Oblivious Computation

Part II - Oblivious Sorts

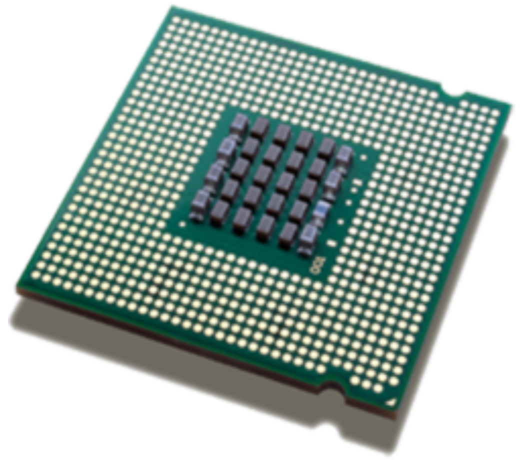
Gilad Asharov

Bar-Ilan University

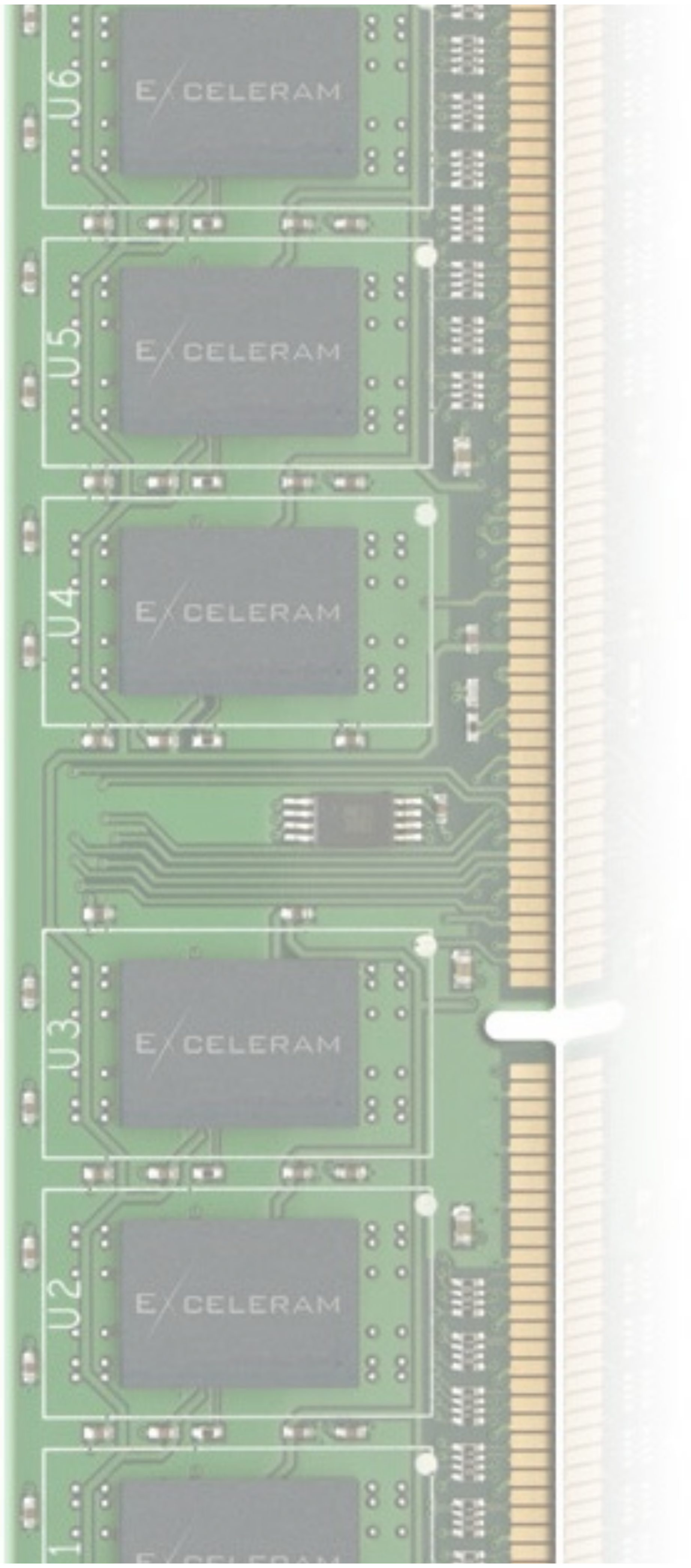
The 12th Bar-Ilan Winter School on Cryptography
Advances in Secure Computation

Center for Research in Applied
Cryptography and Cyber Security

Access Patterns Reveal Information!

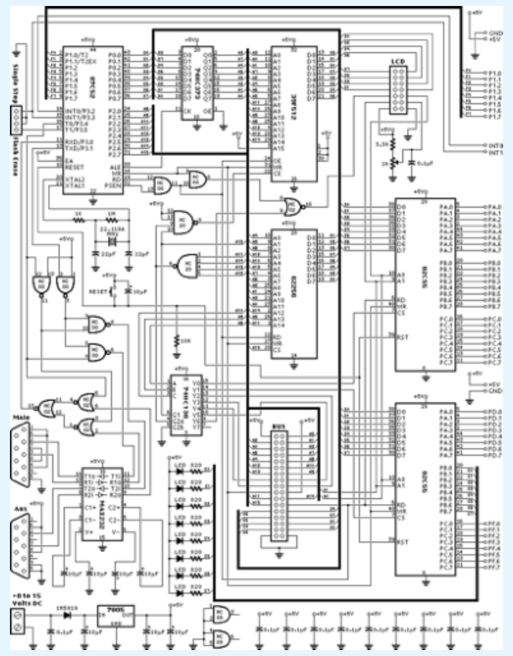


secure processor



Models of Computation

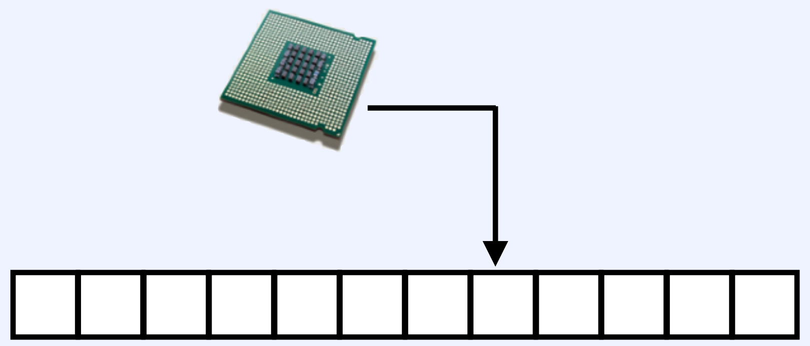
Circuits



Emulate easily →

← $T^3 \log T$
[CR73,PF79]
$(T \gg N)$

RAM Model



Random Access Machine

CPU Operation

Memory access

Metrics:

Size (how many wires, gates)
Depth (parallelism)

Time T
Size of the memory N

Oblivious RAM Compiler: State of the Art

Lower bound: $\Omega(\log N)$

[GoldreichOstrovsky'96, LarsenNeilsen'18]

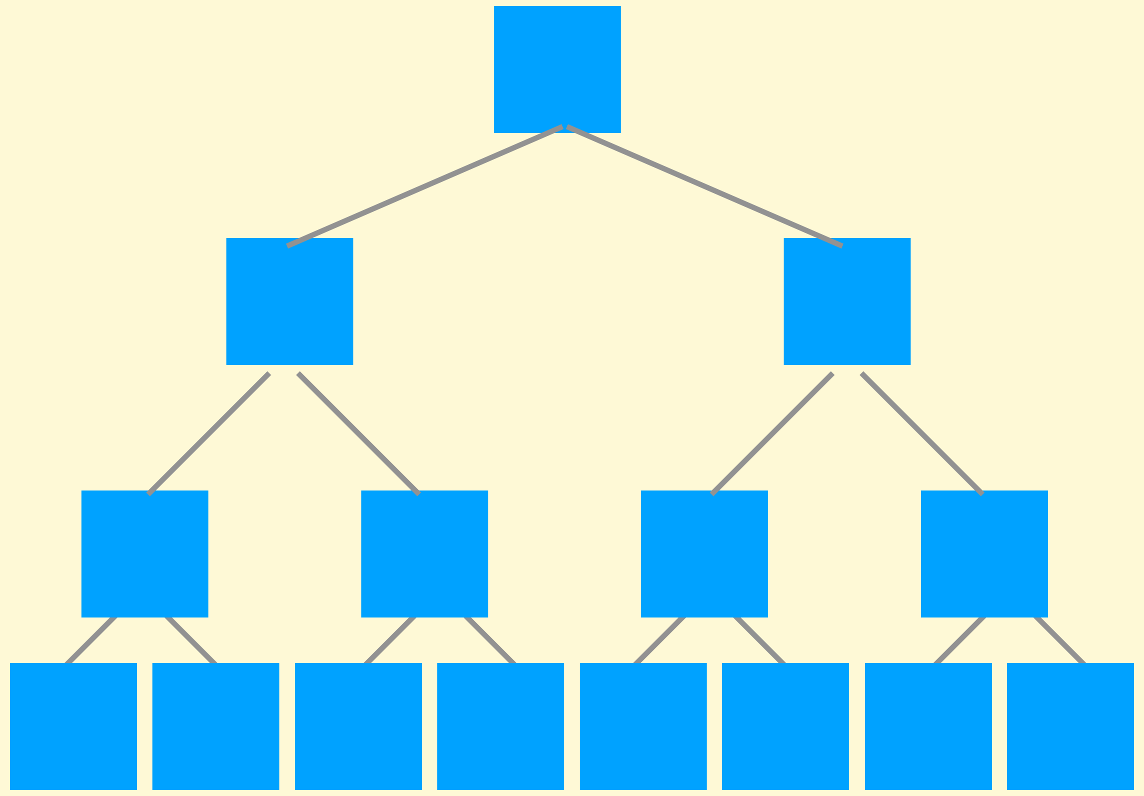
Hierarchical

[O90,GO96]

$$O(\log N)$$

Computational security

[OptORAMa'20]



Tree based ORAM

[Shi,Chan,Stefanov11]

$$O(\log^2 N)$$

Statistical security

[PathORAM,CircuitORAM]

Example: Oblivious Sorts

- **Merge sort:** $O(n \log n)$
  - non oblivious
- **Bubble sort:** $O(n^2)$
  - oblivious
- **Other oblivious sorts?**

Merge((1,2,3),(4,5,6)**)**

1,2,3 4,5,6
1,2,3 4,5,6
1,2,3 4,5,6
1,2,3 4,5,6
1,2,3 4,5,6
1,2,3 4,5,6

Merge((1,3,5),(2,4,6)**)**

1,3,5 2,4,6
1,3,5 2,4,6
1,3,5 2,4,6
1,3,5 2,4,6
1,3,5 2,4,6
1,3,5 2,4,6

BubbleSort(1,2,3,4**)**

1,2,3,4
1,2,3,4
1,2,3,4
1,2,3,4
1,2,3,4
1,2,3,4
1,2,3,4

BubbleSort(4,3,2,1**)**

4,3,2,1
3,4,2,1
3,2,4,1
3,2,1,4
2,3,1,4
2,1,3,4
1,2,3,4

Oblivious Sorts

In the RAM Model

Algorithm		Oblivious	Client Storage	Runtime	
Merge sort	[vonNeumann'45]	No	$O(1)$	$2n \log n$	
Bitonic sort	[Batcher'68]	Yes	$O(1)$	$n \log^2 n$	0
AKS sort	[AKS'83]	Yes	$O(1)$	$5.4 \times 10^7 \times n \log n$	0
Zig-zag sort	[Goodrich'14]	Yes	$O(1)$	$8 \times 10^4 \times n \log n$	0

Circuit Model!

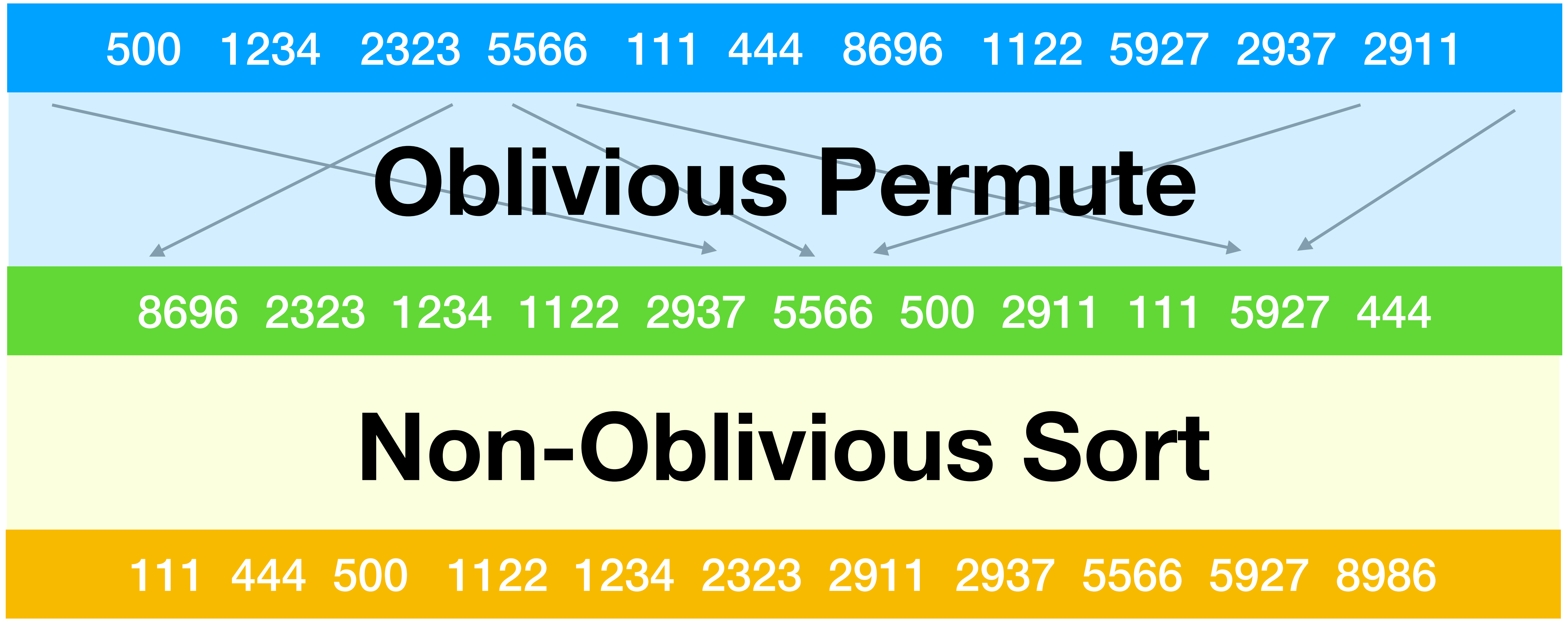
Most practical

*constants of AKS, Zig-zag are from [Goodrich'14]
Z: poly log k

For n>2^{30}
x5 faster than Bitonic

BIU Center for Research in Applied Cryptography and Cyber Security

Bucket Oblivious Sort

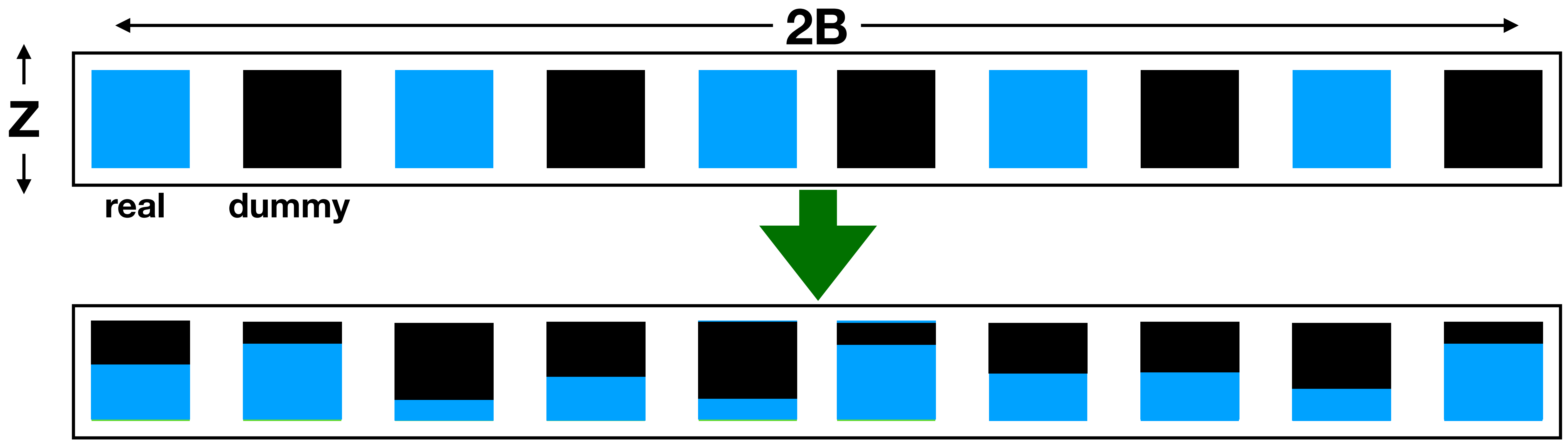


Oblivious Permute + Non-Oblivious Sort = Oblivious Sort

? **Merge-Sort**

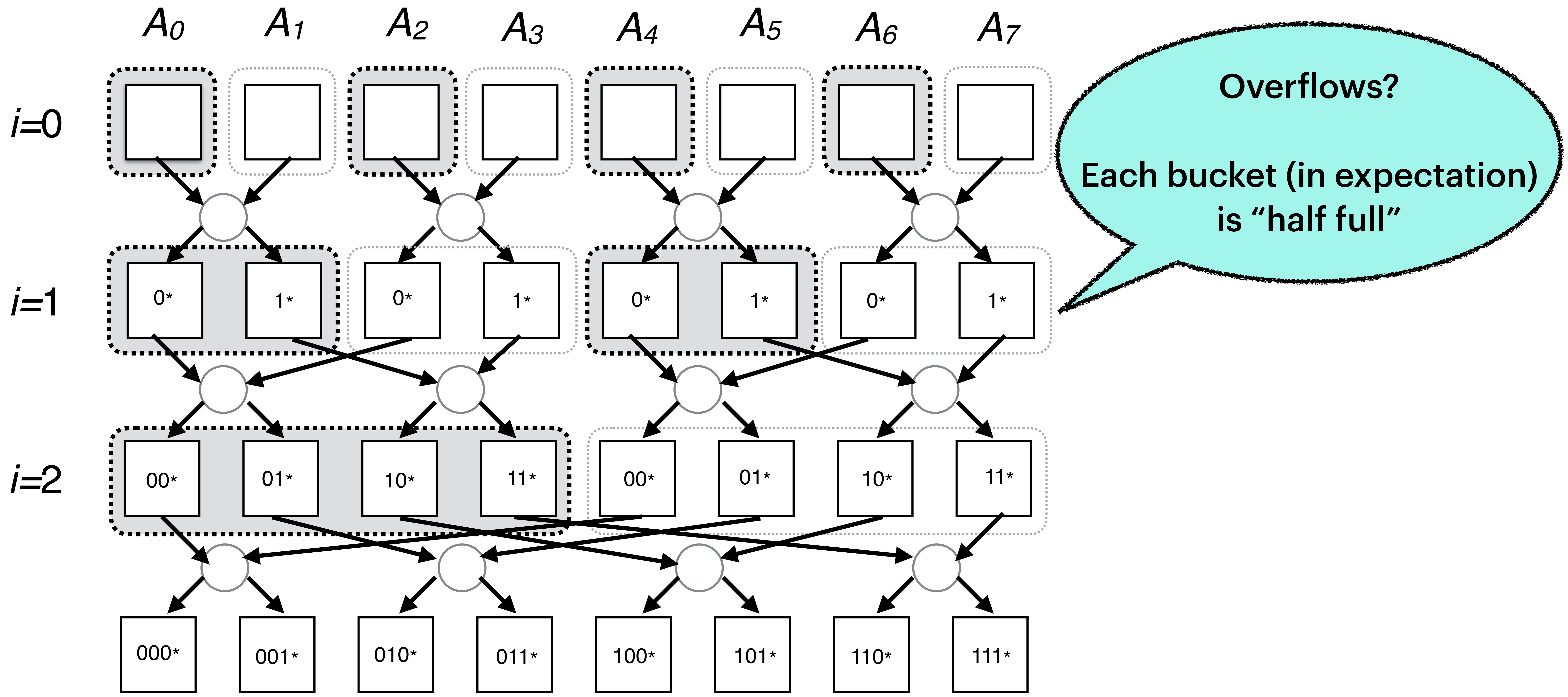
Bucket Oblivious Permute

- Interpret the input array as **B** buckets of size **Z** each
  (**Z**=**poly** log k, **B**=N/**Z**, k is the security parameter)
  - Assign to each element a *random* destination bin [1,…,B]
  - Add dummy bins



(We later remove these dummy elements using the non-oblivious sort)

Bucket Oblivious Permute



MergeSplit

Bucket

MergeSplit - takes all read elements in input buckets and distribute them to output buckets according to the i^{th} MSB

Oblivious Sorts

In the RAM Model

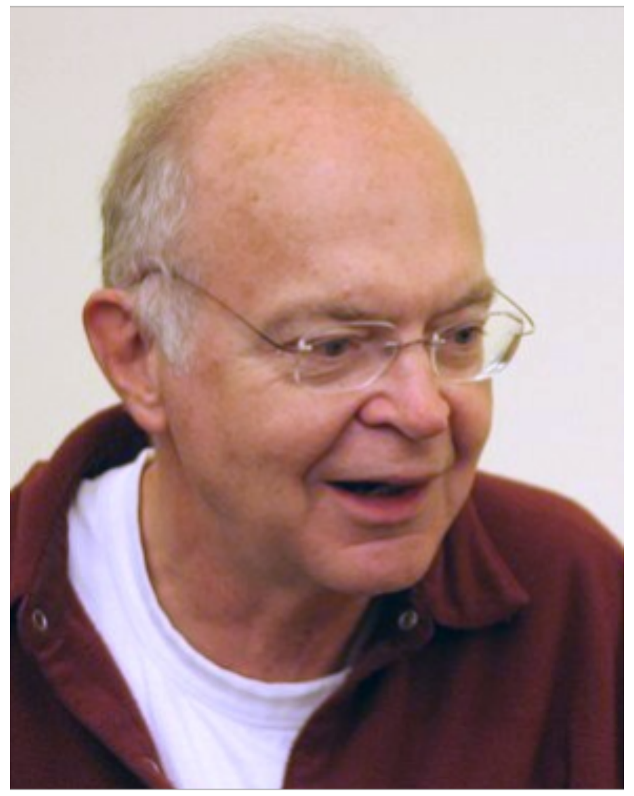
Algorithm		Oblivious	Client Storage	Runtime	Error Probability
Merge sort	[vonNeumann'45]	No	$O(1)$	$2n \log n$	0
Bitonic sort	[Batcher'68]	Yes	$O(1)$	$n \log^2 n$	0
AKS sort	[AKS'83]	Yes	$O(1)$	$5.4 \times 10^7 \times n \log n$	0
Zig-zag sort	[Goodrich'14]	Yes	$O(1)$	$8 \times 10^4 \times n \log n$	0
Bucket oblivious sort	[**A**CNPRS'20]	Yes	$2Z$	$6n \log n$	$\approx e^{-Z/6}$
Bucket oblivious sort	[**A**CNPRS'20]	Yes	$O(1)$	$\approx 2n \log n \log^2 Z$	$\approx e^{-Z/6}$

*constants of AKS, Zig-zag are from [Goodrich'14]

Center for Research in Applied Cryptography and Cyber Security

(Oblivious) Sorting Faster Than $O(n \log n)$?

(Oblivious) **Sorting Faster than $O(n \log n)$?**



"**No!** Such a result is **not** possible with **comparison-based**"

Knuth73: The Art of Computer Programming

RAM Model:

Non-comparison based sorts? (k - length of the key)

Radix-sort $O(k \cdot n)$, counting sort $O(2^k + n)$

Circuit Model:

Radix sort, counting sort - make input-dependent memory accesses
Do not have equally efficient counterparts in the circuit model

n: number of elements
k: length of each key (#bits)
w: payload (#bits)

What about the Circuit Model?

Can we go lower than $(k + w) \cdot \Omega(n \log n)$ circuit-size?

Comparator based?

- Any comparator-based sorting circuit must consume $\Omega(n \log n)$ comparators
- Even for $k = 1$ long key!

Stability?

- Stable sort requires $\Omega(n \log n)$ *selector* gates in the *indivisible* model, even for $k = 1$ [Lin,Shi,Xie19]

Assuming a well-known network conjecture,
sorting circuits of size $(k + w) \cdot o(n \log n)$ **do not exist** for general k
[Afshani, Freksen, Kamma, Larsen19]

No unconditional lower bound is known

Center for Research in Applied Cryptography and Cyber Security

n: number of elements

k: length of each key (#bits)

w: payload (#bits)

AKS'83 $(k + w) \cdot O(n \log n)$

Better for $k \in o(\log n)$

ALS'21 $(k + w) \cdot O(nk)$ *ignoring polylog* terms

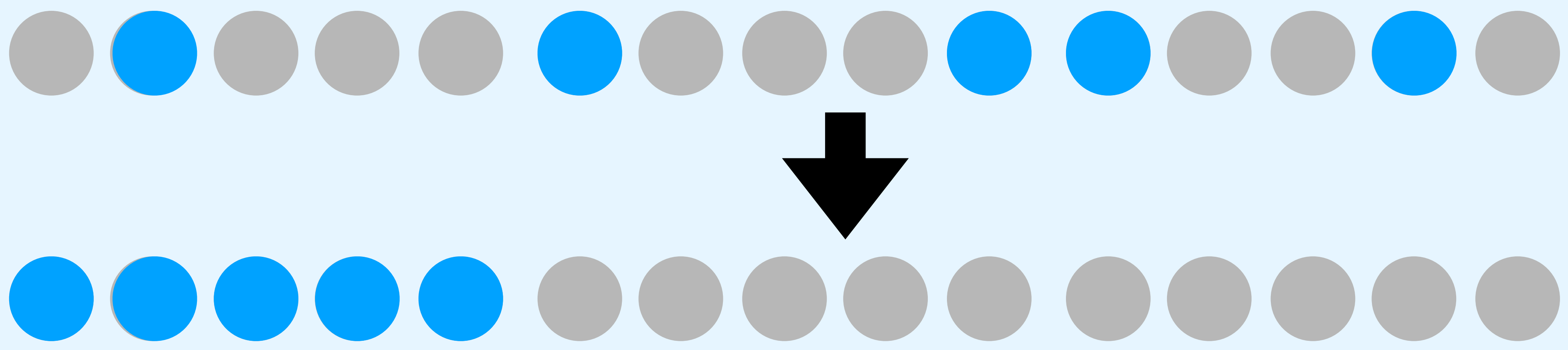
Not Comparison Based!

Not Stable!

Indeed worse than $n \log n$ for general k

Tight Compaction
A Central Problem!

Tight Compaction



Circuit Model

Linear size circuit
(Ignoring polylog* factors)

(Almost) Linear size sorting circuit
(short keys) [**A**LS'21]

RAM Model

Linear time oblivious compaction

Optimal Oblivious RAM compiler
[OptORAMa,**A**KLNPS'20]

Center for Research in Applied Cryptography and Cyber Security

Tight Compaction

- **Input**: An array of size n where each element is marked 0 or 1
- **Output**: all 0-elements appear before 1-elements

- RAM model? Trivial in O(n)

- Oblivious RAM model?
  - Deterministic: $O(n \log n)$ [AKS'83]
  - Open question from [LeightonMaSuel'95]
    - Reveals the number of 0's, randomized
    - Randomized: $O(n \log \log n)$ [MZ'14, LST'18] (negl error prob.)
  - lower bound: stable, indivisible, $\Omega(n \log n)$ [LST'18]

- OptORAMa [AsharovKomargodskiLinNayakPesericoShi20]: Deterministic, $O(n)$, very large constant
  - Better constant [DittmerOstrovksy20]
- $O(n)$ work, depth $O(\log n)$ [AsharovKomargodskiLinPesericoShi20]

Circuit Model

Linear size circuit

(Ignoring polylog* factors)

$O(nw)$*poly $\log^* n$ size circuit

Compacting n balls of size w each

RAM Model

Linear time oblivious compaction

$O(n)$ time **deterministic** algorithm for compacting n balls, each of size word-size

(Assuming word size $O(\log N)$)

Not comparison based

Not stable

Balls and bins model

Metadata is computed using bit-slicing tricks

Next...

1) **Simple, randomized oblivious tight compaction (RAM model) in $O(n \log \log k)$**

   Lin, Shi, Xie: *Can we Overcome the $n \log n$ Barrier for Oblivious Sorting?* SODA'19

2) **Linear size compaction circuit**

   Asharov, Lin, Shi: *Sorting Short Keys in Circuits of Size $o(n \log n)$*, SODA'21

3) **Linear time algorithm for oblivious tight compaction in the RAM model**

   Asharov, Komargodski, Lin, Nayak, Peserico and Shi: OptORAMa, Optimal Oblivious RAM, EUROCRYPT'20

Simple Oblivious Tight Compaction

[Lin,Shi,Xie, SODA'19]

0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1

1) Interpret the array as n/Z bins of size Z each

Simple Oblivious Tight Compaction

[Lin,Shi,Xie, SODA'19]

0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 1

1) Interpret the array as n/Z bins of size Z each

0 1 1 0 0 1 0
1 0 0 1 1 0 0
0 1 0 0 1 0 1
1 0 0 1 0 1 0
1 0 1 0 1 1 1

$Z \in \text{polylog}(k) = \log^6 k$

2) Random shift cycle for each row

Simple Oblivious Tight Compaction
[Lin,Shi,Xie, SODA'19]

0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 1

1) Interpret the array as n/Z bins of size Z each

0 1 1 0 | 0 1 0
1 | 0 0 1 1 0 0
0 1 0 0 1 0 | 1
1 0 | 0 1 0 1 0
1 0 1 | 0 1 1 1

$Z \in \text{polylog}(k) = \log^6 k$

2) For each row, perform random shift cycle

0 1 0 0 1 1 0
0 0 1 1 0 0 1
1 0 1 0 0 1 0
0 1 0 1 0 1 0
0 1 1 1 1 0 1

$O(n)$

Simple Oblivious Tight Compaction

[Lin,Shi,Xie, SODA'19]

0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 1

1) Interpret the array as n/Z bins of size Z each

$Z = \log^6 k$

2) For each row, perform random shift cycle

3) Sort each **column** independently

$$\frac{n}{Z} \cdot Z \log^2 Z = O(n \log^2 Z) \in O(n \log \log k)$$

0	1	0	0	1	1	0
0	0	1	1	0	0	1
1	0	1	0	0	1	0
0	1	0	1	0	1	0
0	1	1	1	1	0	1

1	1	1	1	1	1	1
0	1	1	1	1	1	1
0	1	1	1	0	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Simple Oblivious Tight Compaction
[Lin,Shi,Xie, SODA'19]

Claim:

W.h.p, there exists a mixed stripe of size $Z/\log^2 k = \log^4 k$ rows

$$Z = \log^6 k \quad \begin{array}{c|ccccccc} & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \hline \log^4 k & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ \hline & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$n/Z = n/\log^6 k$

1) Interpret the array as n/Z bins of size Z each

2) For each row, perform random shift cycle

3) Sort each **column** independently

4) copy the mixed stripe to some working array of size $\dfrac{n}{\log^6 k} \cdot \log^4 k = n/\log^2 k$

5) Sort (obliviously!) the mixed stripe; write it back

0 1 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 0 1 0 1 1 1

$$\begin{matrix}0&1&1&0&0&1&0\\1&0&0&1&1&0&0\\0&1&0&0&1&0&1\\1&0&0&1&0&1&0\\1&0&1&0&1&1&1\end{matrix} \rightarrow \begin{matrix}0&1&1&0&0&1&0\\1&0&0&1&1&0&0\\0&1&0&0&1&0&1\\1&0&0&1&0&1&0\\1&0&1&0&1&1&1\end{matrix} \rightarrow \begin{matrix}0&1&0&0&1&1&0\\0&0&1&1&0&0&1\\1&0&1&0&0&1&0\\0&1&0&1&0&1&0\\0&1&1&1&1&0&1\end{matrix} \rightarrow \begin{matrix}1&1&1&1&1&1&1\\0&1&1&1&1&1&1\\0&1&1&1&0&1&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}$$

$$\begin{matrix}1&1&1&1&1&1&1\\ \hline 0&1&1&1&1&1&1\\0&1&1&1&0&1&0\\ \hline 0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix} \rightarrow \begin{matrix}0&1&1&1&1&1&1\\0&1&1&1&0&1&0\end{matrix} \rightarrow \begin{matrix}1&1&1&1&1&1&1\\0&0&0&0&1&1&1\end{matrix} \rightarrow \begin{matrix}1&1&1&1&1&1&1\\ \hline 1&1&1&1&1&1&1\\0&0&0&0&1&1&1\\ \hline 0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{matrix}$$

1) Interpret the array as n/Z bins of size Z each

2) For each row, perform random shift cycle

3) Sort each **column** independently

4) Copy the mixed stripe to some working array of size $\frac{n}{\log^6 k} \cdot \log^4 k = n/\log^2 k$

5) Sort (obliviously!) the mixed stripe; write it back

When $n \in O(k)$, the algorithm is statistically secure and runs in $O(n \log \log n)$

Next…

1) **Simple, randomized oblivious tight compaction (RAM model) in $O(n \log \log k)$**

    Lin, Shi, Xie: *Can we Overcome the $n \log n$ Barrier for Oblivious Sorting?* SODA'19

2) **Linear size compaction circuit**

    Asharov, Lin, Shi: *Sorting Short Keys in Circuits of Size $o(n \log n)$*, SODA'21

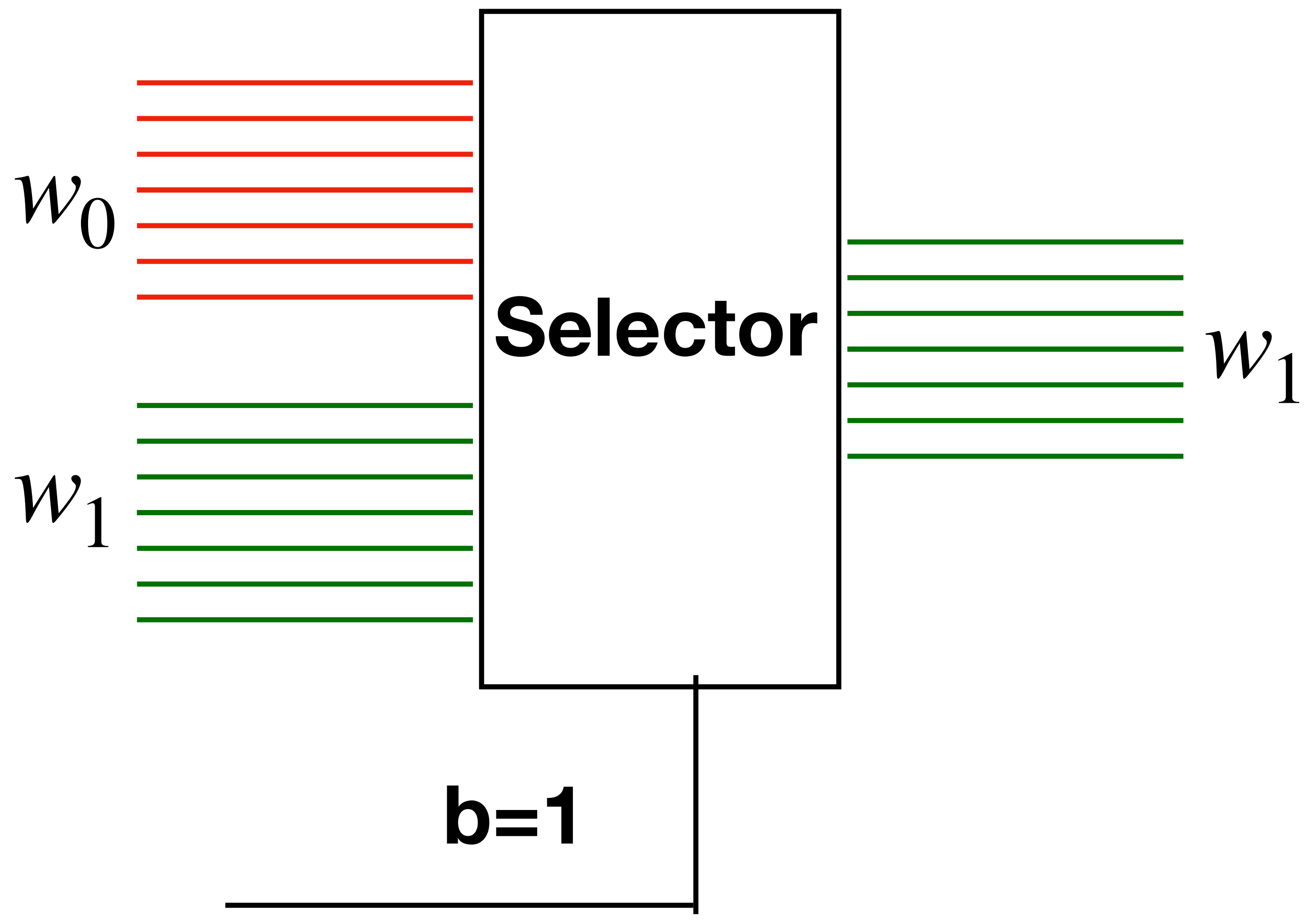
3) **Linear time algorithm for oblivious tight compaction in the RAM model**

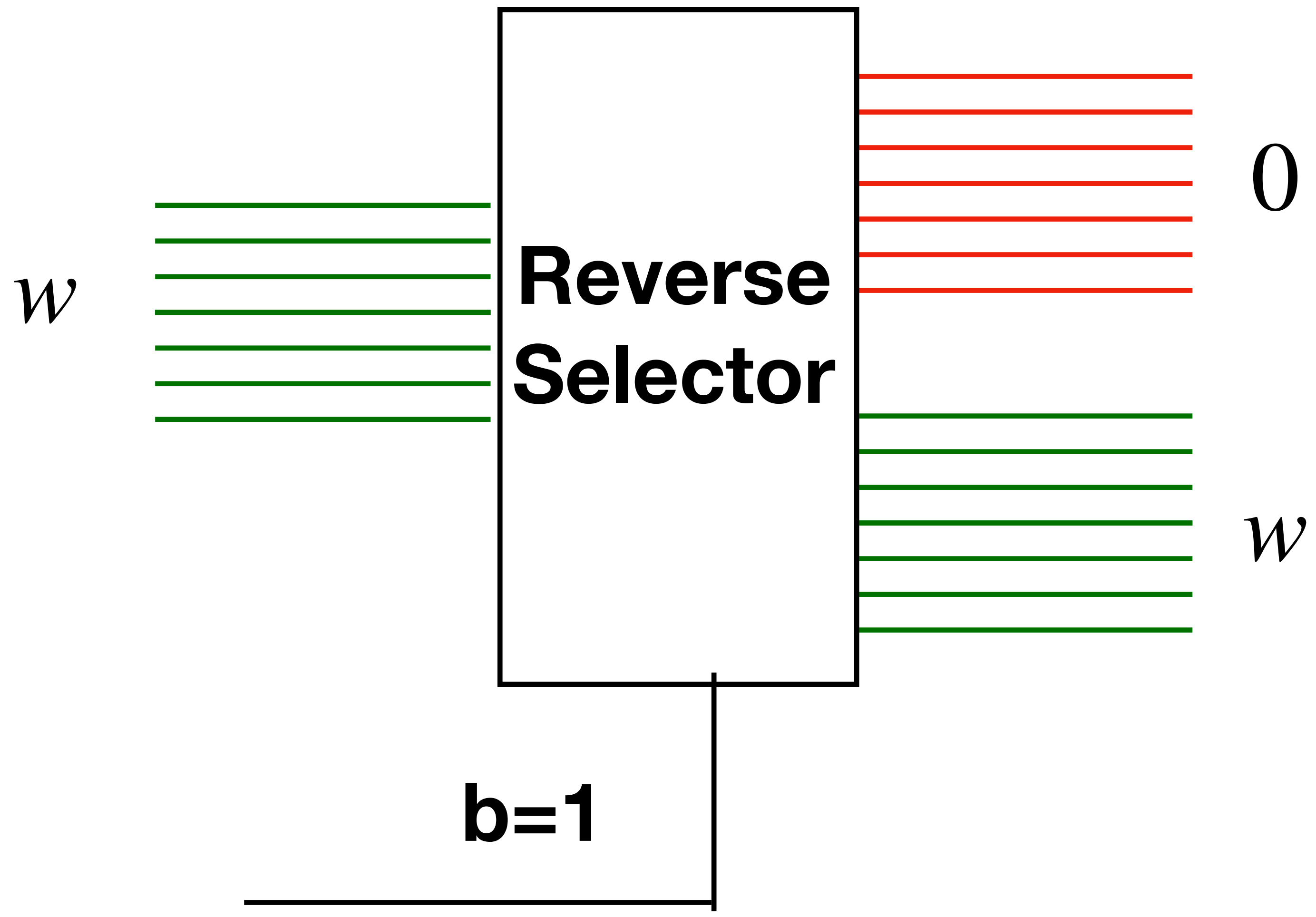
    Asharov, Komargodski, Lin, Nayak, Peserico and Shi: OptORAMa, Optimal Oblivious RAM, EUROCRYPT'20

Our Cost Model

Generalized Boolean gates

w-selector gates





“Reverse” selector

Oblivious Tight Compaction

- **Input**: An array of size n where each element is marked 0 or 1
- **Output**: all 0-elements appear before 1-elements

0	1	1	0	0	1	0	1	0	0	1	1	0	0	0	1

(1) Count the number of balls marked 0

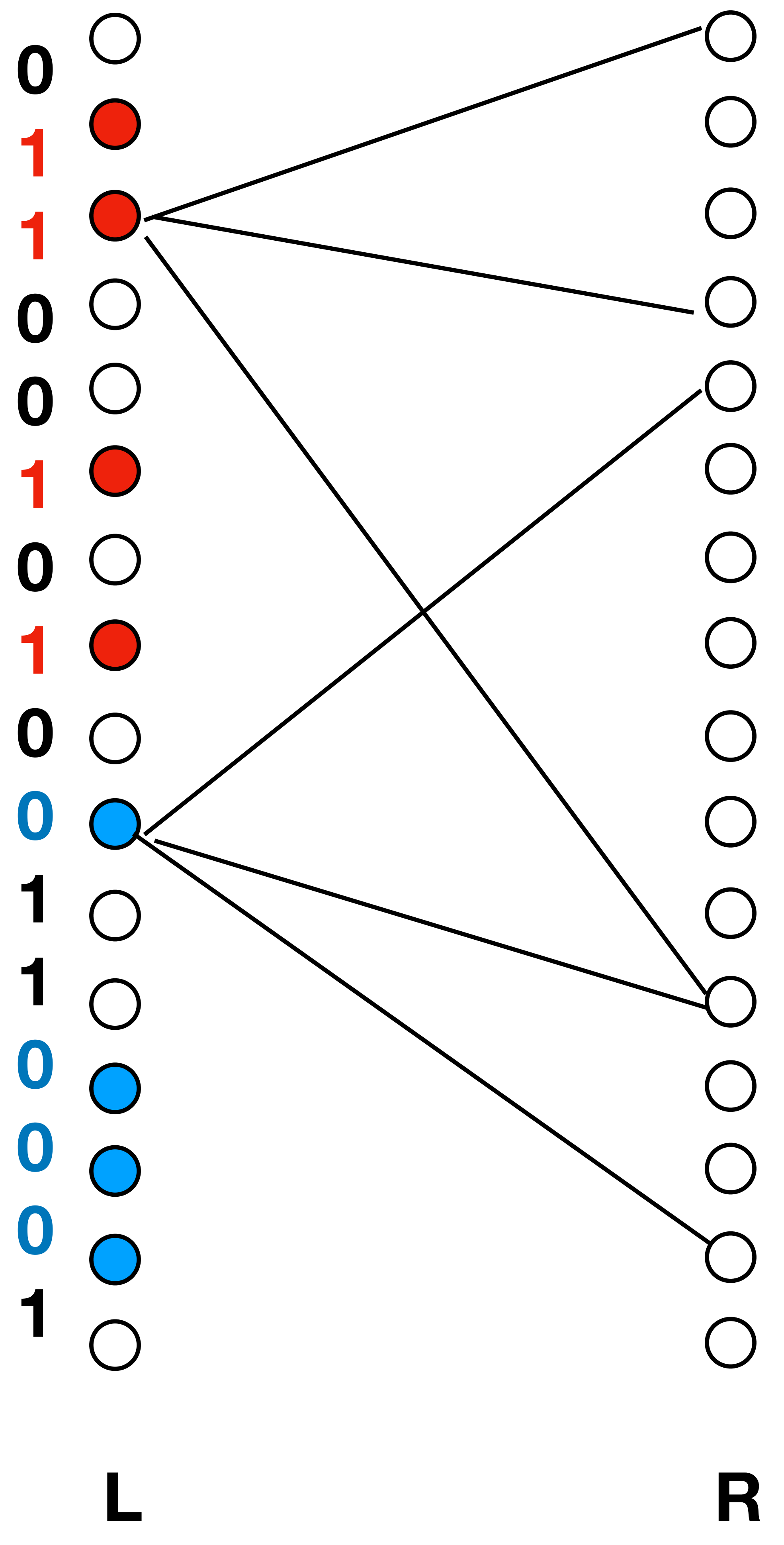
0	1	1	0	0	1	0	1	0	0	1	1	0	0	0	1

(2) Mark the elements that are “misplaced”

0	1	1	0	0	1	0	1	0	0	1	1	0	0	0	1

Observation: number of reds always equals number of blues
We just have to swap them!

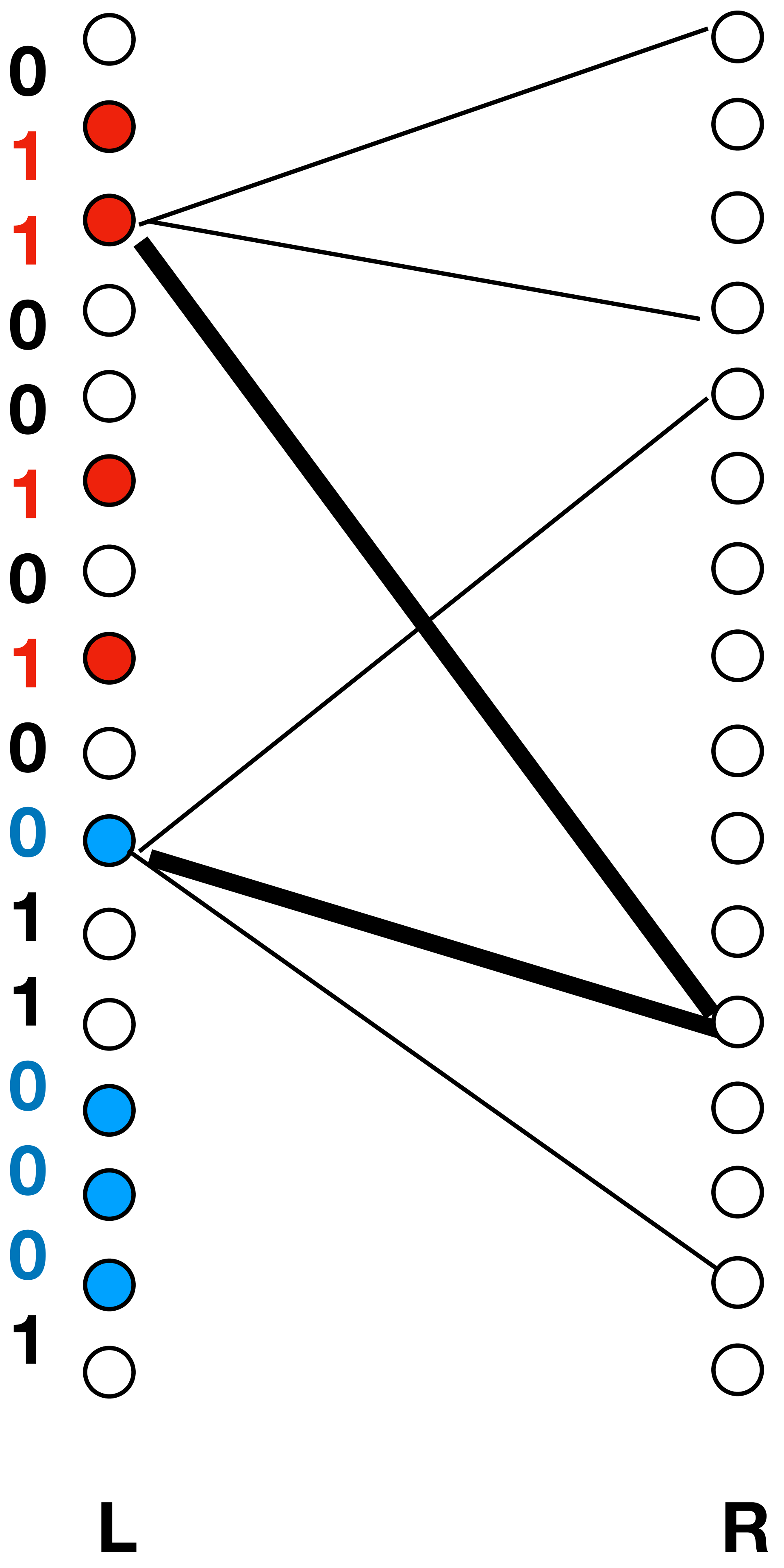
Loose Swap



Bipartite Expander Graph

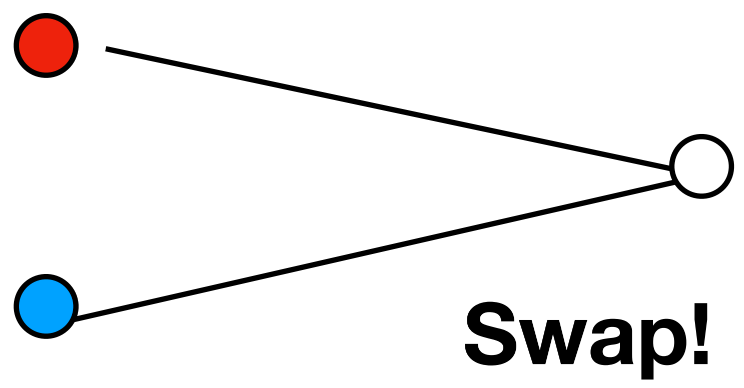
- $O(1)$ degree
- Constant spectral expansion

Loose Swap



- 1 that wants to swap with 0
- 0 that wants to swap with 1

For every pair of



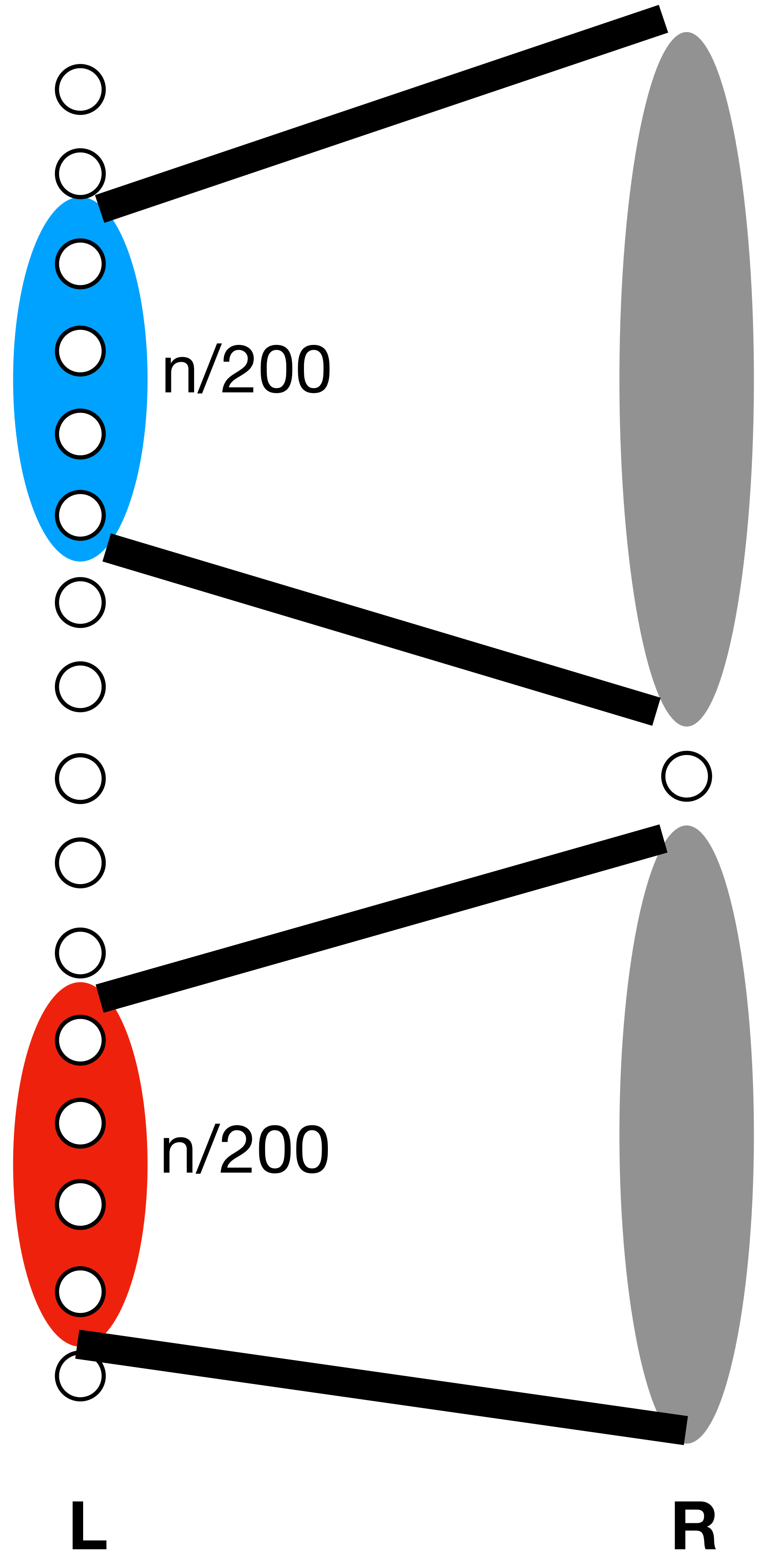
$O(nd^2)$ **boolean gates**
$O(n \cdot d^2)$ w**-selector gates**

$O(nw)$ **gates total**

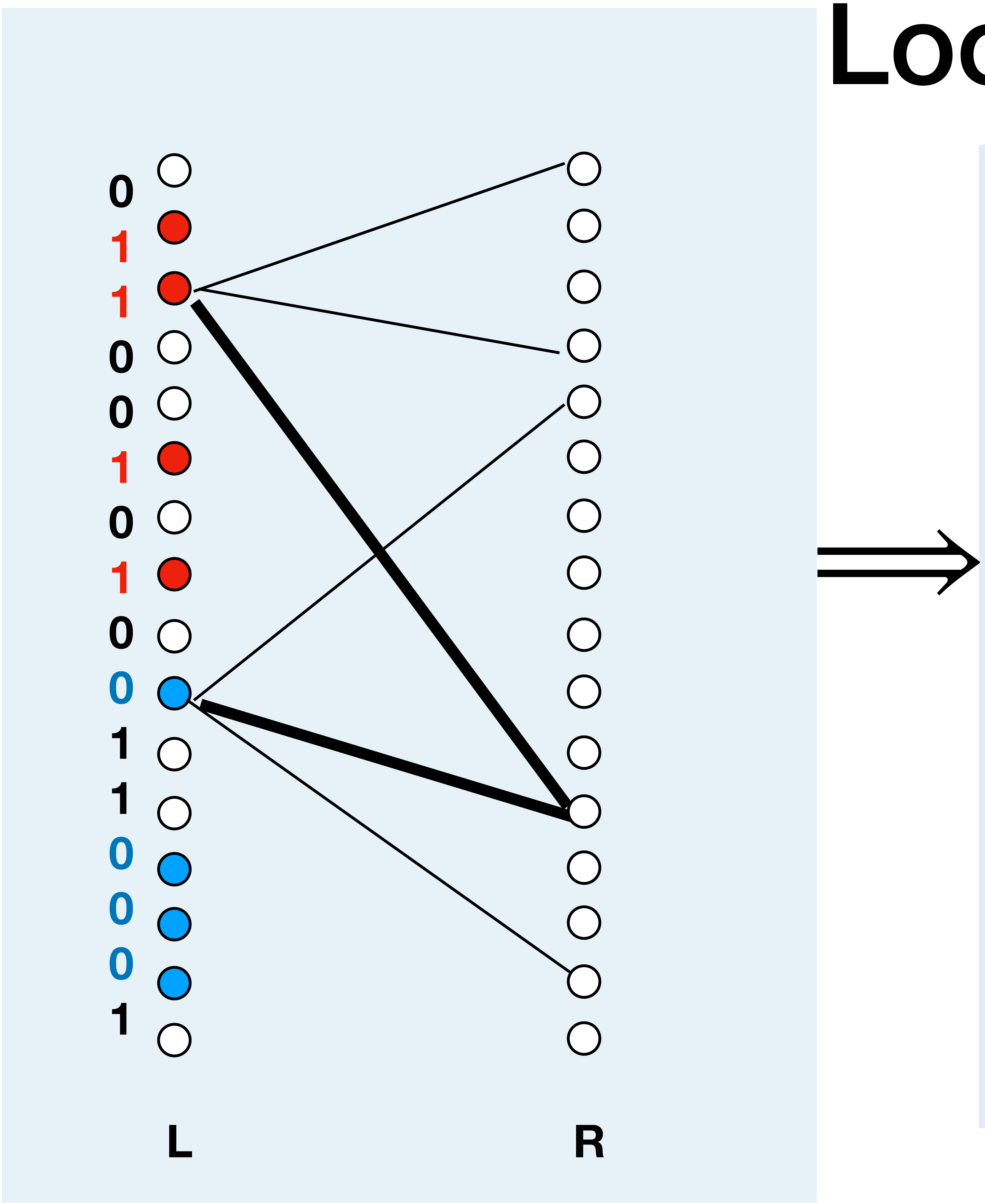
Claim

At the end of this procedure, there are no more than n/100 remaining swaps

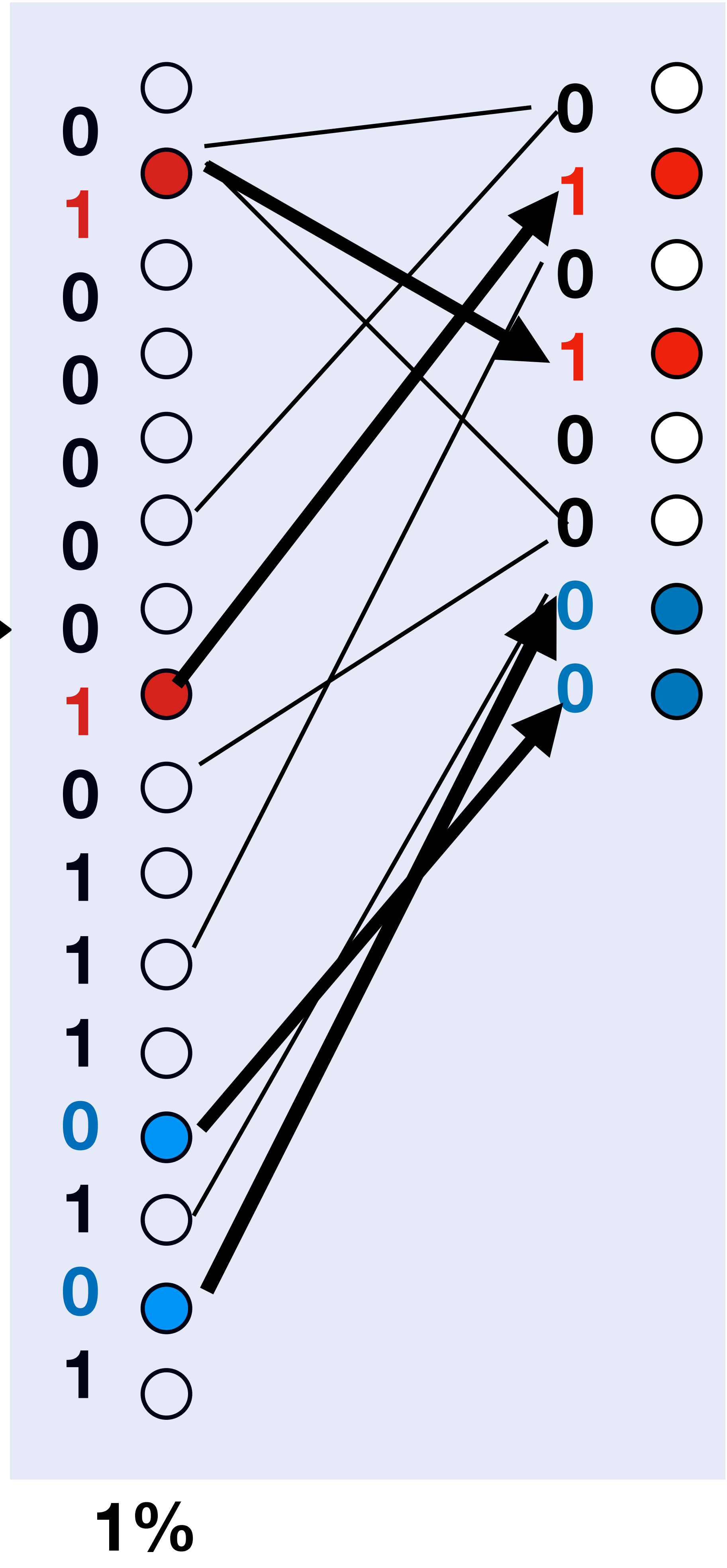
- Consider the sets of “survivors”
  - Each of size > n/200 (Recall #reds=#blues)
  - Their sets of neighbors must be disjoint
- **Expansion property**: for any set of size > n/200, number of neighbors > n/2

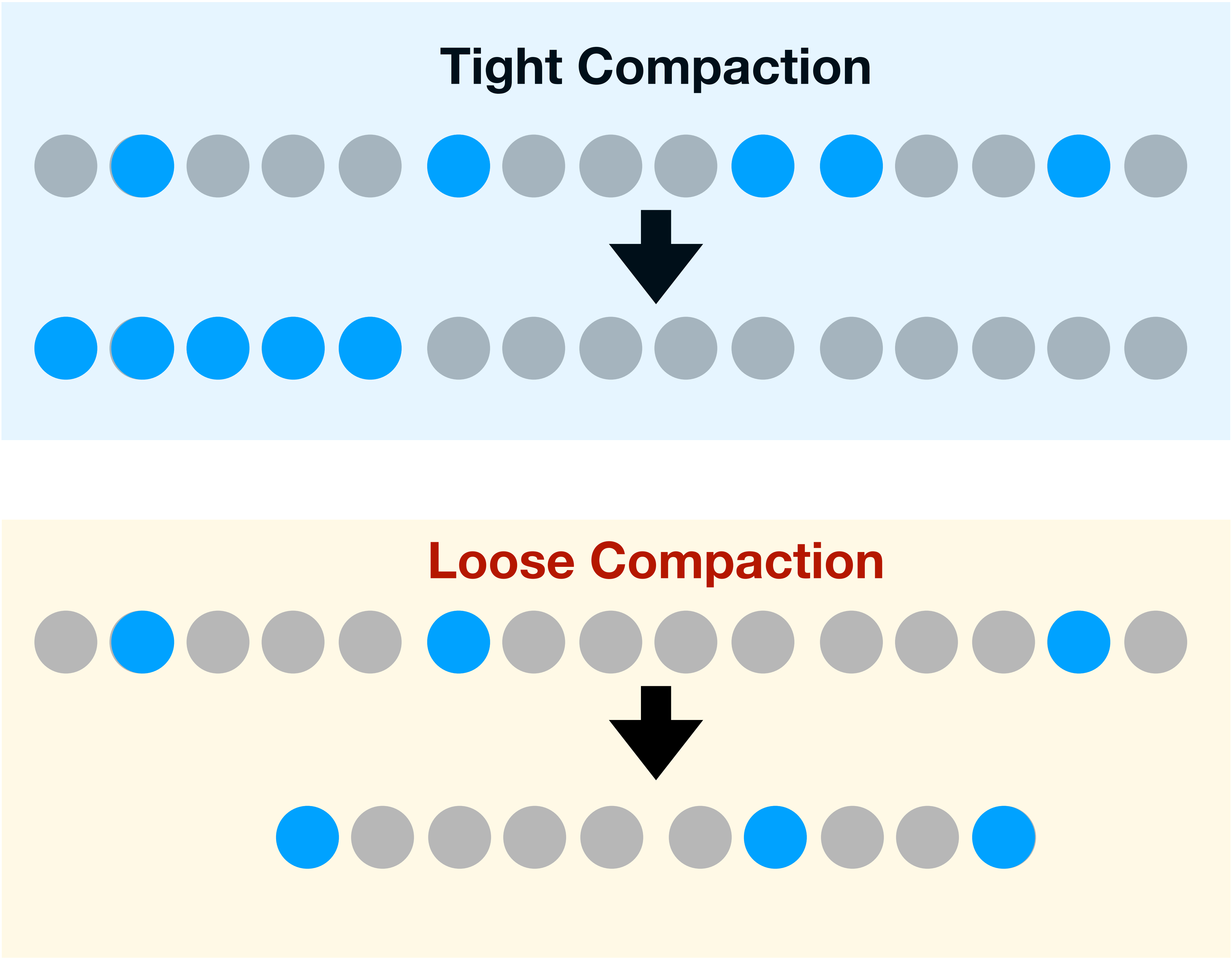


Loose Swap



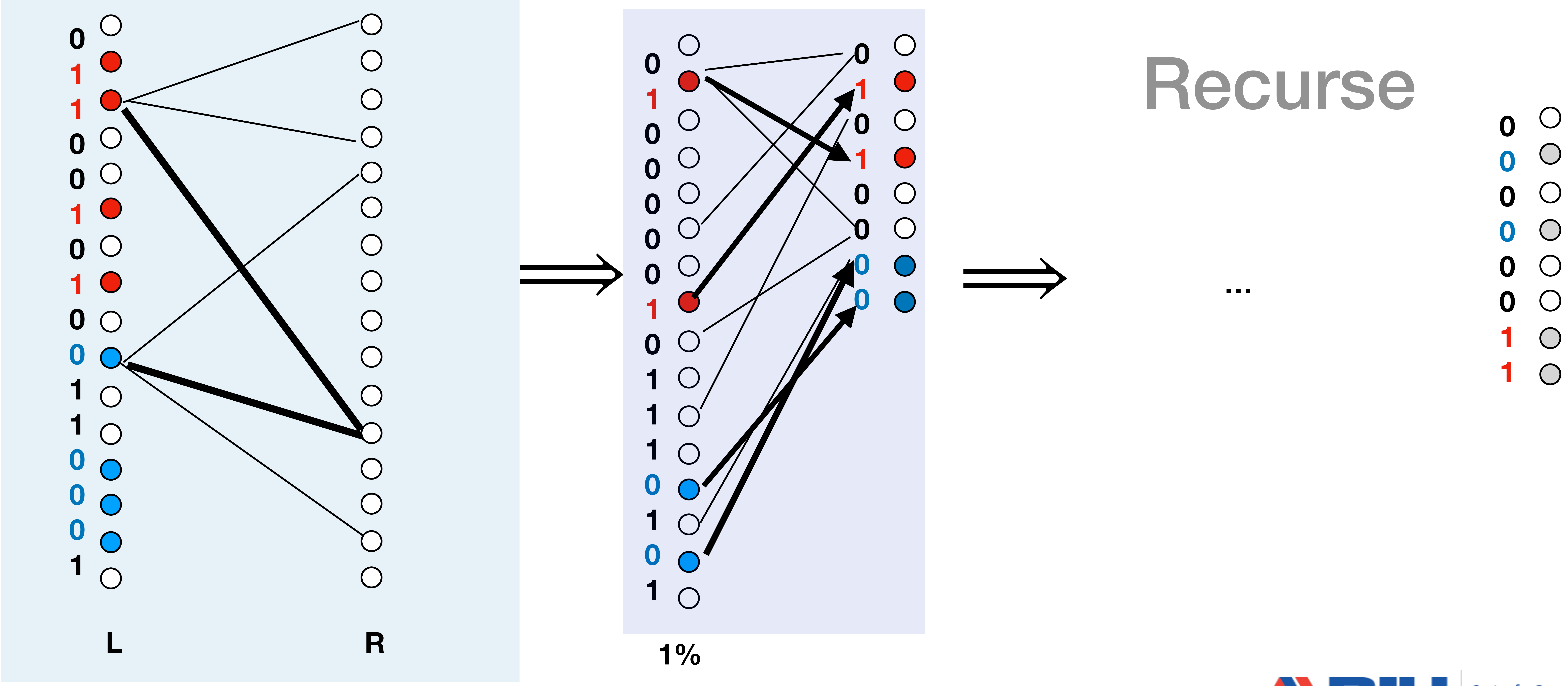
Loose Compactor



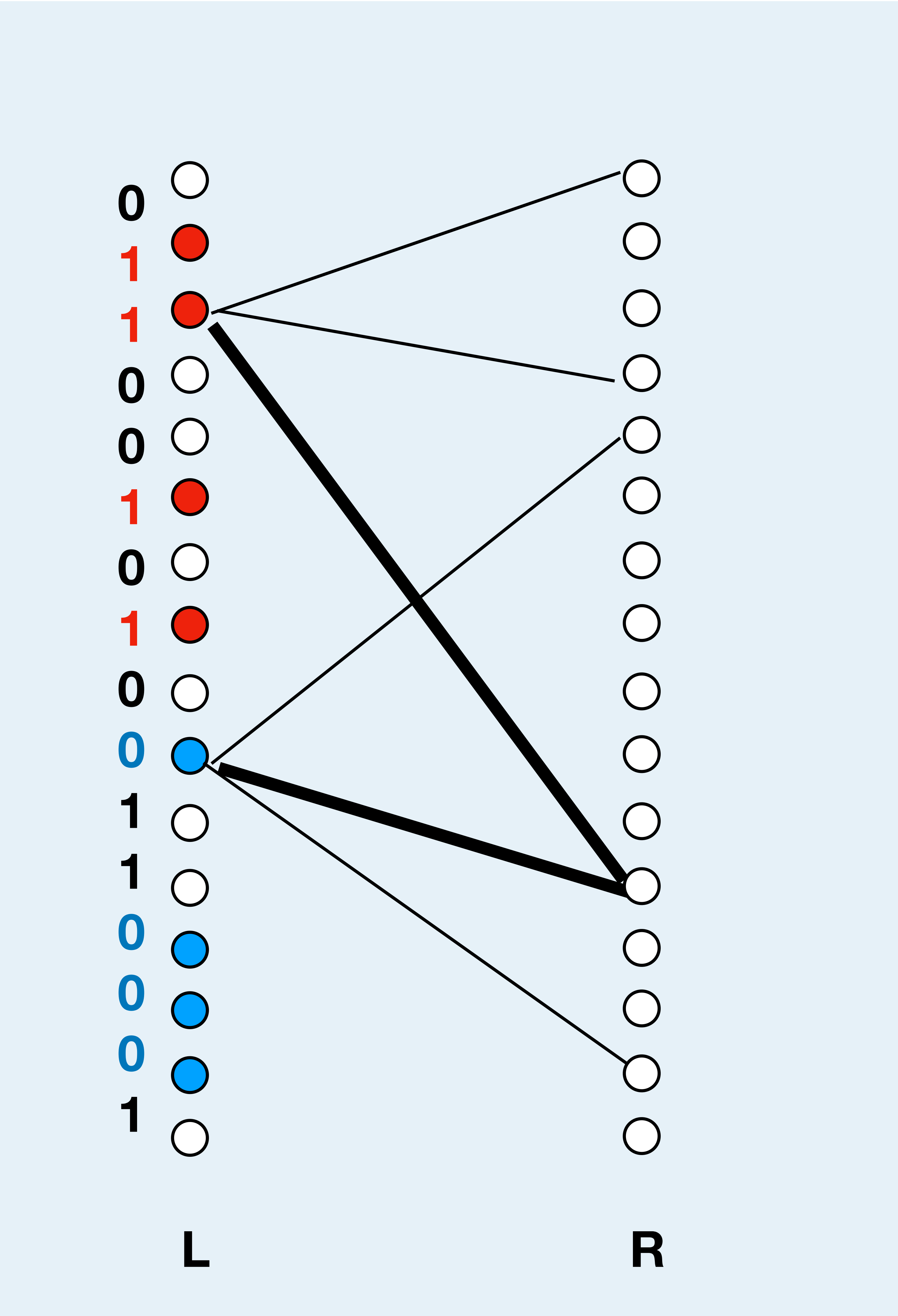


Loose Swap

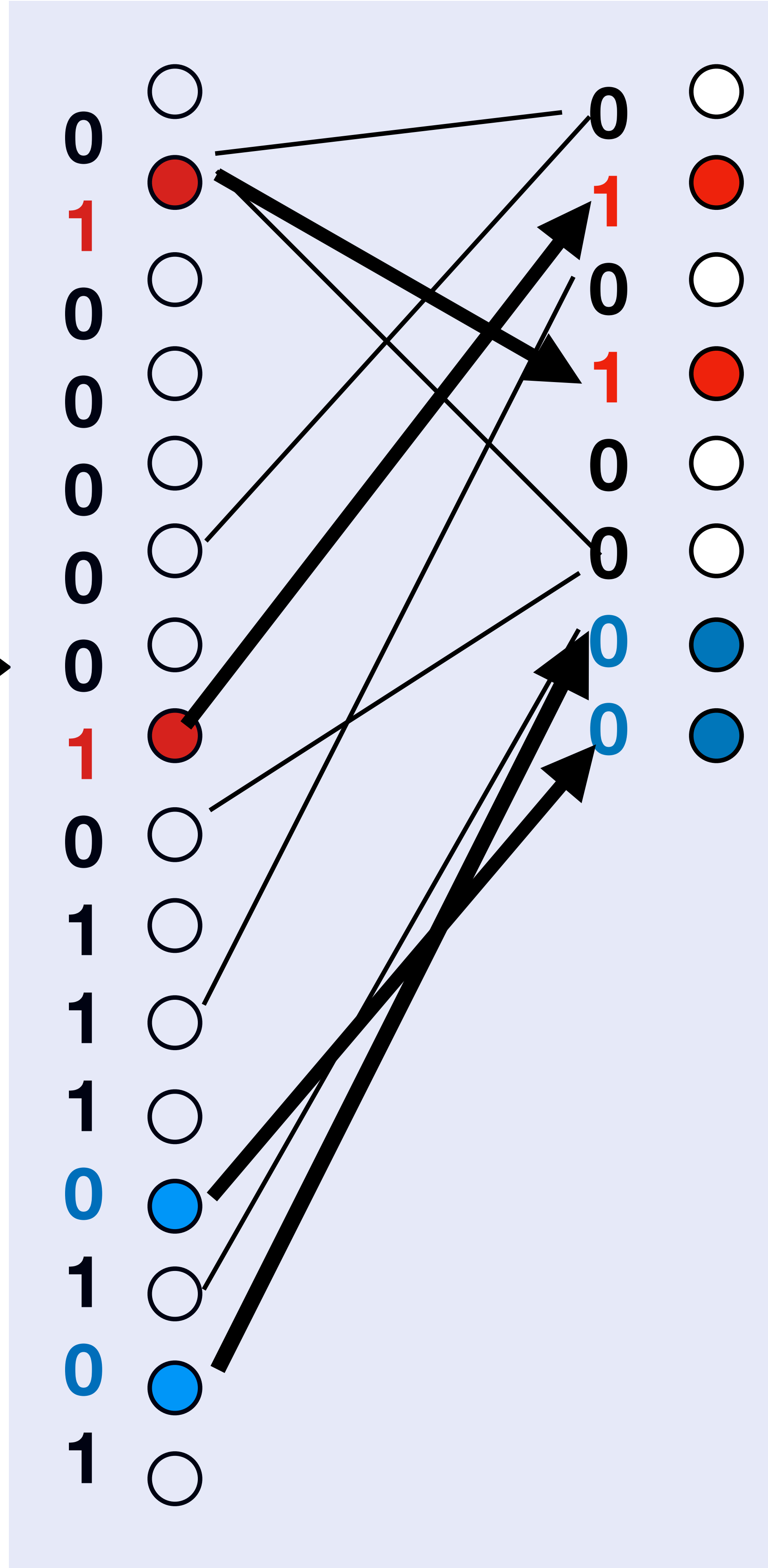
Loose Compactor



Loose Swap



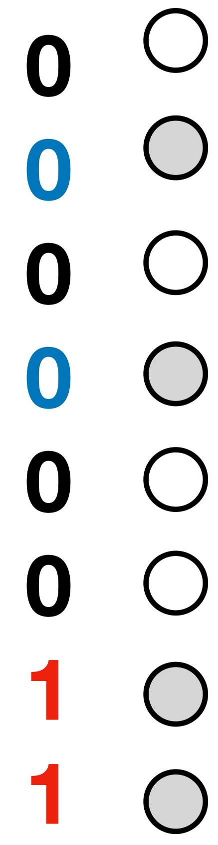
Loose Compactor



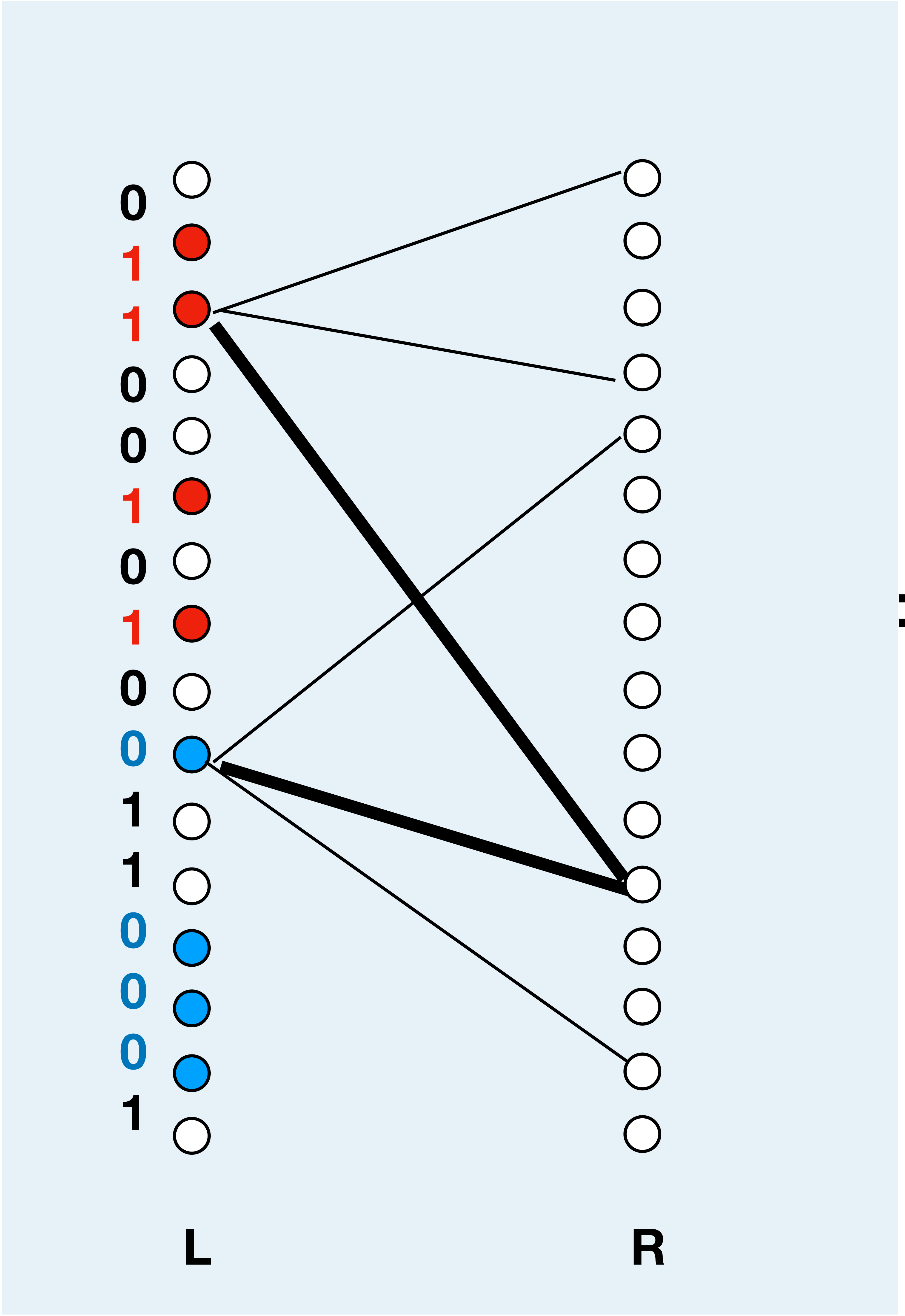
1%

Recurse

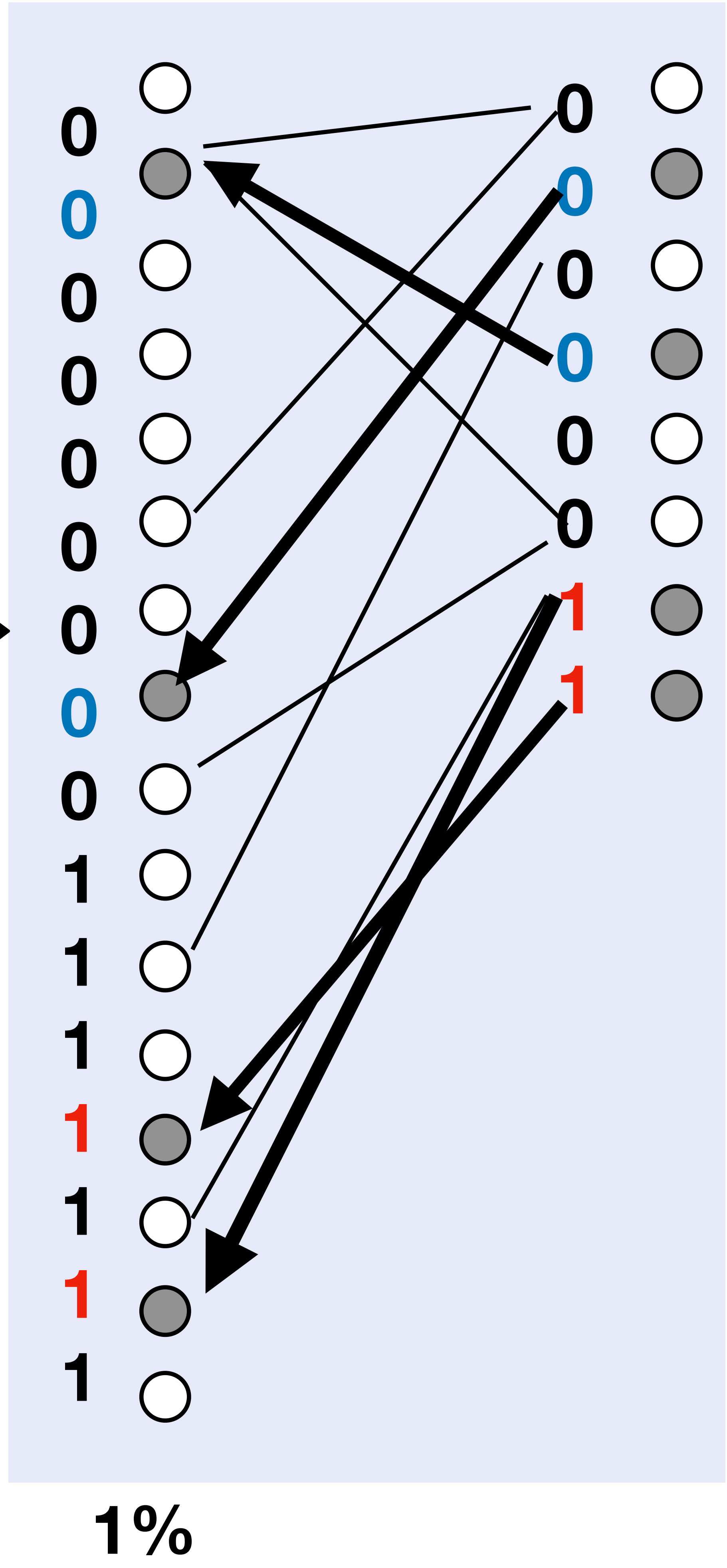
...



Loose Swap

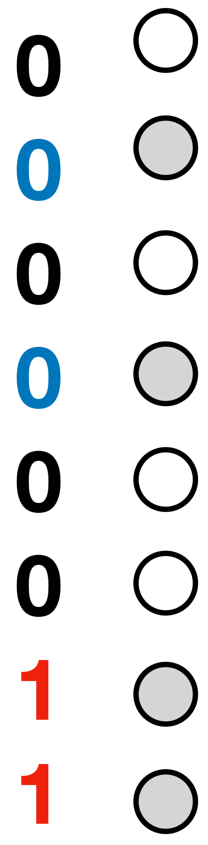


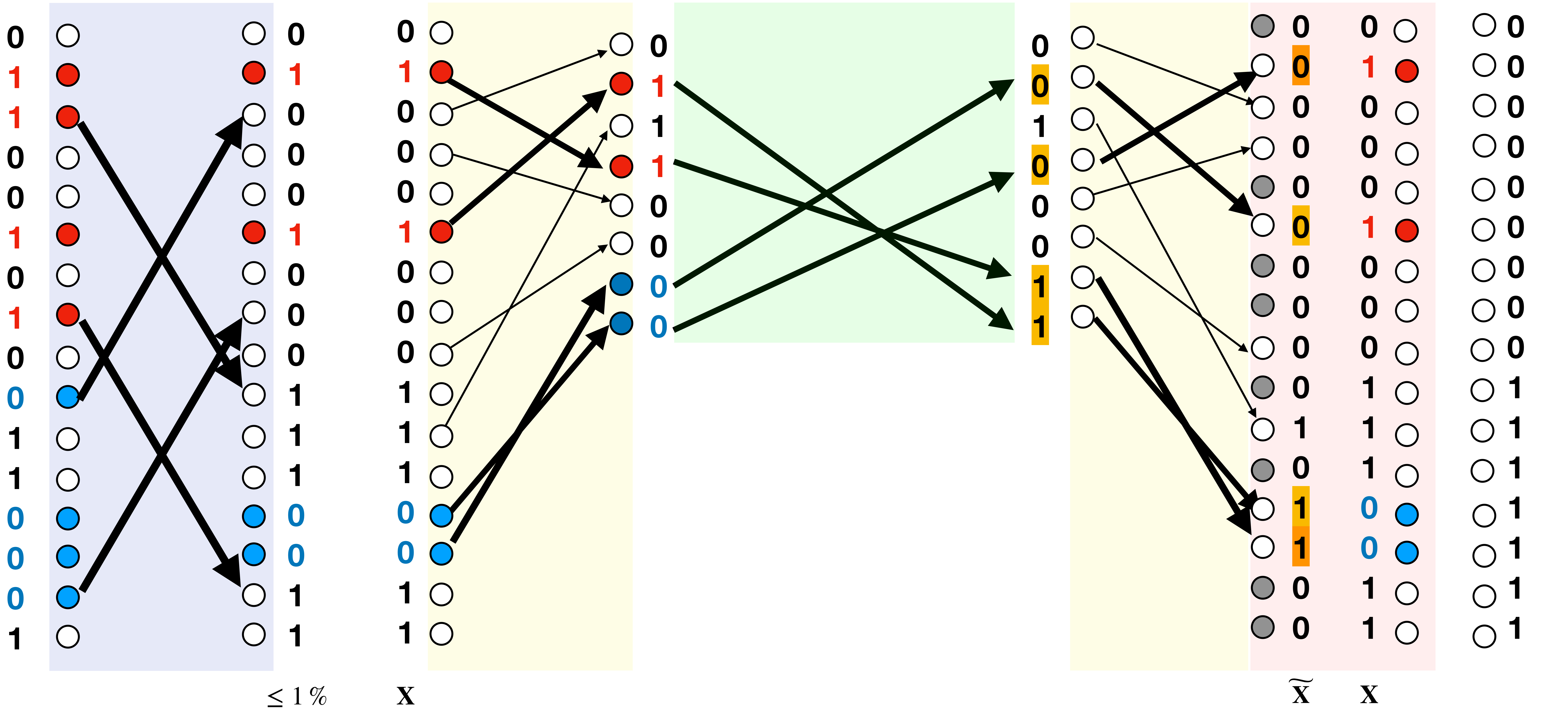
Reverse Route



Recurse

...





(1) Loose Swapper

(2) Loose Compactor

(3) $\mathsf{Swap}(\mathbf{Y})$ (Recurse)

(4) Reverse Route Loose Compactor

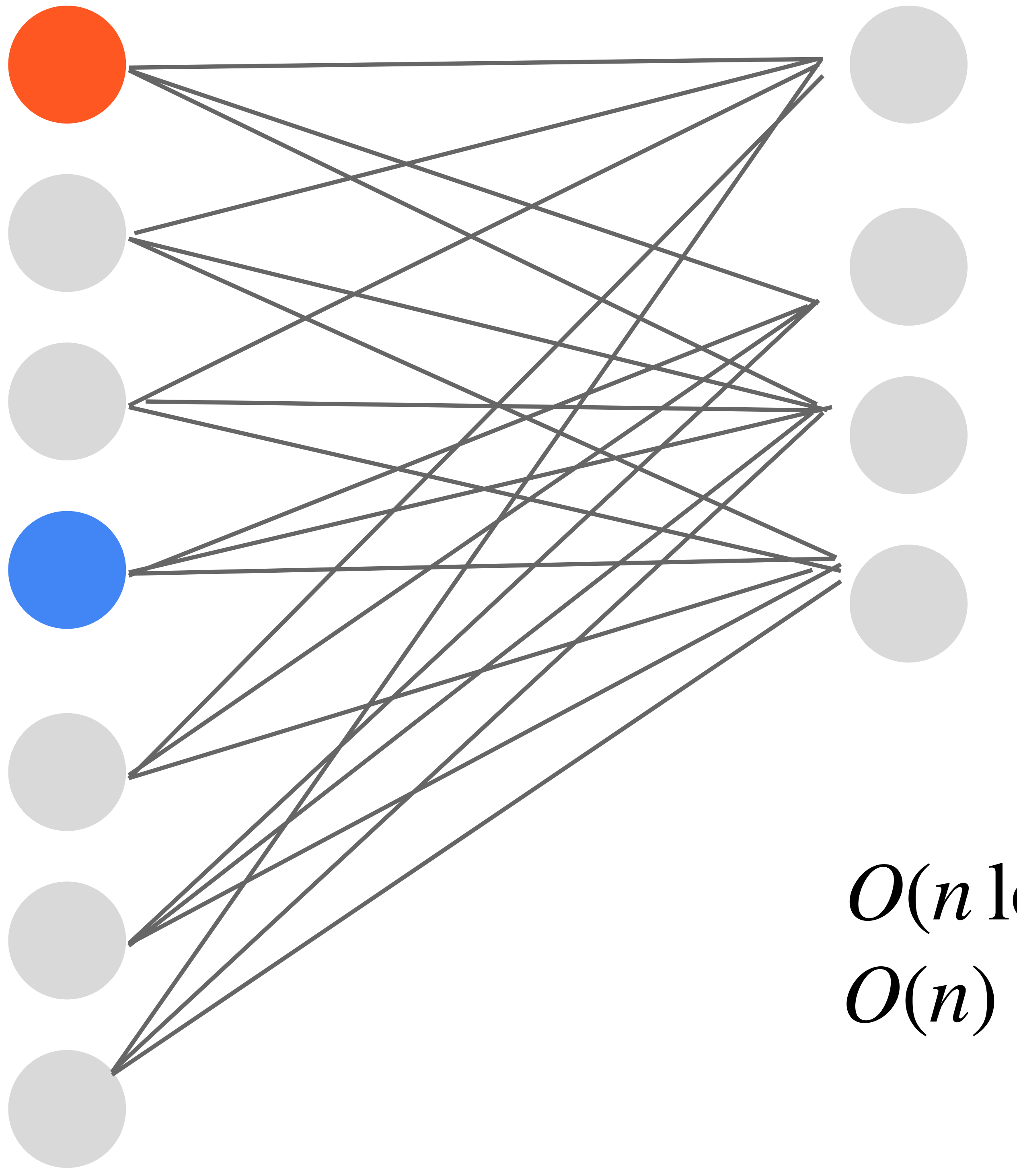
(5) Coordinate-wise select - $\mathbf{X}$ and $\widetilde{\mathbf{X}}$

What Do We Have So Far?

Loose Swap + Loose Compactor $\implies$ Tight Compactor

$O(nw)$ $\quad$ $O(n \cdot f(n) + nw)$ $\quad$ $O(n \cdot f(n) + nw)$

Loose Compactor



$\leq 1\,\%$ **are marked**

Bipartite Expander Graph

- $O(1)$ degree
- Constant spectral expansion

$O(n \log n)$ **boolean gates**
$O(n)$ w**-selector gates**

Two stages:

- Find which edges will be chosen
  - Find a matching (routes)
- Route elements on those edges

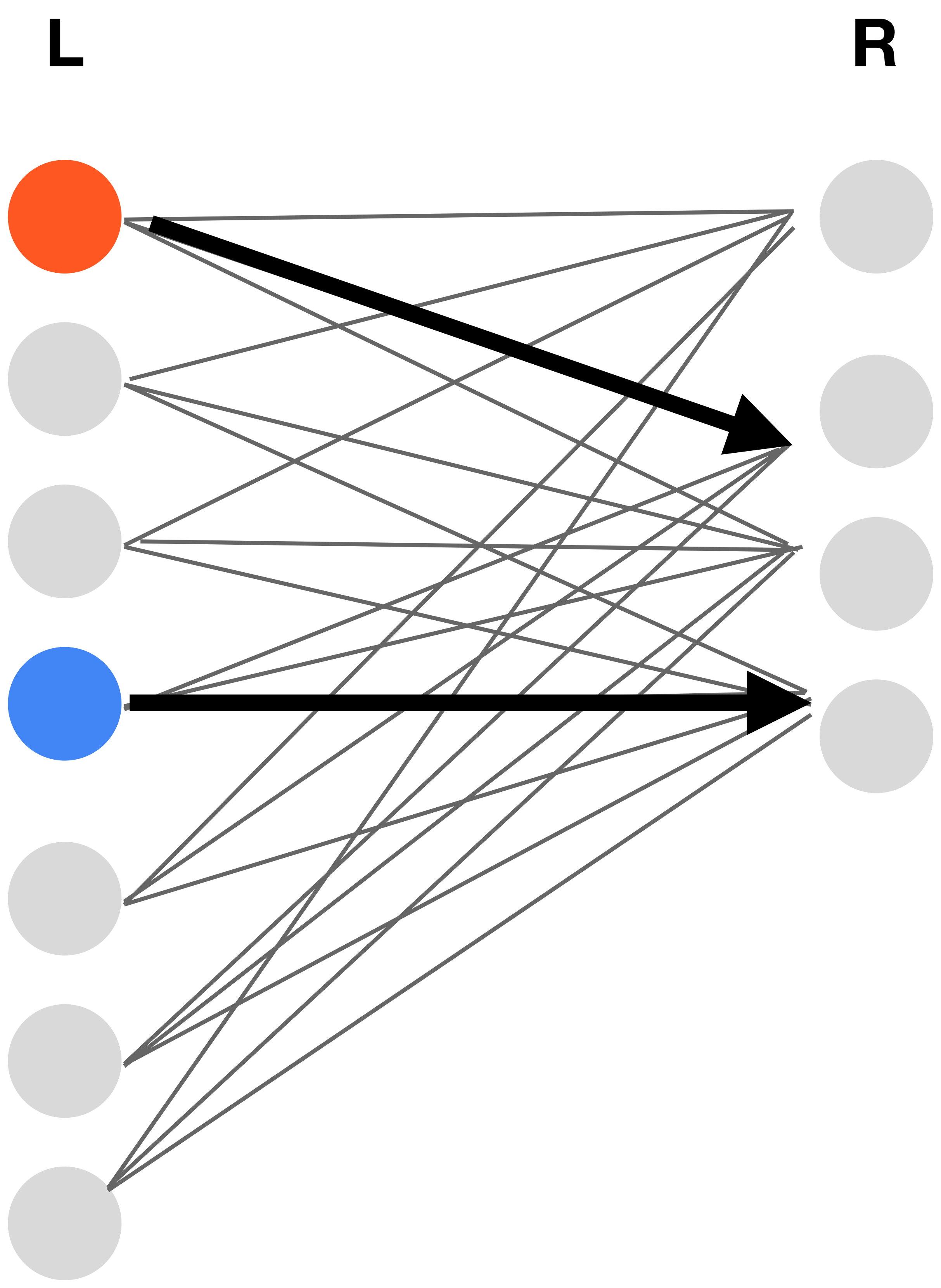
$O(n \log n + nw)$ **boolean gates**

Non comparison based!

[ALM'90,Pippenger'93]

Find a Matching

Offline route-finding does not depend on the payload!



$\leq 1\,\%$ **are marked**

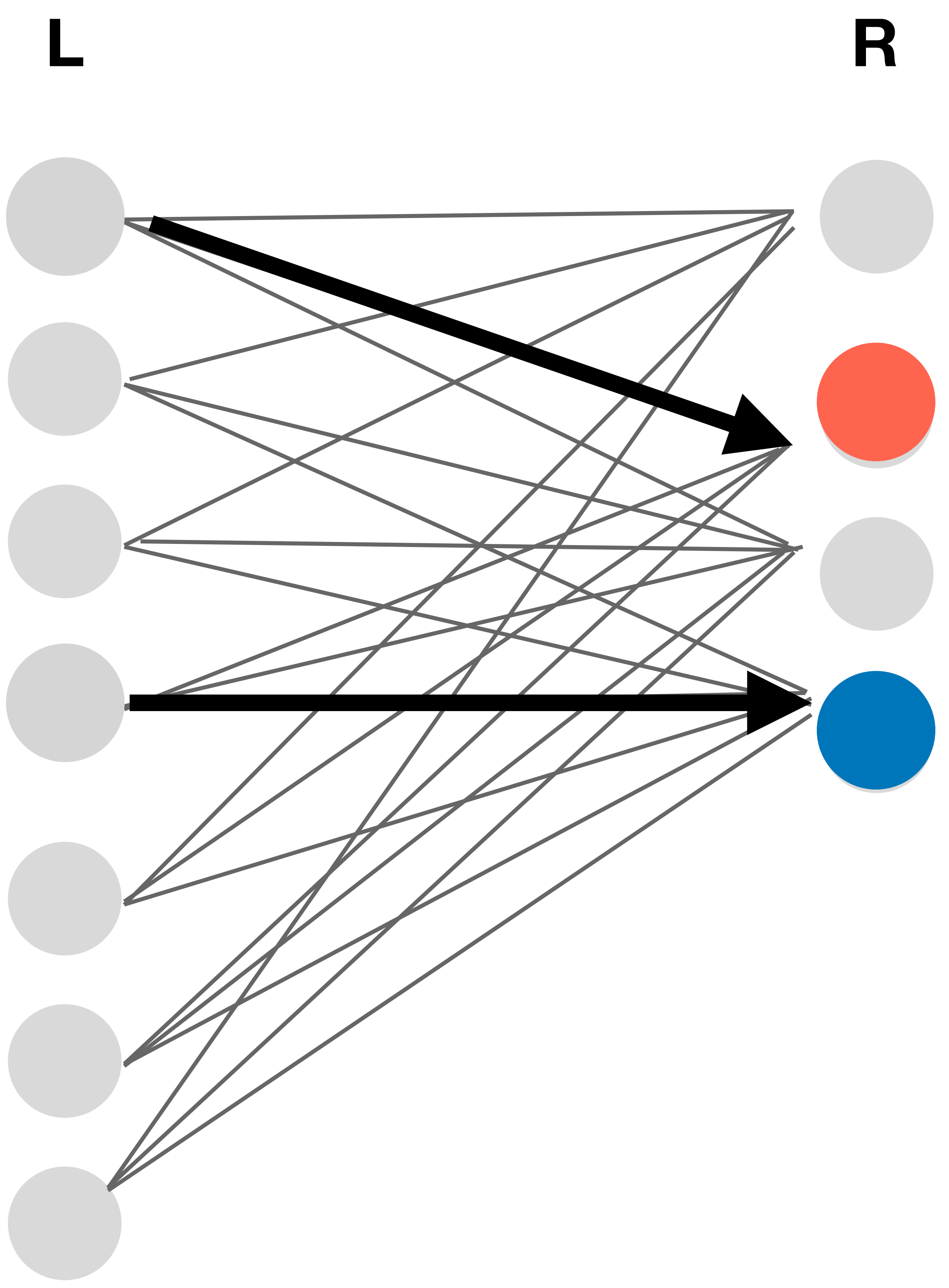
Repeat $\log n$ times:

- **All** ● ● **in L:** propose to all incident vertices
- **R:** accept only if 1 proposal received, else reject all
- **All** ● ● **in L:** if received "accept" become ●

$O(n \log n)$ **boolean gates**

Each edge has an indicator (a wire) whether to "route" an element on it

Route



$\leq 1\%$ **are marked**

On all marked edges:

- Move an element

Requires $O(n)$ w-selector gates

$O(n \log n)$ **boolean gates**

Overall circuit requires $O(n \log n + nw)$ boolean gates

What Do We Have So Far?

Loose Compactor $O(n \log n + nw)$

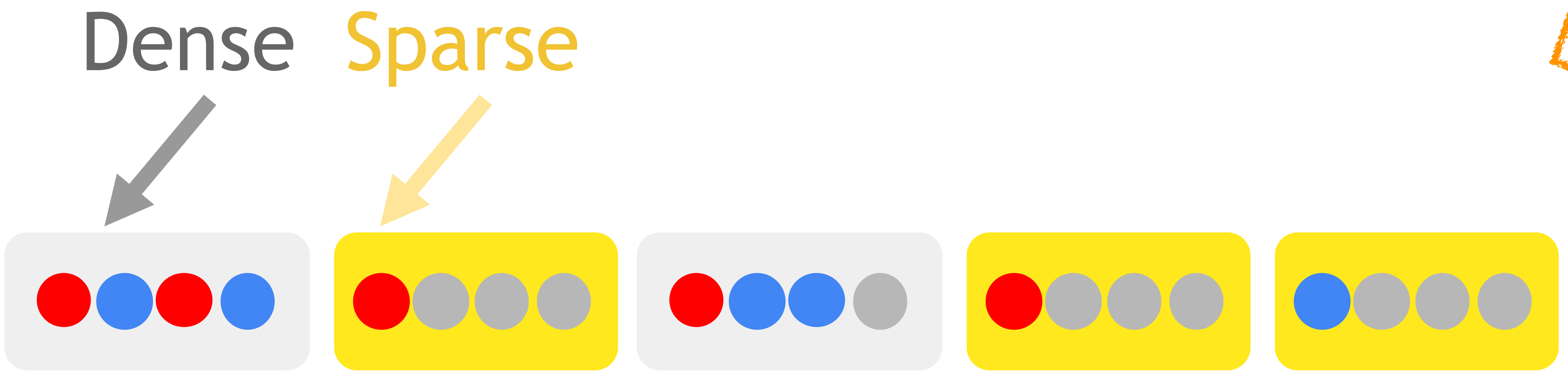
Loose Swap + Loose Compactor $\Longrightarrow$ Tight Compactor

$O(nw)$ $\quad$ $O(n \cdot f(n) + nw)$ $\quad$ $O(n \cdot f(n) + nw)$

Tight Compactor $O(n \log n + nw)$

Key insight: don't stop here

Tight Compactor $O(n \log n + nw)$ $\Longrightarrow$ Loose Compactor $O(n \log \log n + nw)$

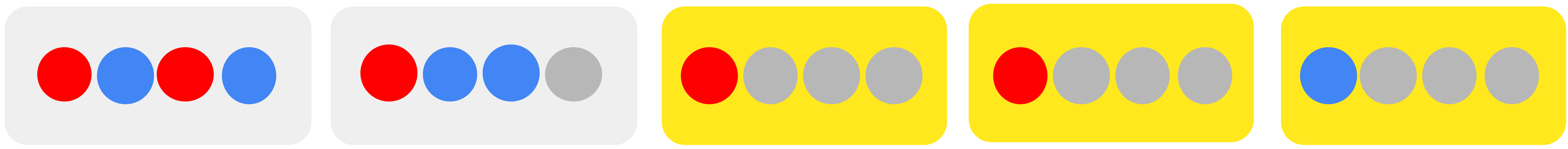


Very few chunks are dense

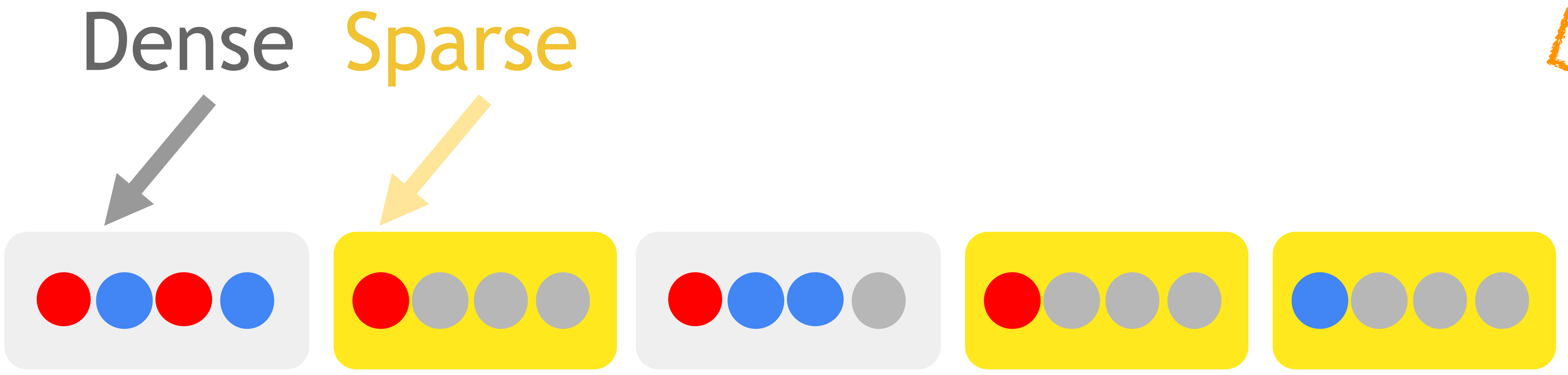
Chunk: $\log n$ **balls** **(each ball** w **bits long)**

$n/\log n$ **chunks,** $w \log n$ **bit payload each**

Step 1: **Run tight compact to move all "dense" chunks to the front** $O(nw)$ **circuit**

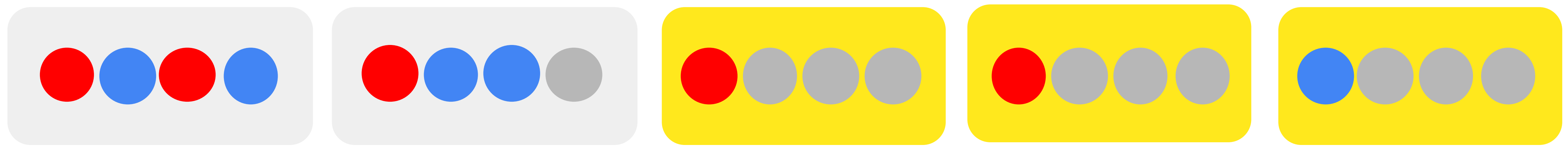


Tight Compactor $O(n \log n + nw)$ $\Longrightarrow$ Loose Compactor $O(n \log \log n + nw)$

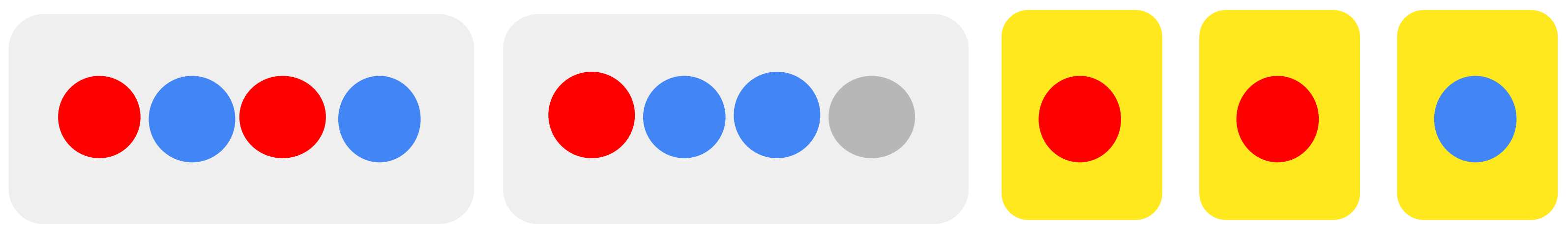


Very few chunks are dense

Step 1: **Run tight compact to move all "dense" chunks to the front**



Step 2: **Run tight compact on all sparse chunks**



$n / \log n$

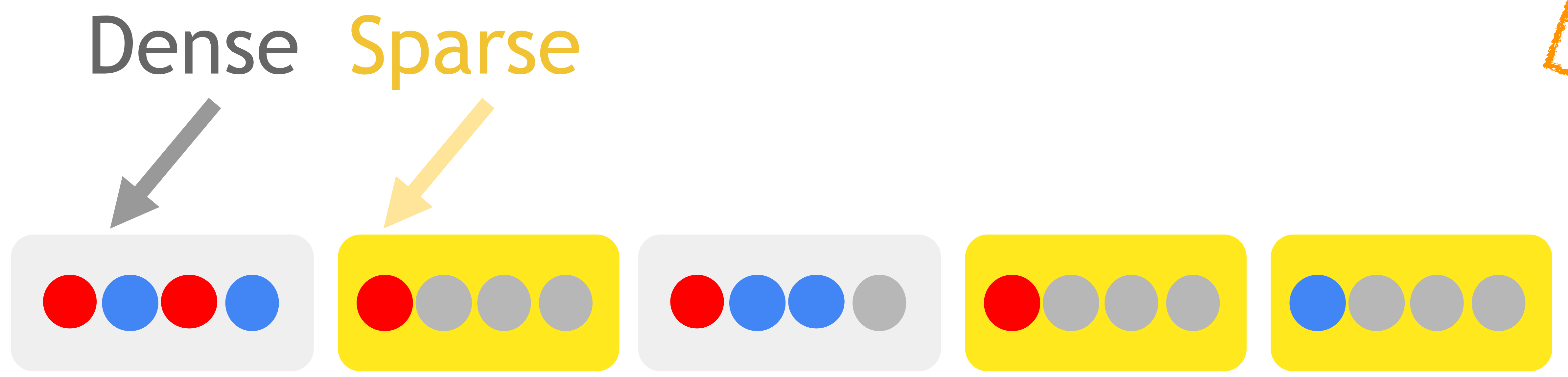
$\times$

$\log n$ **balls of size** w**-bit each**

Tight compact: $\log n \cdot \log \log n + \log n \cdot w$

$O(n \log \log n + nw)$ **size circuit**

Tight Compactor $O(n \cdot f(n) + nw)$ $\Longrightarrow$ Loose Compactor $O(n \cdot f(f(n)) + nw)$

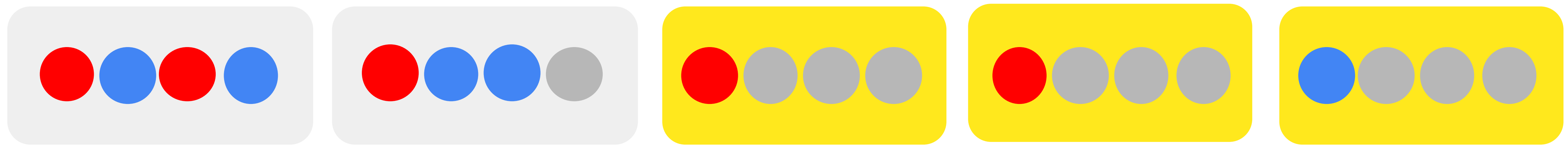


Very few chunks are dense

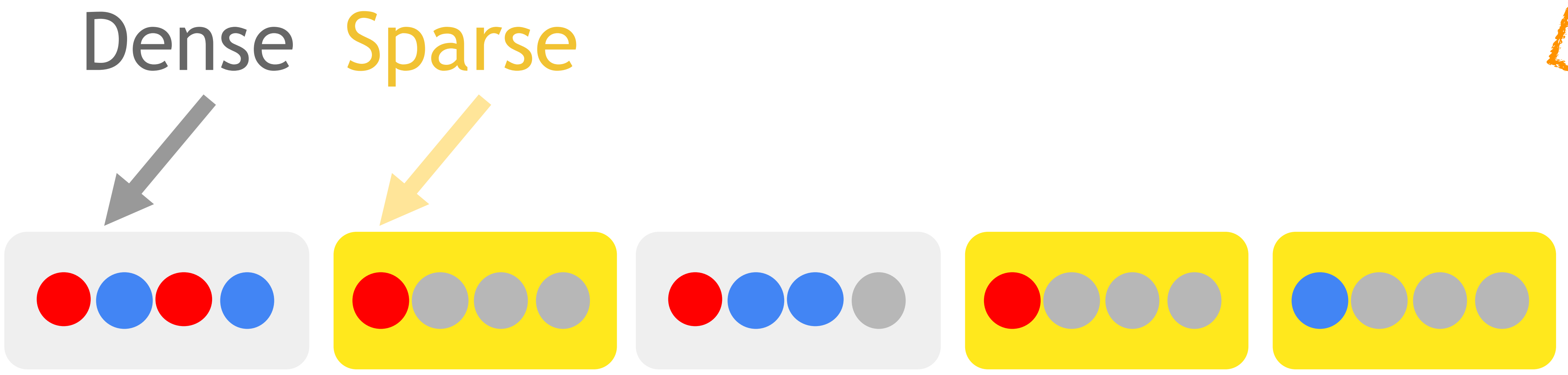
Chunk: $f(n)$ **balls** **(**$w \cdot f(n)$ **bits)**

$n/f(n)$ **chunks,** $w \cdot f(n)$ **bit payload each**

Step 1: **Run tight compact to move all "dense" chunks to the front** $O(nw)$ **circuit**

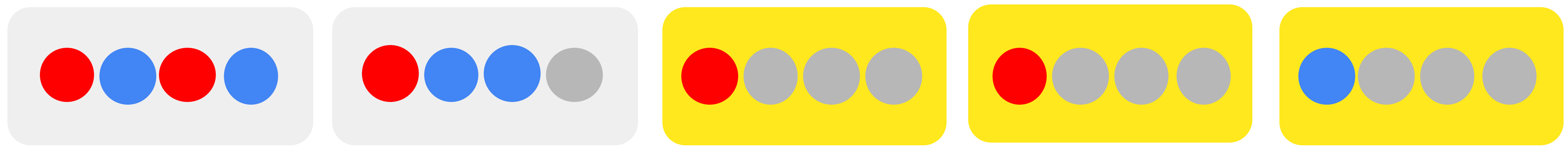


Tight Compactor $O(n \cdot f(n) + nw) \implies$ Loose Compactor $O(n \cdot f(f(n)) + nw)$

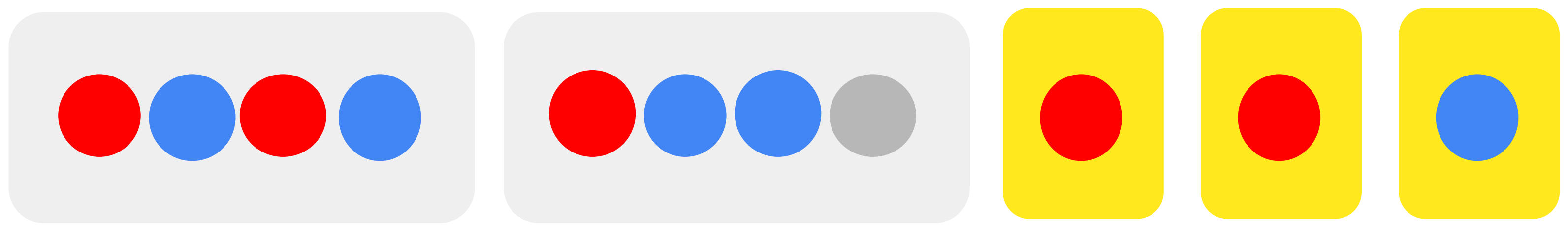


Very few chunks are dense

Step 1: **Run tight compact to move all "dense" chunks to the front**



Step 2: **Run tight compact on all sparse chunks**



$n/f(n)$

$\times$

$f(n)$ **balls of size w-bit each**

Tight compact: $f(n) \cdot f(f(n)) + f(n) \cdot w$

$O(n \cdot f(f(n)) + nw)$ **size circuit**

What Do We Have So Far?

Loose Compactor $O(n \log n + nw)$

Loose Compactor $O(n \cdot f(n) + nw)$ $\Longrightarrow$ Tight Compactor $O(n \cdot f(n) + nw)$ $O(n \log n + nw)$

Tight Compactor $O(n \cdot f(n) + nw)$ $\Longrightarrow$ Loose Compactor $O(n \cdot f(f(n)) + nw)$

Tight Compactor $O(n \cdot f(n) + nw)$ $\Longrightarrow$ Tight Compactor $O(n \cdot f(f(n)) + nw)$

Bootstrapping!

Tight Compactor $O(n \log n + nw)$

Tight Compactor $O(n \cdot f(n) + nw)$ $\Longrightarrow$ Tight Compactor $O(n \cdot f(f(n)) + nw)$

Tight Compactor $O(n \log n + nw)$ $\Longrightarrow$ Tight Compactor $O(n \log \log n + nw)$ $\Longrightarrow$ Tight Compactor $O(n \log^{(4)} n + nw)$ $\Longrightarrow$ Tight Compactor $O(n \log^{(8)} n + nw)$

...

Constant blowups exponentially....

$$C^d \cdot \left(n \cdot \log^{(2^d)} n + nw\right)$$

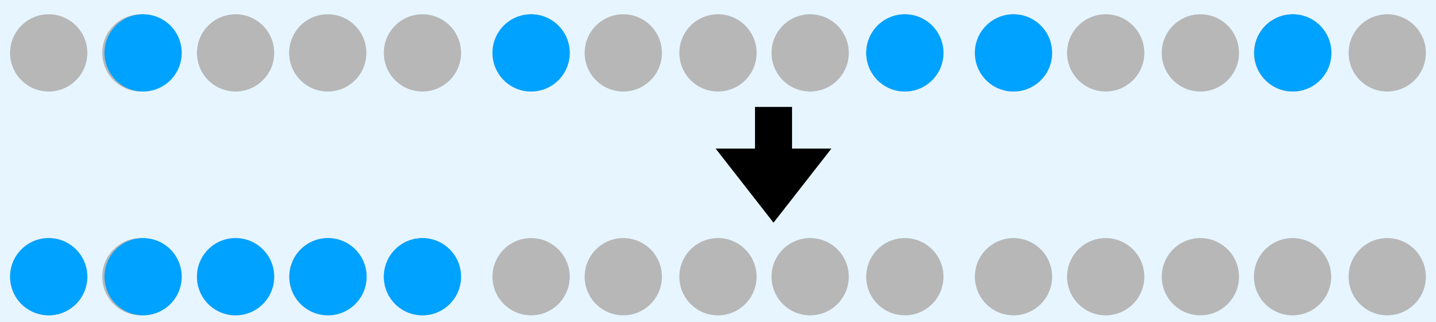
$$\log^{(2^d)} n = w$$

$$d = \log(\log^* n - \log^* w)$$

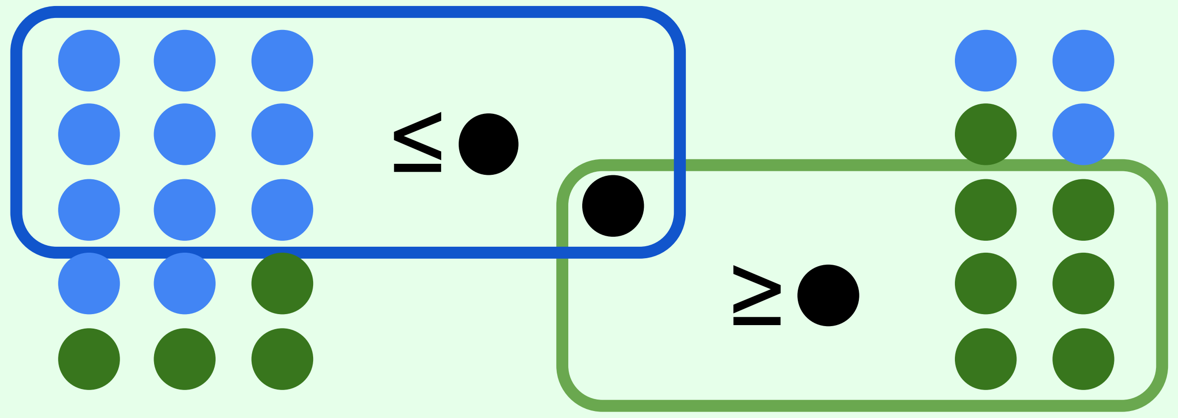
A tight compactor of size: $\mathbf{poly}(\log^* n - \log^* w) \cdot O(nw)$

Center for Research in Applied Cryptography and Cyber Security

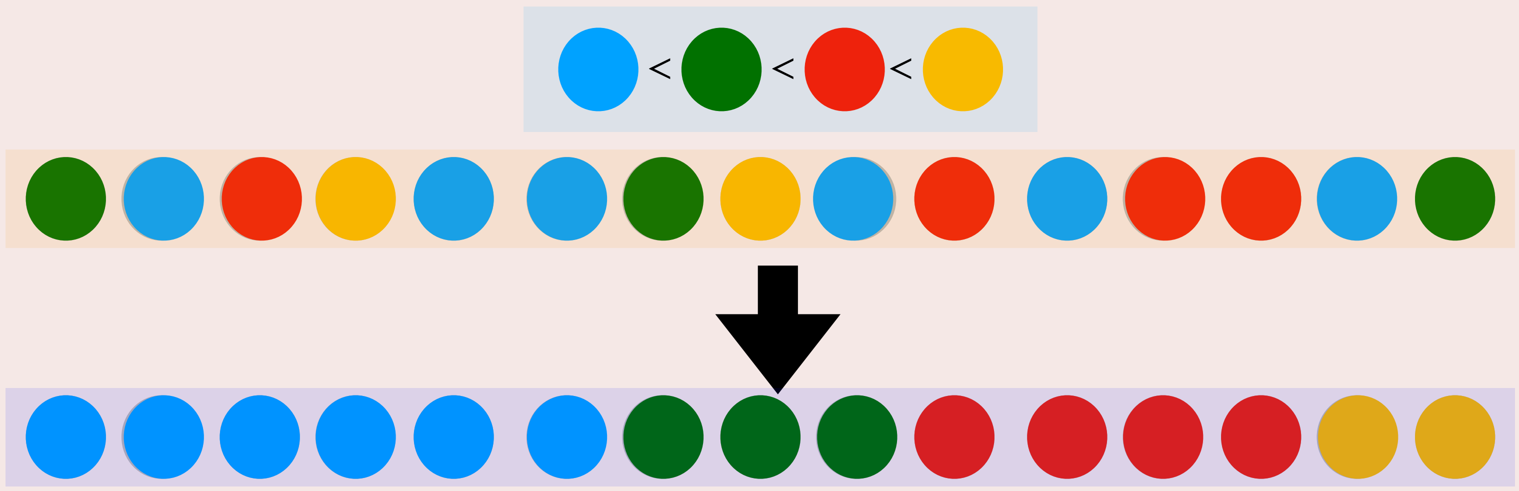
Tight Compaction



Selection

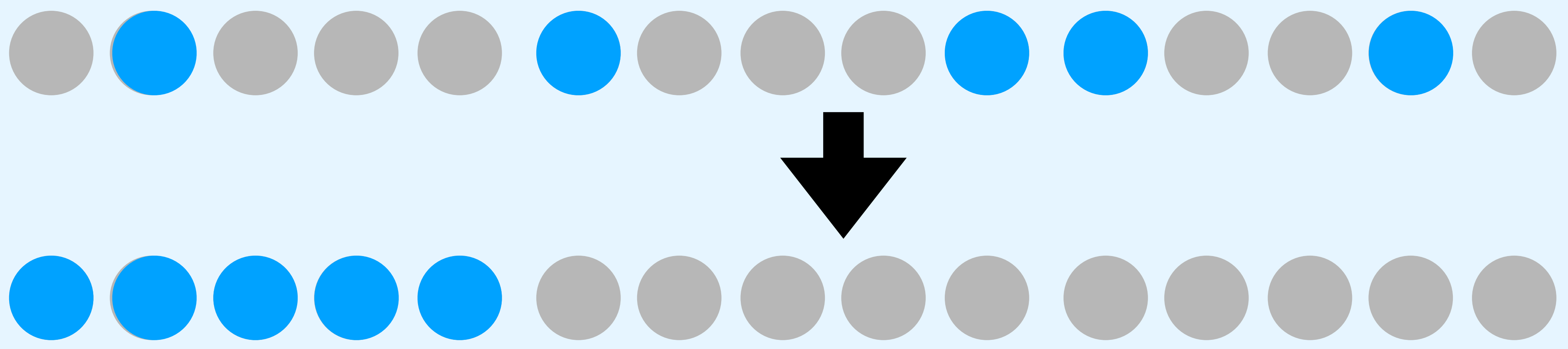


Sorting



Tight Compaction
A Central Problem!

Tight Compaction



Circuit Model

Linear size circuit
(Ignoring polylog* factors)

(Almost) Linear size sorting circuit
(short keys) [**A**LS'21]

RAM Model

Linear time compaction

Optimal Oblivious RAM compiler
[OptORAMa,**A**KLNPS'20]

Oblivious RAM Compiler: State of the Art

Lower bound: $\Omega(\log N)$

[GoldreichOstrovsky'96, LarsenNeilsen'18]

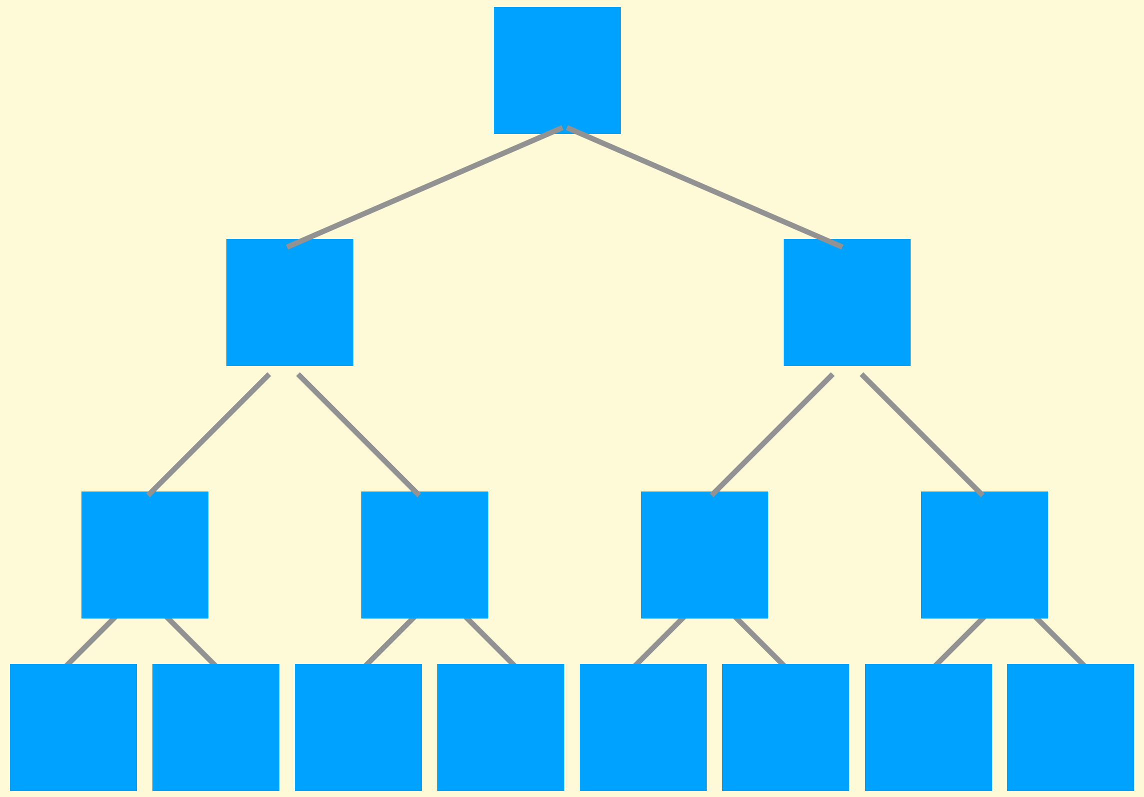
Hierarchical

[O90,GO96]

$$O(\log N)$$

Computational security

[OptORAMa'20]



Tree based ORAM

[Shi,Chan,Stefanov11]

$$O(\log^2 N)$$

Statistical security

[PathORAM,CircuitORAM]

Thank You!