x_6)	$x_1 \vee x_2 \vee \overline{x_3}$	$\overline{x_1} \vee \overline{x_2} \vee x_4$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}$	$x_1 \vee x_5 \vee x_6$
$(0,0,0)$	1	1	0	1
$(0,0,1)$	1	1	0	0
$(0,1,0)$	1	1	0	0
$(0,1,1)$	1	1	0	0
$(1,0,0)$	1	0	0	1
$(1,0,1)$	1	0	0	0
$(1,1,0)$	1	0	0	0
$(1,1,1)$	1	0	1	0

SETH $\Longrightarrow$ OV. Example

(x_1, x_2, x_3)	$x_1 \vee x_2 \vee \overline{x_3}$	$\overline{x_1} \vee \overline{x_2} \vee x_4$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}$	$x_1 \vee x_5 \vee x_6$
$(0,0,0)$	0	0	1	1
$(0,0,1)$	1	0	1	1
$(0,1,0)$	0	0	1	1
$(0,1,1)$	0	0	1	1
$(1,0,0)$	0	0	1	0
$(1,0,1)$	0	0	1	0
$(1,1,0)$	0	1	1	0
$(1,1,1)$	0	1	1	0

(x_4, x_5, x_6)	$x_1 \vee x_2 \vee \overline{x_3}$	$\overline{x_1} \vee \overline{x_2} \vee x_4$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_6}$	$x_1 \vee x_5 \vee x_6$
$(0,0,0)$	1	1	0	1
$(0,0,1)$	1	1	0	0
$(0,1,0)$	1	1	0	0
$(0,1,1)$	1	1	0	0
$(1,0,0)$	1	0	0	1
$(1,0,1)$	1	0	0	0
$(1,1,0)$	1	0	0	0
$(1,1,1)$	1	0	1	0

SETH $\implies$ OV

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

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

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

$$\forall i \in [m]: \quad x_i \cdot y_i = 0$$

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

$$\forall i \in [m]: \quad x_i \cdot y_i = 0$$

- An $N^{2-\varepsilon}$ algorithm for OV gives an algorithm for SAT with run time

$$N + N^{2-\varepsilon}$$

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

$$\forall i \in [m]: \quad x_i \cdot y_i = 0$$

- An $N^{2-\varepsilon}$ algorithm for OV gives an algorithm for SAT with run time

$$N + N^{2-\varepsilon} = (2^{n/2})^{2-\varepsilon} = 2^{n(1-\varepsilon/2)}$$

OV Conjecture

Orthogonal Vectors Conjecture

For every $\varepsilon > 0$, there exists $c \geq 1$ such that no algorithm solves Orthogonal Vectors on dimension $d = c \log n$ in time $n^{2-\varepsilon}$

OV Conjecture

Orthogonal Vectors Conjecture

For every $\varepsilon > 0$, there exists $c \geq 1$ such that no algorithm solves Orthogonal Vectors on dimension $d = c \log n$ in time $n^{2-\varepsilon}$

- P vs NP is too hard

OV Conjecture

Orthogonal Vectors Conjecture

For every $\varepsilon > 0$, there exists $c \geq 1$ such that no algorithm solves Orthogonal Vectors on dimension $d = c \log n$ in time $n^{2-\varepsilon}$

- P vs NP is too hard
- Is SETH False? Beating brute force for SAT doesn't sound that crazy?

OV Conjecture

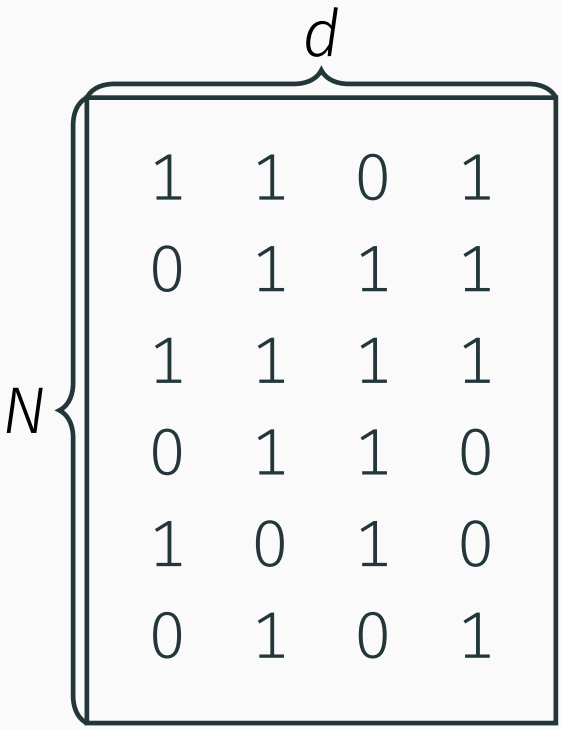
Orthogonal Vectors Conjecture

For every $\varepsilon > 0$, there exists $c \geq 1$ such that no algorithm solves Orthogonal Vectors on dimension $d = c \log n$ in time $n^{2-\varepsilon}$

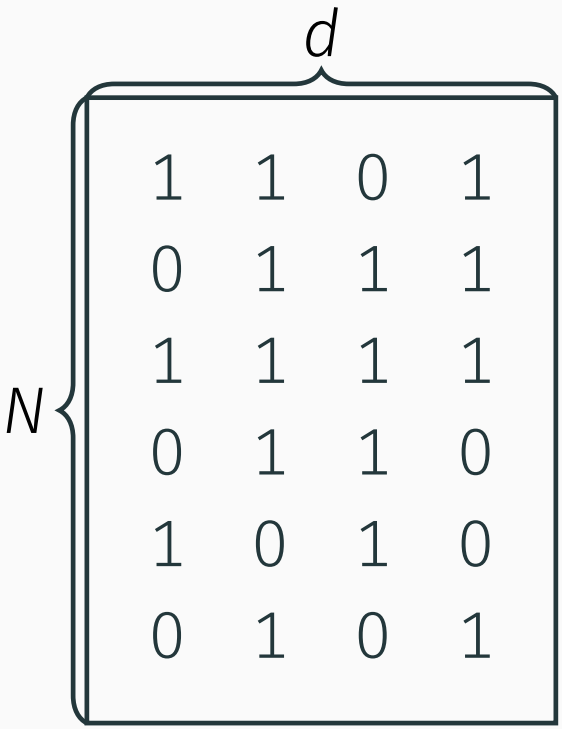
- P vs NP is too hard
- Is SETH False? Beating brute force for SAT doesn't sound that crazy?
- Best algorithm for OV run in time $O(n^{2-1/\log c})$ [AWY14]

(Online) Orthogonal Vectors

Online Orthogonal Vectors Problem

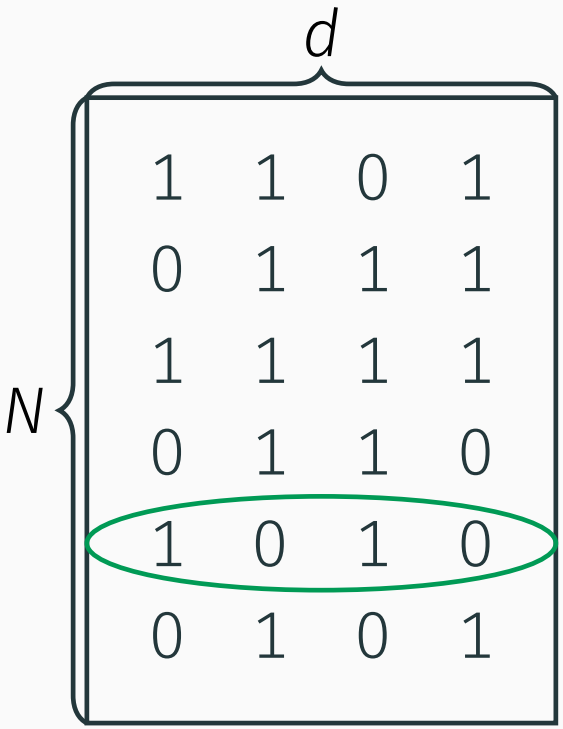


Online Orthogonal Vectors Problem



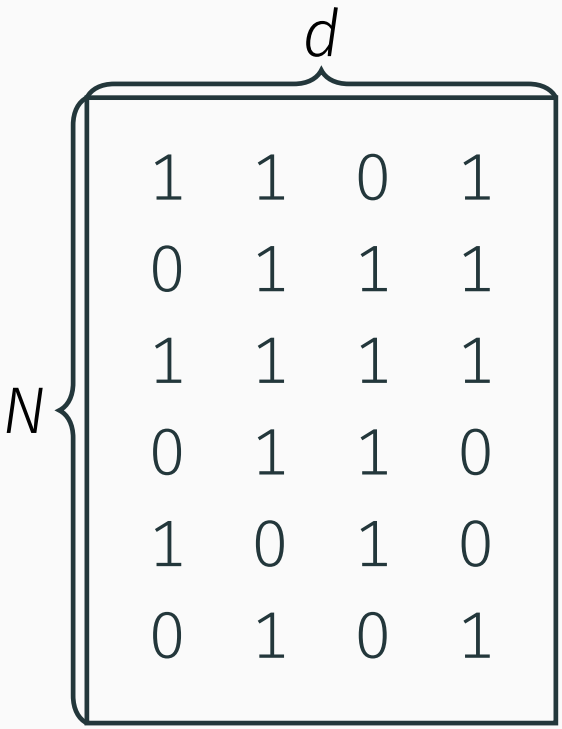
$q = 0100$

Online Orthogonal Vectors Problem



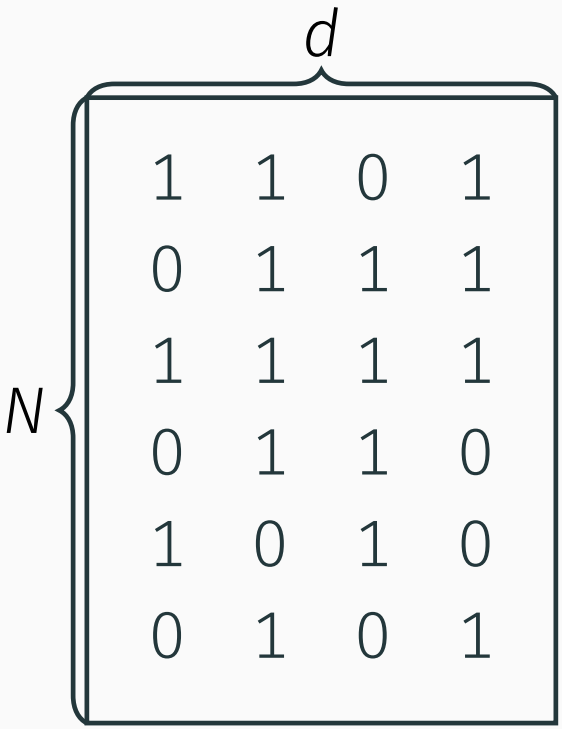
$q = 0100$

Online Orthogonal Vectors Problem



$q = 1011$

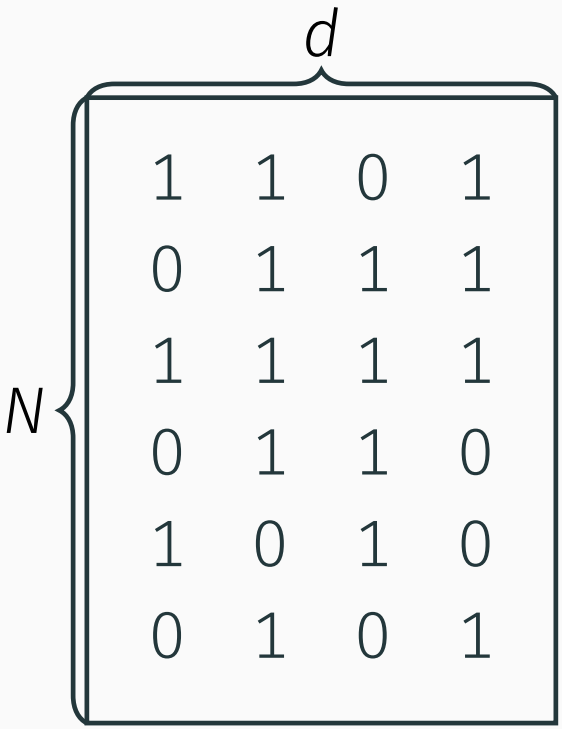
Online Orthogonal Vectors Problem



$q = 1011$

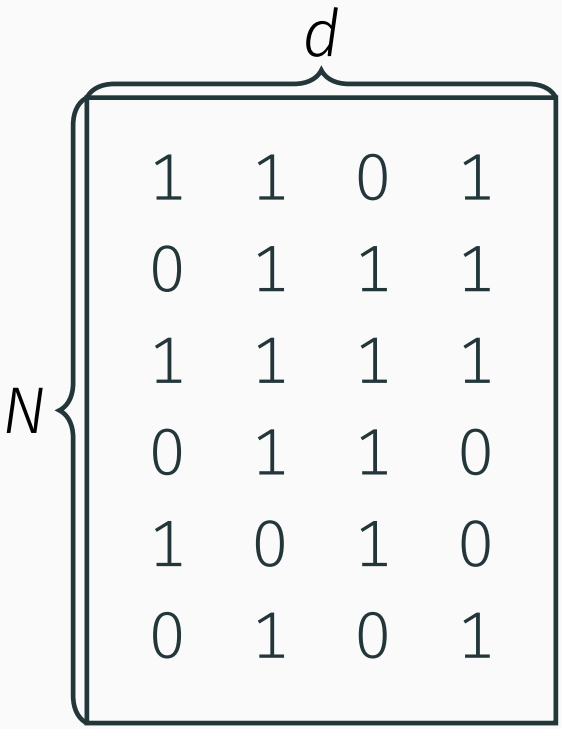
No orthogonal vector

Online Orthogonal Vectors Problem



q

Online Orthogonal Vectors Problem



q

Does there exist input vector whose set of 0s contains q's 1s?

Equivalent Formulations

- SubsetQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : q \subseteq S_i$.

Equivalent Formulations

- SubsetQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : q \subseteq S_i$.
- ContainmentQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : S_i \subseteq q$.

Equivalent Formulations

- SubsetQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : q \subseteq S_i$.
- ContainmentQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : S_i \subseteq q$.
- PartialMatch: preprocess n vectors $x_1, \ldots, x_n \in \{0,1\}^d$. For a query $q \in \{0,1,*\}^d$, check if $\exists i :$ for all j, $q[j] = *$ or $q[j] = x_i[j]$.

Equivalent Formulations

- SubsetQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : q \subseteq S_i$.
- ContainmentQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : S_i \subseteq q$.
- PartialMatch: preprocess n vectors $x_1, \ldots, x_n \in \{0,1\}^d$. For a query $q \in \{0,1,*\}^d$, check if $\exists i :$ for all $j, q[j] = *$ or $q[j] = x_i[j]$.
- DNF Evaluation: preprocess a DNF φ with n clauses and d variables. For a query $x \in \{0,1\}^d$, evaluate $\varphi(x)$.

Equivalent Formulations

- SubsetQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : q \subseteq S_i$.
- ContainmentQuery: preprocess n sets $S_1, \ldots, S_n \subseteq [d]$. For a query $q \subseteq [d]$, check if $\exists i : S_i \subseteq q$.
- PartialMatch: preprocess n vectors $x_1, \ldots, x_n \in \{0,1\}^d$. For a query $q \in \{0,1,*\}^d$, check if $\exists i :$ for all $j, q[j] = *$ or $q[j] = x_i[j]$.
- DNF Evaluation: preprocess a DNF φ with n clauses and d variables. For a query $x \in \{0,1\}^d$, evaluate $\varphi(x)$.
- OnlineInnerProduct, OrthogonalRangeSearch, ApproximateNearestNeighbors, ...

Time-Space Tradeoffs

- Preprocess the N input vectors from $\{0,1\}^d$ into data structure $\sigma \in \{0,1\}^S$, $d = c \log N$

Time-Space Tradeoffs

- Preprocess the N input vectors from $\{0,1\}^d$ into data structure $\sigma \in \{0,1\}^S$, $d = c \log N$
- Answer each query in time T

Time-Space Tradeoffs

- Preprocess the N input vectors from $\{0,1\}^d$ into data structure $\sigma \in \{0,1\}^S$, $d = c \log N$
- Answer each query in time T
- Two trivial solutions:

Time-Space Tradeoffs

- Preprocess the N input vectors from $\{0,1\}^d$ into data structure $\sigma \in \{0,1\}^S$, $d = c \log N$
- Answer each query in time T
- Two trivial solutions:
  - $S = N \cdot d$, $T = N \cdot d$

Time-Space Tradeoffs

- Preprocess the N input vectors from $\{0,1\}^d$ into data structure $\sigma \in \{0,1\}^S$, $d = c \log N$
- Answer each query in time T
- Two trivial solutions:
  - $S = N \cdot d$, $T = N \cdot d$
  - $S = N^c$, $T = d$

Time-Space Tradeoffs

- Preprocess the N input vectors from $\{0,1\}^d$ into data structure $\sigma \in \{0,1\}^S$, $d = c \log N$
- Answer each query in time T
- Two trivial solutions:
  - $S = N \cdot d$, $T = N \cdot d$
  - $S = N^c$, $T = d$
- Can we simultaneously have $T \ll N$ and $S \ll N^c$?

Conjectures

Conjecture [Riv74]

If $S = O(Nd)$, then $T \geq N^{1-O(s/d)}$ for an average query of sparsity s.

Conjectures

Conjecture [Riv74]

If $S = O(Nd)$, then $T \geq N^{1-O(s/d)}$ for an average query of sparsity s.

Conjecture [BOR99,Ind01]

For $d = \omega(\log N)$, either $S = 2^{\Omega(d)}$ or $T = \Omega(N)$.

Conjectures

Conjecture [Riv74]

If $S = O(Nd)$, then $T \geq N^{1-O(s/d)}$ for an average query of sparsity s.

Conjecture [GKLP17]

Either $S \geq \widetilde{\Omega}(N^2)$ or $T \geq \widetilde{\Omega}(N)$ even for $d = O(\log N)$.

Conjecture [BOR99,Ind01]

For $d = \omega(\log N)$, either $S = 2^{\Omega(d)}$ or $T = \Omega(N)$.

Conjectures

Conjecture [GLP17]

For $d = c \log N$, either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Conjecture [GKLP17]

Either $S \geq \widetilde{\Omega}(N^2)$ or $T \geq \widetilde{\Omega}(N)$ even for $d = O(\log$

Conjecture [BOR99,Ind01]

For $d = \omega(\log N)$, either $S = 2^{\Omega(d)}$ or $T = \Omega(N)$.

Conjectures

Conjecture [GLP17]

For $d = c \log N$, either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Conjecture [GLP17]

For $d = c \log N$, either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$ even on average-case inputs.

Conjecture [BOR99,Ind01]

For $d = \omega(\log N)$, either $S = 2^{\Omega(d)}$ or $T = \Omega(N)$.

Conjectures

Conjecture [GLP17]

For $d = c \log N$, either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Conjecture [GLP17]

For $d = c \log N$, either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$ even on average-case inputs.

Conjecture [JPP25]

For $d = N^{\varepsilon}$ and polynomial-time preprocessing, amortized $T = \widetilde{\Omega}(N^{1+\varepsilon})$.

Known Algorithms: $d = c \log N$

Conjecture [GLP17]

Either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Known Algorithms: $d = c \log N$

Conjecture [GLP17]

Either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Algorithms

- [GLP17]: Algorithms with $S = N^{c-0.3}$ and $T = N^{1-\varepsilon}$.

Known Algorithms: $d = c \log N$

Conjecture [GLP17]

Either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Algorithms

- [GLP17]: Algorithms with $S = N^{c-0.3}$ and $T = N^{1-\varepsilon}$.
- [GLP17]: Algorithms with $S = N^{c-0.99}$ and $T = N^{1-\varepsilon}$ in the average case.

Known Algorithms: $d = c \log N$

Conjecture [GLP17]

Either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Algorithms

- [GLP17]: Algorithms with $S = N^{c-0.3}$ and $T = N^{1-\varepsilon}$.
- [GLP17]: Algorithms with $S = N^{c-0.99}$ and $T = N^{1-\varepsilon}$ in the average case.
- [Cha17, AJW25]: Randomized algorithms with $S = N^{1.1}$ and $T = N^{1-\varepsilon}$.

Known Algorithms: $d = c \log N$

Conjecture [GLP17]

Either $S \geq \widetilde{\Omega}(N^{c-1})$ or $T \geq \widetilde{\Omega}(N)$.

Algorithms

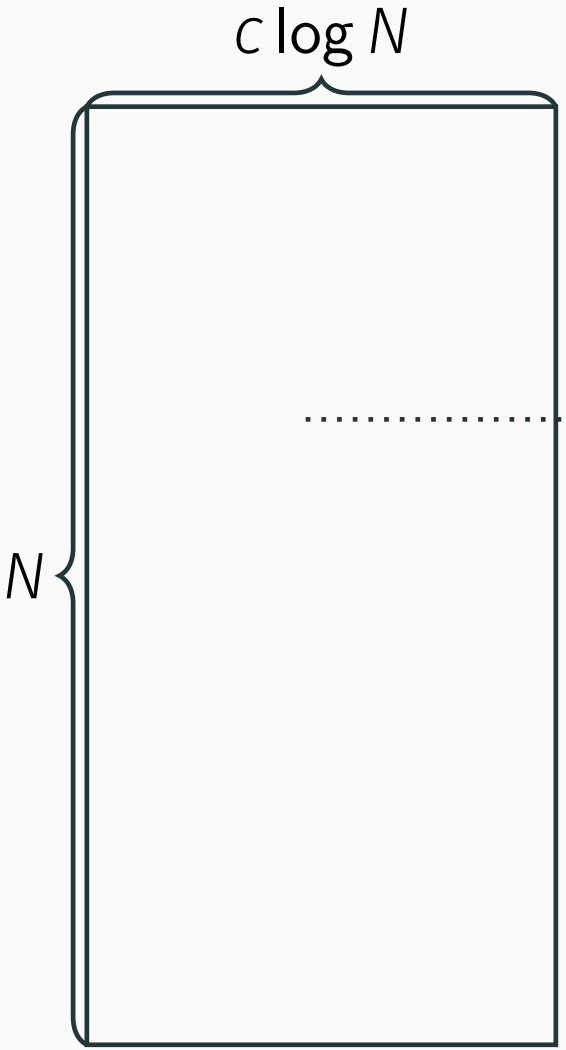
- [GLP17]: Algorithms with $S = N^{c-0.3}$ and $T = N^{1-\varepsilon}$.
- [GLP17]: Algorithms with $S = N^{c-0.99}$ and $T = N^{1-\varepsilon}$ in the average case.
- [Cha17, AJW25]: Randomized algorithms with $S = N^{1.1}$ and $T = N^{1-\varepsilon}$.
- [CIP02]: Algorithms for $d \gg \log N$.

Outline

- Online Orthogonal Vectors
- Algorithms
- Lower Bounds

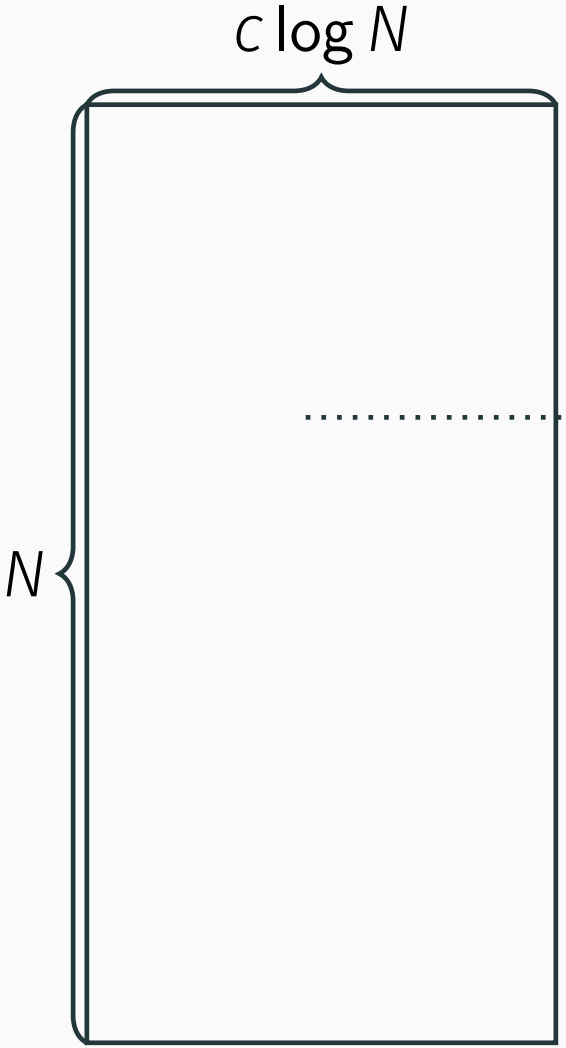
Average-case OnlineOV: Toy Algorithm

Average-case OnlineOV: Toy Algorithm



a typical vector has $\approx c \log N/2$ zeros

Average-case OnlineOV: Toy Algorithm



a typical vector has $\approx c \log N/2$ zeros

it has $\lesssim 2^{c \log N/2} = N^{c/2}$ orthogonal queries

Average-case OnlineOV: Toy Algorithm

a typical vector has $\approx c \log N/2$ zeros

it has $\lesssim 2^{c \log N/2} = N^{c/2}$ orthogonal queries

store all queries q that have an orthogonal vector in the input in space $S \lesssim N^{c/2+1} \ll N^{c-1}$

Average-case OnlineOV: Toy Algorithm

a typical vector has $\approx c \log N/2$ zeros

it has $\lesssim 2^{c \log N/2} = N^{c/2}$ orthogonal queries

store all queries q that have an orthogonal vector in the input in space $S \lesssim N^{c/2+1} \ll N^{c-1}$

look up each query in time $T \ll N$

Average-case OnlineOV

Average-case OnlineOV

sparse query

$$q = \underbrace{1111}_{< \varepsilon \log N} \ldots\ldots\ldots$$

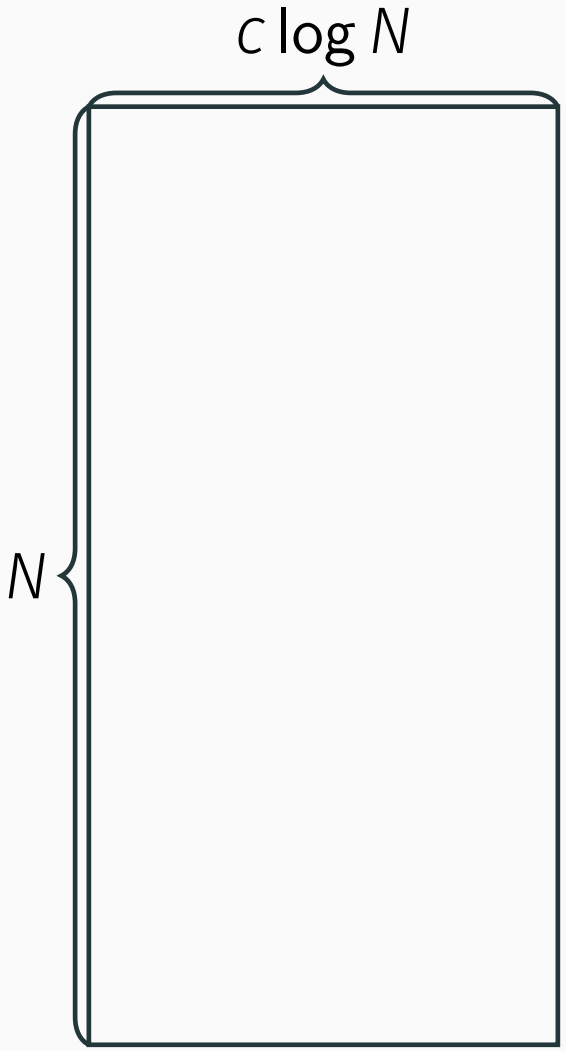
Average-case OnlineOV

sparse query

$$q = \underbrace{1111}_{< \varepsilon \log N}.....$$

Store the answer for each such query

Average-case OnlineOV



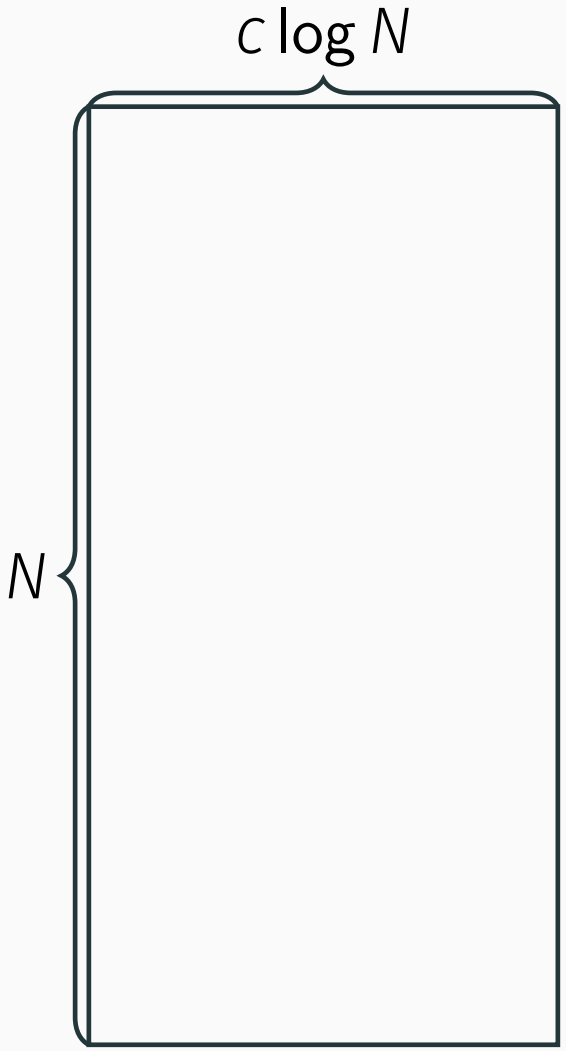
sparse query

$$q = \underbrace{1111}_{< \varepsilon \log N}.....$$

Store the answer for each such query

$$S \leq \binom{c \log N}{\varepsilon \log N} \ll N$$

Average-case OnlineOV



sparse query

$$q = \underbrace{1111}_{< \varepsilon \log N}.....$$

Store the answer for each such query

$$S \leq \binom{c \log N}{\varepsilon \log N} \ll N$$

$$T = d$$

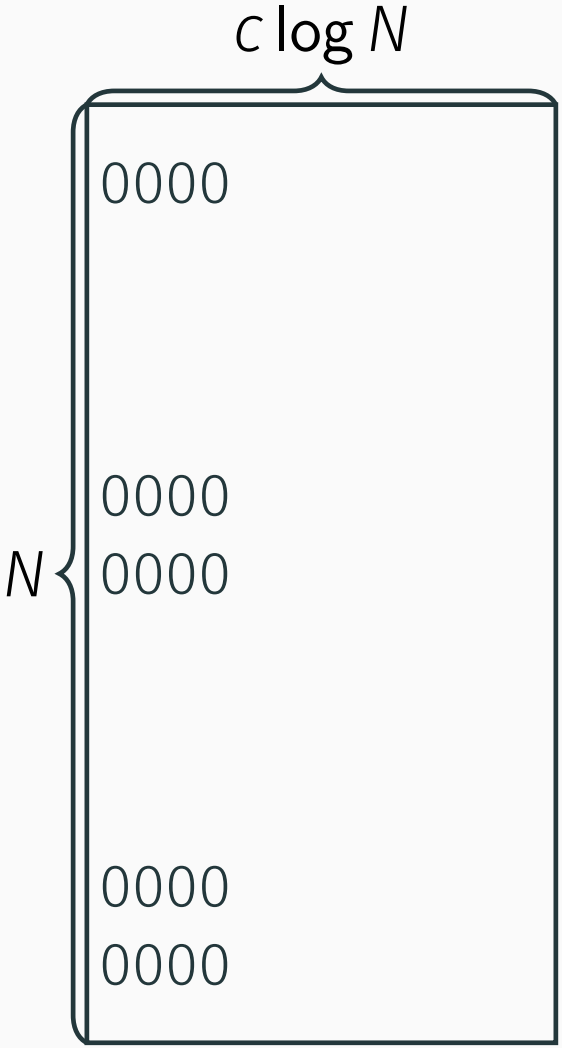
Average-case OnlineOV

Average-case OnlineOV

dense query

$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

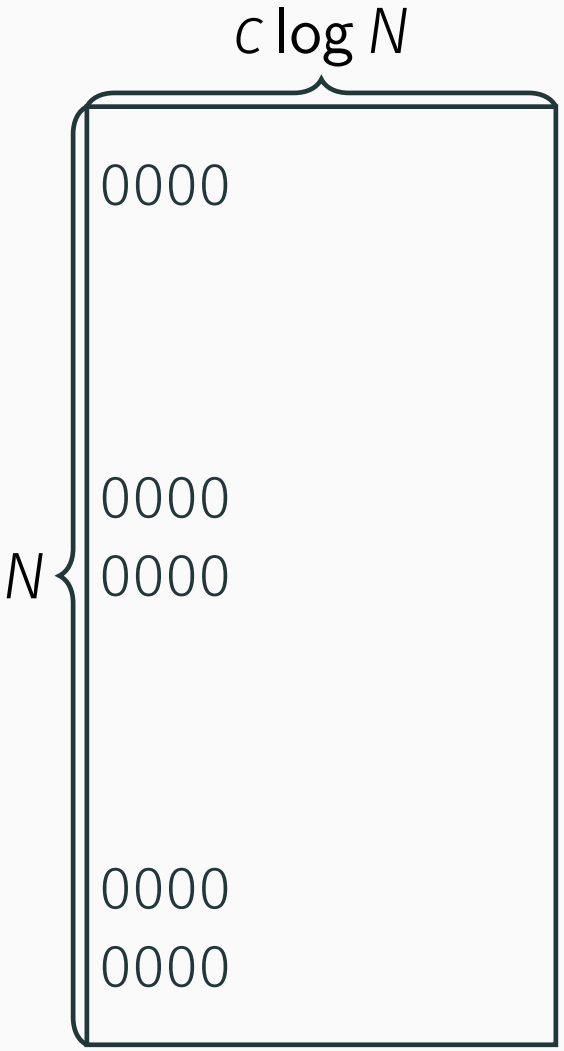
Average-case OnlineOV



dense query

$$q = \underbrace{1111}_{\varepsilon \log N}\ldots\ldots$$

Average-case OnlineOV

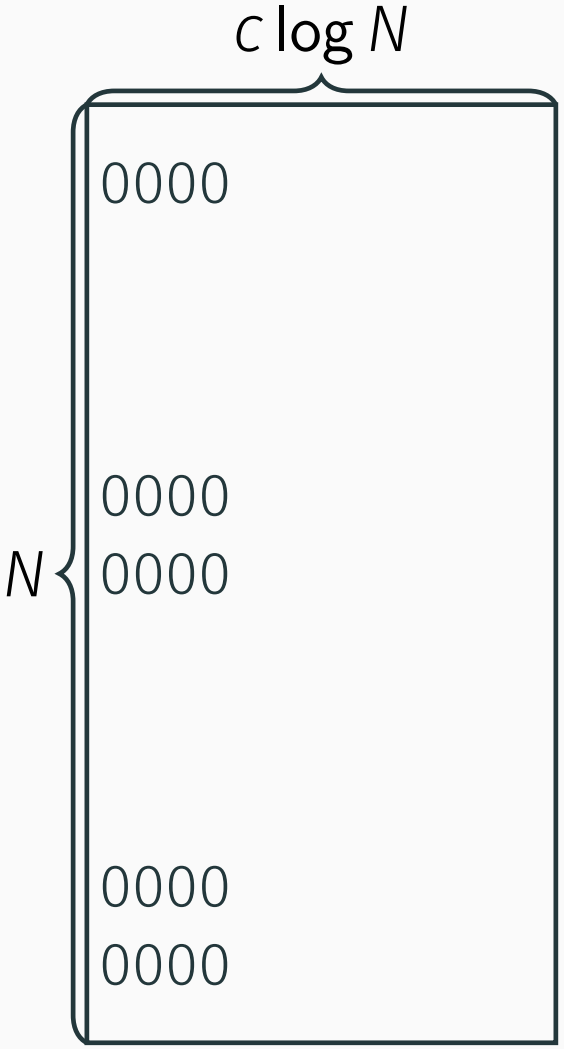


dense query

$$q = \underbrace{1111}_{\varepsilon \log N}\ldots\ldots$$

# of such vectors in random input is $N/2^{\varepsilon \log N} = N^{1-\varepsilon}$

Average-case OnlineOV



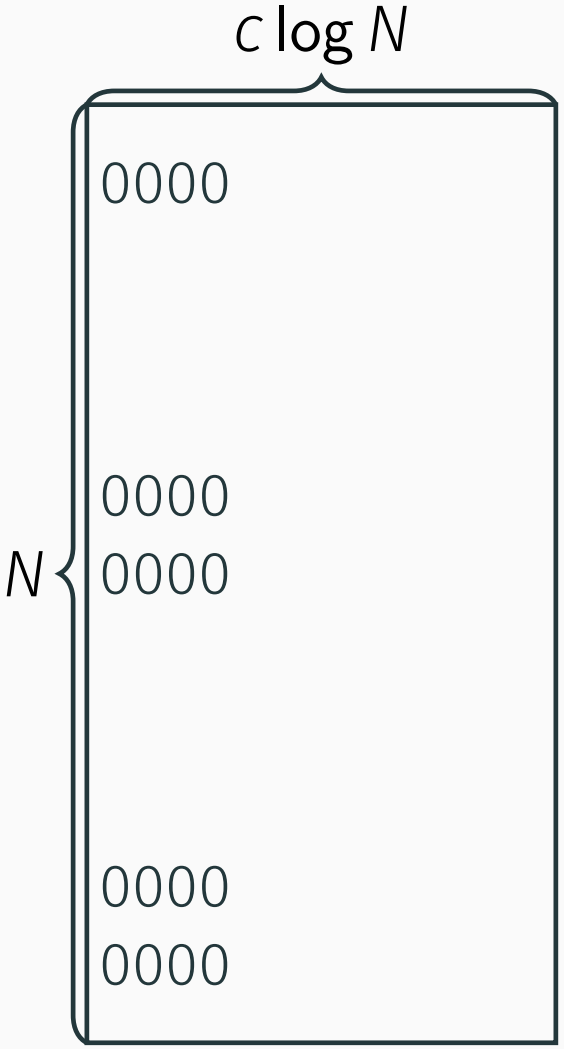
dense query

$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

# of such vectors in random input is $N/2^{\varepsilon \log N} = N^{1-\varepsilon}$

$\forall$ set of $\varepsilon \log N$ coordinates, store all such input vectors

Average-case OnlineOV



dense query

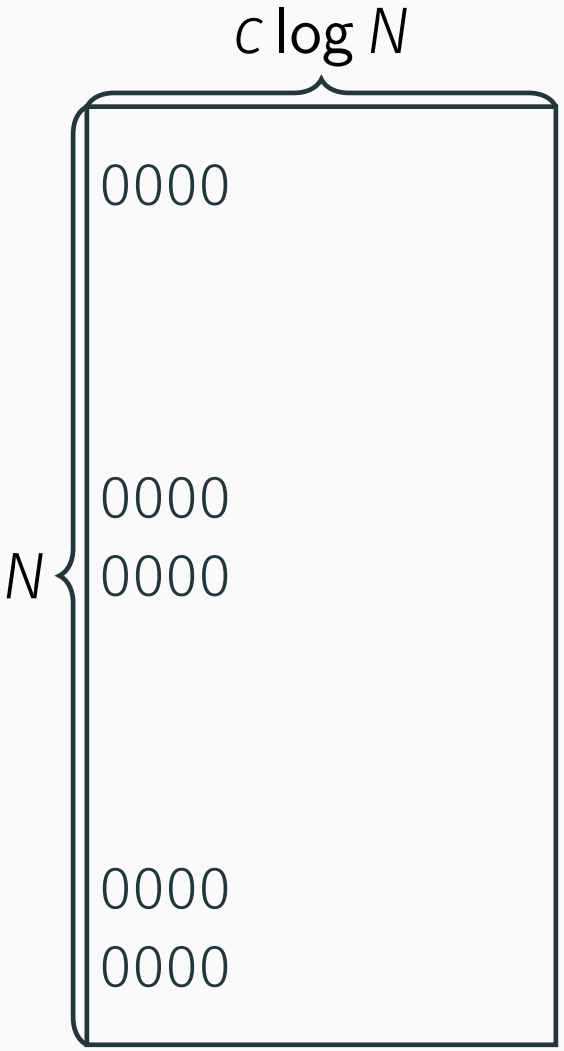
$$q = \underbrace{1111}_{\varepsilon \log N}\ldots\ldots$$

# of such vectors in random input is $N/2^{\varepsilon \log N} = N^{1-\varepsilon}$

$\forall$ set of $\varepsilon \log N$ coordinates, store all such input vectors

$S \leq \binom{c \log N}{\varepsilon \log N} \cdot N^{1-\varepsilon} \leq N^{1.1}$

Average-case OnlineOV



dense query

$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

# of such vectors in random input is $N/2^{\varepsilon \log N} = N^{1-\varepsilon}$

$\forall$ set of $\varepsilon \log N$ coordinates, store all such input vectors

$$S \leq \binom{c \log N}{\varepsilon \log N} \cdot N^{1-\varepsilon} \leq N^{1.1}$$

$$T \leq N^{1-\varepsilon}$$

Applications

- Data structure for average-case OnlineOV with $S = N^{1.1}$ and $T = N^{1-1/\log c}$

Applications

- Data structure for average-case OnlineOV with $S = N^{1.1}$ and $T = N^{1-1/\log c}$

- Even the preprocessing is efficient $T_p = N^{1.1}$

Applications

- Data structure for average-case OnlineOV with $S = N^{1.1}$ and $T = N^{1-1/\log c}$
- Even the preprocessing is efficient $T_p = N^{1.1}$
- Solves average-case Offline OV in time $N^{2-1/\log c}$

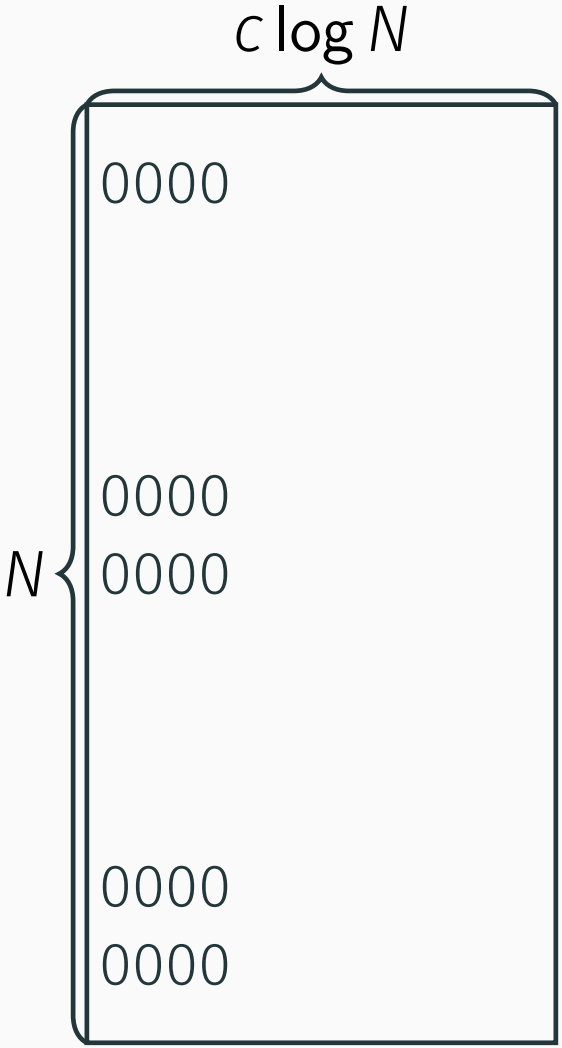
Worst-Case OnlineOV: Structure vs Randomness

Worst-Case OnlineOV: Structure vs Randomness

dense query

$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

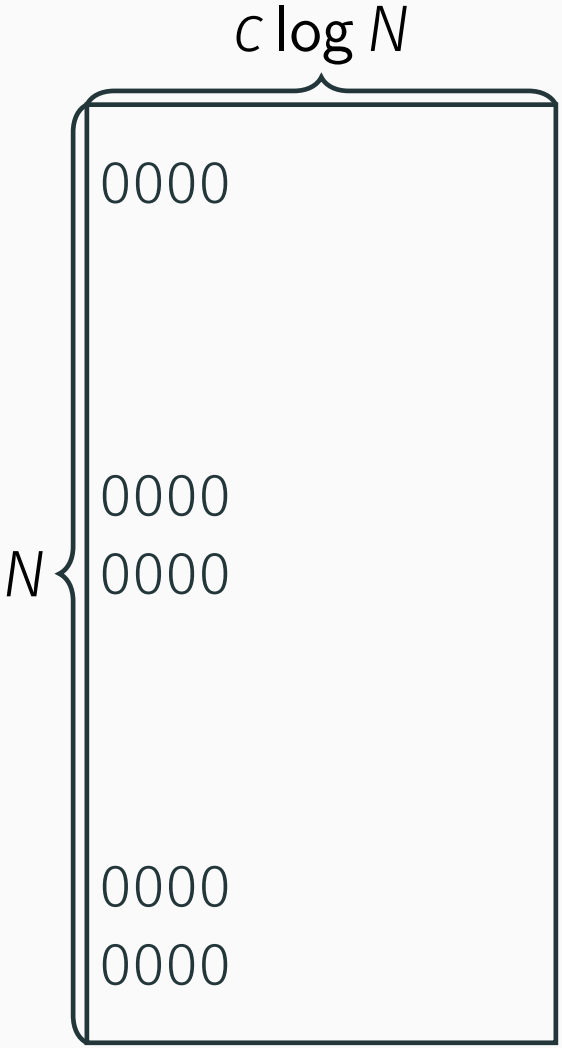
Worst-Case OnlineOV: Structure vs Randomness



dense query

$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

Worst-Case OnlineOV: Structure vs Randomness

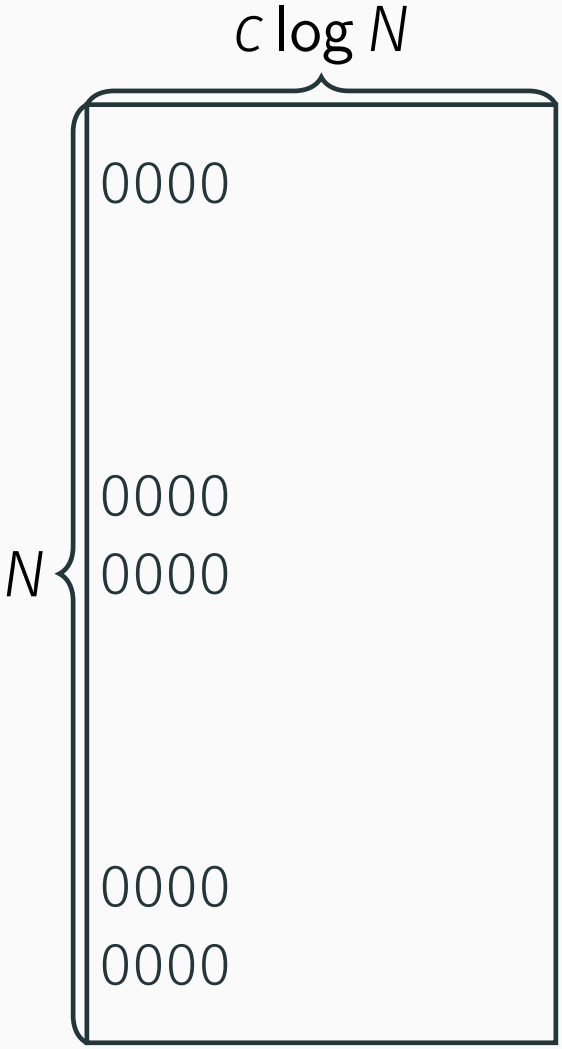


dense query

$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

# of such vectors may be large: $> N^{1-\varepsilon}$

Worst-Case OnlineOV: Structure vs Randomness



dense query

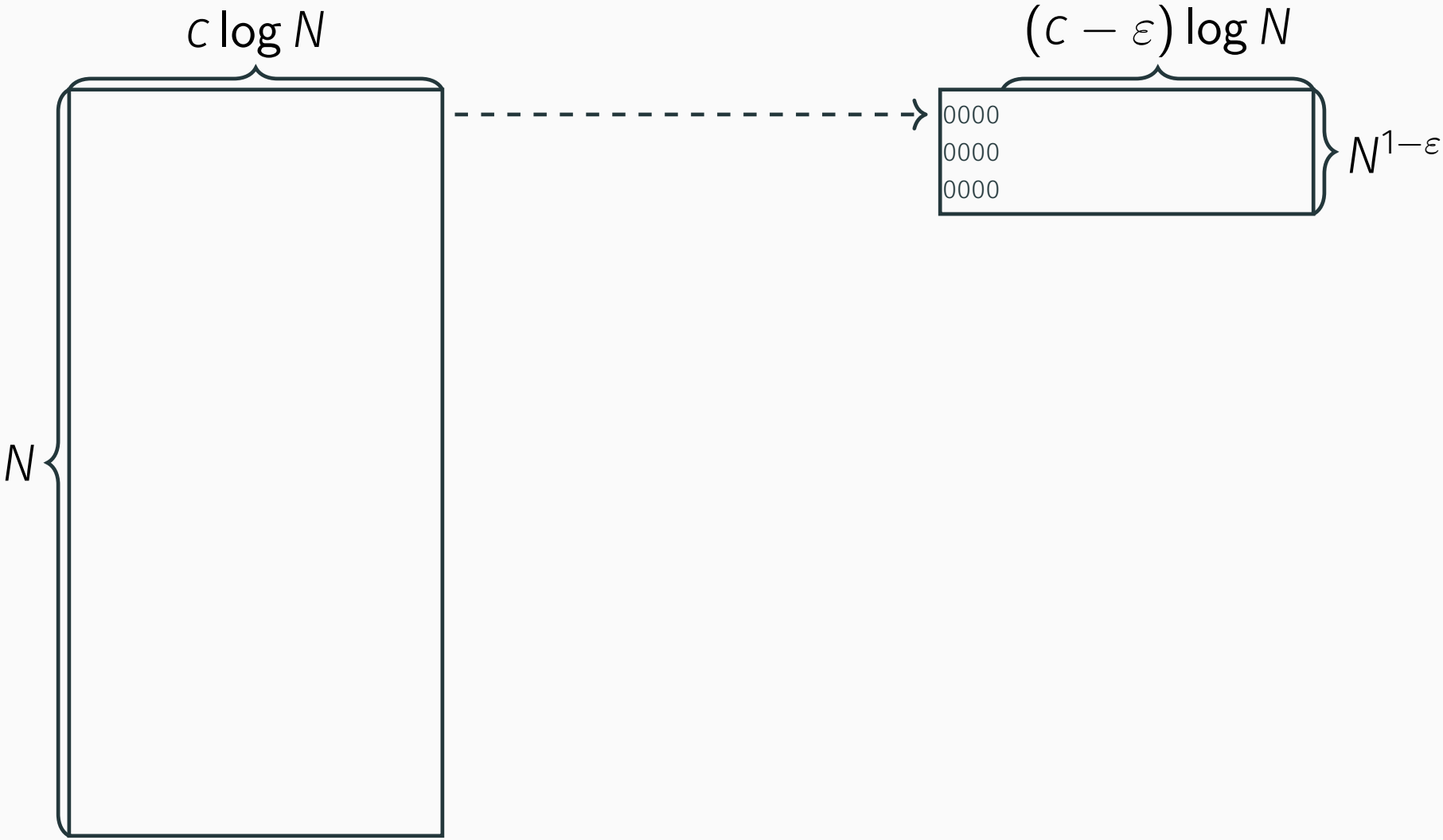
$$q = \underbrace{1111}_{\varepsilon \log N}.....$$

# of such vectors may be large: $> N^{1-\varepsilon}$

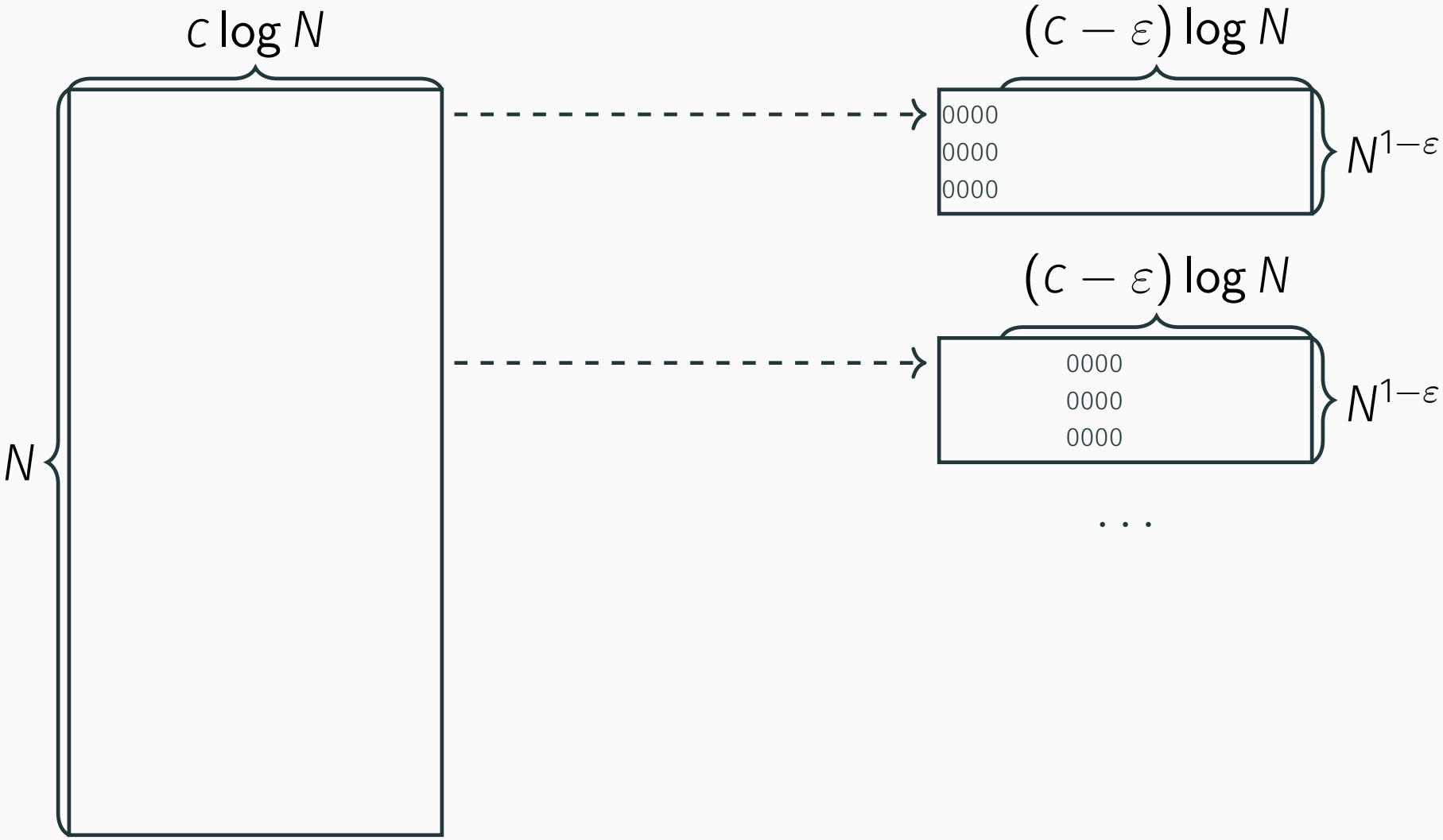
this structure gives us a subinstance of Online OV in $(c - \varepsilon) \log N$ dimensions!

Worst-Case OnlineOV: Structure vs Randomness

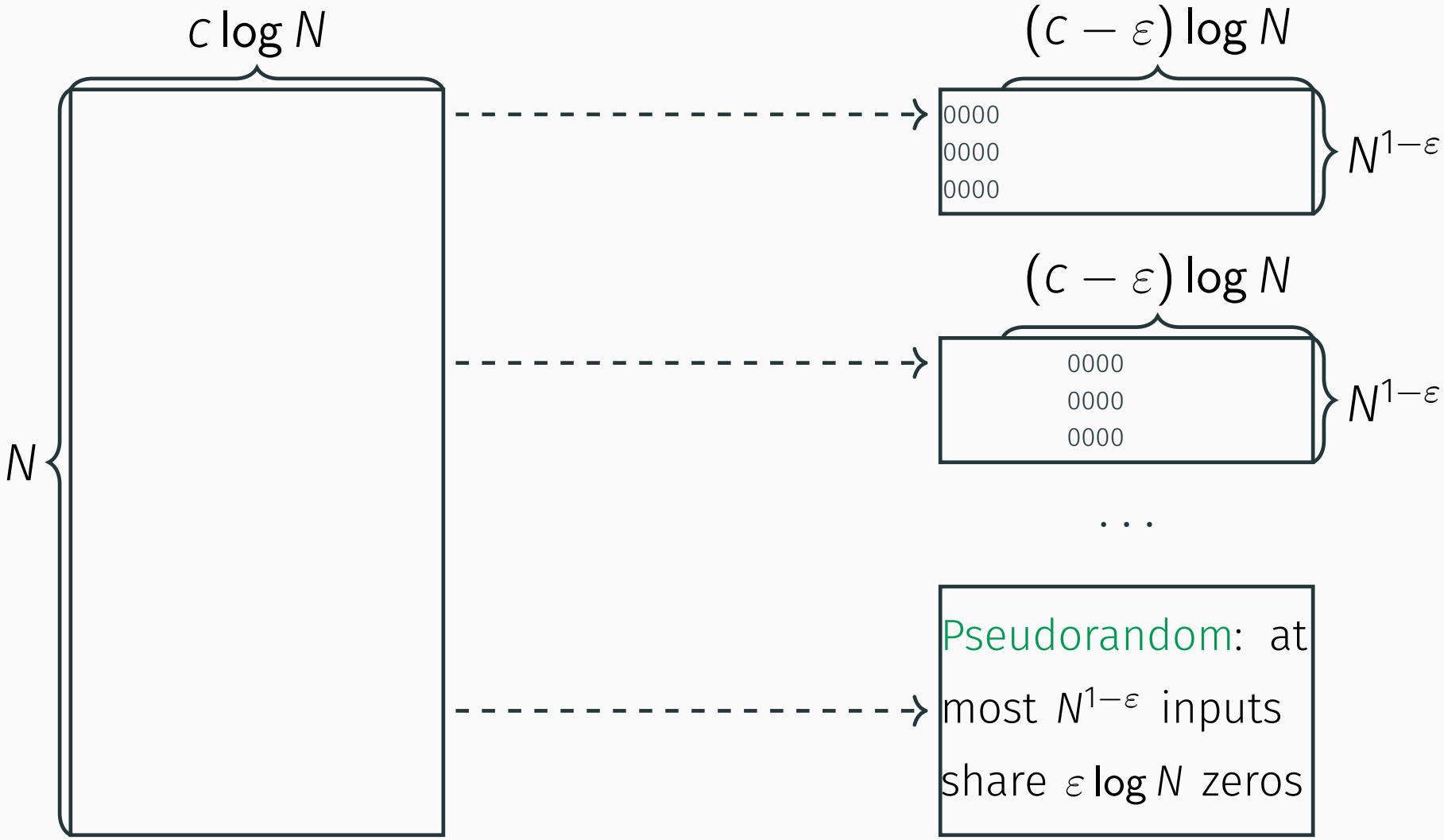
Worst-Case OnlineOV: Structure vs Randomness



Worst-Case OnlineOV: Structure vs Randomness



Worst-Case OnlineOV: Structure vs Randomness



Worst-Case OnlineOV: Structure vs Randomness

- Partition N input vectors into $X_1, \ldots, X_{N^\varepsilon}$ and X
  - X is Pseudorandom

Worst-Case OnlineOV: Structure vs Randomness

- Partition N input vectors into $X_1, \ldots, X_{N^\varepsilon}$ and X
  - X is Pseudorandom
  - Each X_i is of size $N^{1-\varepsilon}$, lives in smaller dimension

Worst-Case OnlineOV: Structure vs Randomness

- Partition N input vectors into $X_1, \ldots, X_{N^\varepsilon}$ and X
  - X is Pseudorandom
  - Each X_i is of size $N^{1-\varepsilon}$, lives in smaller dimension
- Solve X using Average-case algorithm

Worst-Case OnlineOV: Structure vs Randomness

- Partition N input vectors into $X_1, \ldots, X_{N^\varepsilon}$ and X
  - X is Pseudorandom
  - Each X_i is of size $N^{1-\varepsilon}$, lives in smaller dimension
- Solve X using Average-case algorithm

- Solve each X_i recursively

Applications

- Data structure for OnlineOV with $S = T_p = N^{1.1}$ and $T = N^{1-1/(c \log c)}$

Applications

- Data structure for OnlineOV with $S = T_p = N^{1.1}$ and $T = N^{1-1/(c \log c)}$
- For $d = c \log N$, matches the best known randomized data structures, refutes [GLP17] conjecture

Applications

- Data structure for OnlineOV with $S = T_p = N^{1.1}$ and $T = N^{1-1/(c \log c)}$
- For $d = c \log N$, matches the best known randomized data structures, refutes [GLP17] conjecture
- For $d = N^{o(1)}$, improves on known data structures, refutes the curse of dimensionality conjecture

Applications

- Data structure for OnlineOV with $S = T_p = N^{1.1}$ and $T = N^{1-1/(c \log c)}$
- For $d = c \log N$, matches the best known randomized data structures, refutes [GLP17] conjecture
- For $d = N^{o(1)}$, improves on known data structures, refutes the curse of dimensionality conjecture
- Implies new data structures for PartialMatch, DNFEvaluation, SubsetQuery, ContainmentQuery, OrthogonalRangeSearch, ...

Applications

- Data structure for OnlineOV with $S = T_p = N^{1.1}$ and $T = N^{1-1/(c \log c)}$
- For $d = c \log N$, matches the best known randomized data structures, refutes [GLP17] conjecture
- For $d = N^{o(1)}$, improves on known data structures, refutes the curse of dimensionality conjecture
- Implies new data structures for PartialMatch, DNFEvaluation, SubsetQuery, ContainmentQuery, OrthogonalRangeSearch, ...
- Solves Offline OV in time $N^{2-1/(c \log c)}$

Outline

- Online Orthogonal Vectors
- Algorithms
- Lower Bounds

Lower Bounds

- The best lower bounds:

Lower Bounds

- The best lower bounds:
  - For linear space $S = O(N \cdot d)$, $T = \Omega(\log N)$

Lower Bounds

- The best lower bounds:
  - For linear space $S = O(N \cdot d)$, $T = \Omega(\log N)$
  - For super-linear space $S = N^{1.1}$, only $T = \Omega(d)$

Lower Bounds

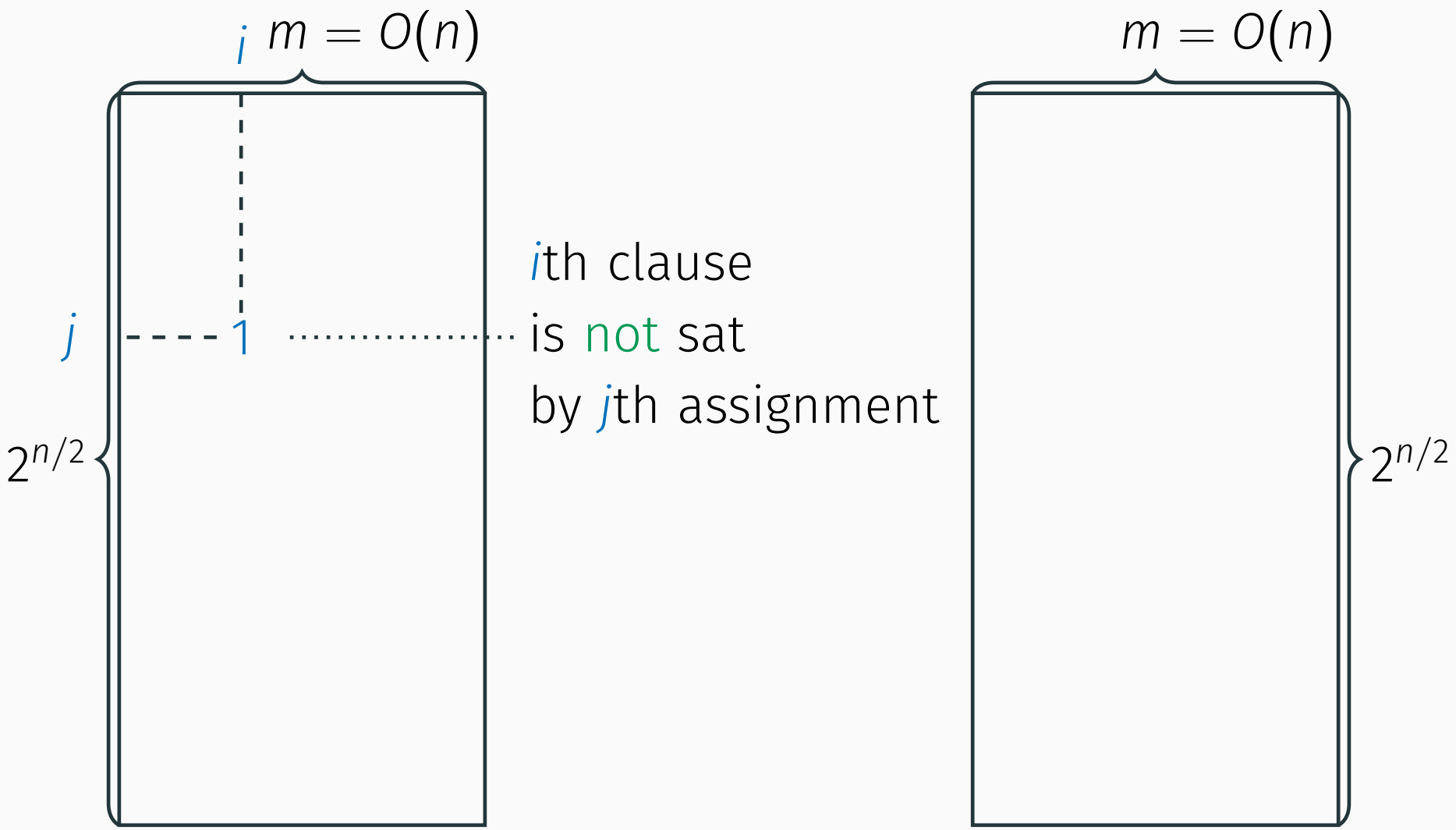
- The best lower bounds:
  - For linear space $S = O(N \cdot d)$, $T = \Omega(\log N)$
  - For super-linear space $S = N^{1.1}$, only $T = \Omega(d)$
- No better lower bound is known for any data structure problem

Lower Bounds

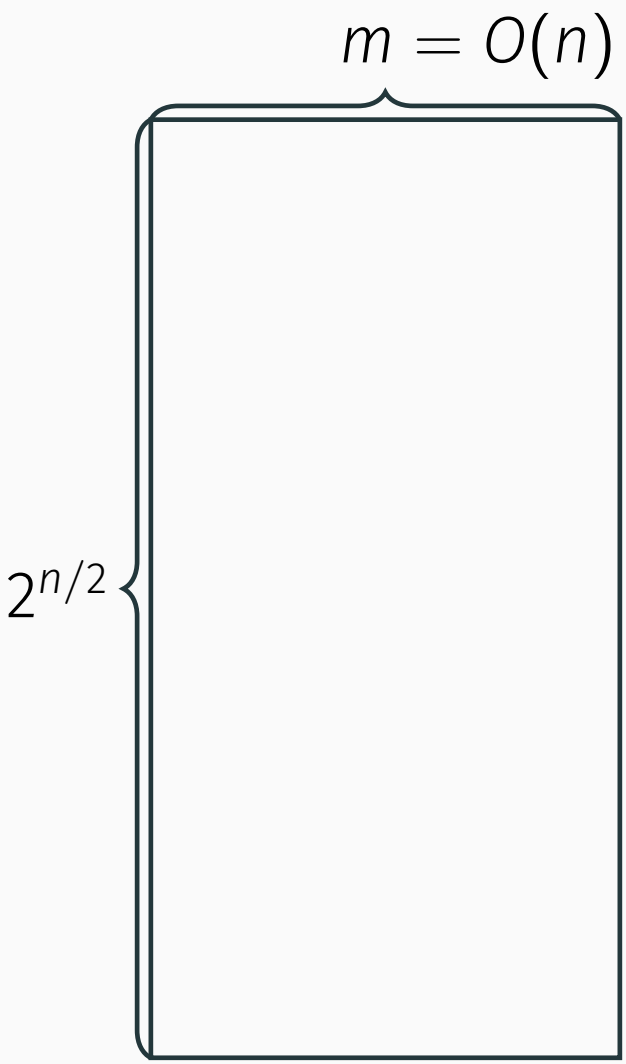
- The best lower bounds:
  - For linear space $S = O(N \cdot d)$, $T = \Omega(\log N)$
  - For super-linear space $S = N^{1.1}$, only $T = \Omega(d)$
- No better lower bound is known for any data structure problem

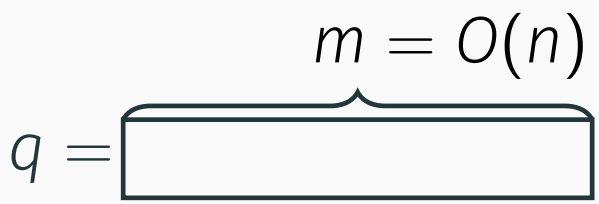
- Holy grail: $S = O(N \cdot d)$, $T \geq N^{\varepsilon}$

SETH Lower Bound

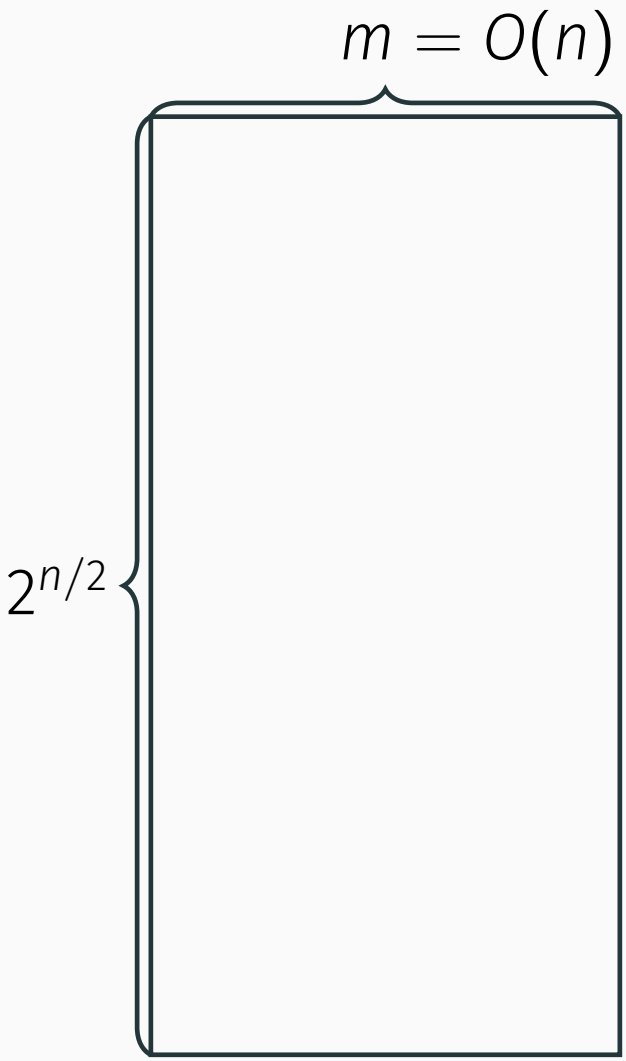


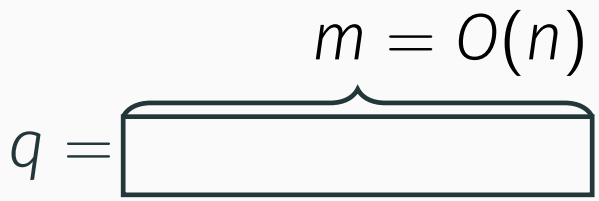
SETH Lower Bound





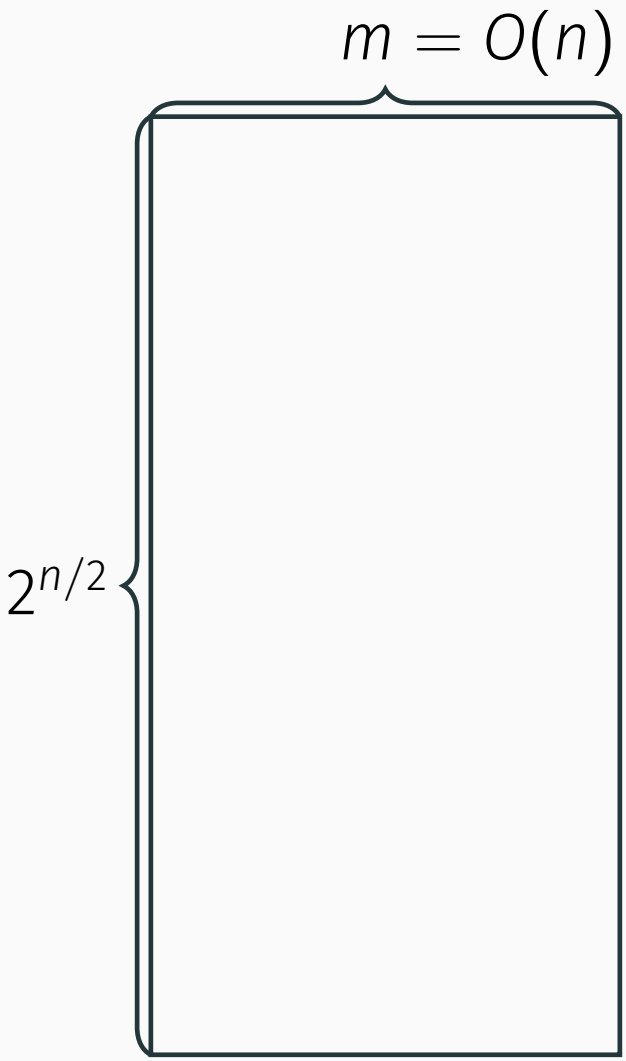
SETH Lower Bound

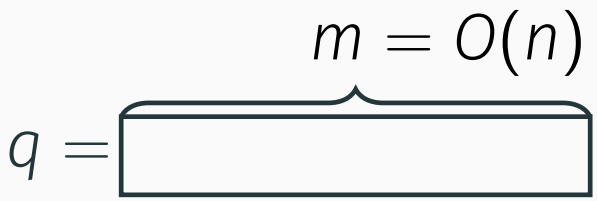




Data structure with $T_p = N^{2-\varepsilon}$ and $T = N^{1-\varepsilon}$ would refute SETH

SETH Lower Bound

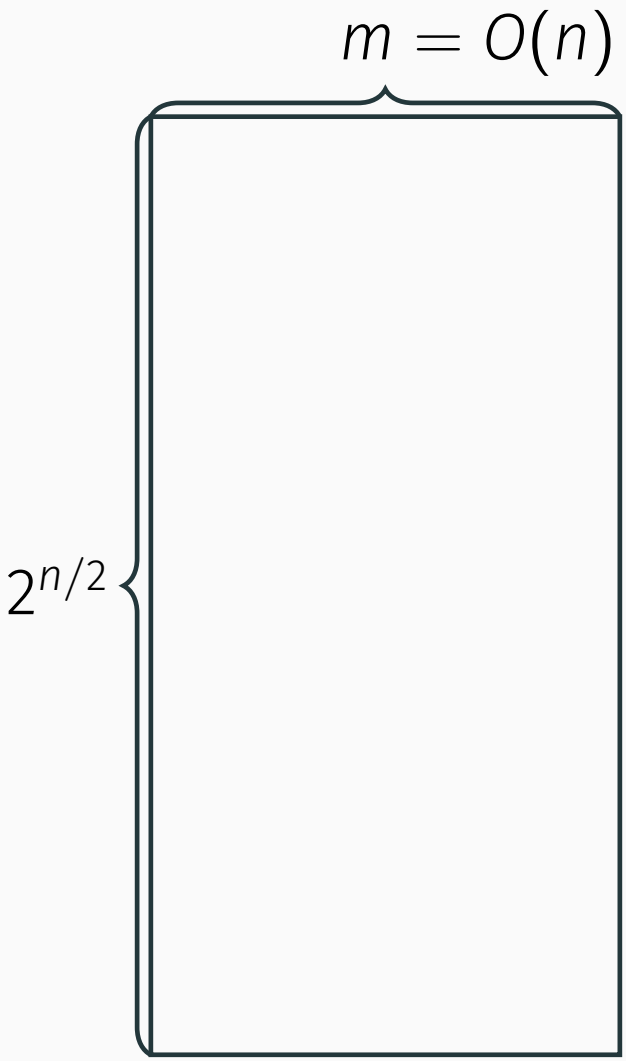


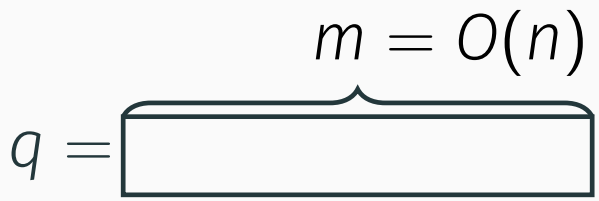


Data structure with $T_p = N^{2-\varepsilon}$ and $T = N^{1-\varepsilon}$ would refute SETH

Data structure with $T_p = N^{C}$ and $T = N^{1-\varepsilon}$ would refute SETH

SETH Lower Bound





Data structure with $T_p = N^{2-\varepsilon}$ and $T = N^{1-\varepsilon}$ would refute SETH

Data structure with $T_p = N^C$ and $T = N^{1-\varepsilon}$ would refute SETH

Not a time-space tradeoff!

Non-Uniform SETH

- Non-uniform algorithms: have an advice string for each n (not for each input)

Non-Uniform SETH

- Non-uniform algorithms: have an advice string for each n (not for each input)
- SAT can be solved in time $O(2^n \cdot \mathbf{poly}(|\varphi|))$

Non-Uniform SETH

- Non-uniform algorithms: have an advice string for each n (not for each input)
- SAT can be solved in time $O(2^n \cdot \mathbf{poly}(|\varphi|))$
- We don't know how to solve SAT exponentially faster: in time $2^{(1-\varepsilon)n}$ even when $m = O(n)$, even by non-uniform algorithms

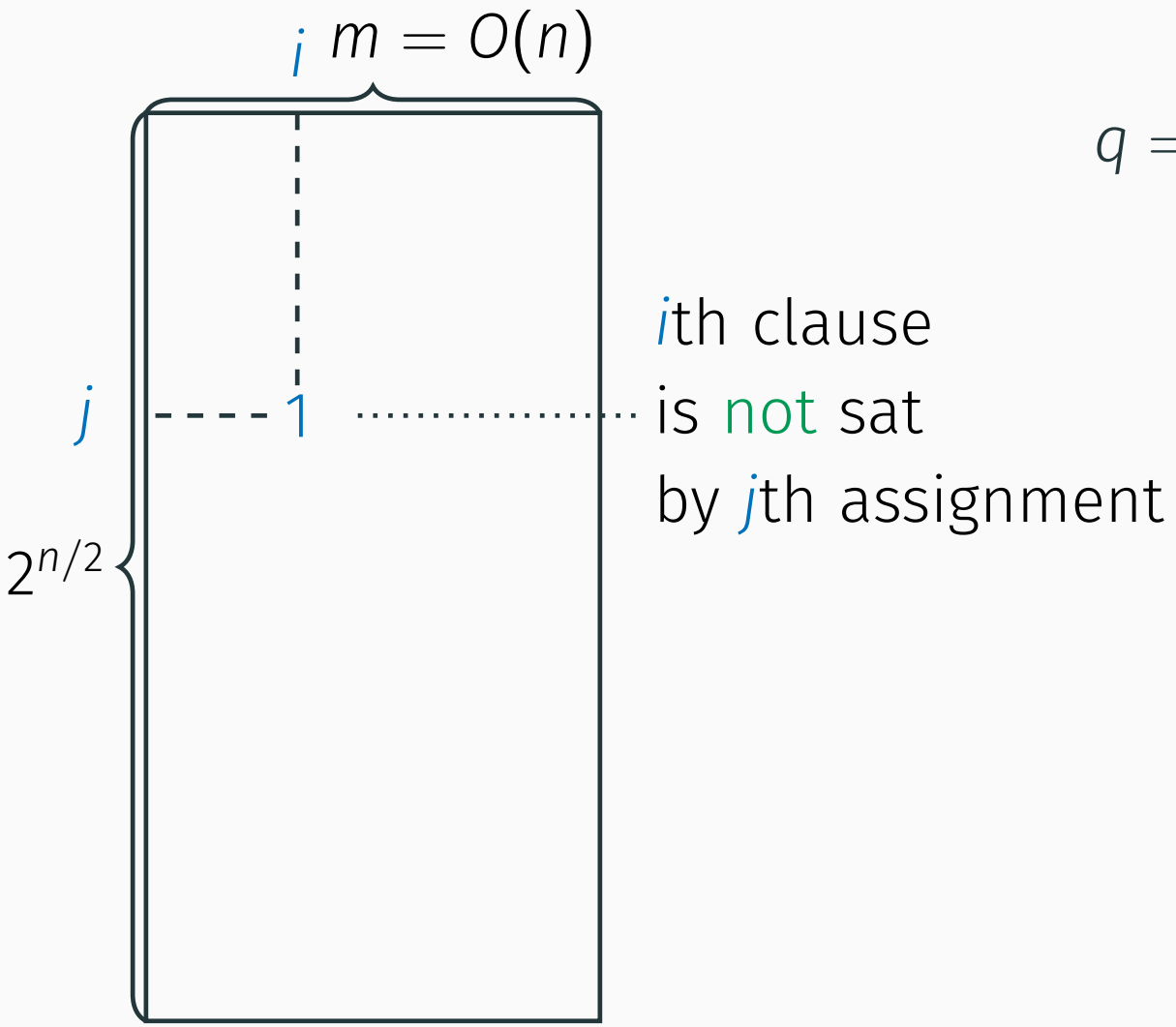
Non-Uniform SETH

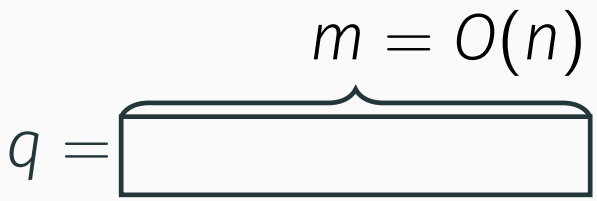
- Non-uniform algorithms: have an advice string for each n (not for each input)
- SAT can be solved in time $O(2^n \cdot \mathbf{poly}(|\varphi|))$
- We don't know how to solve SAT exponentially faster: in time $2^{(1-\varepsilon)n}$ even when $m = O(n)$, even by non-uniform algorithms

Non-Uniform Strong Exponential Time Hypothesis (NUSETH)

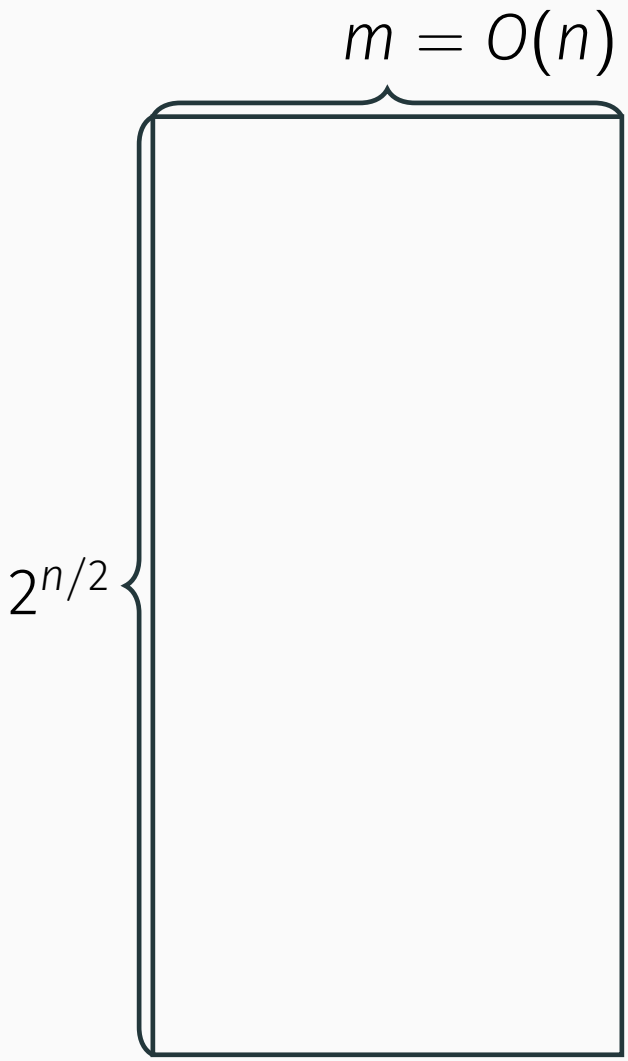
For every $\varepsilon > 0$, there exists k such that no non-uniform algorithm solves k-SAT on formulas with n variables in time $2^{(1-\varepsilon)n}$ (with advice of length $2^{(1-\varepsilon)n}$)

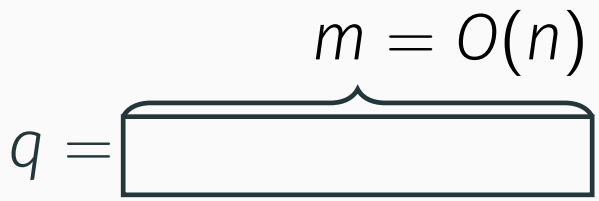
NUSETH Lower Bound





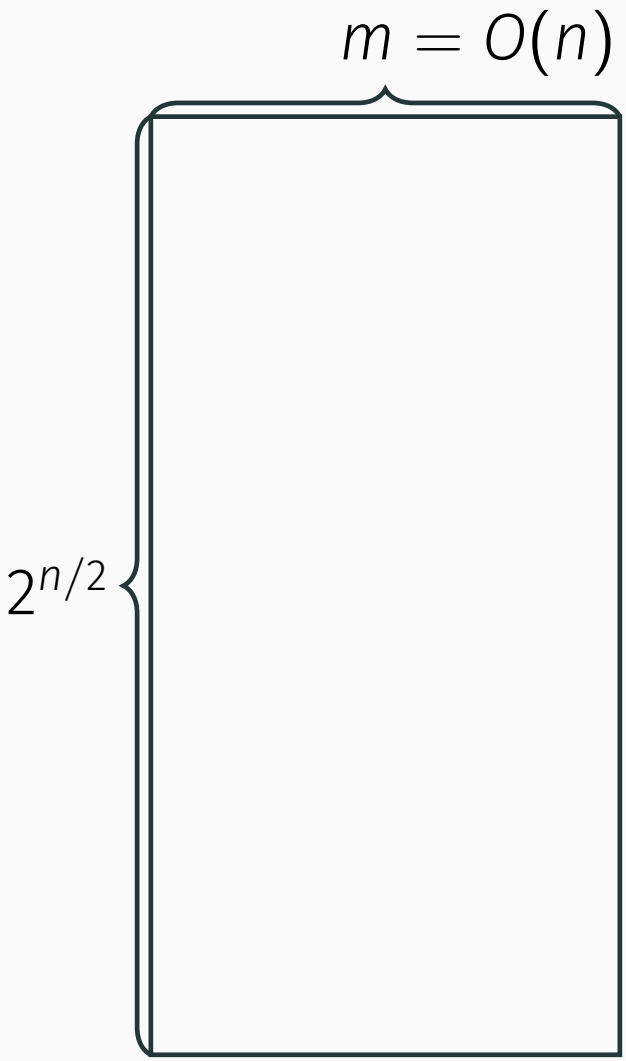
NUSETH Lower Bound

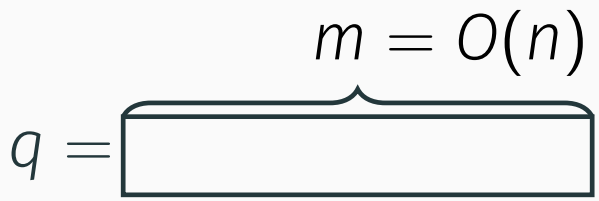




The OnlineOV instance depends on φ, not on n

NUSETH Lower Bound

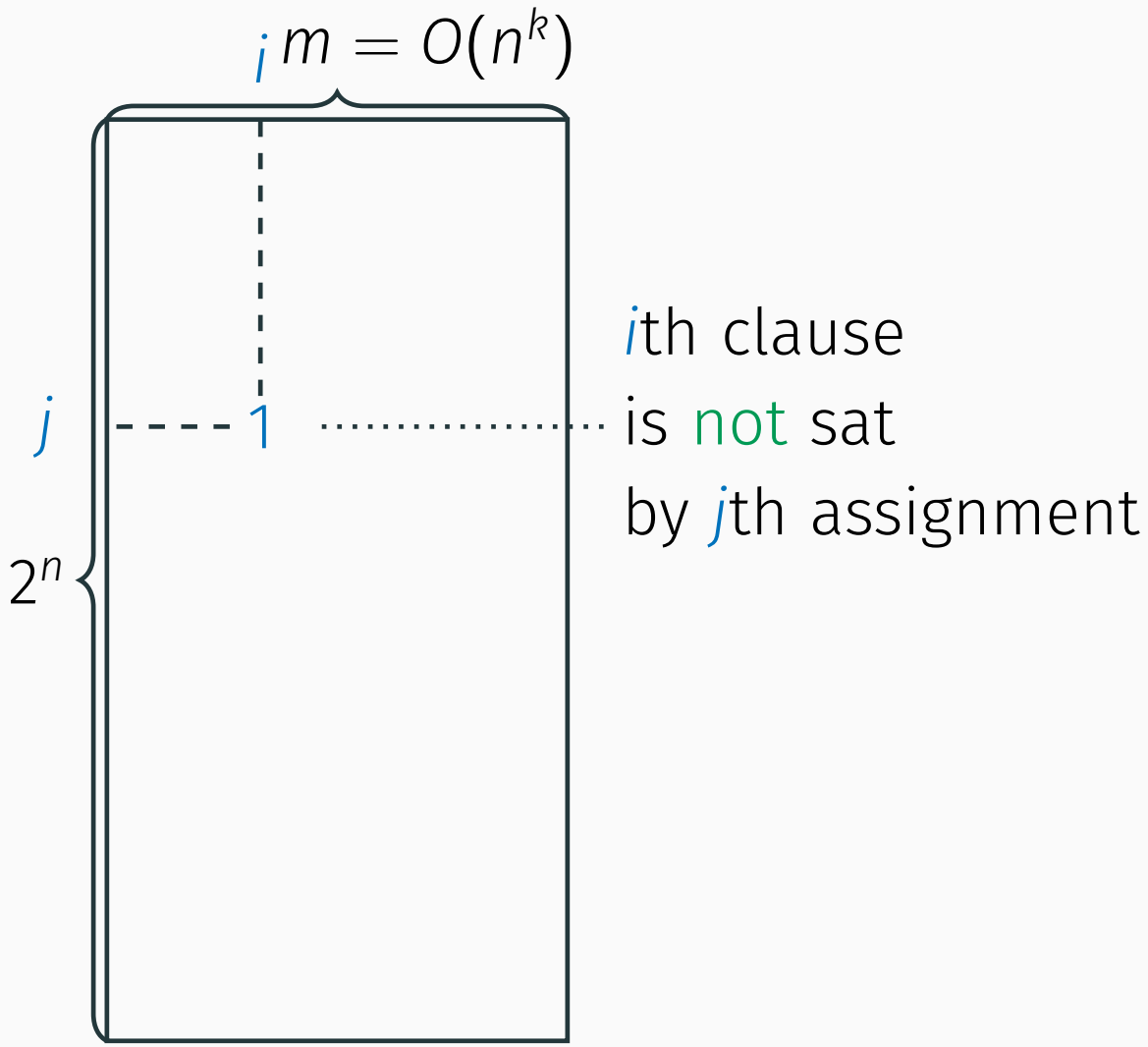




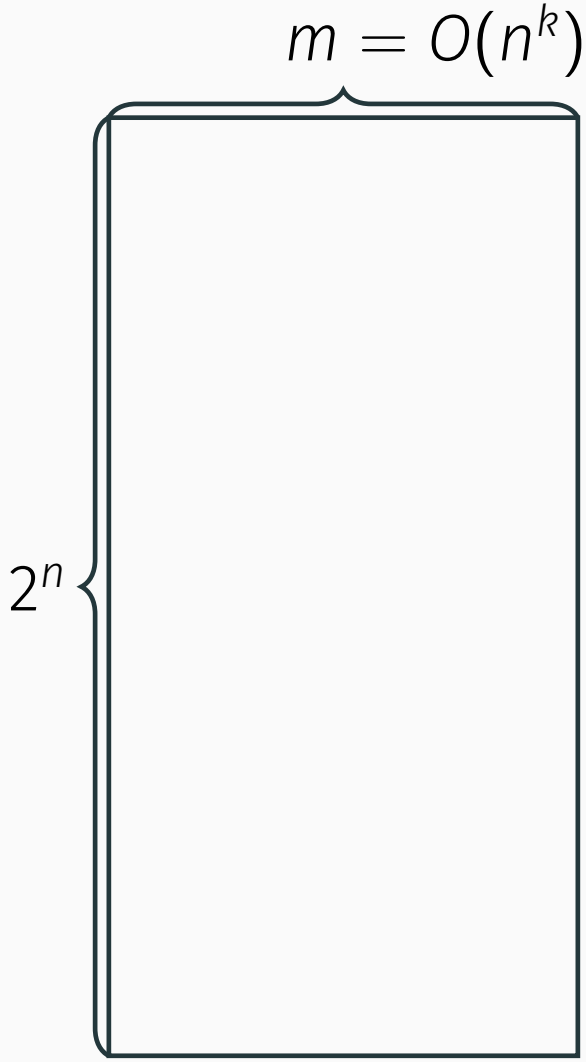
The OnlineOV instance depends on φ, not on n

We need one instance of OnlineOV that solves all φ's

Hardest Instance of OnlineOV



Hardest Instance of OnlineOV

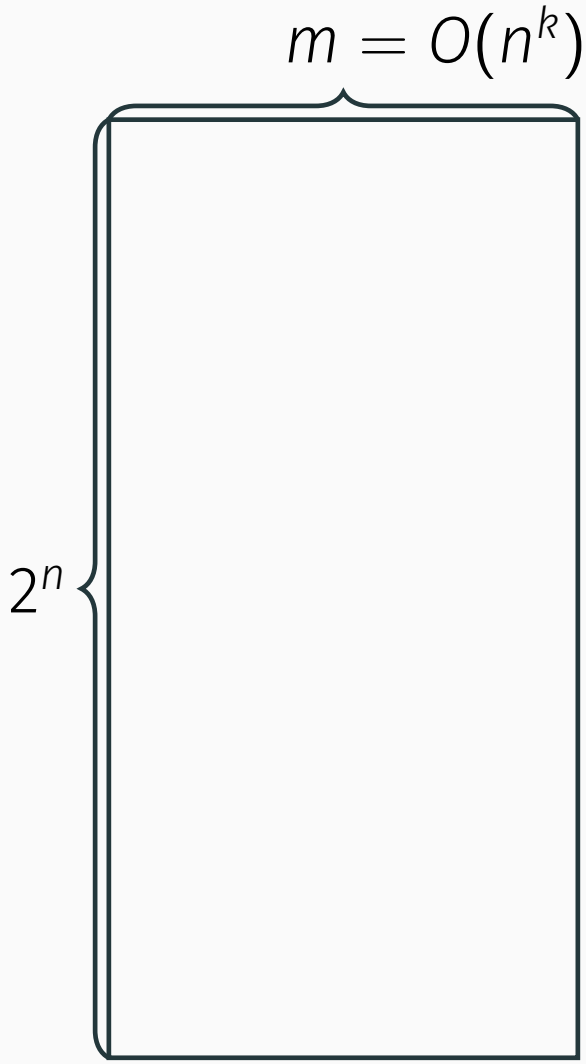


Given k-CNF formula φ, create query q

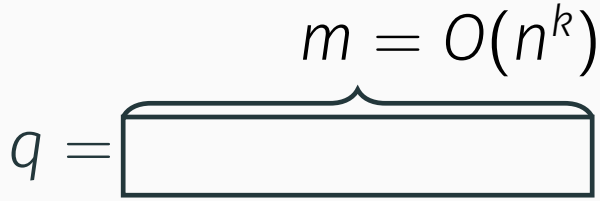
$q =$ 

Indicator vector of all clauses in φ

Hardest Instance of OnlineOV



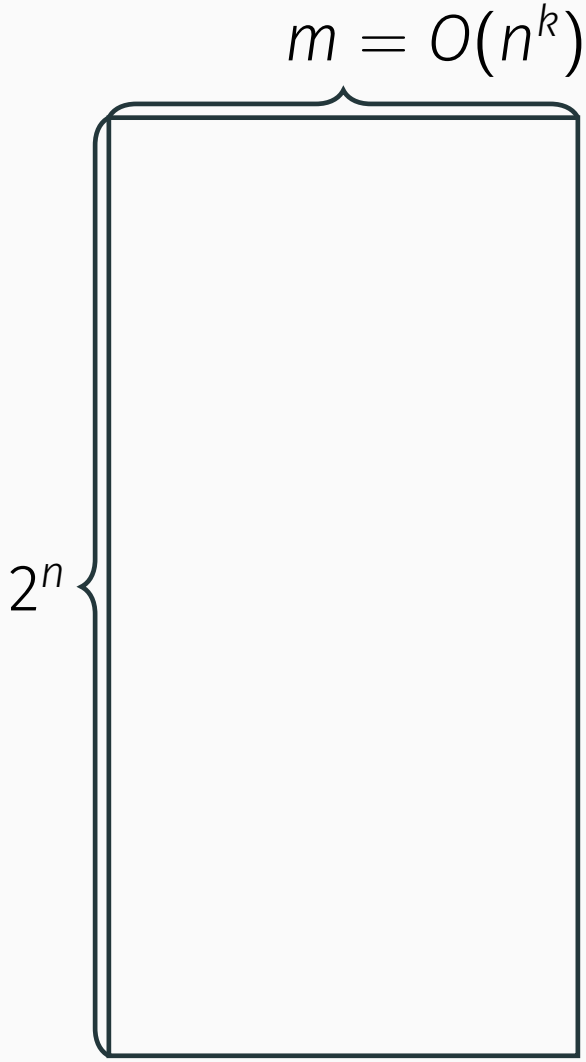
Given k-CNF formula φ, create query q



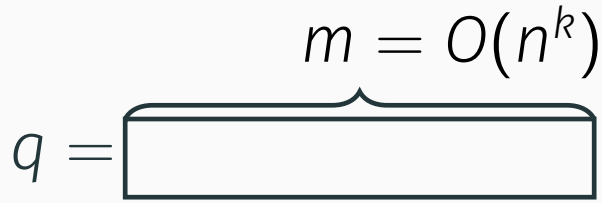
Indicator vector of all clauses in φ

φ is satisfiable iff q is orthogonal to a vector in the table

Hardest Instance of OnlineOV



Given k-CNF formula φ, create query q



Indicator vector of all clauses in φ

φ is satisfiable iff q is orthogonal to a vector in the table

Use downward self-reducibility to get $2^{n/2}$ formulas over $n/2$ variables.

Generalizations

- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for OnlineOV in $d = \mathsf{poly}(\log N)$ dimensions. ([AV21]).

Generalizations

- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for OnlineOV in $d = \mathbf{poly}(\log N)$ dimensions. ([AV21]).
- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for Sparse-OnlineOV in $d = N^\delta$ dimensions

Generalizations

- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for OnlineOV in $d = \mathbf{poly}(\log N)$ dimensions. ([AV21]).
- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for Sparse-OnlineOV in $d = N^\delta$ dimensions
- Under Non-Uniform Hampath, no $(N^{1.08}, N^{1.08})$-data structure with unbounded preprocessing for 3-SUM with preprocessing

Generalizations

- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for OnlineOV in $d = \mathbf{poly}(\log N)$ dimensions. ([AV21]).
- Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for Sparse-OnlineOV in $d = N^\delta$ dimensions
- Under Non-Uniform Hampath, no $(N^{1.08}, N^{1.08})$-data structure with unbounded preprocessing for 3-SUM with preprocessing
- Lower bounds for PartialMatch, SubsetQuery, ContainmentQuery, OrthogonalRangeSearch, DNFEvaluation, OnlineIP, ApproxNearestNeighbors, ...

Thanks!

Theorem (Upper Bound)

There is deterministic (S, T)-data structure for OnlineOV in $d = c \log N$ dimensions with $S = T_p = N^{1.1}$ and $T = N^{1-\varepsilon}$

Theorem (Lower Bound)

Under NUSETH, there is no $(N^c, N^{1-\varepsilon})$-data structure with unbounded preprocessing for OnlineOV in $d = \mathrm{poly}(\log N)$ dimensions