Communication Complexity of Collision

Mika Göös (EPFL), Siddhartha Jain (EPFL → UT Austin)

Main result

CC lower bound for NATURAL two-party Collision

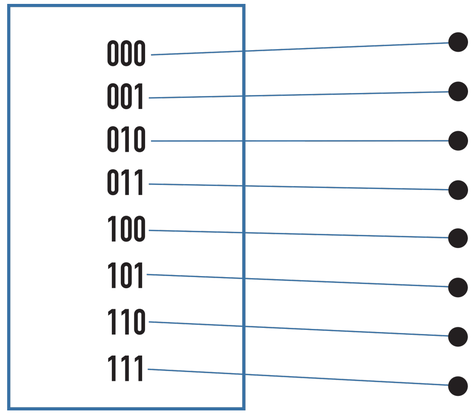
Collision

Collision

Given a list of N numbers in $[N]$, say z. $COL_N(z)$ is a partial function only defined when z is

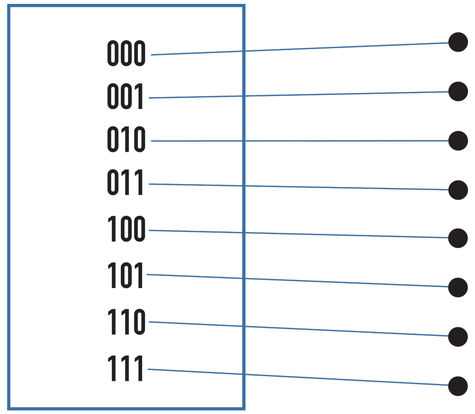
Collision

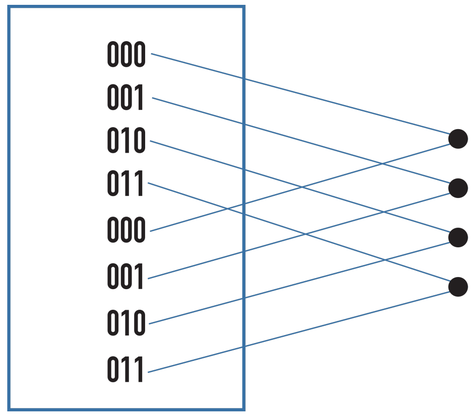
Given a list of N numbers in $[N]$, say z. $COL_N(z)$ is a partial function only defined when z is 1-1



Collision

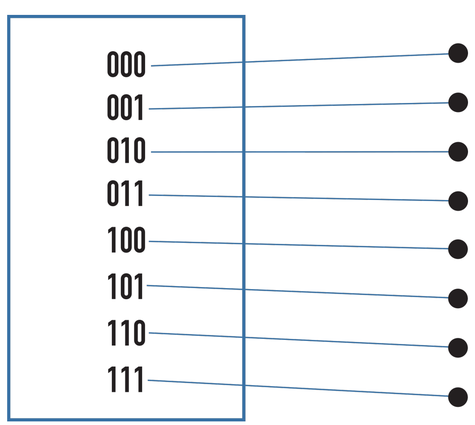
Given a list of N numbers in $[N]$, say z. $COL_N(z)$ is a partial function only defined when z is 1-1 or 2-1. Well studied, motivated by cryptanalysis.

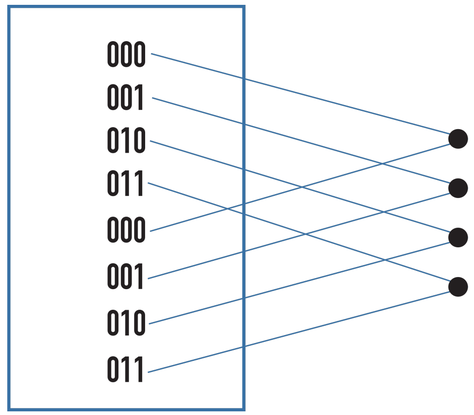




Collision

Given a list of N numbers in $[N]$, say z. $COL_N(z)$ is a partial function only defined when z is 1-1 or 2-1. Well studied, motivated by cryptanalysis.

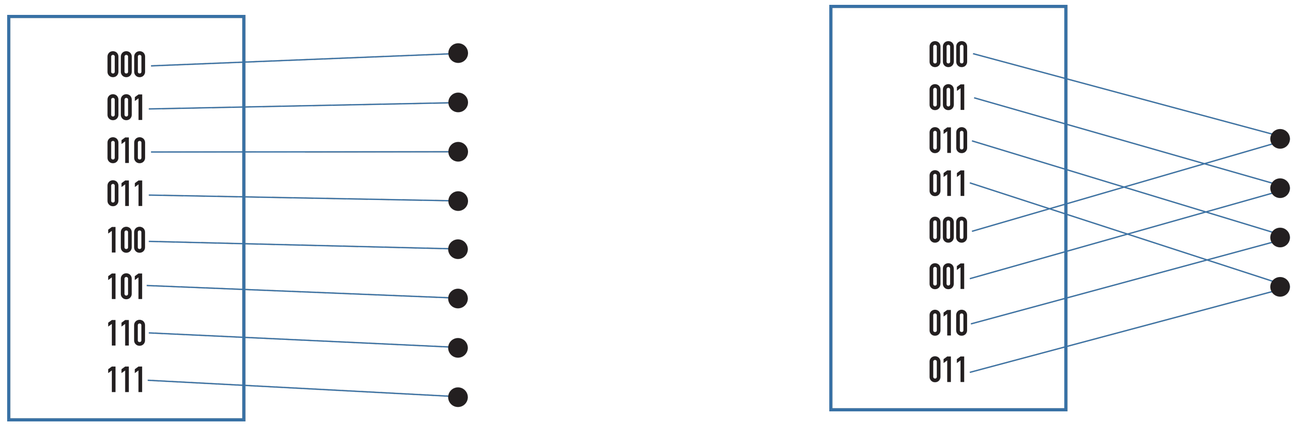




- Randomised query complexity = $\theta(\sqrt{n})$ (folklore)

Collision

Given a list of N numbers in $[N]$, say z. $COL_N(z)$ is a partial function only defined when z is 1-1 or 2-1. Well studied, motivated by cryptanalysis.



- Randomised query complexity = $\theta(\sqrt{n})$ (folklore)
- Quantum query complexity = $\theta(n^{1/3})$ [Aar02, AS04]

Bipartite Collision

Image taken from https://github.com/greeenway/alicebobandeve

Bipartite Collision

Now we split the binary encoding of each number in z to get two lists a, b.
$BICOL_N(a, b)$ is a partial function only defined when z is 1-1 or 2-1.

Bipartite Collision

Now we split the binary encoding of each number in z to get two lists a, b.
$BICOL_N(a, b)$ is a partial function only defined when z is 1-1 or 2-1.

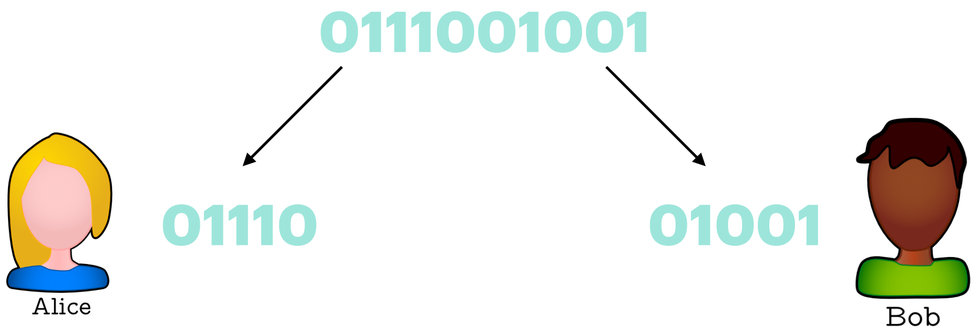
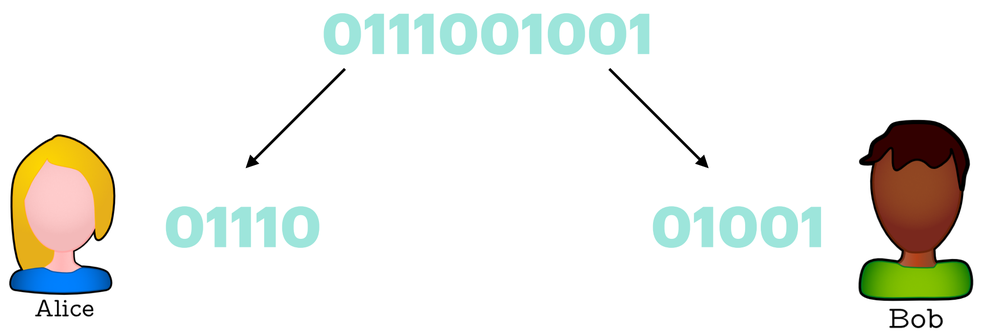


Image taken from https://github.com/greeenway/alicebobandeve

Bipartite Collision

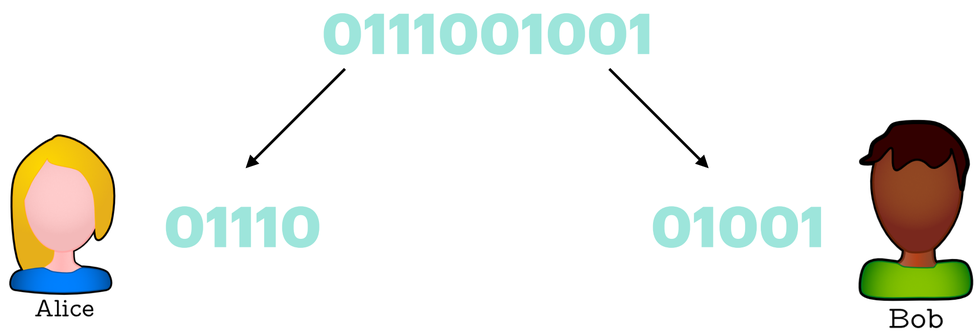
Now we split the binary encoding of each number in z to get two lists a, b. $BICOL_N(a, b)$ is a partial function only defined when z is 1-1 or 2-1.



How much communication is needed between A and B to decide $BICOL_N$?

Bipartite Collision

Now we split the binary encoding of each number in z to get two lists a, b. $BICOL_N(a, b)$ is a partial function only defined when z is 1-1 or 2-1.



How much communication is needed between A and B to decide $BICOL_N$?

Main theorem. $BICOL_N$ has randomised (and even quantum) communication complexity $\Omega(N^{1/12})$.

Image taken from https://github.com/greeenway/alicebobandeve

Communication and lifting

Communication and lifting

Technical barrier. Popular method to show communication lower bounds is *lifting.* Given f for which we know a query lower bound, we wish to compose with a small “gadget" g to create a two-party problem.

Communication and lifting

Technical barrier. Popular method to show communication lower bounds is *lifting.* Given f for which we know a query lower bound, we wish to compose with a small “gadget" g to create a two-party problem.

$$(f \circ g)(x, y) \;:=\; f(g(x_1, y_1), \ldots, g(x_n, y_n))$$

Communication and lifting

Technical barrier. Popular method to show communication lower bounds is *lifting.* Given f for which we know a query lower bound, we wish to compose with a small “gadget" g to create a two-party problem.

$$(f \circ g)(x, y) \;:=\; f(g(x_1, y_1), \ldots, g(x_n, y_n))$$

↖ artificial problem

Communication and lifting

Technical barrier. Popular method to show communication lower bounds is *lifting.* Given f for which we know a query lower bound, we wish to compose with a small "gadget" g to create a two-party problem.

$$(f \circ g)(x, y) \;:=\; f(g(x_1, y_1), \ldots, g(x_n, y_n))$$

↖ artificial problem

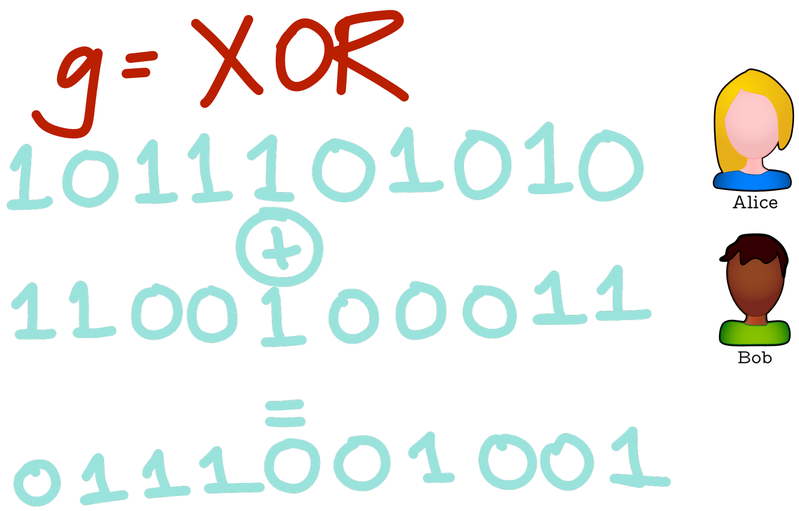
$g = \mathrm{XOR}$

Communication and lifting

Technical barrier. Popular method to show communication lower bounds is *lifting.* Given f for which we know a query lower bound, we wish to compose with a small "gadget" g to create a two-party problem.

$$(f \circ g)(x, y) \;:=\; f(g(x_1, y_1), \ldots, g(x_n, y_n))$$

↖ artificial problem



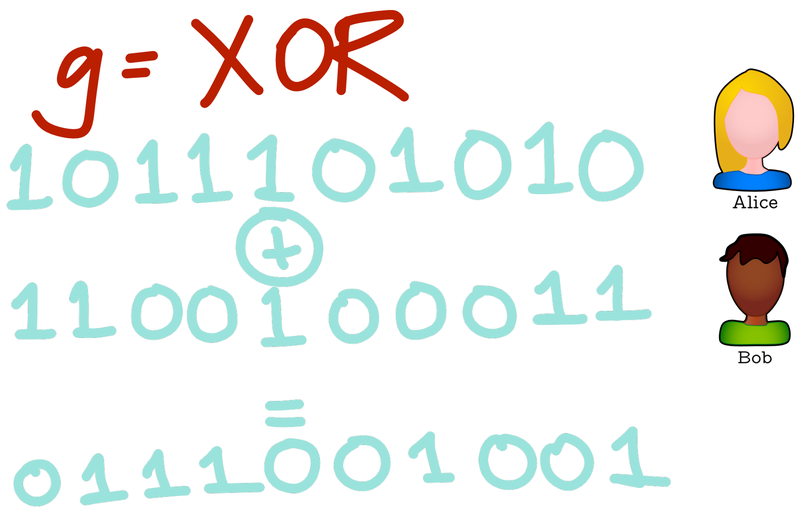
Communication and lifting

Technical barrier. Popular method to show communication lower bounds is *lifting.* Given f for which we know a query lower bound, we wish to compose with a small "gadget" g to create a two-party problem.

$$(f \circ g)(x, y) \;:=\; f(g(x_1, y_1), \ldots, g(x_n, y_n))$$

↖ artificial problem

v. s. natural problem $BICOL_N$



XOR

Image taken from https://github.com/greeenway/alicebobandeve

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]

Image taken from https://github.com/greeenway/alicebobandeve

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]

Alice

Bob

a_1	b_1
a_2	b_2
…	…
a_i	b_i
…	…
a_j	b_j
…	…
a_m	b_m

$$a_i + b_i = a_j + b_j$$

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]

a_1	b_1
a_2	b_2
…	…
a_i	b_i
…	…
a_j	b_j
…	…
a_m	b_m

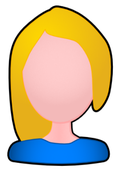
$$a_i + b_i = a_j + b_j$$

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]


Alice


Bob


Alice


Bob

a_1	b_1	$[a_1 + z]$	$[b_1 + z]$
a_2	b_2	$[a_2 + z]$	$[b_2 + z]$
…	…	…	…
a_i	b_i	$[a_i + z]$	$[b_i + z]$
…	…	…	…
a_j	b_j	$[a_j + z]$	$[b_j + z]$
…	…	…	…
a_m	b_m	$[a_m + z]$	$[b_m + z]$

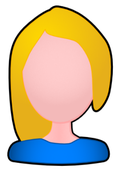
$$a_i + b_i = a_j + b_j$$

$$(a_i + z_0, b_i + z_0) = (a_j + z_1, b_j + z_1)$$

Image taken from https://github.com/greeenway/alicebobandeve

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]



$$
\begin{array}{cc}
a_1 & b_1 \\
a_2 & b_2 \\
\dots & \dots \\
a_i & b_i \\
\dots & \dots \\
a_j & b_j \\
\dots & \dots \\
a_m & b_m
\end{array}
\longrightarrow
\begin{array}{cc}
[a_1 + z] & [b_1 + z] \\
[a_2 + z] & [b_2 + z] \\
\dots & \dots \\
[a_i + z] & [b_i + z] \\
\dots & \dots \\
[a_j + z] & [b_j + z] \\
\dots & \dots \\
[a_m + z] & [b_m + z]
\end{array}
$$

$a_i + z_0 = a_j + z_1,$
$b_i + z_0 = b_j + z_1$

$\Longrightarrow$

$a_i + b_i = a_j + b_j$

z_0, z_1 unique pair!

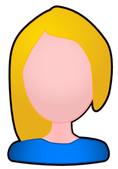
$a_i + b_i = a_j + b_j$

$(a_i + z_0, b_i + z_0)$
$= (a_j + z_1, b_j + z_1)$

Image taken from https://github.com/greeenway/alicebobandeve

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]



$$
\begin{array}{cc}
a_1 & b_1 \\
a_2 & b_2 \\
\dots & \dots \\
a_i & b_i \\
\dots & \dots \\
a_j & b_j \\
\dots & \dots \\
a_m & b_m
\end{array}
\longrightarrow
\begin{array}{cc}
[a_1+z] & [b_1+z] \\
[a_2+z] & [b_2+z] \\
\dots & \dots \\
[a_i+z] & [b_i+z] \\
\dots & \dots \\
[a_j+z] & [b_j+z] \\
\dots & \dots \\
[a_m+z] & [b_m+z]
\end{array}
$$

$a_i + z_0 = a_j + z_1,$
$b_i + z_0 = b_j + z_1$

$\Longrightarrow$

$a_i + b_i = a_j + b_j$

z_0, z_1 unique pair!

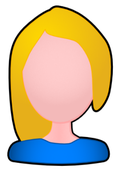
$a_i + b_i = a_j + b_j$

$(a_i + z_0, b_i + z_0) = (a_j + z_1, b_j + z_1)$

No lifting thms for XOR :(

XOR

A clever reduction for the case of XOR by Itsykson & Riazanov [IR20]



$$
\begin{array}{cc}
a_1 & b_1 \\
a_2 & b_2 \\
\dots & \dots \\
a_i & b_i \\
\dots & \dots \\
a_j & b_j \\
\dots & \dots \\
a_m & b_m
\end{array}
\longrightarrow
\begin{array}{cc}
[a_1+z] & [b_1+z] \\
[a_2+z] & [b_2+z] \\
\dots & \dots \\
[a_i+z] & [b_i+z] \\
\dots & \dots \\
[a_j+z] & [b_j+z] \\
\dots & \dots \\
[a_m+z] & [b_m+z]
\end{array}
$$

$a_i + z_0 = a_j + z_1,$
$b_i + z_0 = b_j + z_1$

$\Longrightarrow$

$a_i + b_i = a_j + b_j$

z_0, z_1 unique pair!

$a_i + b_i = a_j + b_j$

$(a_i + z_0, b_i + z_0) = (a_j + z_1, b_j + z_1)$

No lifting thms for XOR :(

CLAIM

Only using the "regularity" of XOR

Image taken from https://github.com/greeenway/alicebobandeve

Regular gadgets

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

Regular functions. A bipartite function is said to be regular if there is a group acting on its domain such that:

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

Regular functions. A bipartite function is said to be regular if there is a group acting on its domain such that:

- The orbit of any $(x, y) \in f^{-1}(b)$ is exactly the pre-image $f^{-1}(b)$.

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

Regular functions. A bipartite function is said to be regular if there is a group acting on its domain such that:

- The orbit of any $(x, y) \in f^{-1}(b)$ is exactly the pre-image $f^{-1}(b)$.
- For any two (possibly equal) elements of the set, there is a unique group element taking the first to the second.

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

XOR is Regular

Regular functions. A bipartite function is said to be regular group acting on its domain such that:

- The orbit of any $(x, y) \in f^{-1}(b)$ is exactly the pre-image $f^{-1}(b)$.
- For any two (possibly equal) elements of the set, there is a unique group element taking the first to the second.

Regular gadgets

Main contribution. If g is a constant-sized *regular function* $\Longrightarrow$ $COL \circ g \leq BICOL$ with some polynomial blowup.

Regular functions. A bipartite function is said to be regular group acting on its domain such that:

- The orbit of any $(x, y) \in f^{-1}(b)$ is exactly the pre-image $f^{-1}(b)$.
- For any two (possibly equal) elements of the set, there is a unique group element taking the first to the second.

XOR is Regular

$(x, y) \mapsto (x, y)$
$(x, y) \mapsto (\neg x, \neg y)$

VER

VER

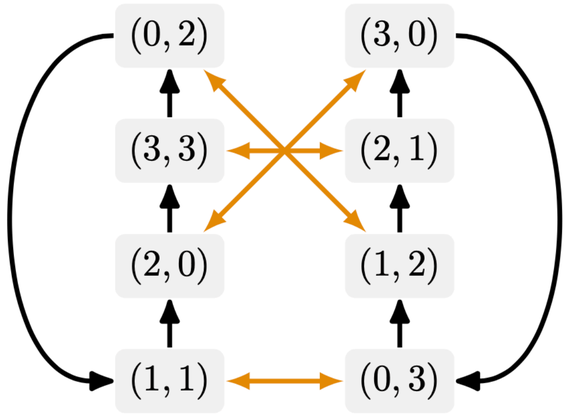
Bonus! Sherstov [She11] proved that approx degree lifts to approx rank with VER. We note that VER is a regular function! Proof by picture.

VER

Bonus! Sherstov [She11] proved that approx degree lifts to approx rank with VER. We note that VER is a regular function! Proof by picture.

	0	**1**	**2**	**3**
0	0	0	1	1
1	0	1	1	0
2	1	1	0	0
3	1	0	0	1

(a)



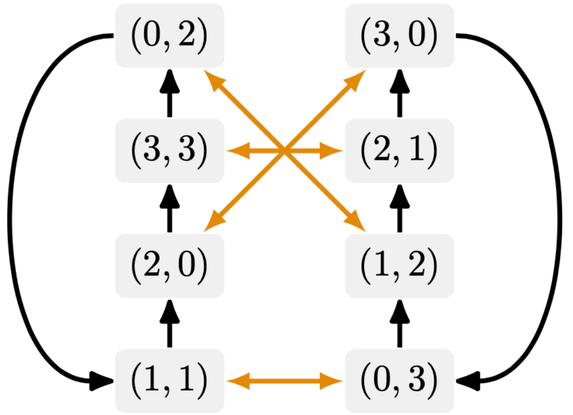
(b)

VER

Bonus! Sherstov [She11] proved that approx degree lifts to approx rank with VER. We note that VER is a regular function! Proof by picture.

	0	**1**	**2**	**3**
0	0	0	1	1
1	0	1	1	0
2	1	1	0	0
3	1	0	0	1

(a)



(b)

$VER : \mathbb{Z}_4 \times \mathbb{Z}_4 \mapsto \{0,1\}$

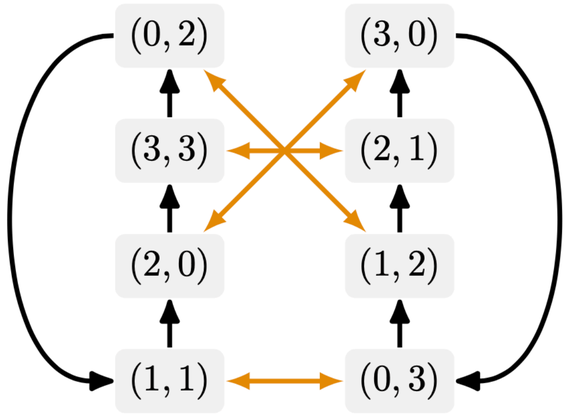
Generators on $VER^{-1}(1)$

VER

Bonus! Sherstov [She11] proved that approx degree lifts to approx rank with VER. We note that VER is a regular function! Proof by picture.

	0	**1**	**2**	**3**
0	0	0	1	1
1	0	1	1	0
2	1	1	0	0
3	1	0	0	1

(a)



(b)

$VER : \mathbb{Z}_4 \times \mathbb{Z}_4 \mapsto \{0,1\}$

Generators on $VER^{-1}(1)$

(black) $(x, y) \mapsto (x + 1, y - 1)$

(orange) $(x, y) \mapsto (1 - x, -y)$

Application

Application

Proof complexity. We show a similar lower bound for the *search* problem of natural bipartite analogue for the Pigeonhole Principle.

Application

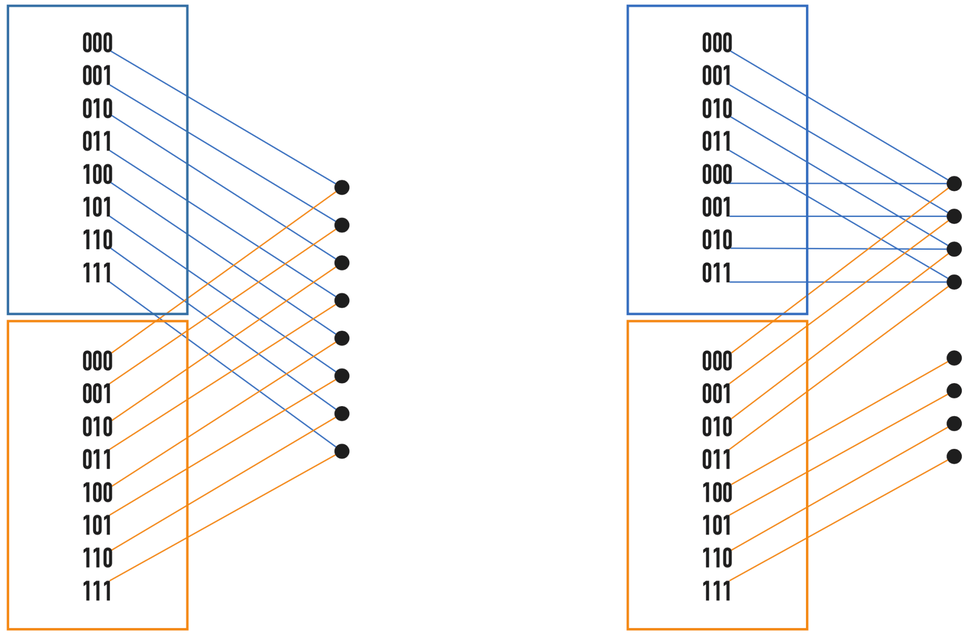
Proof complexity. We show a similar lower bound for the *search* problem of natural bipartite analogue for the Pigeonhole Principle.

We do this by a simple *search-to-decision* reduction, which involves appending a 1-1 list to our input. We illustrate the property below.

Application

Proof complexity. We show a similar lower bound for the *search* problem of natural bipartite analogue for the Pigeonhole Principle.

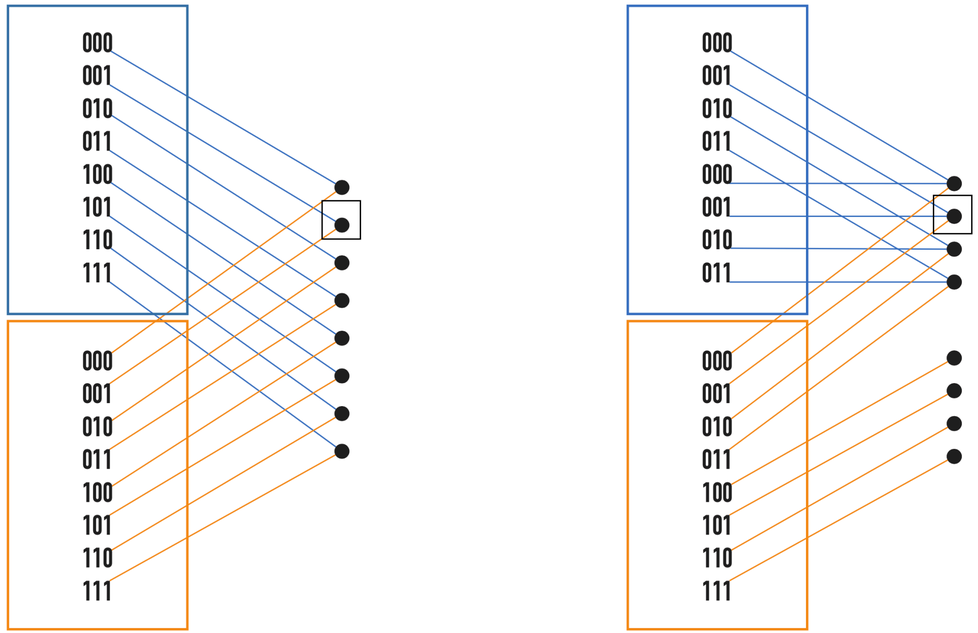
We do this by a simple *search-to-decision* reduction, which involves appending a 1-1 list to our input. We illustrate the property below.



Application

Proof complexity. We show a similar lower bound for the *search* problem of natural bipartite analogue for the Pigeonhole Principle.

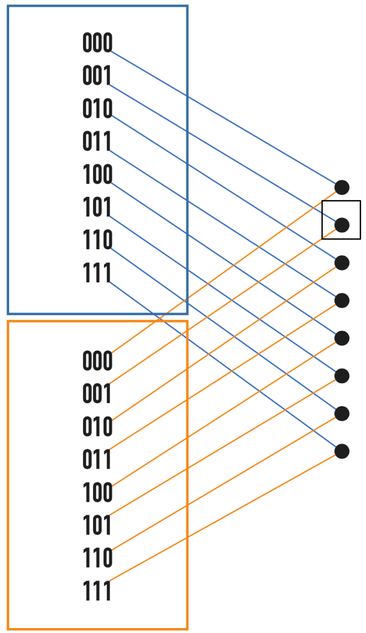
We do this by a simple *search-to-decision* reduction, which involves appending a 1-1 list to our input. We illustrate the property below.

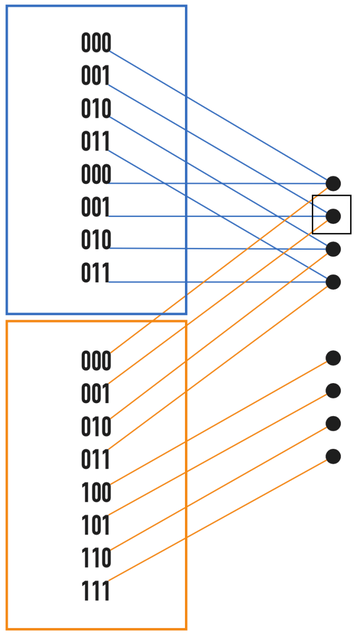


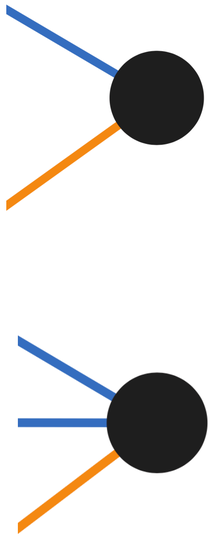
Application

Proof complexity. We show a similar lower bound for the *search* problem of natural bipartite analogue for the Pigeonhole Principle.

We do this by a simple *search-to-decision* reduction, which involves appending a 1-1 list to our input. We illustrate the property below.







Thanks for listening!

Au revoir