Lower bounds for Streaming Problems

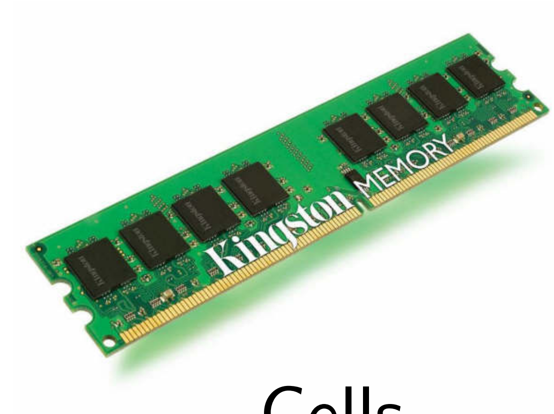
Raphaël Clifford

Joint work with

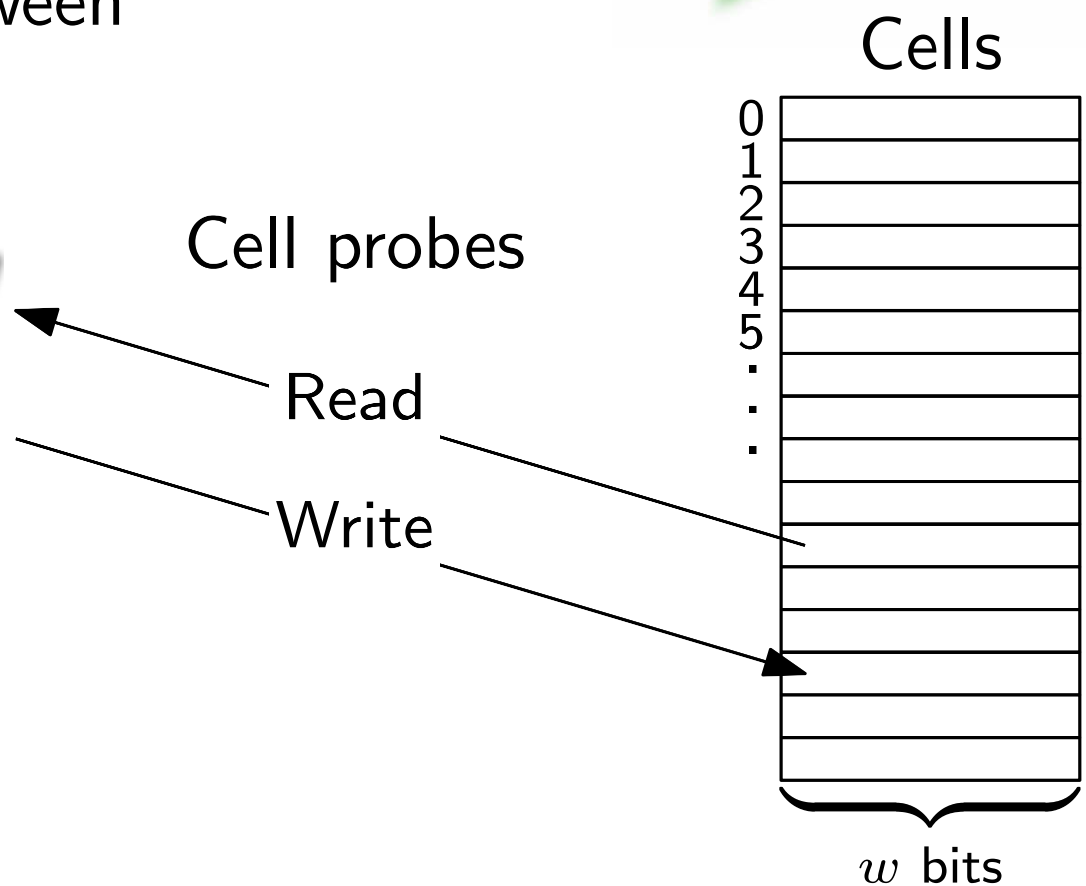
Markus Jalsenius and Benjamin Sach

University of BRISTOL

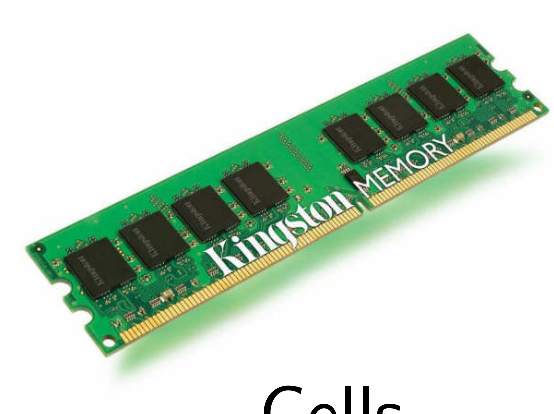
Cell-probe model



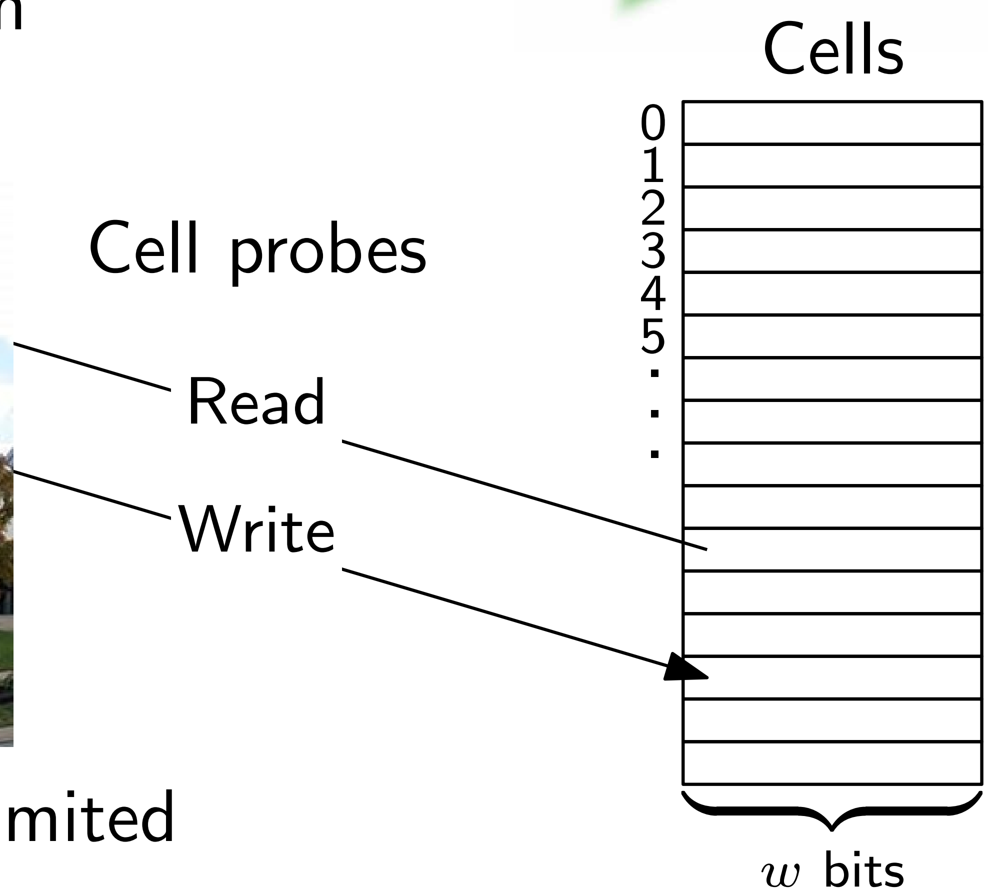
The CPU does not remember anything in between operations.



Cell-probe model



The CPU does not remember anything in between operations.



The CPU has unlimited computational power.

Data Structure Lower Bounds

Yao - FOCS '78

Predecessor (static)

- Ajtai - Combinatorica '88 (incorrect) (Communication complexity)
- Miltersen - STOC' 94
- Miltersen, Nisan, Safra, Wigdersen - STOC '95
- Beame, Fich - STOC '99
- Sen - ICALP '01

Dynamic problems (partial sums, union find)

- Fredman, Saks - STOC '89 (Chronogram technique)
- Ben-Amram, Galil - FOCS '91
- Miltersen, Subramanian, Vitter, Tamassia - TCS '94
- Husfeldt, Rauhe, Skyum - SWAT '96 (planar connectivity)
- Fredman, Henzinger - Algorithmica '98 (non-determinism)
- Alstrup, Husfeldt, Rauhe - FOCS '98 (marked ancestor)
- Alstrup, Husfeldt, Rauhe - SODA '01 (2D NN)
- Alstrup, Ben-Amram, Rauhe - STOC '99 (union-find)

Data Structure Lower Bounds

Yao - FOCS '78

Predecessor (static)

- Ajtai - Combinatorica '88 (incorrect) (Communication complexity)
- Miltersen - STOC' 94
- Miltersen, Nisan, Safra, Wigdersen - STOC '95
- Beame, Fich - STOC '99
- Sen - ICALP '01

Dynamic problems (partial sums, union find)

- Fredman, Saks - STOC '89 (Chronogram technique)
- Ben-Amram, Galil - F
- Miltersen, Subramania … '94
- Husfeldt, Rauhe, Skyu … nectivity)
- Fredman, Henzinger - … rminism)
- Alstrup, Husfeldt, Rau … cestor)
- Alstrup, Husfeldt, Rauhe - SODA '01 (2D NN)
- Alstrup, Ben-Amram, Rauhe - STOC '99 (union-find)

Best lower bound

$$\Omega\left(\frac{\log n}{\log\log n}\right)$$

Data Structure Lower Bounds

Yao - FOCS '78

Predecessor (static)

- Ajtai - Combinatorica '88 (incorrect) (Communication complexity)
- Miltersen - STOC' 94
- Miltersen, Nisan, Safra, Wigdersen - STOC '95
- Be
- Se

First $\Omega(\log n)$ lower bound using *information transfer*.

M. Pătrașcu and E. Demaine
Tight bounds for the partial-sums problem
SODA 2004

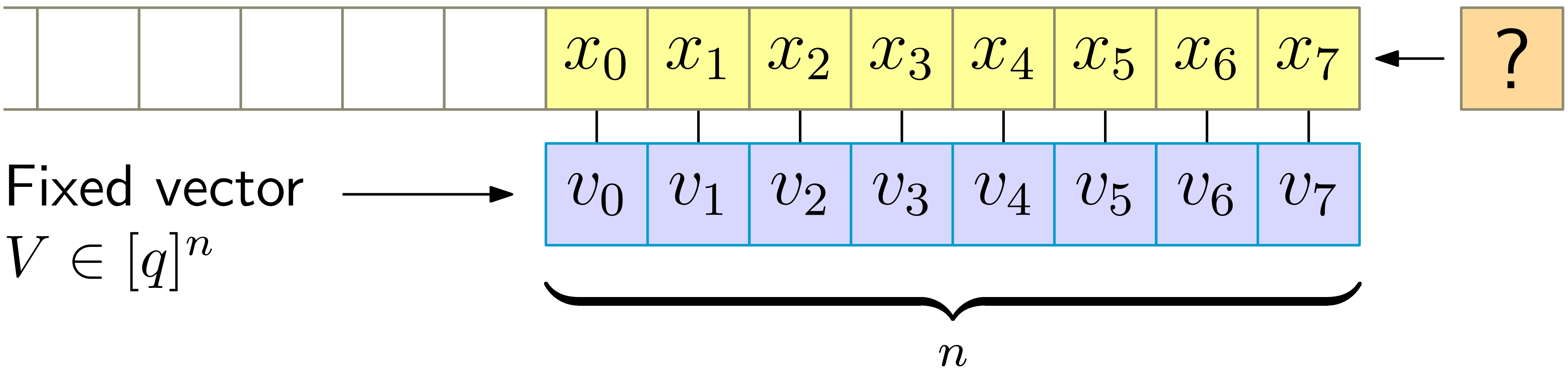
Dynam

- Fre
- Be
- Mi
- Hu
- Fredman, Henzinger - Algorithmica '98 (non-determinism)
- Alstrup, Husfeldt, Rauhe - FOCS '98 (marked ancestor)
- Alstrup, Husfeldt, Rauhe - SODA '01 (2D NN)
- Alstrup, Ben-Amram, Rauhe - STOC '99 (union-find)

Convolution

Stream of numbers from $[q]$

Fixed vector $V \in [q]^n$

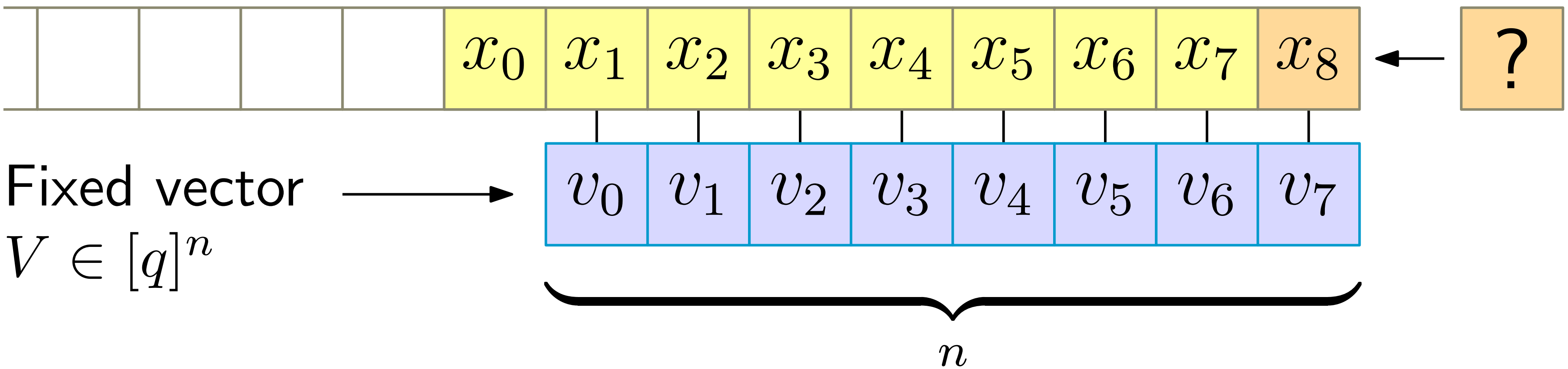


Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$

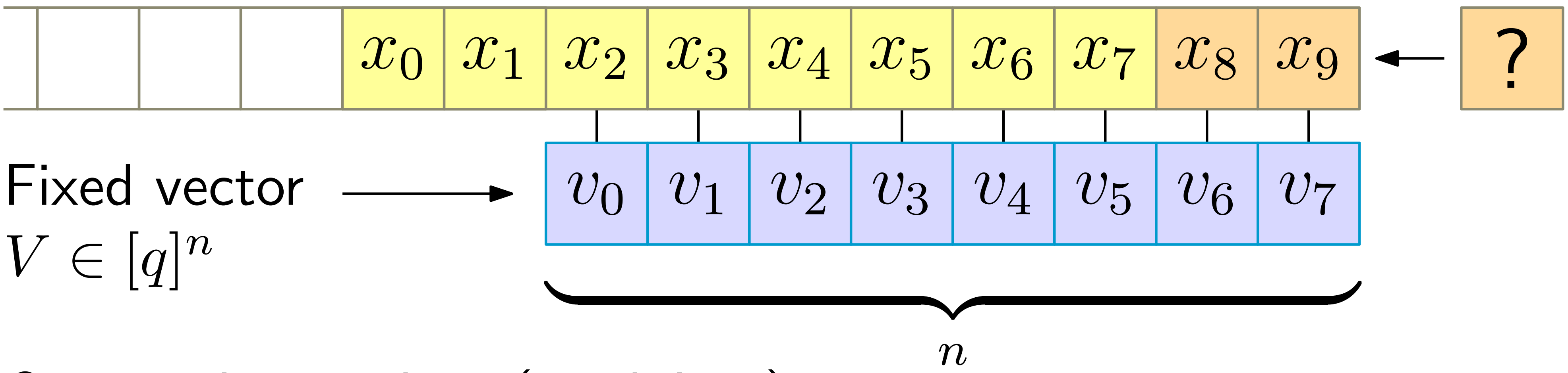


Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$

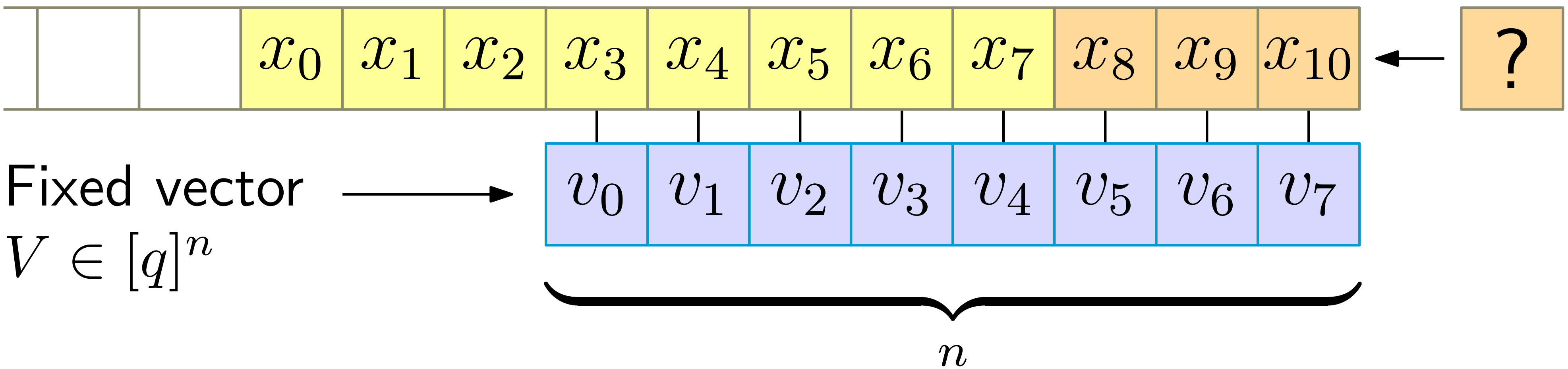


Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$

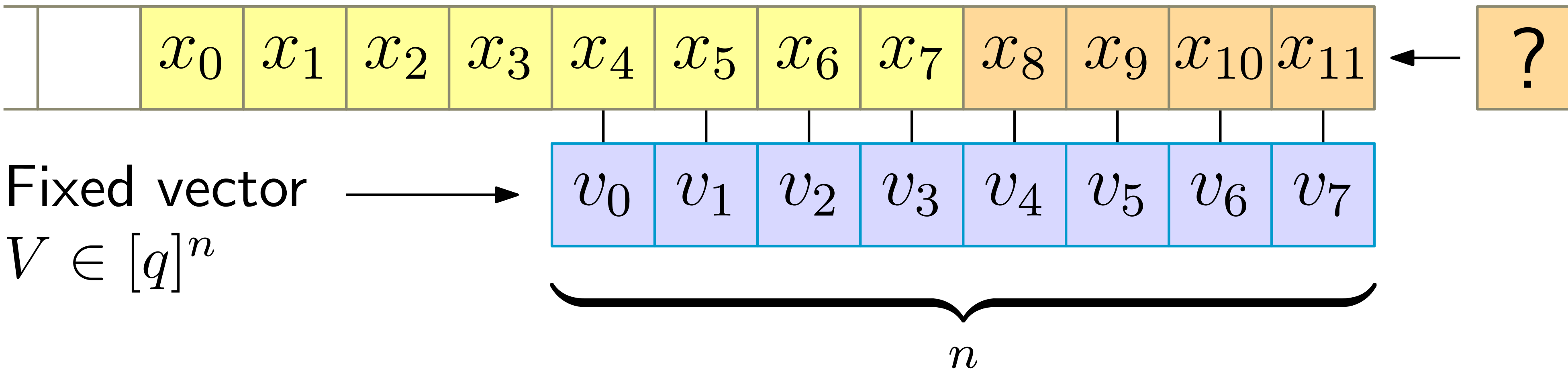


Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$

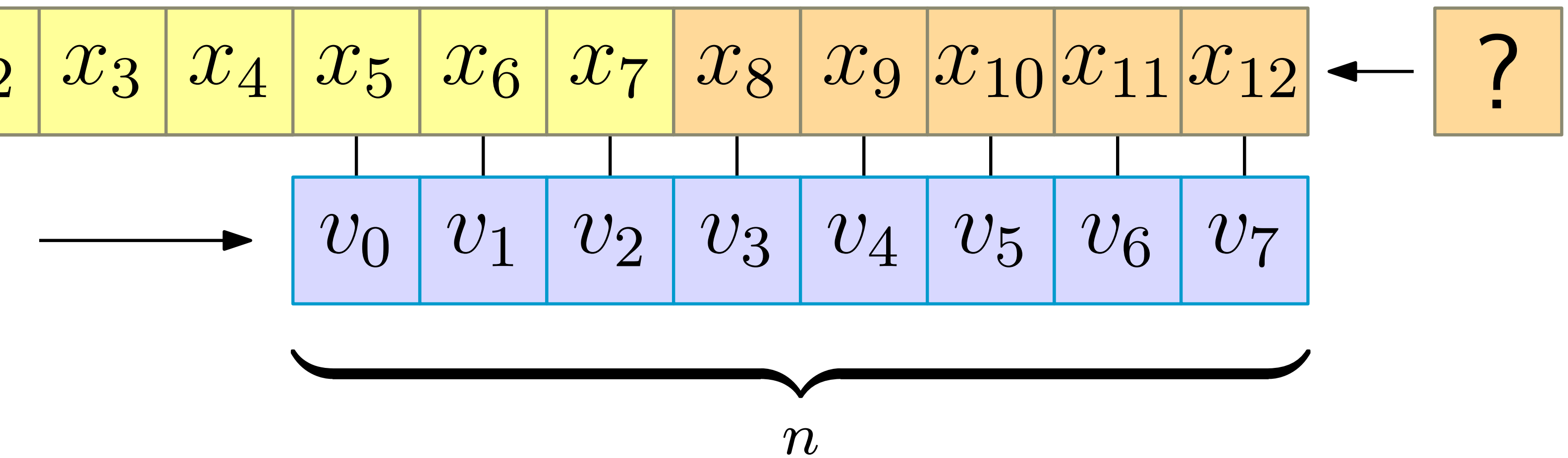


Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$



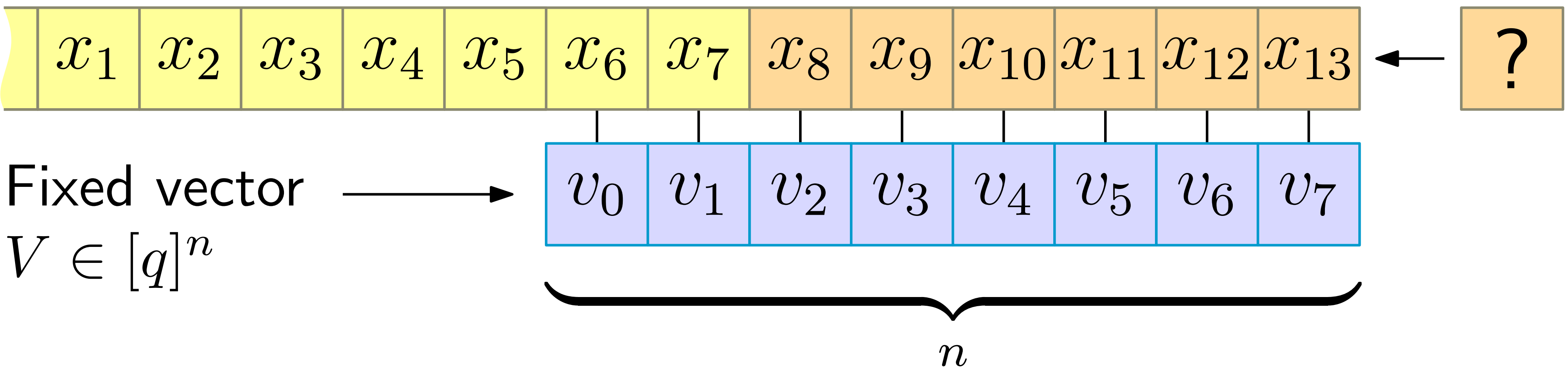
Fixed vector $V \in [q]^n$

Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$

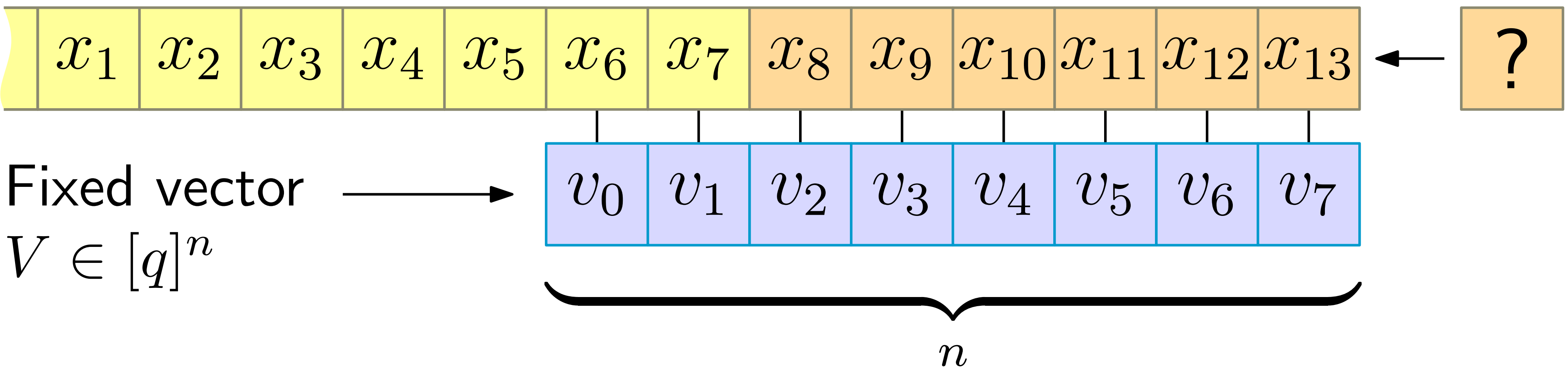


Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Convolution

Stream of numbers from $[q]$



Fixed vector
$V \in [q]^n$

Output dot product (modulo q):

$$V \cdot (\text{last } n \text{ digits of stream}) = \sum_{i=0}^{n-1} v_i x_{(i + \text{leftmost aligned index})}$$

Lower bound: $\Omega\left(\frac{\delta}{w}\log n\right)$

$\delta = \log q$, word size w.

C., Jalsenius. Lower Bounds for Online Integer Multiplication and Convolution in the Cell-Probe Mode. ICALP 2011

Previous bounds

M. J. Fischer and L. J. Stockmeyer
Fast on-line integer multiplication
STOC '73

C., K. Efremenko, B. Porat and E. Porat
A black box for online approximate pattern matching
Information and Computation 209(4):731–736, 2011

- $O(\log^2 n)$ time per arriving symbol (pair)

Previous bounds

M. J. Fischer and L. J. Stockmeyer
Fast on-line integer multiplication
STOC '73

C., K. Efremenko, B. Porat and E. Porat
A black box for online approximate pattern matching
Information and Computation 209(4):731–736, 2011

- $O(\log^2 n)$ time per arriving symbol (pair)

Offline cell probe complexity is linear!

$$\Rightarrow$$

online upper bound of $O(\log n)$

Previous bounds

M. J. Fischer and L. J. Stockmeyer
Fast on-line integer multiplication
STOC '73

C., K. Efremenko, B. Porat and E. Porat
A black box for online approximate pattern matching
Information and Computation 209(4):731–736, 2011

- $O(\log^2 n)$ time per arriving symbol (pair)

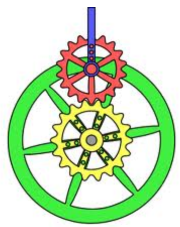
Better online lower bound

$\Rightarrow$

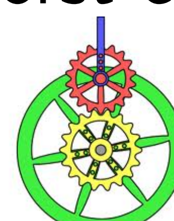
super linear lower bound for
offline convolution and multiplication

Yao's minimax principle

A lower bound on the expected running time for

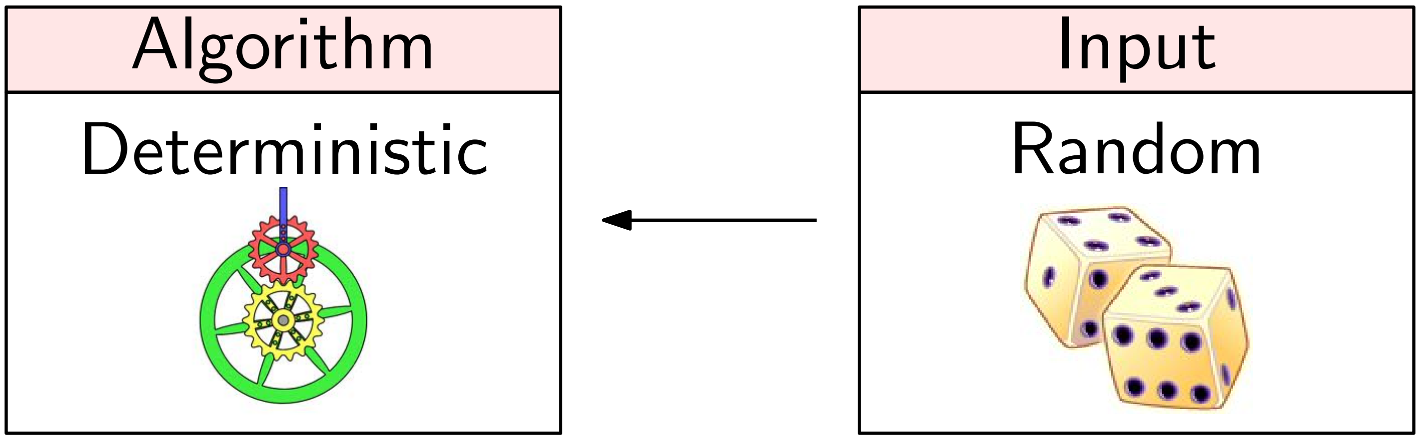
Algorithm		Input
Deterministic 	←	Random 

implies that the same lower bound holds for

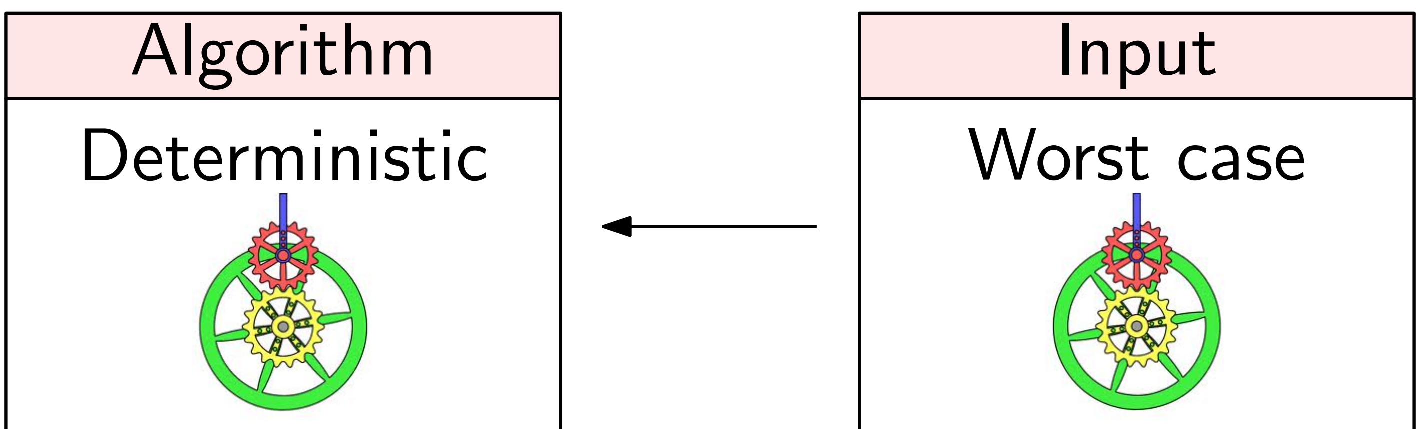
Algorithm		Input
Randomised 	←	Worst case 

Yao's minimax principle

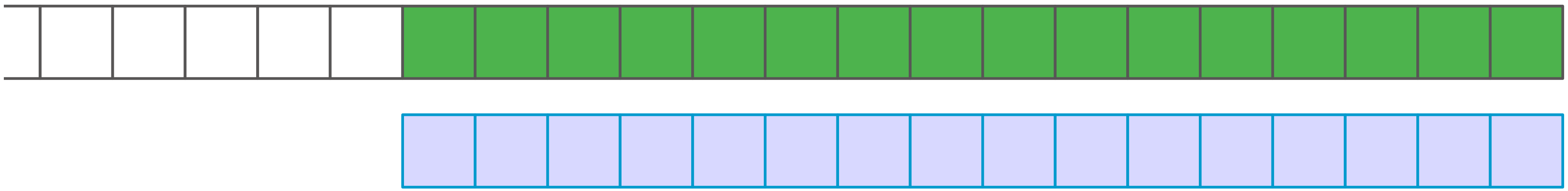
A lower bound on the expected running time for

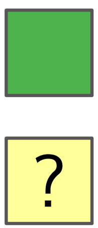


implies that the same lower bound holds for



Information transfer

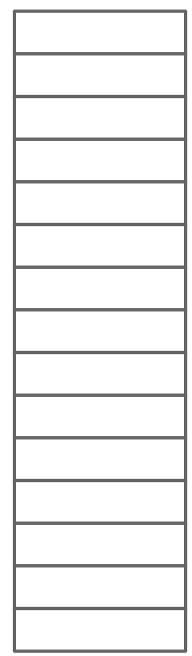




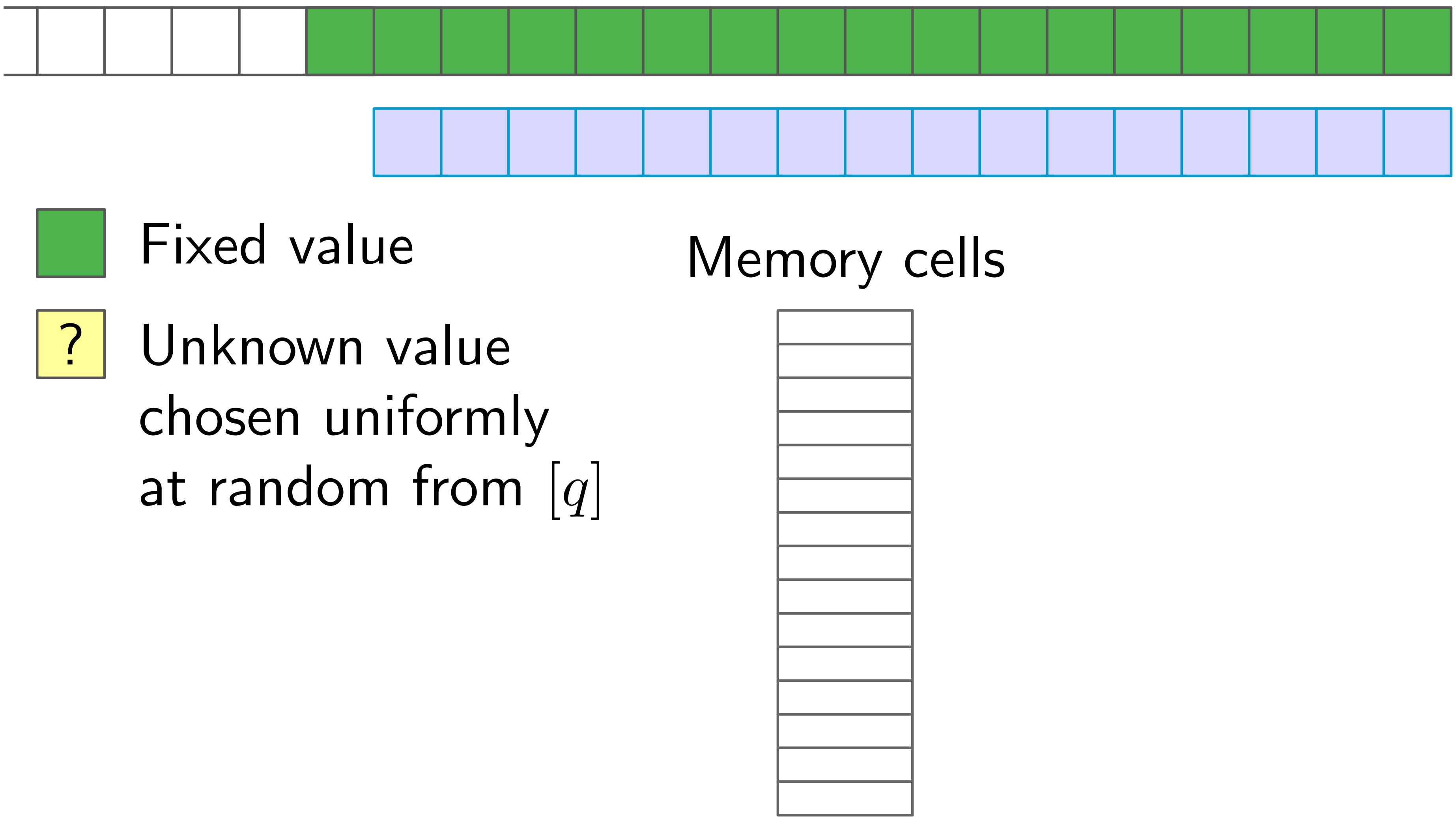
Fixed value

Unknown value chosen uniformly at random from $[q]$

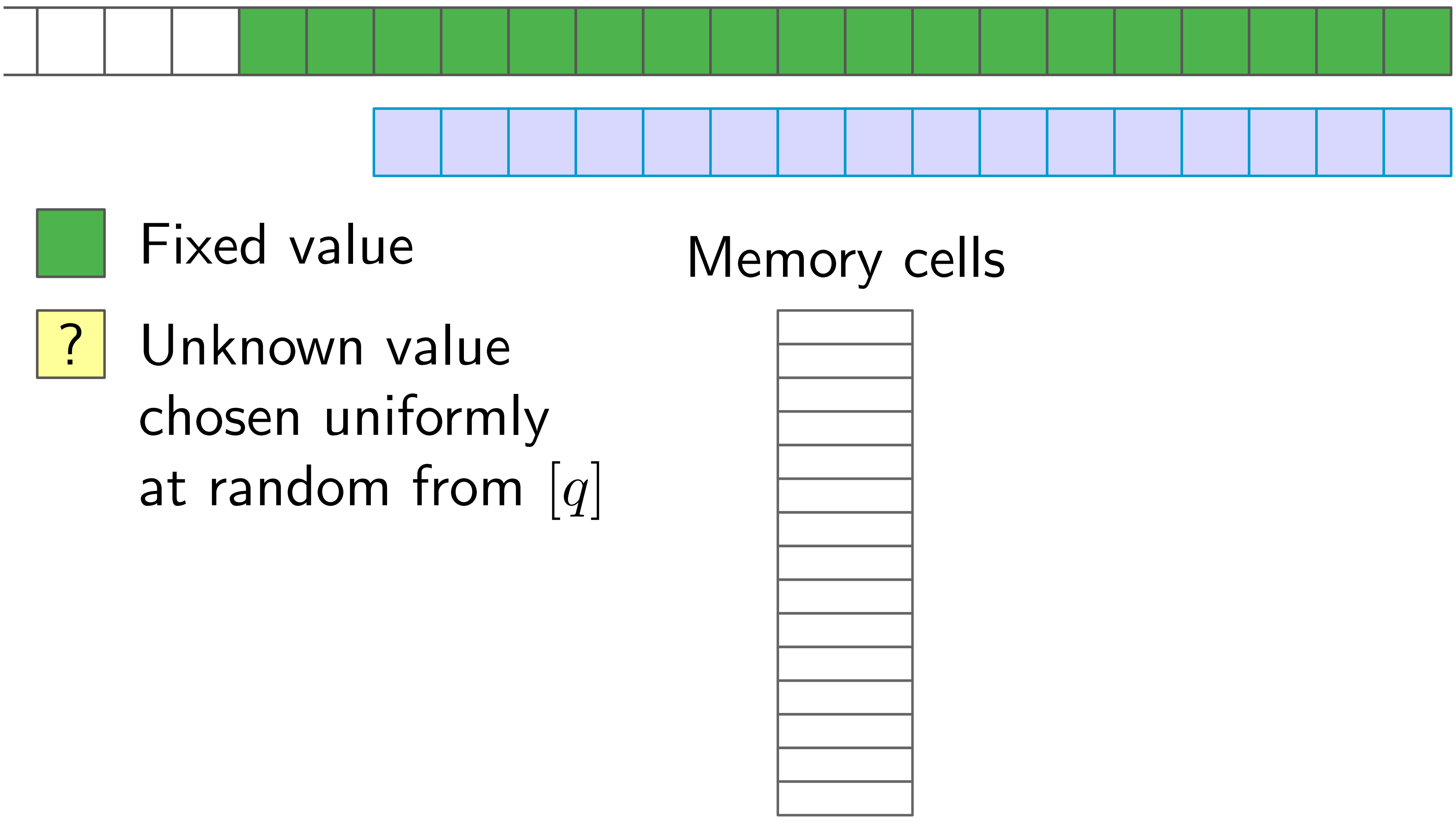
Memory cells



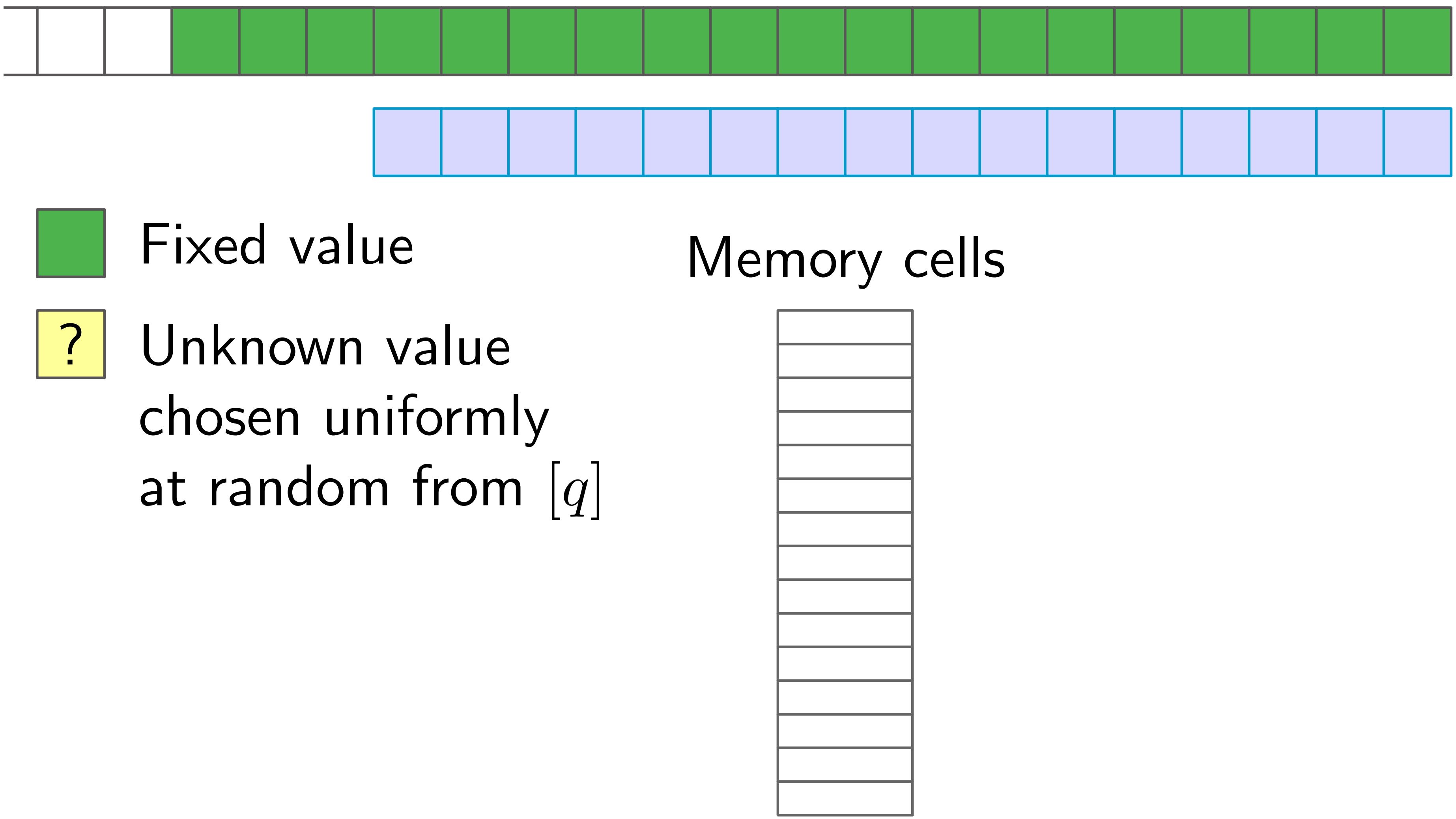
Information transfer



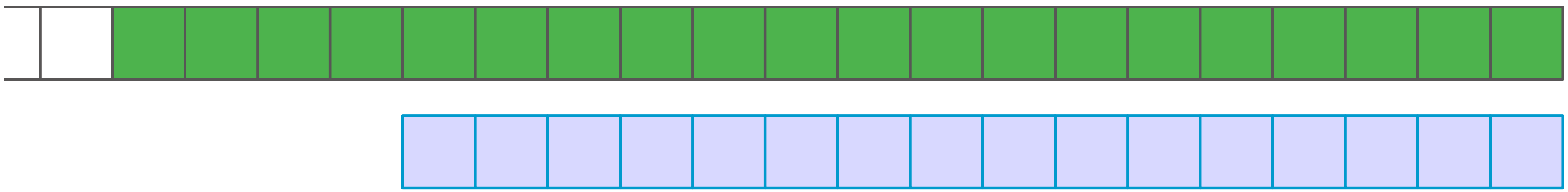
Information transfer



Information transfer



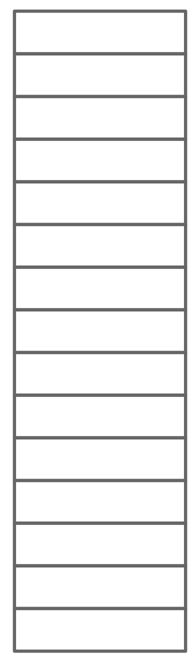
Information transfer



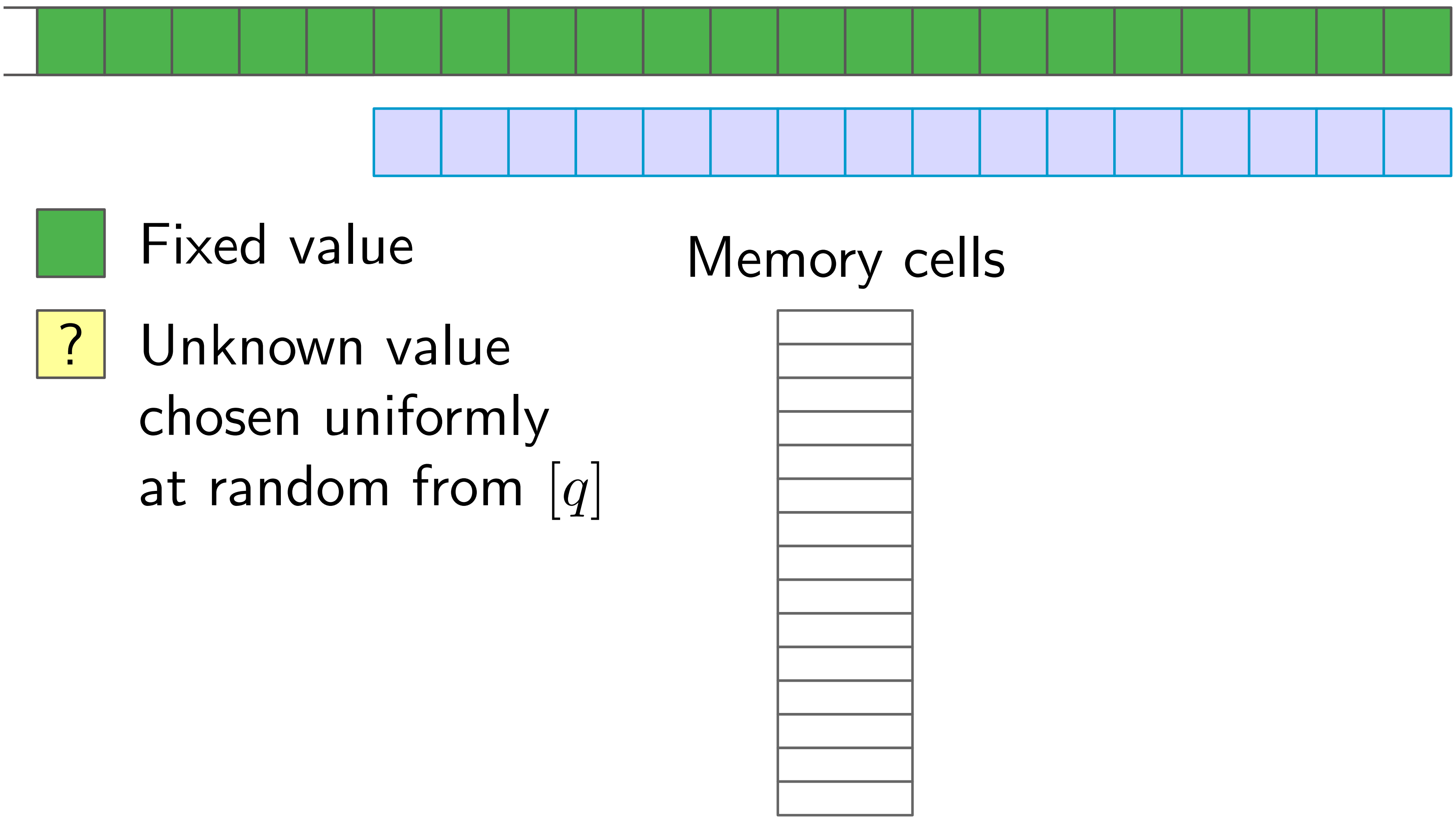
Fixed value

? Unknown value chosen uniformly at random from $[q]$

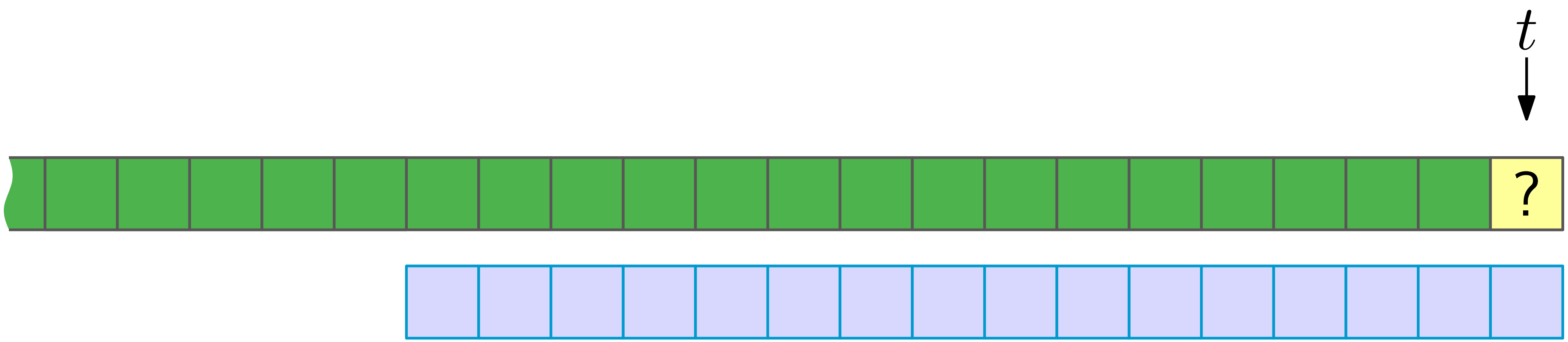
Memory cells



Information transfer



Information transfer

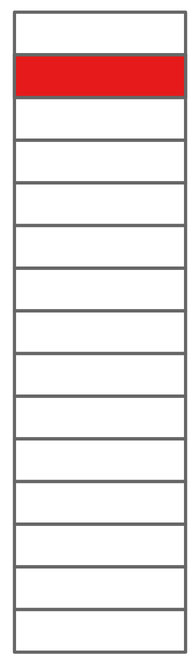


Fixed value

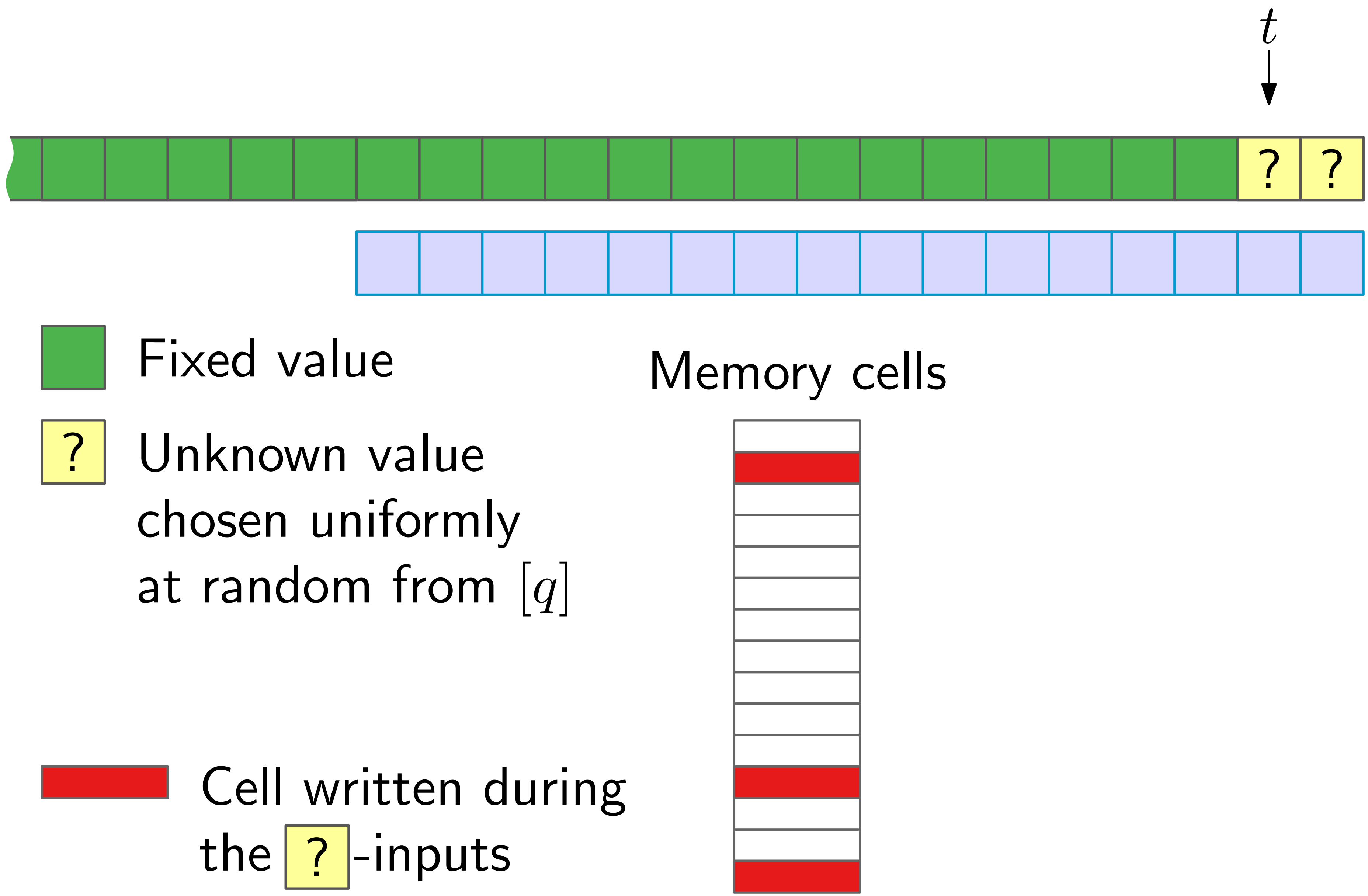
? Unknown value chosen uniformly at random from $[q]$

Cell written during the $\boxed{?}$-inputs

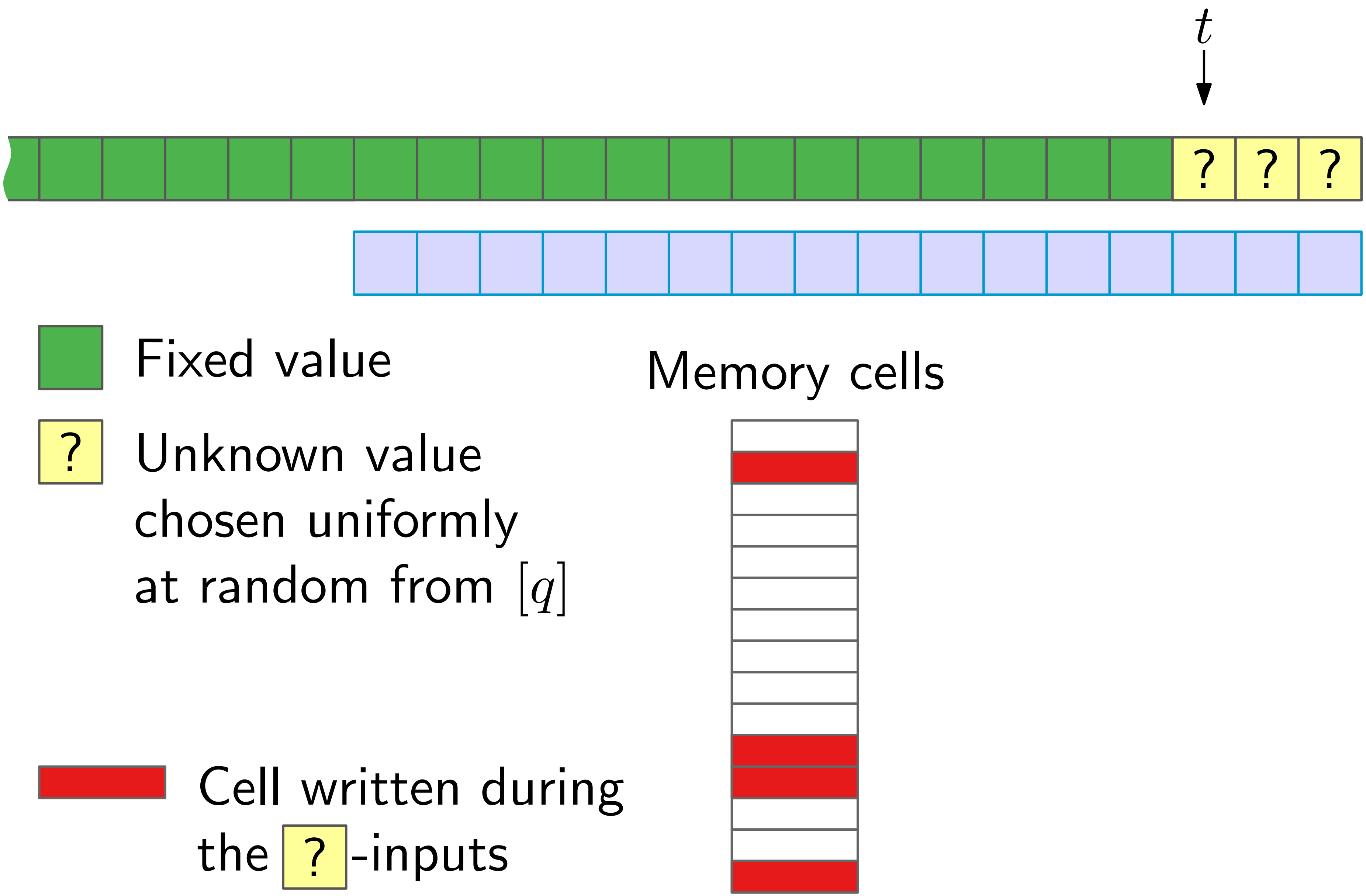
Memory cells



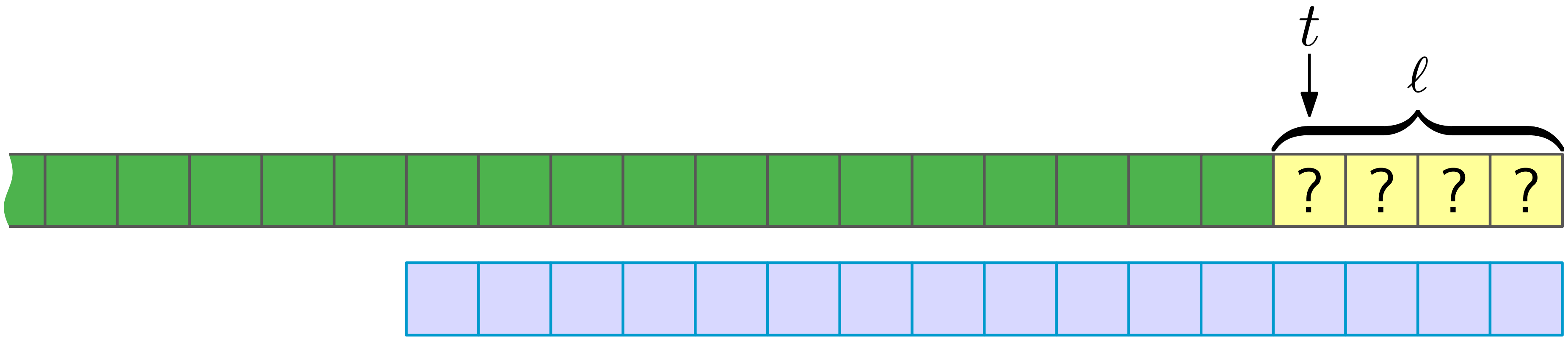
Information transfer



Information transfer



Information transfer

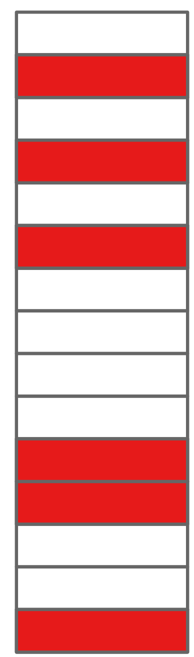


Fixed value

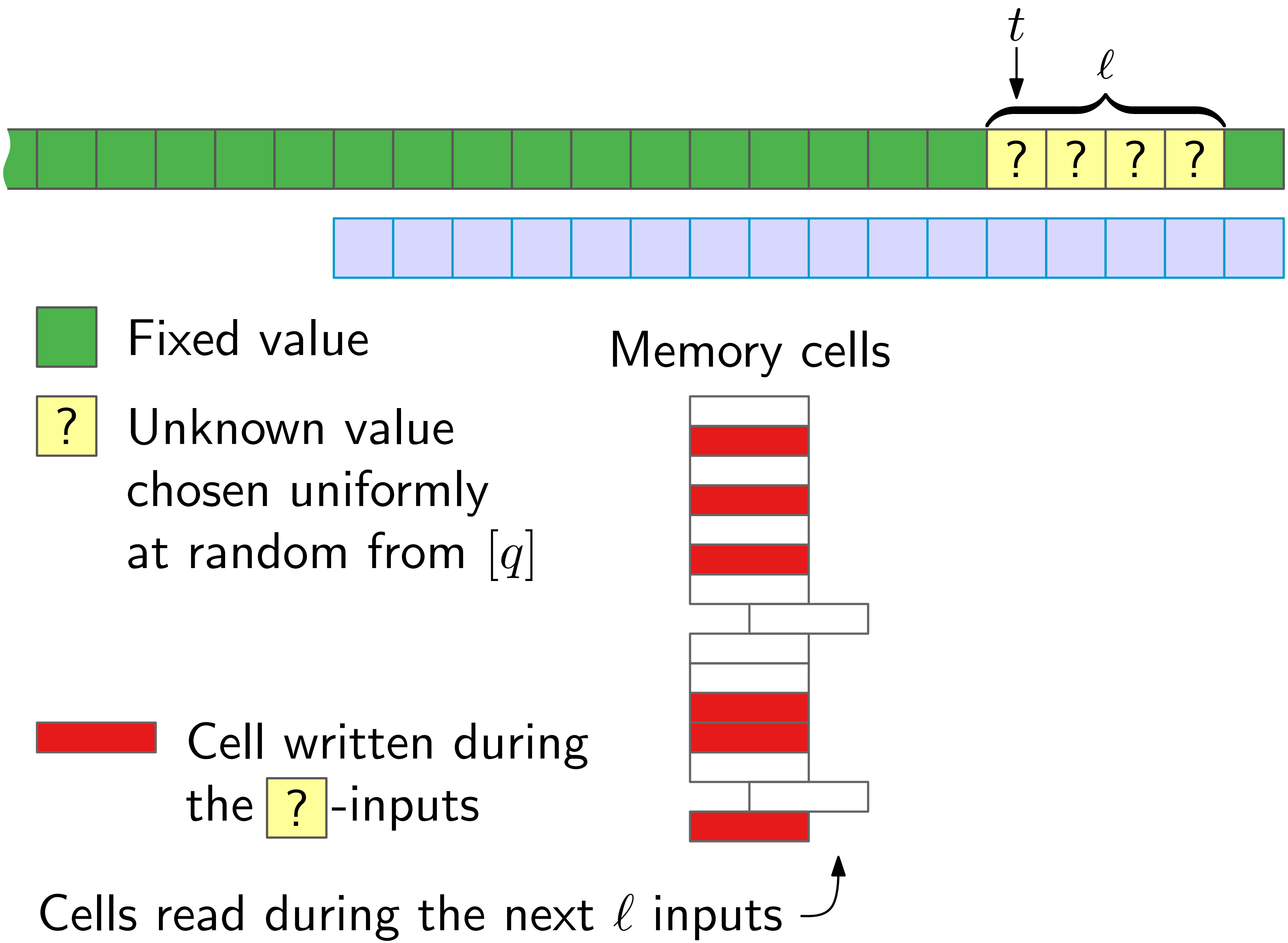
? Unknown value chosen uniformly at random from $[q]$

Cell written during the ? -inputs

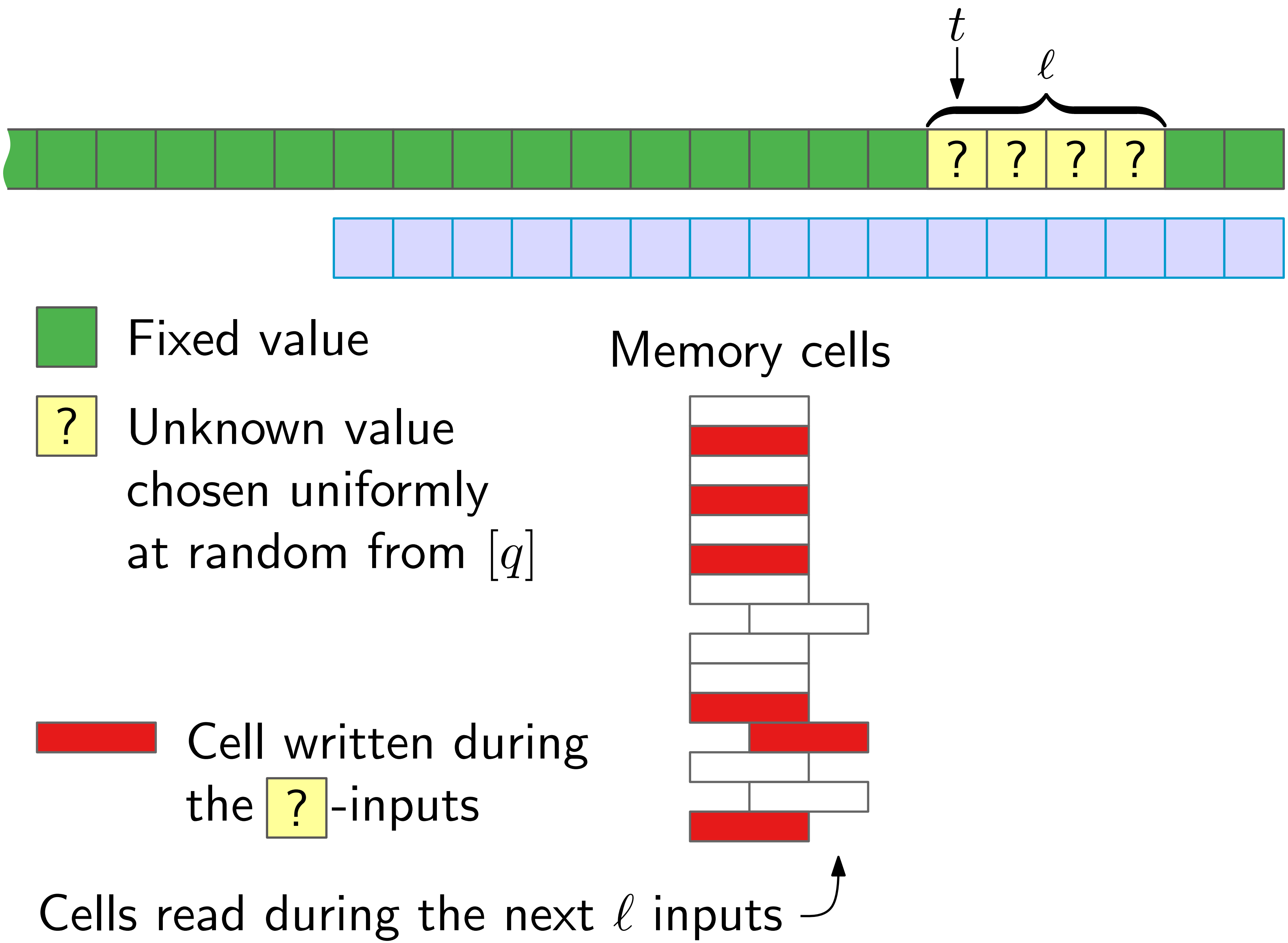
Memory cells



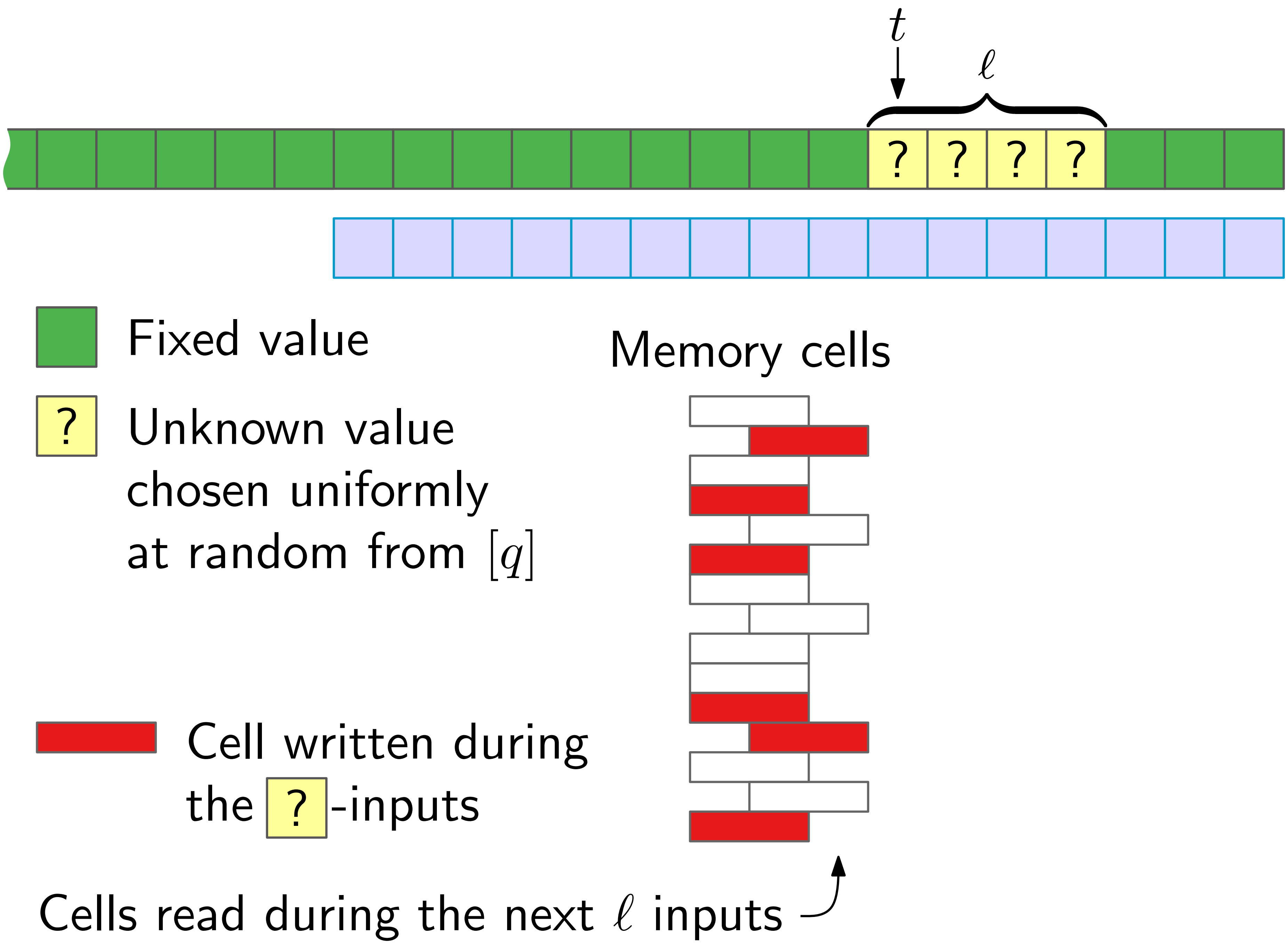
Information transfer



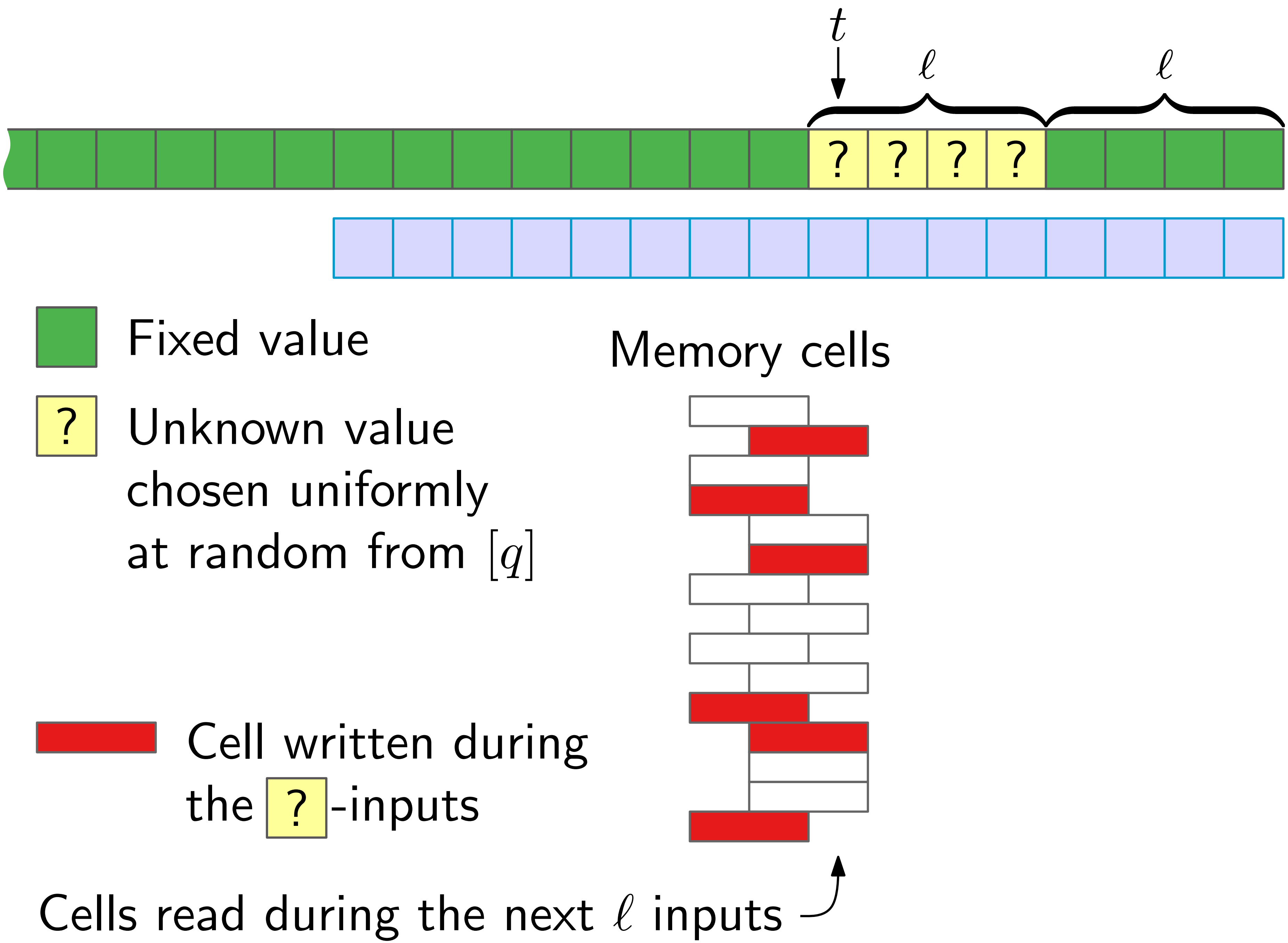
Information transfer



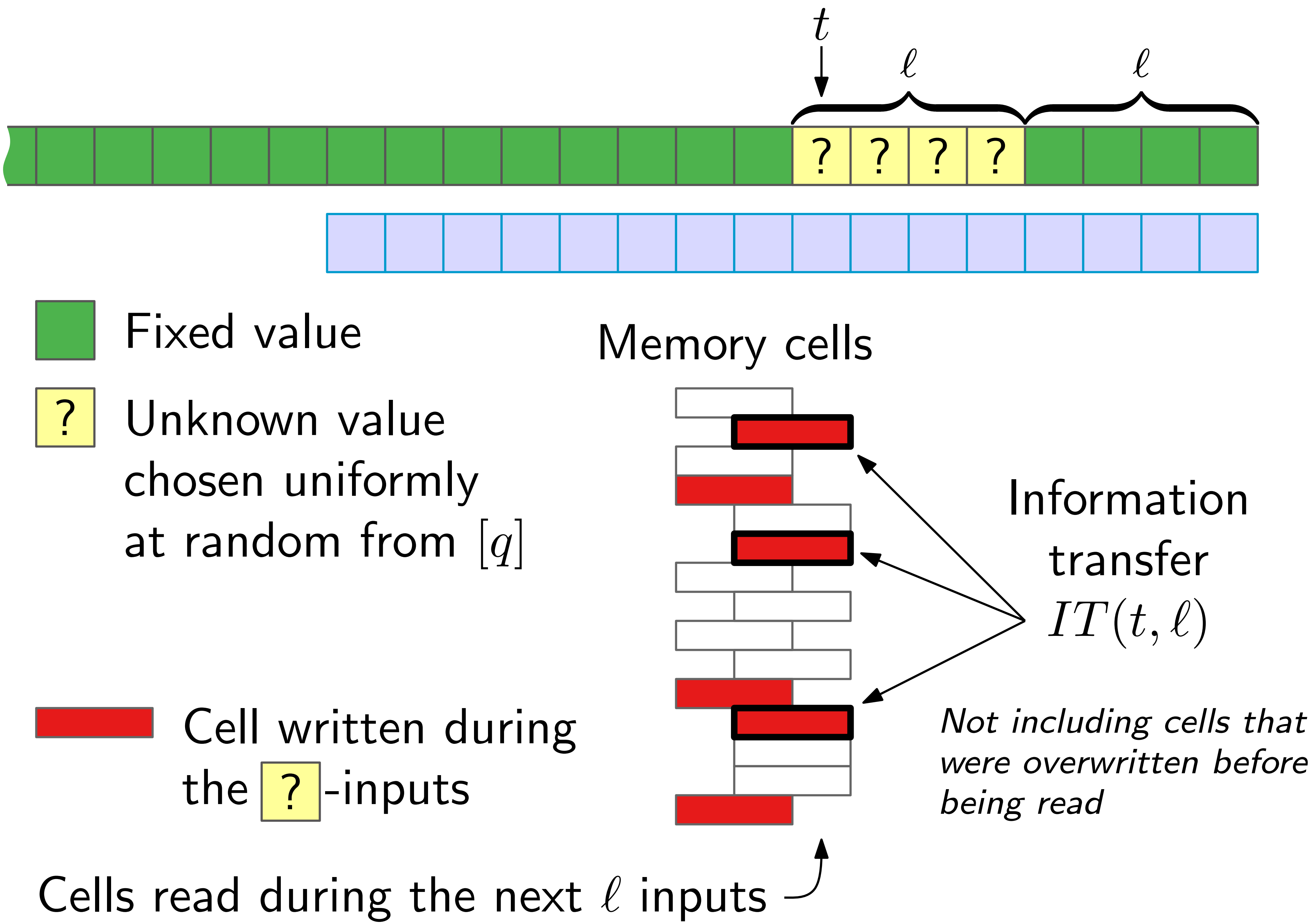
Information transfer



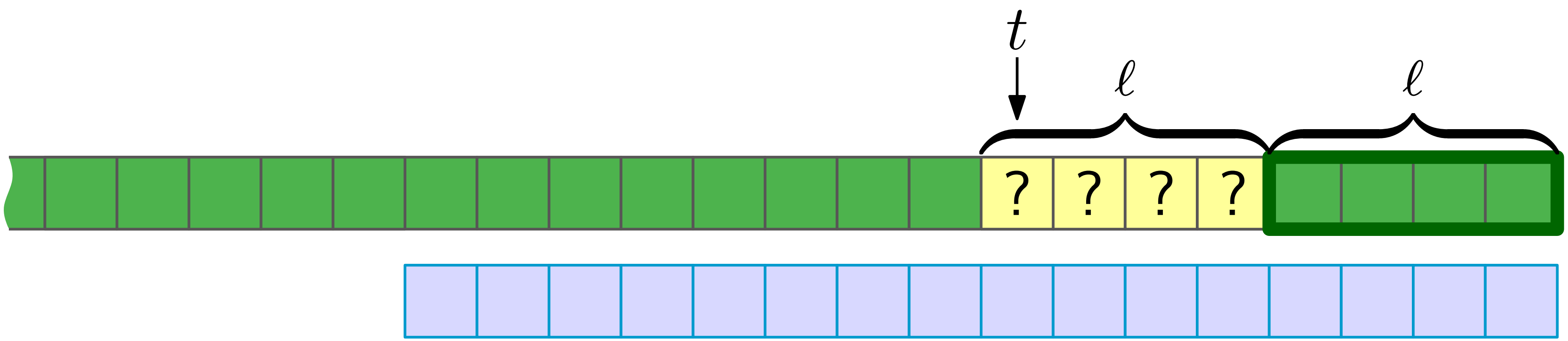
Information transfer



Information transfer

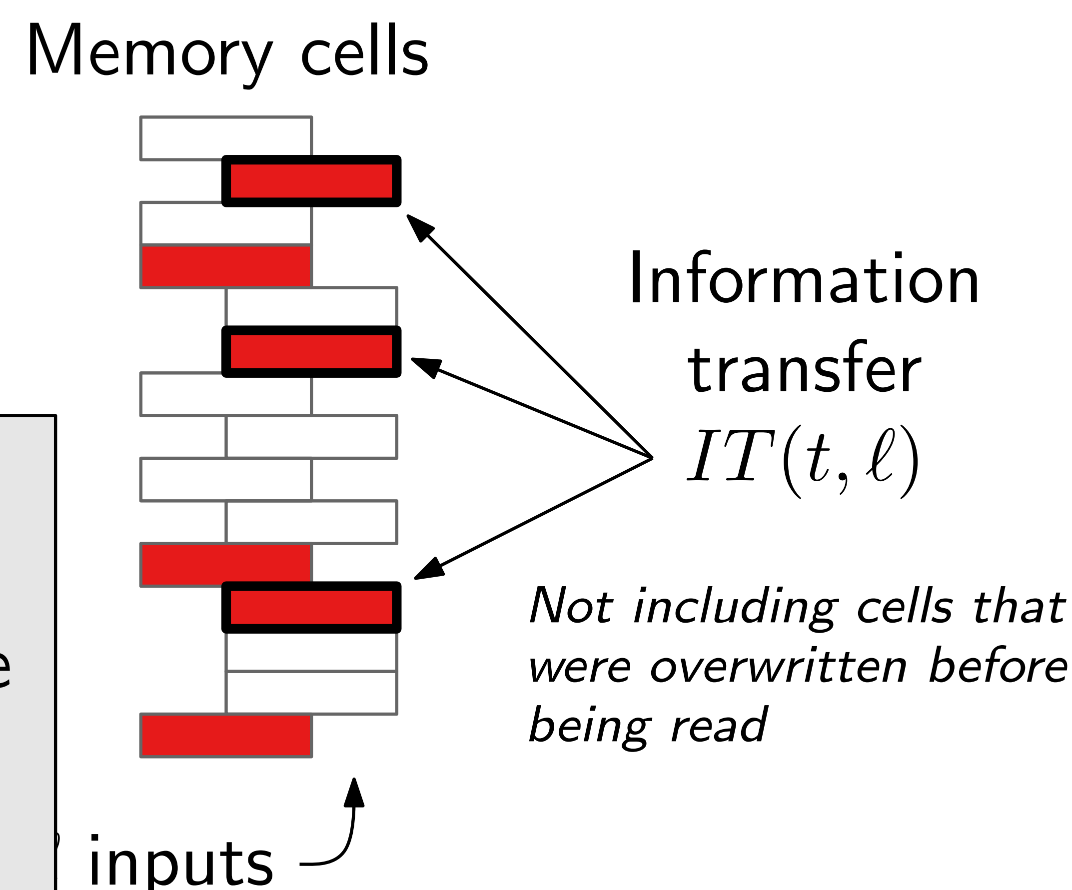


Information transfer



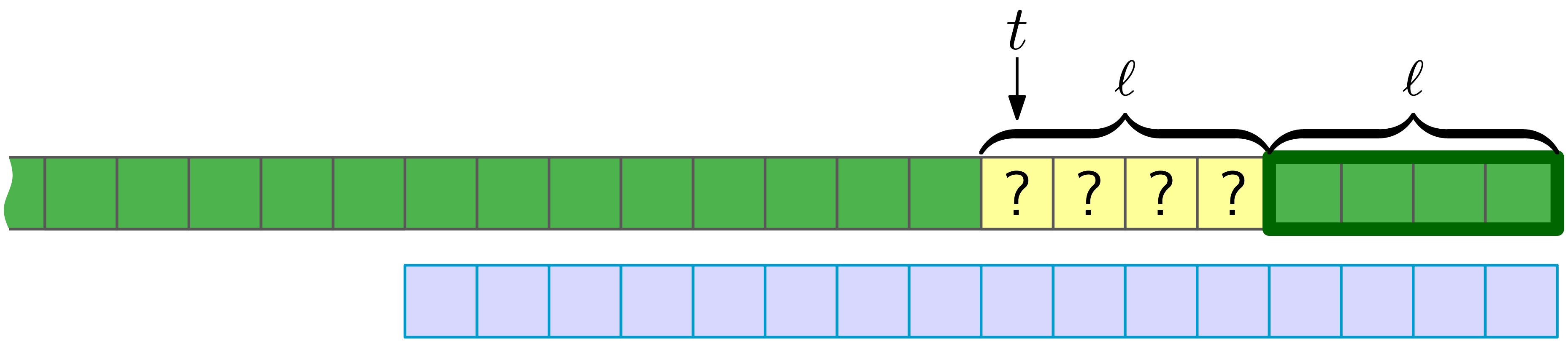
Fixed value

? Unknown value chosen uniformly at random from $[q]$



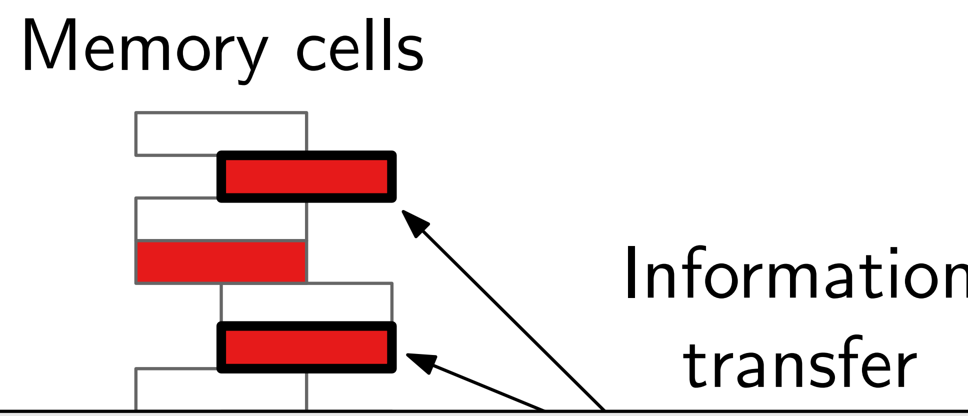
The cells in $IT(t,\ell)$ provide sufficient information in order to give correct output during

Information transfer



Fixed value

? Unknown value chosen uniformly at random from $[q]$



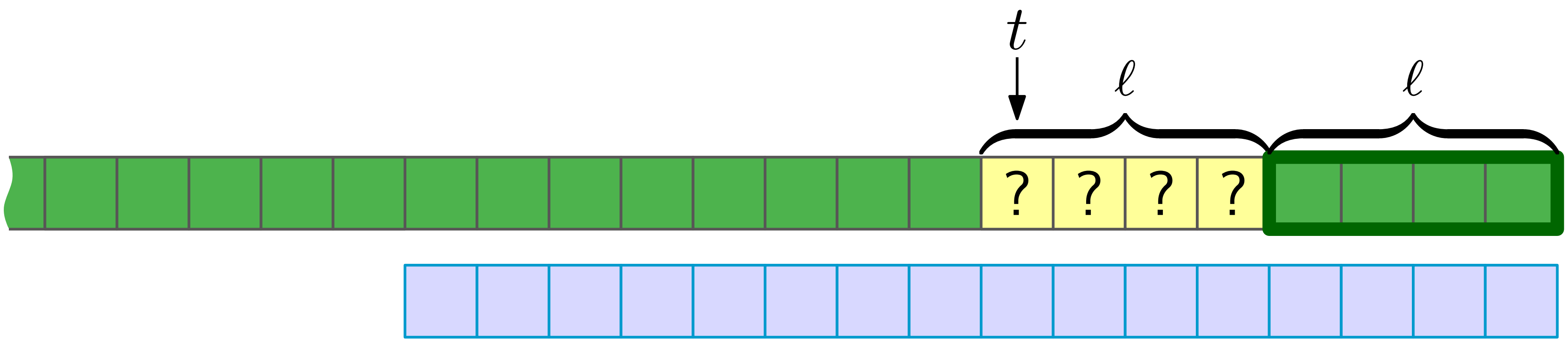
The conditional entropy

$$H(\text{the outputs during} \quad \mid \text{all} \quad \text{fixed})$$

$$\leqslant w + 2w \cdot \mathbb{E}\left[|IT(t,\ell)| \;\mid \text{all} \quad \text{fixed}\right]$$

w bits per cell

Information transfer



Fixed value

? Unknown value chosen uniformly at random from $[q]$

Cell	Address	Contents
	00124	76112
	34123	88819
	92540	01882

$|IT(t,\ell)|$

w bits to encode $|IT(t,\ell)|$

w bits $\quad$ w bits

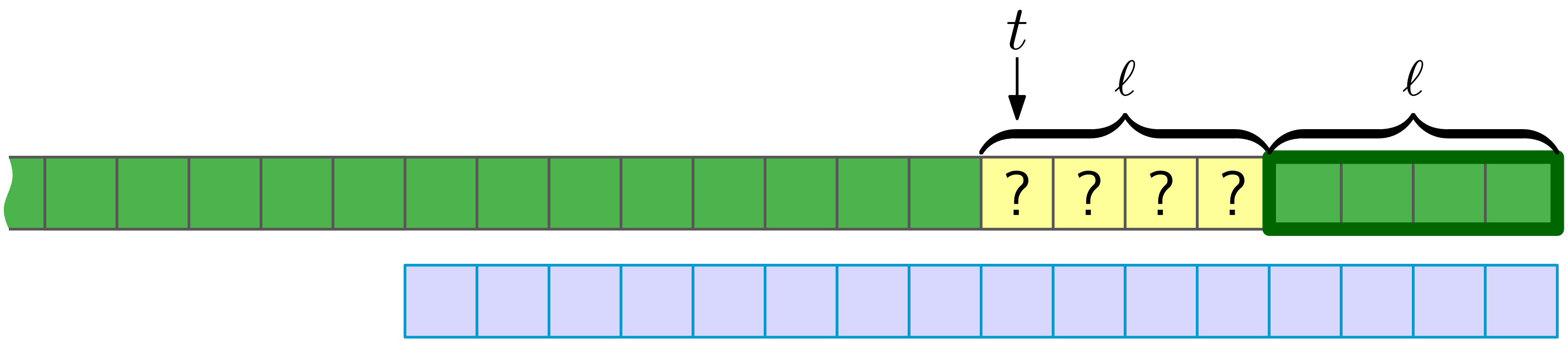
The conditional entropy

$H($the outputs during $\mid$ all fixed$)$

$$\leqslant w + 2w \cdot \mathbb{E}\left[|IT(t,\ell)| \mid \text{all fixed}\right]$$

w bits per cell

Information transfer



Fixed value

? Unknown value chosen uniformly at random from $[q]$

Cell	Address	Contents
	00124	76112
	00000	00000
	92540	01882

$|IT(t,\ell)|$

w bits to encode $|IT(t,\ell)|$

w bits

w bits

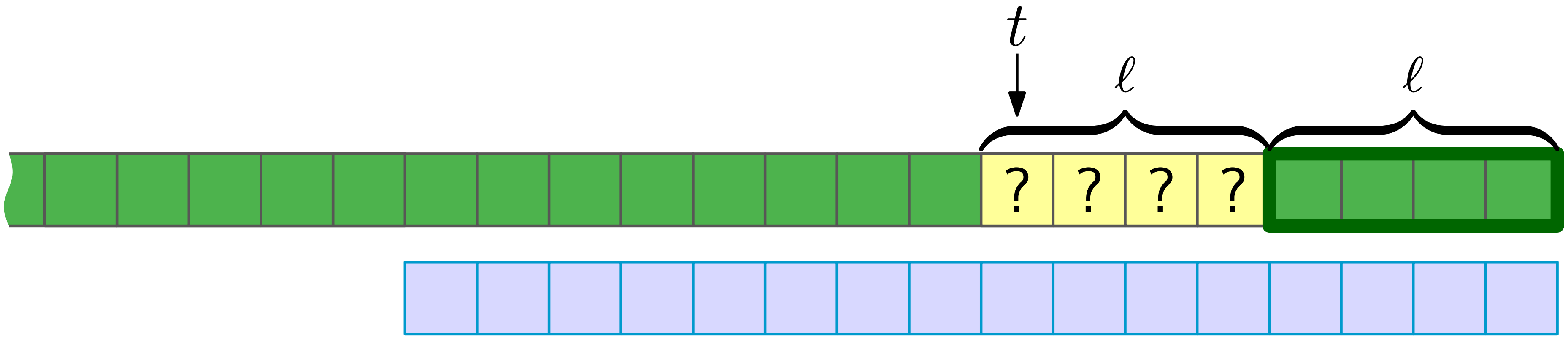
The conditional entropy

H(the outputs during | all fixed)

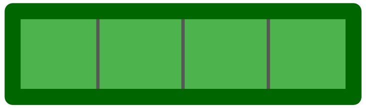
$\leqslant w + 2w \cdot \mathbb{E}\left[|IT(t,\ell)| \mid \text{all fixed}\right]$

w bits per cell

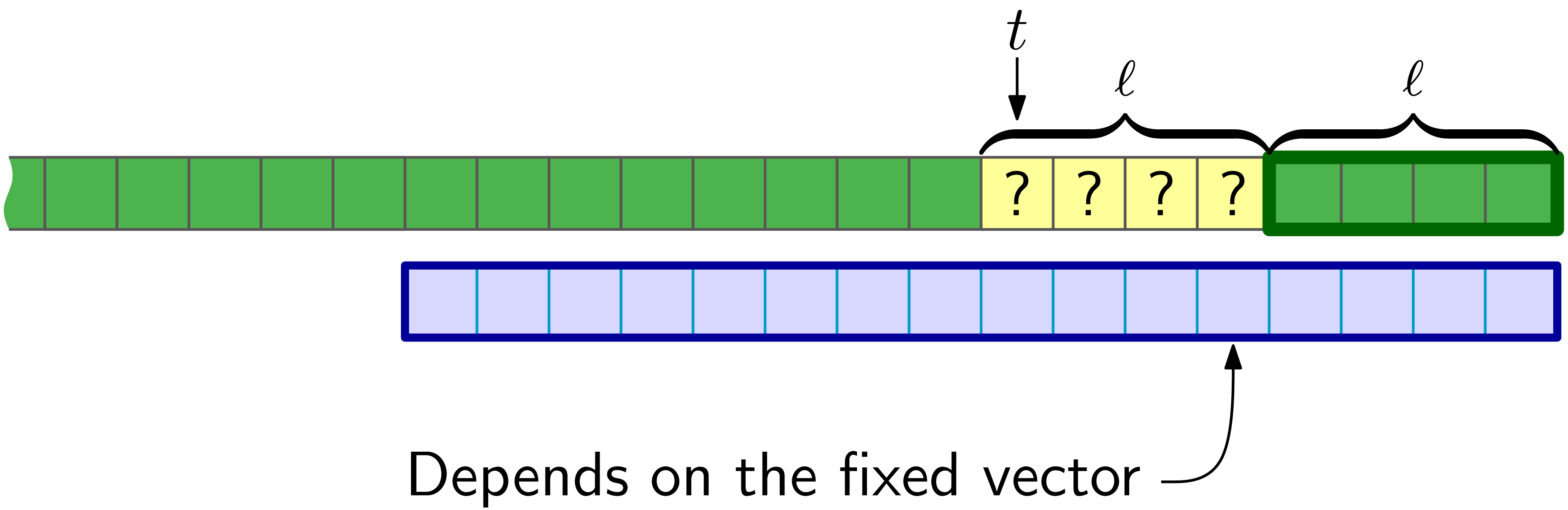
Information transfer



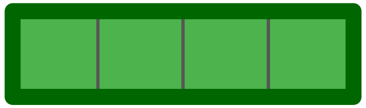
How much information about ? ? ? ? do we **need**

in order to give correct outputs during  ?

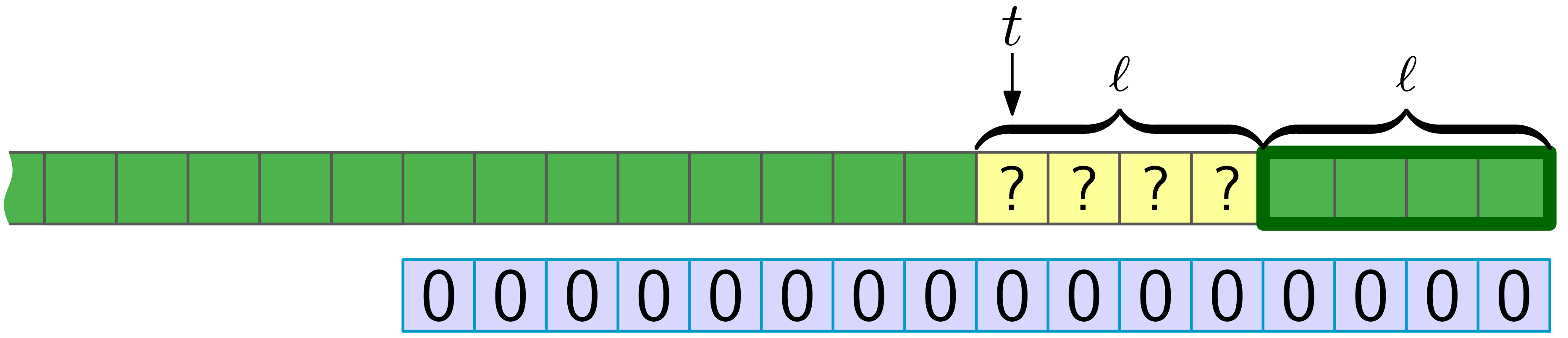
Information transfer



How much information about $\boxed{?\;?\;?\;?}$ do we **need**

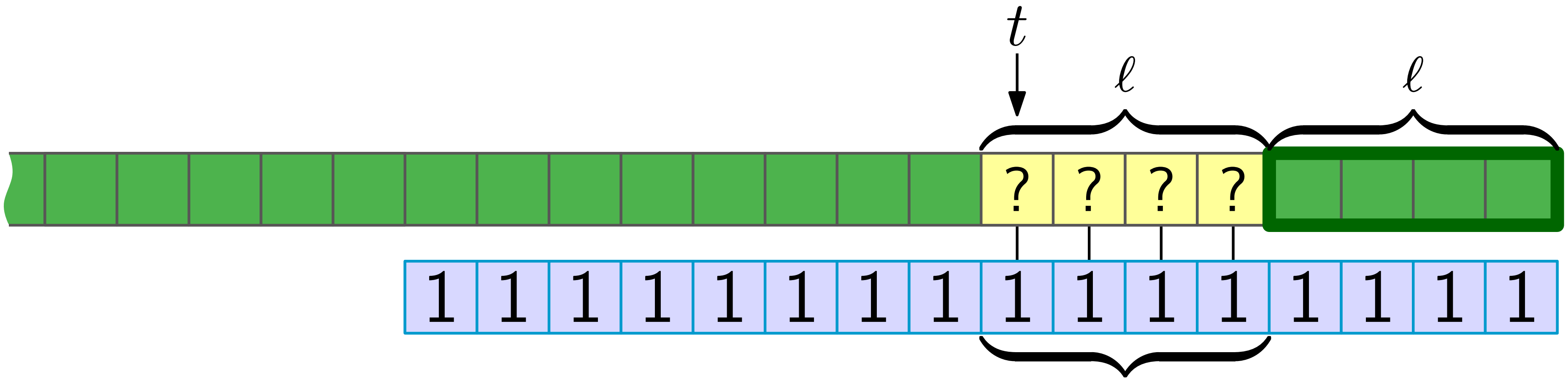
in order to give correct outputs during  ?

Information transfer



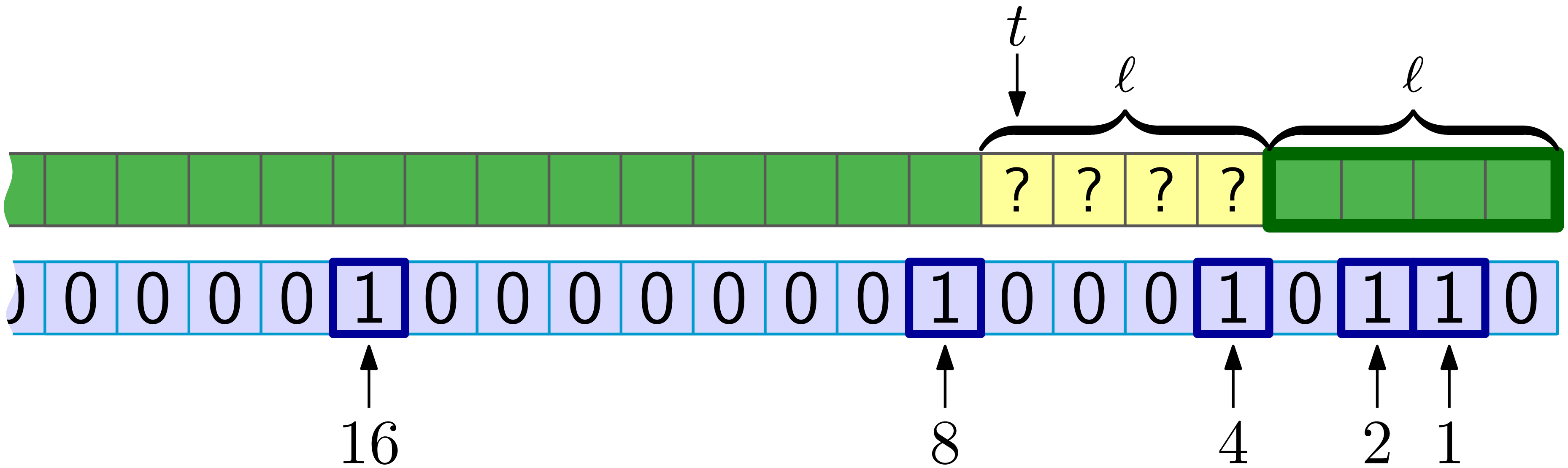
Output is always 0 (no information)

Information transfer



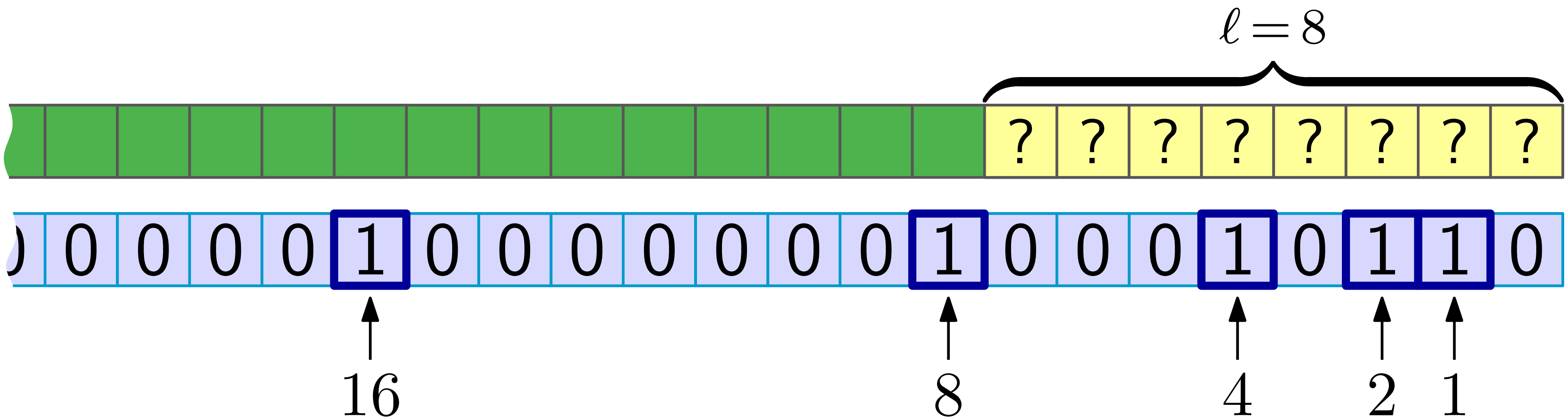
Contributes to the dot product with the same value at each alignment
($\delta = \log q$ bits of information)

Information transfer



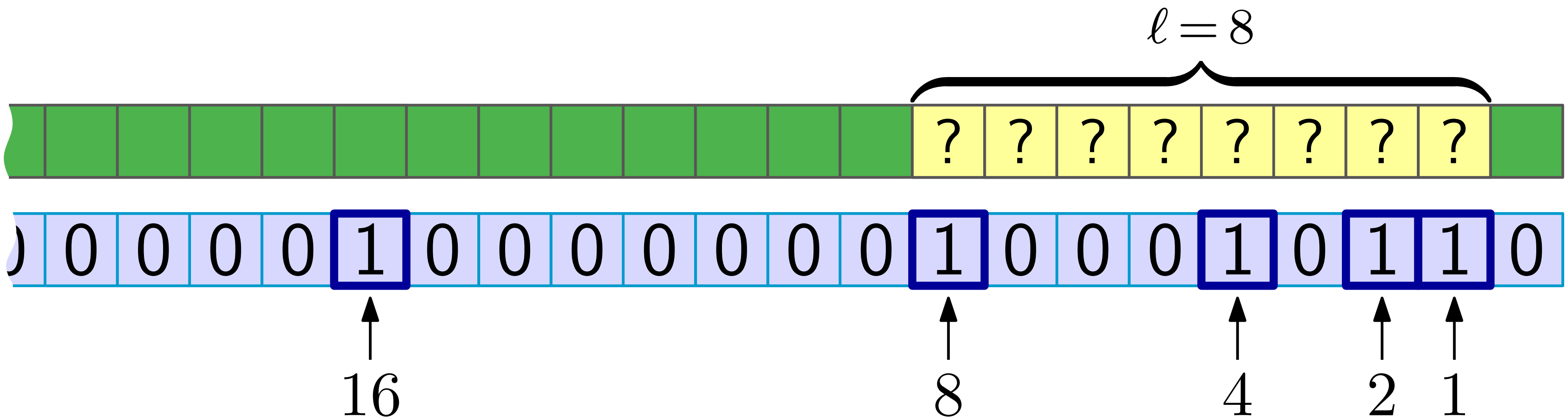
$\boxed{1}$ if the position is a power of 2

Information transfer



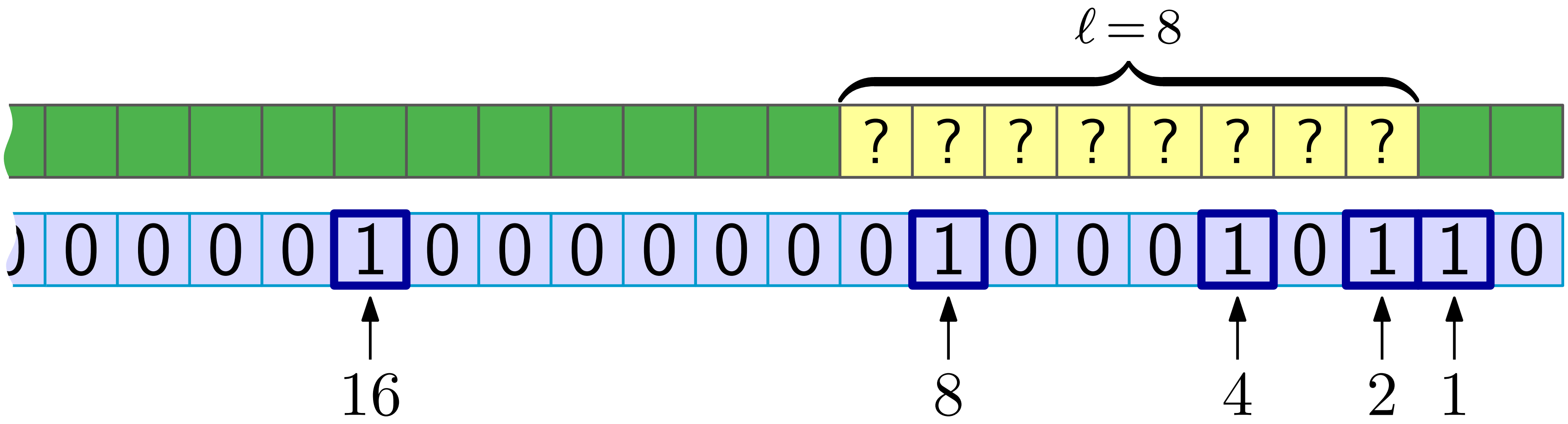
$\boxed{1}$ if the position is a power of 2

Information transfer



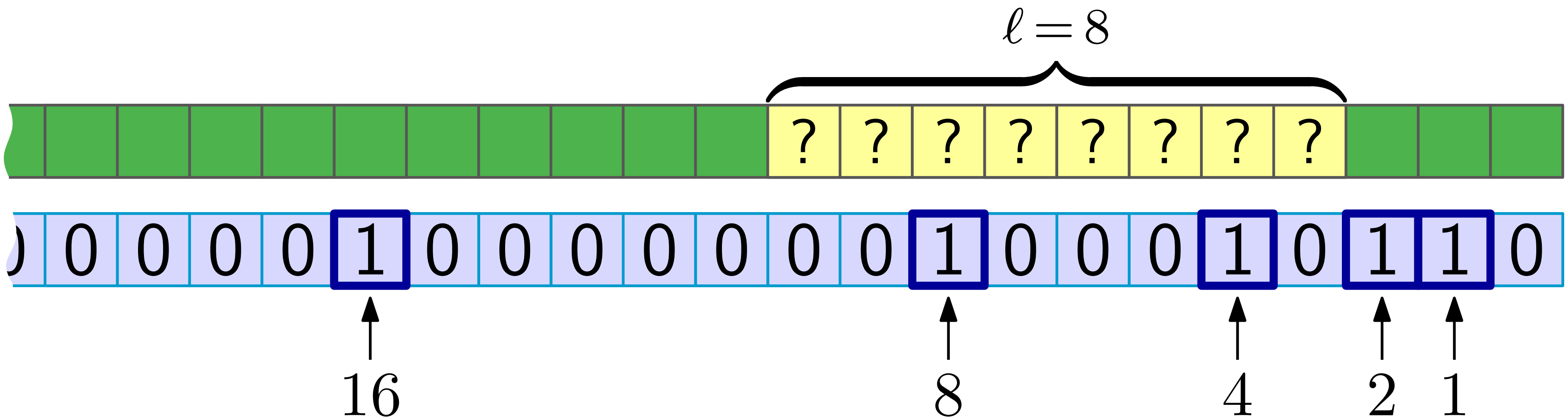
$\boxed{1}$ if the position is a power of 2

Information transfer



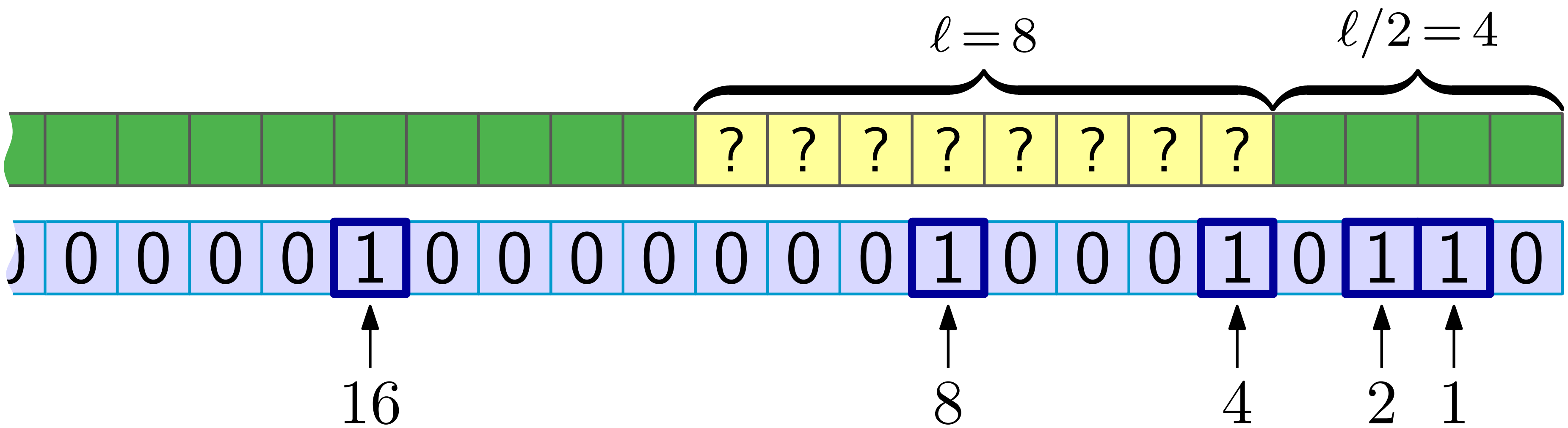
$\boxed{1}$ if the position is a power of 2

Information transfer



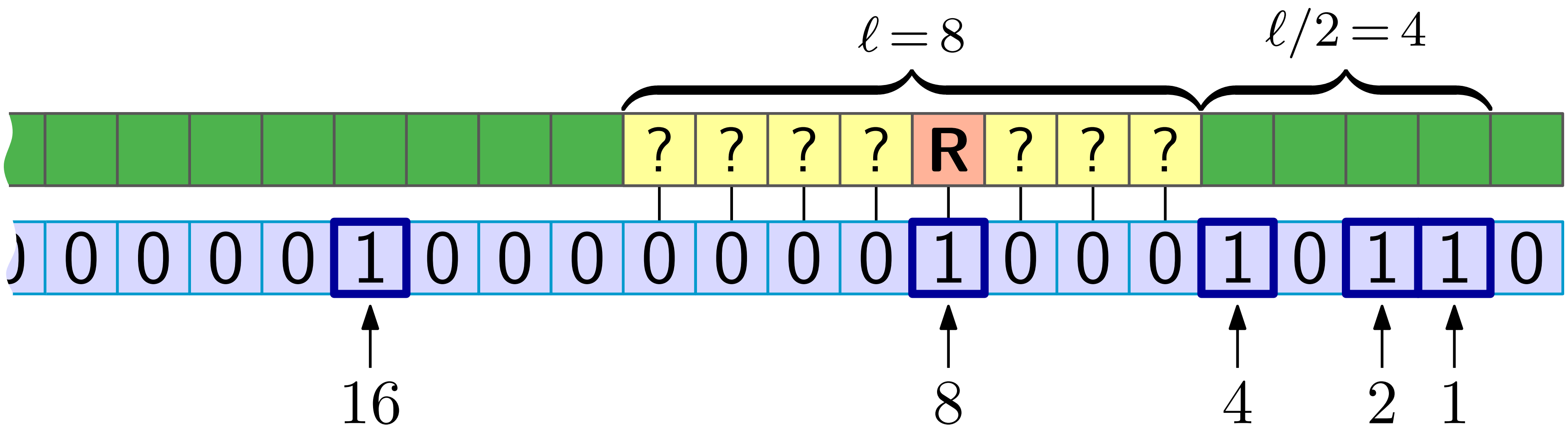
$\boxed{1}$ if the position is a power of 2

Information transfer



$\boxed{1}$ if the position is a power of 2

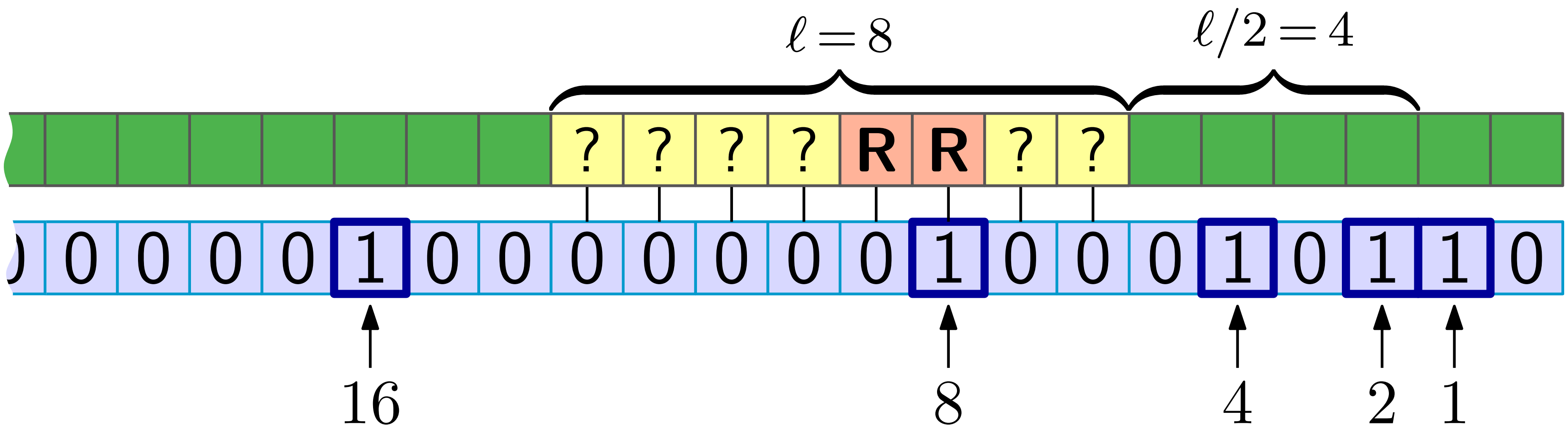
Information transfer



$\boxed{1}$ if the position is a power of 2

R = a recovered value
(recall that $\boxed{?}$ is chosen uniformly at random from $[q]$, hence contributes with $\delta = \log q$ bits to the entropy)

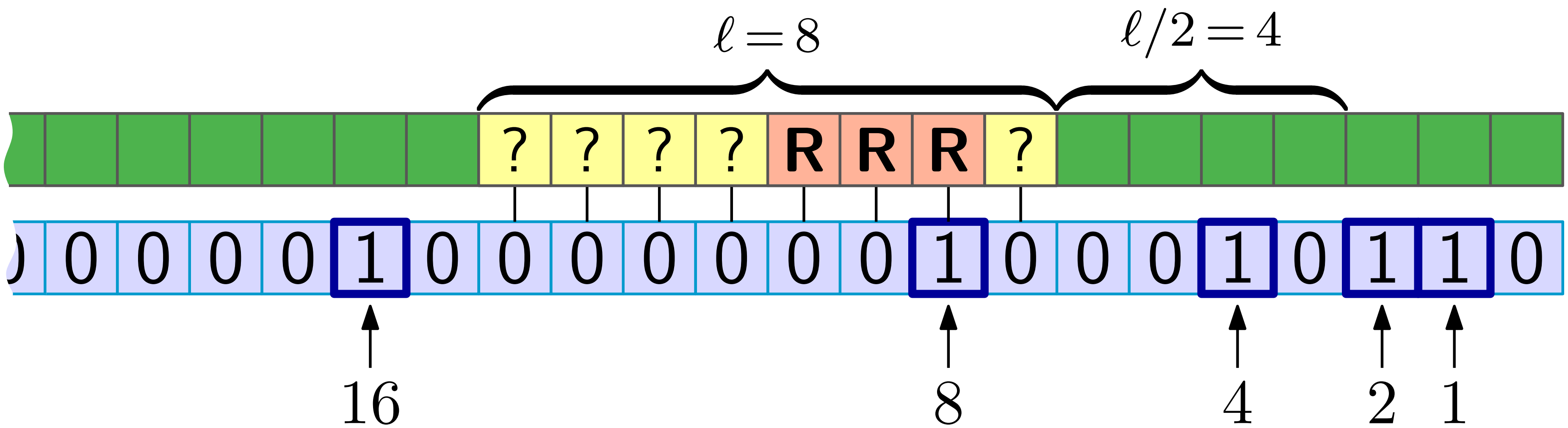
Information transfer



1 if the position is a power of 2

R = a recovered value
(recall that ? is chosen uniformly at random from $[q]$, hence contributes with $\delta = \log q$ bits to the entropy)

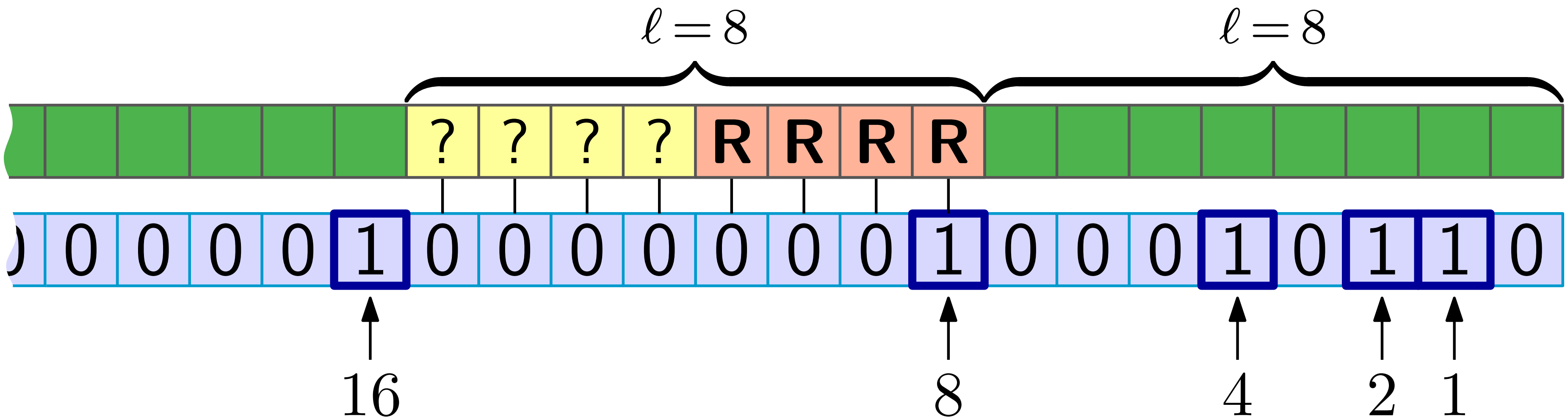
Information transfer



1 if the position is a power of 2

R = a recovered value
(recall that ? is chosen uniformly at random from $[q]$, hence contributes with $\delta = \log q$ bits to the entropy)

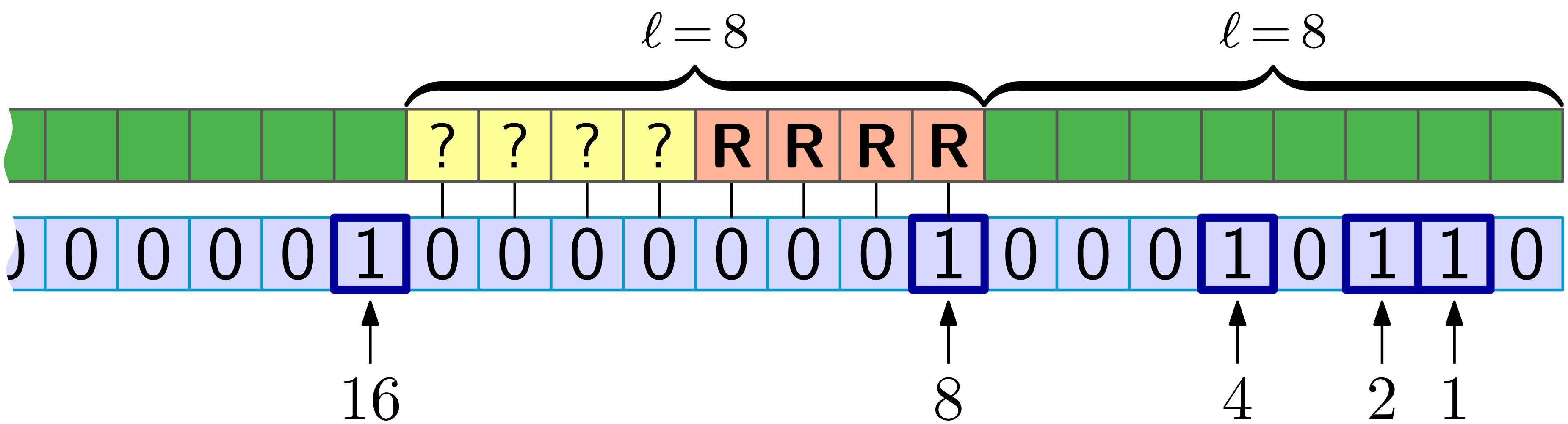
Information transfer



$\boxed{1}$ if the position is a power of 2

$\boxed{\textbf{R}}$ $=$ a recovered value
(recall that $\boxed{?}$ is chosen uniformly at random from $[q]$, hence contributes with $\delta = \log q$ bits to the entropy)

Information transfer

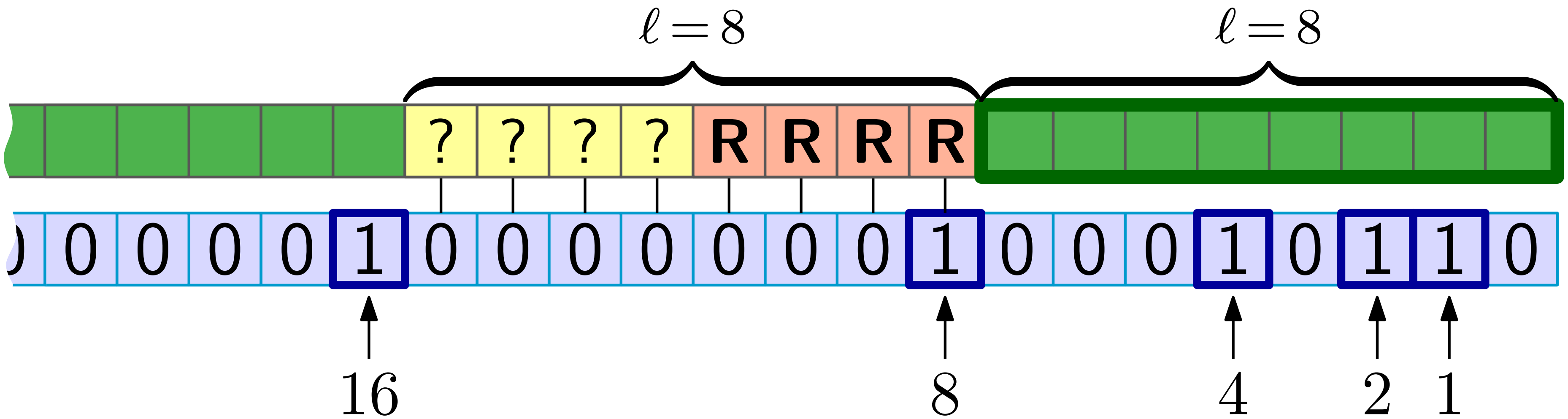


$\boxed{1}$ if the position is a power of 2

R = a recovered value (recall that $\boxed{?}$ is chosen uniformly at random from $[q]$, hence contributes with $\delta = \log q$ bits to the entropy)

Conclusion: If ℓ is a power of 2 then we recover $\frac{\ell}{2}$ values

Information transfer

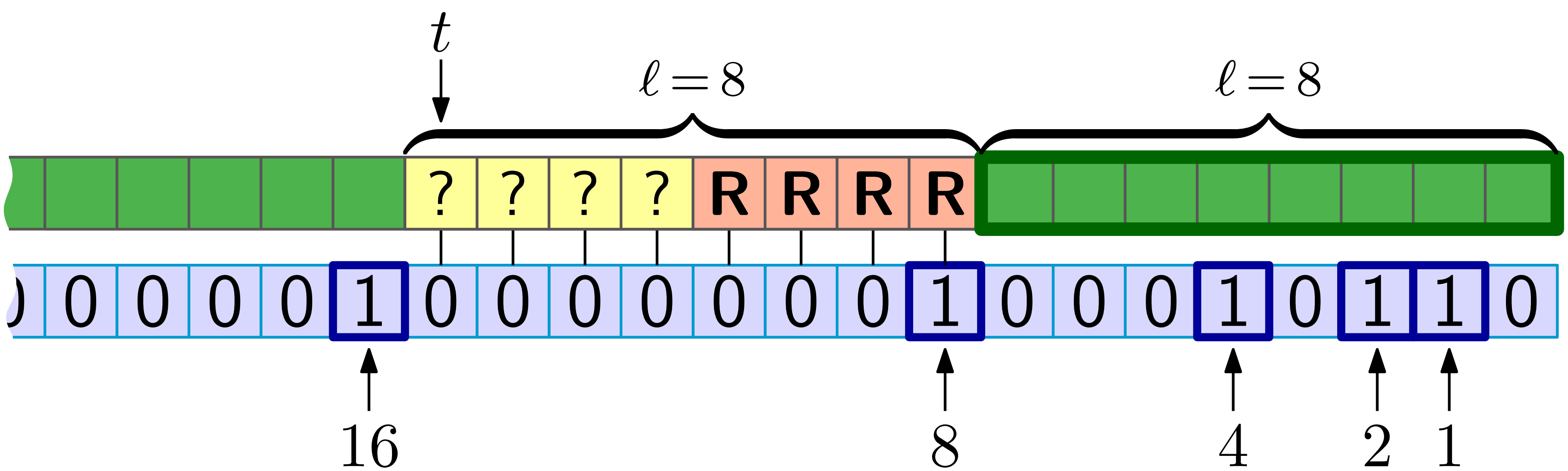


The conditional entropy

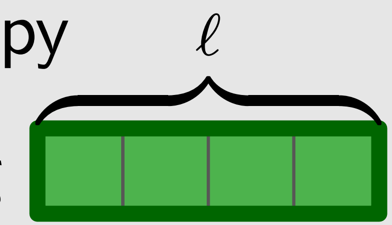
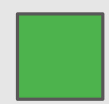
$$H(\text{the outputs during } \overbrace{\boxed{}}^{\ell} \mid \text{all } \boxed{} \text{ fixed}) \geqslant \frac{\ell}{2}\delta$$

Conclusion: If ℓ is a power of 2 then we recover $\frac{\ell}{2}$ values

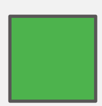
Information transfer



The conditional entropy

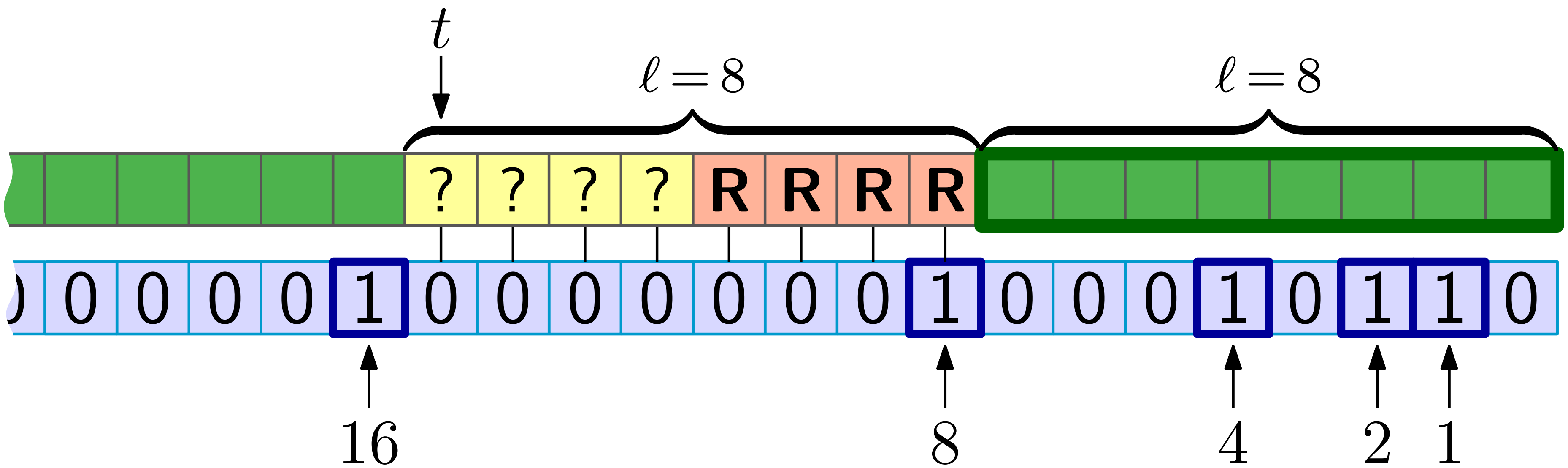
H(the outputs during  $\mid$ all  fixed) $\geqslant \frac{\ell}{2}\delta$

The conditional information transfer

$\mathbb{E}\left[|IT(t,\ell)| \mid \right.$ all  fixed$\left.\right] \geqslant \frac{\delta}{4w}\ell - \frac{1}{2}$

w bits per cell

Information transfer



Suppose that all values (and $\boxed{?}$) from the stream are chosen uniformly at random from $[q]$.

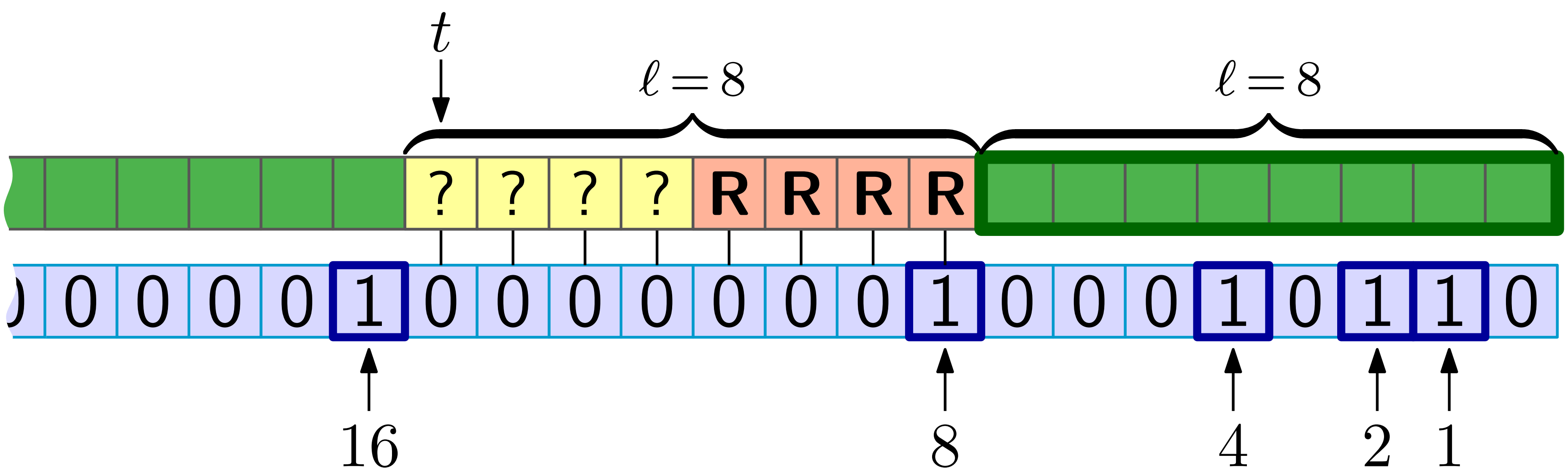
By linearity of expectation. . .

The conditional information transfer

$$\mathbb{E}\left[|IT(t,\ell)| \mid \text{all } \text{ fixed}\right] \geqslant \frac{\delta}{4w}\ell - \frac{1}{2}$$

w bits per cell

Information transfer



Suppose that all values (and ?) from the stream are chosen uniformly at random from $[q]$.

By linearity of expectation. . .

The conditional information transfer

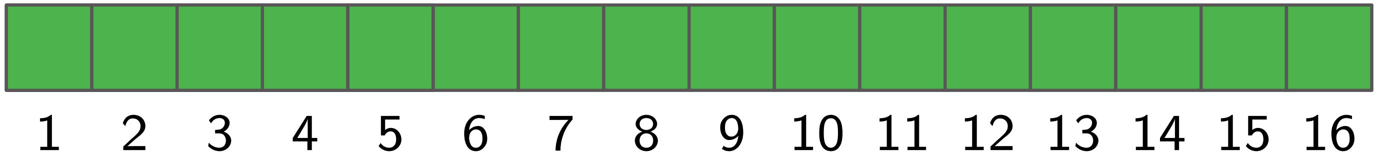
$$\mathbb{E}\left[|IT(t,\ell)| \mid \text{all} \quad \text{fixed}\right] \geqslant \frac{\delta}{4w}\ell - \frac{1}{2}$$

w bits per cell

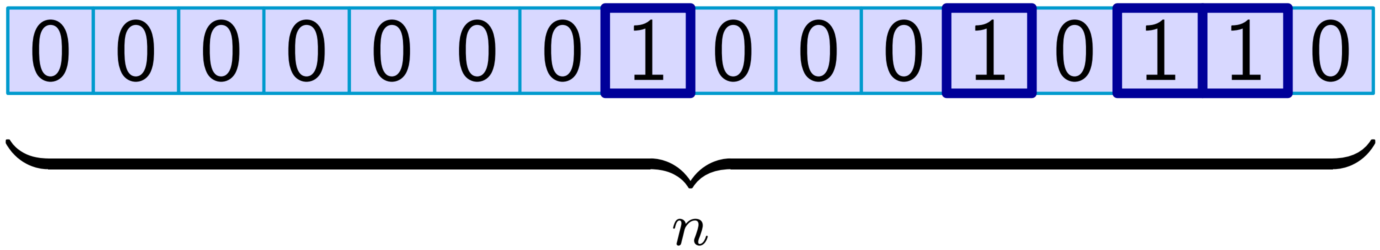
Total number of cell reads

$$\underbrace{0\;0\;0\;0\;0\;0\;0\;\boxed{1}\;0\;0\;0\;\boxed{1}\;0\;\boxed{1}\;\boxed{1}\;0}_{n}$$

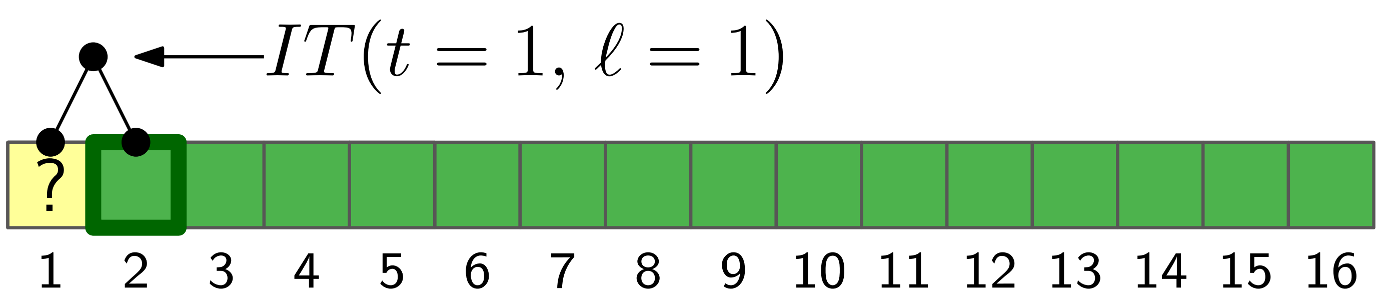
Feed the algorithm with n values chosen uniformly at random from $[q]$.



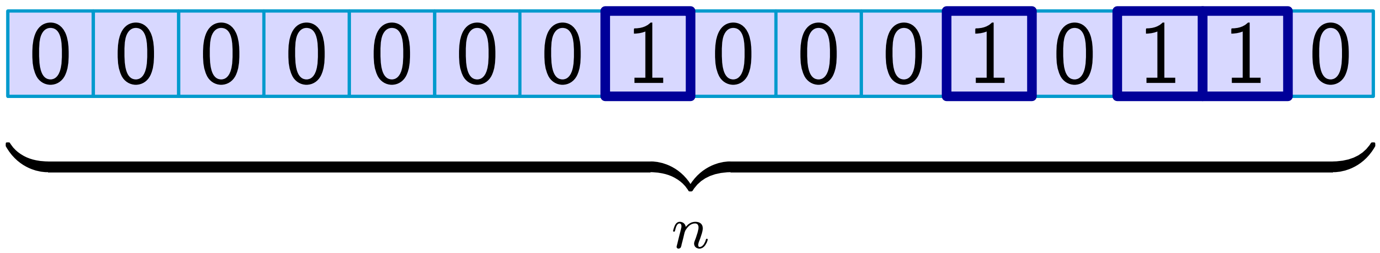
Total number of cell reads



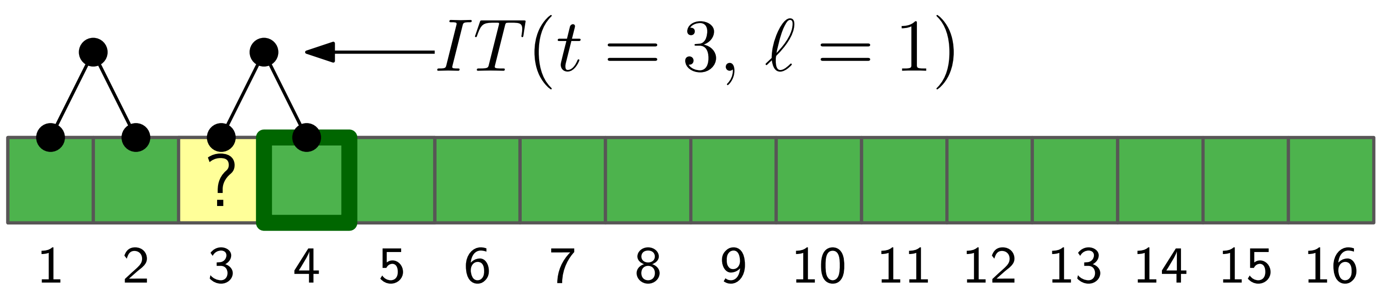
Feed the algorithm with n values chosen uniformly at random from $[q]$.



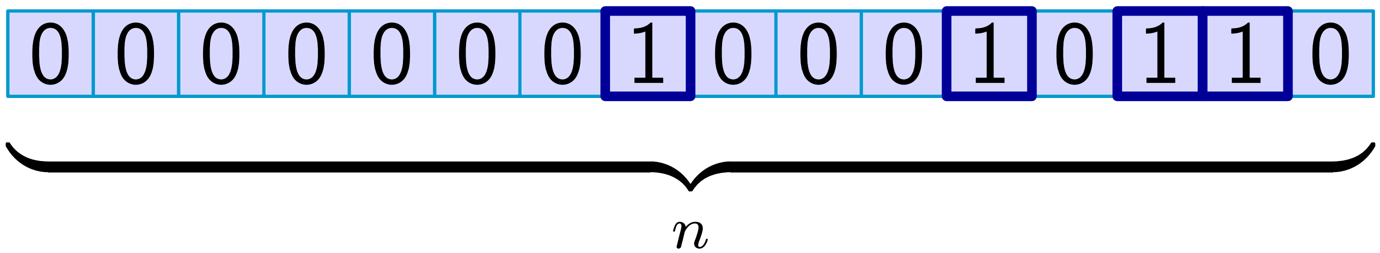
Total number of cell reads



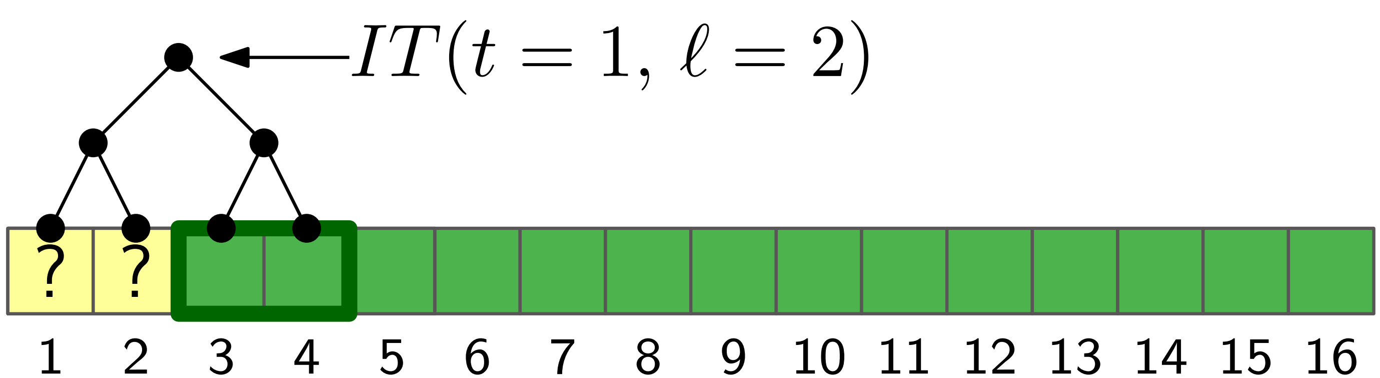
Feed the algorithm with n values chosen uniformly at random from $[q]$.



Total number of cell reads



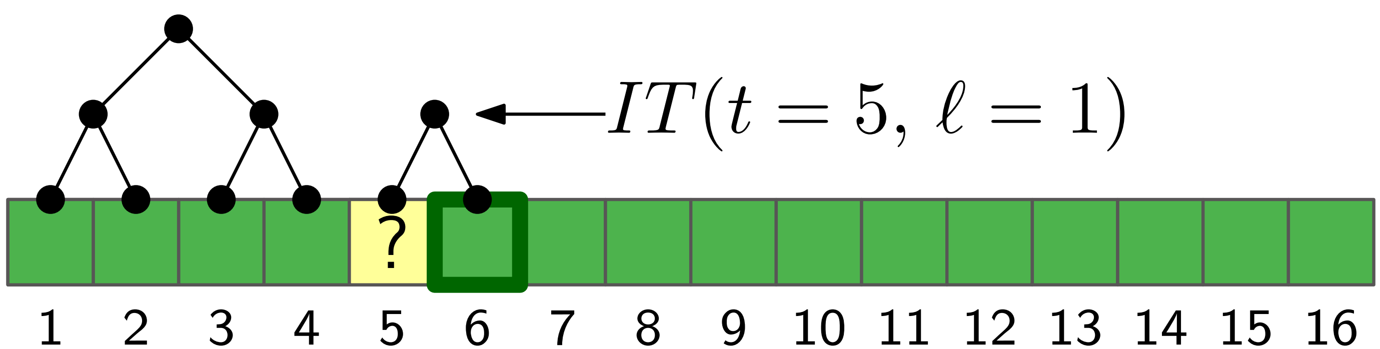
Feed the algorithm with n values chosen uniformly at random from $[q]$.



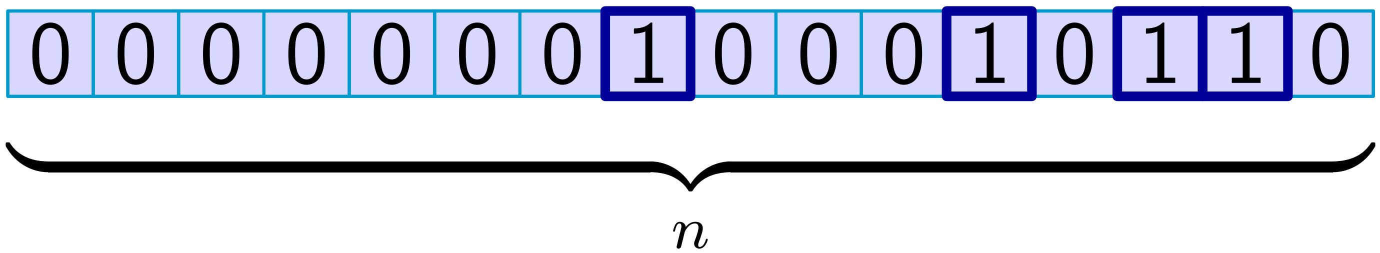
Total number of cell reads

$$\underbrace{0\ 0\ 0\ 0\ 0\ 0\ 0\ \boxed{1}\ 0\ 0\ 0\ \boxed{1}\ 0\ \boxed{1}\ \boxed{1}\ 0}_{n}$$

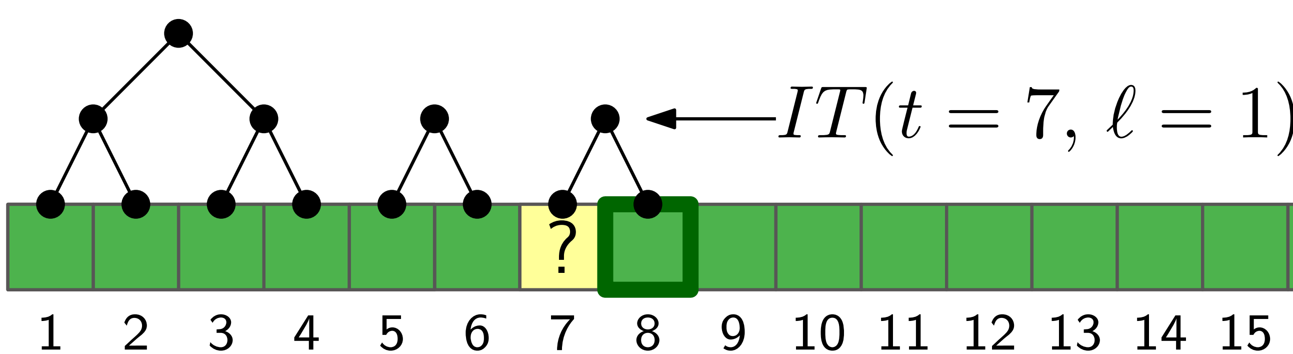
Feed the algorithm with n values chosen uniformly at random from $[q]$.



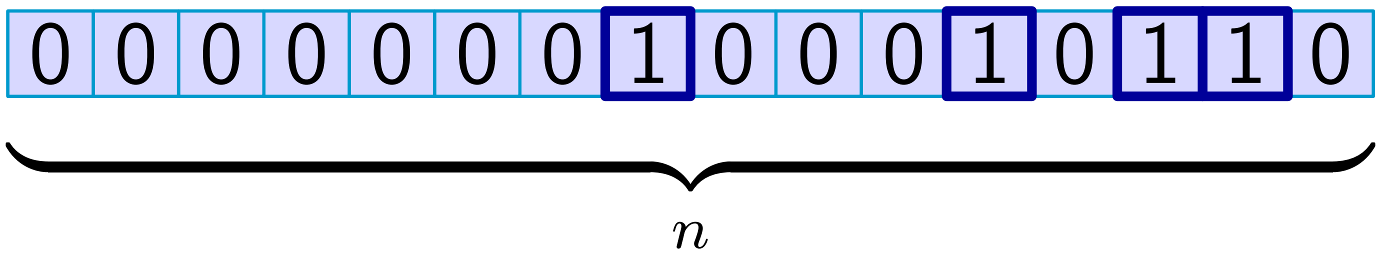
Total number of cell reads



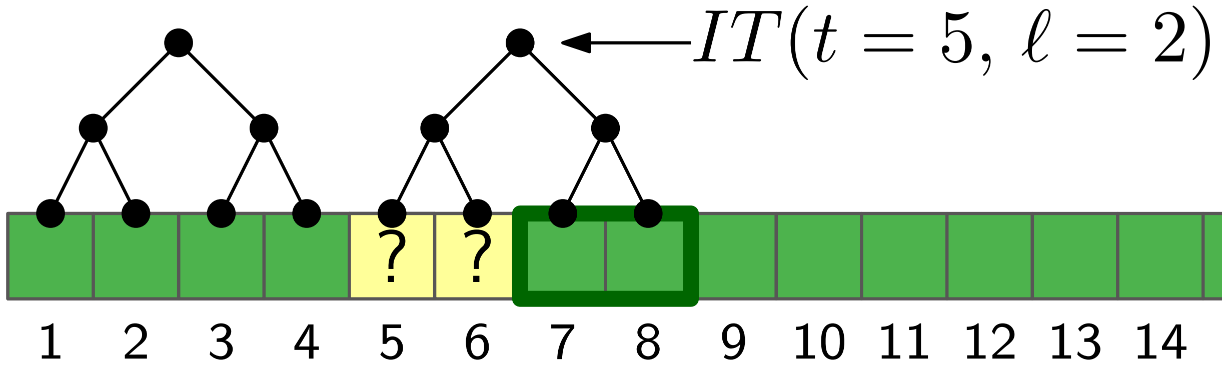
Feed the algorithm with n values chosen uniformly at random from $[q]$.



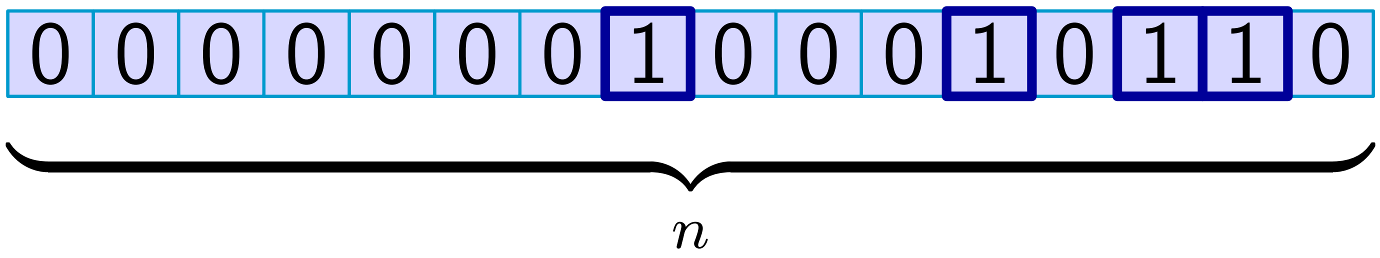
Total number of cell reads



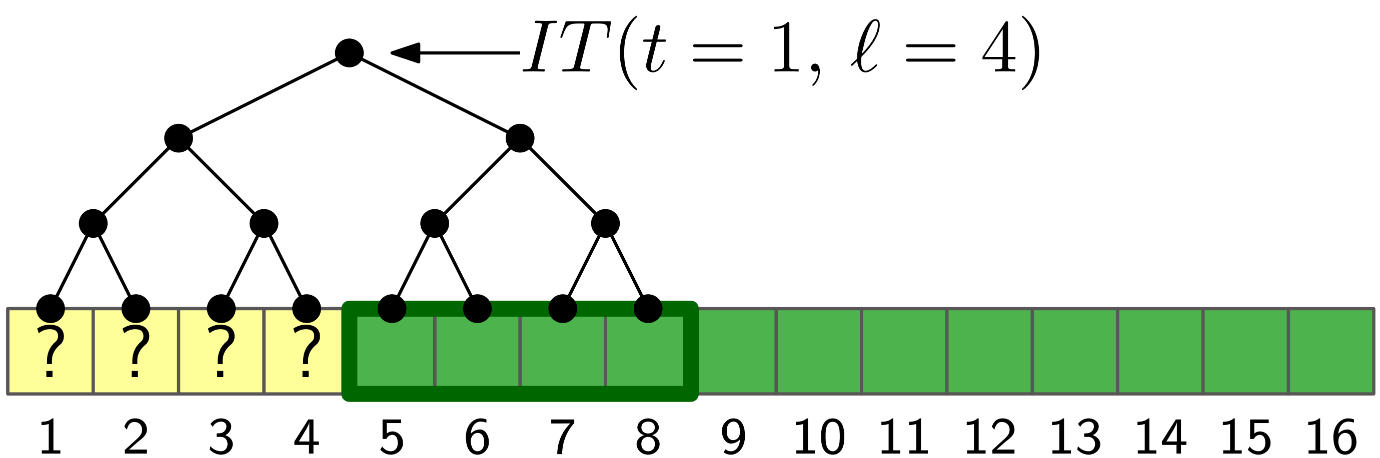
Feed the algorithm with n values chosen uniformly at random from $[q]$.



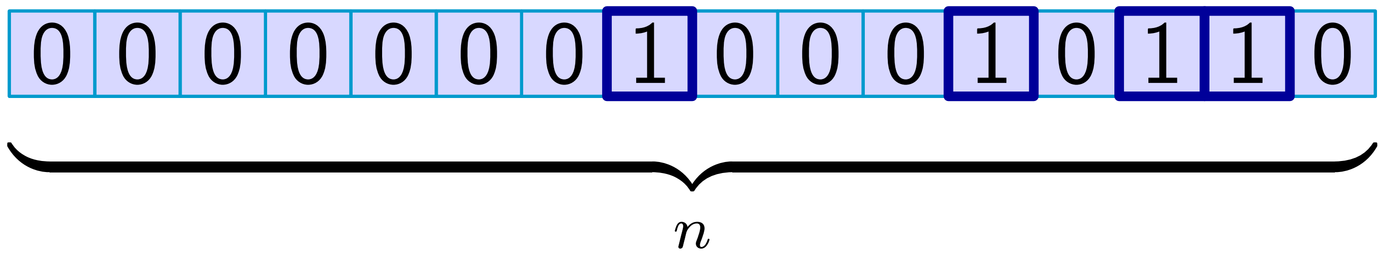
Total number of cell reads



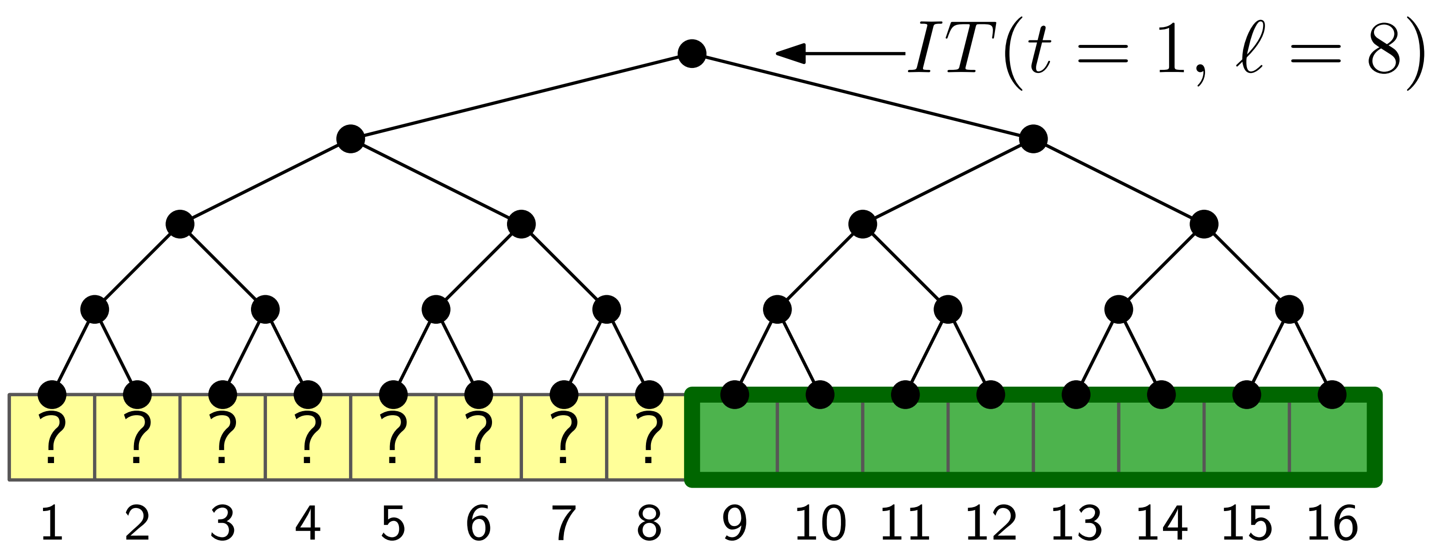
Feed the algorithm with n values chosen uniformly at random from $[q]$.



Total number of cell reads



Feed the algorithm with n values chosen uniformly at random from $[q]$.

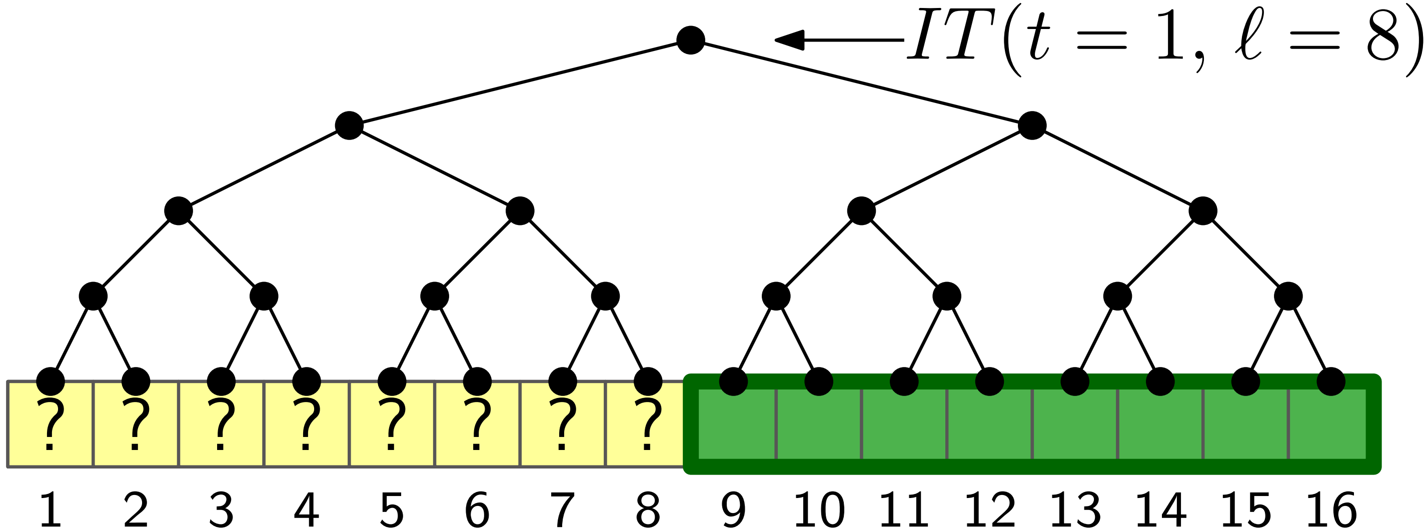


Total number of cell reads

The number of cell reads during the n inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

random from $[q]$.



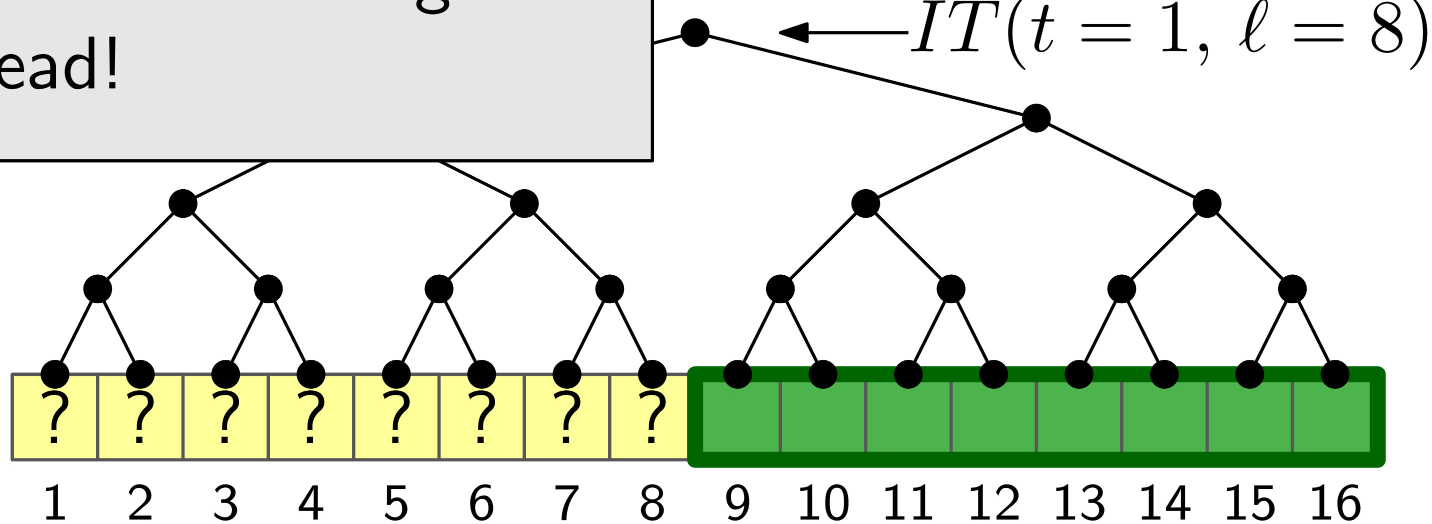
Total number of cell reads

The number of cell reads during the n inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

random from $[q]$.

No double counting of a cell read!



Total number of cell reads

The number of cell reads during the n inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

The expected number of cell reads is at least

$$\begin{aligned}
\mathbb{E}\left[\sum_{\text{internal node } v} |IT(t_v, \ell_v)|\right] &= \sum_{\text{internal node } v} \mathbb{E}\left[|IT(t_v, \ell_v)|\right] \\
&\geqslant \sum_{\text{internal node } v} \frac{\delta}{4w}\ell_v - \frac{1}{2} \\
&= \Omega\left(\frac{\delta}{w} \cdot n \log n\right)
\end{aligned}$$

Total number of cell reads

The number of cell reads during the n inputs is at least

$$\sum_{\text{internal node } v} |IT(t_v, \ell_v)|$$

The expected number of cell reads is at least

$$\mathbb{E}\left[\sum_{\text{internal node } v} |IT(t_v, \ell_v)|\right] = \sum_{\text{internal node } v} \mathbb{E}\left[|IT(t_v, \ell_v)|\right]$$

$$\geqslant \sum_{\text{internal node } v} \frac{\delta}{4w}\ell_v - \frac{1}{2}$$

$$= \Omega\left(\frac{\delta}{w} \cdot n \log n\right)$$

So...
The amortised time lower bound per output is $\Omega\left(\frac{\delta}{w}\log n\right)$

Multiplication in a stream

Paterson, Fischer and Meyer

An Improved Overlap Argument for On-Line Multiplication

SIAM-AMS Proceedings, 1974

For binary numbers on

- Multitape Turing machine: $\Omega(n \log n)$
- BAM or “bounded activity machine”:

$$\Omega\left(\frac{n \log n}{\log \log n}\right)$$

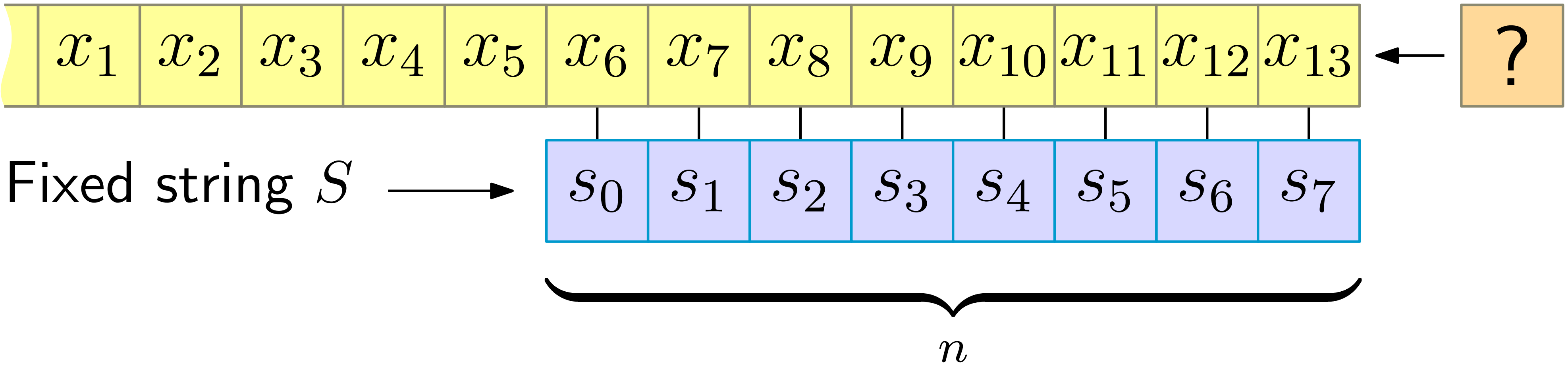
C., Jalsenius

Lower Bounds for Online Integer Multiplication and Convolution in the Cell-Probe Mode. ICALP 2011

Time lower bound: $\Omega\left(\frac{\delta}{w} \cdot n \log n\right)$

Hamming distance

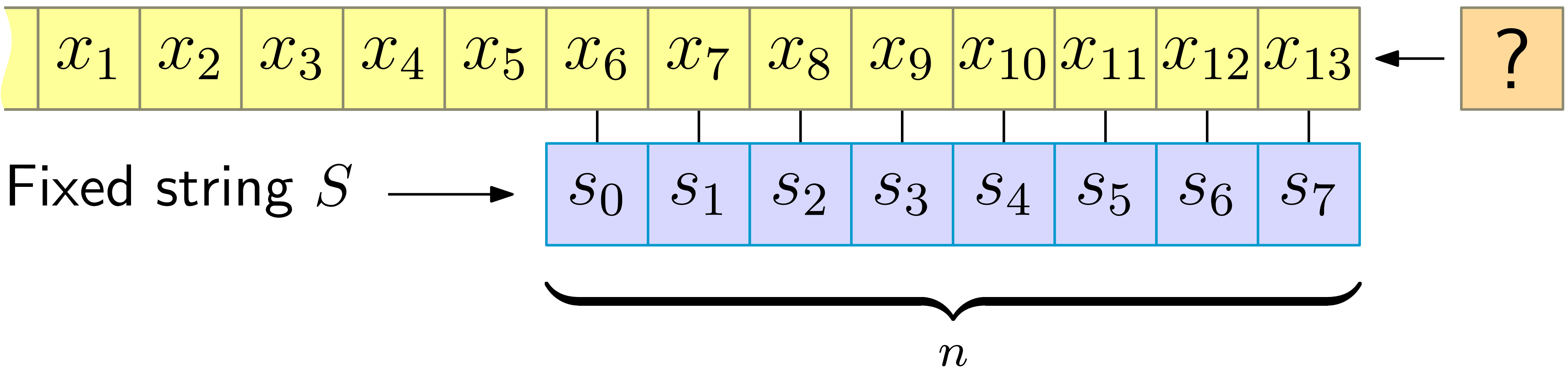
Stream of symbols from alphabet Σ



Output Hamming distance between S and last n symbols of stream.

Hamming distance

Stream of symbols from alphabet Σ



Output Hamming distance between S and last n symbols of stream.

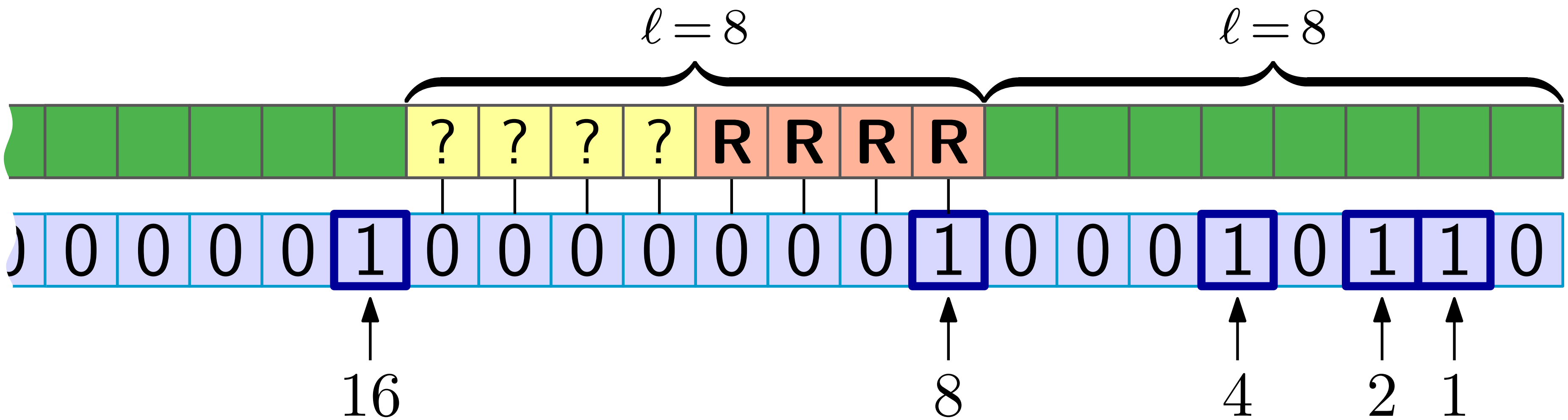
Lower bound: $\Omega\left(\frac{\delta}{w}\log n\right)$

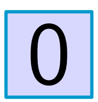
$\delta = \log|\Sigma|$

C., Jalsenius, Sach. Tight Cell-Probe Bounds for Online Hamming Distance Computation. SODA 2013

The hard instance - a first attempt

Try a similar approach to before:





= a symbol occurring only in the fixed string

if the position is a power of 2

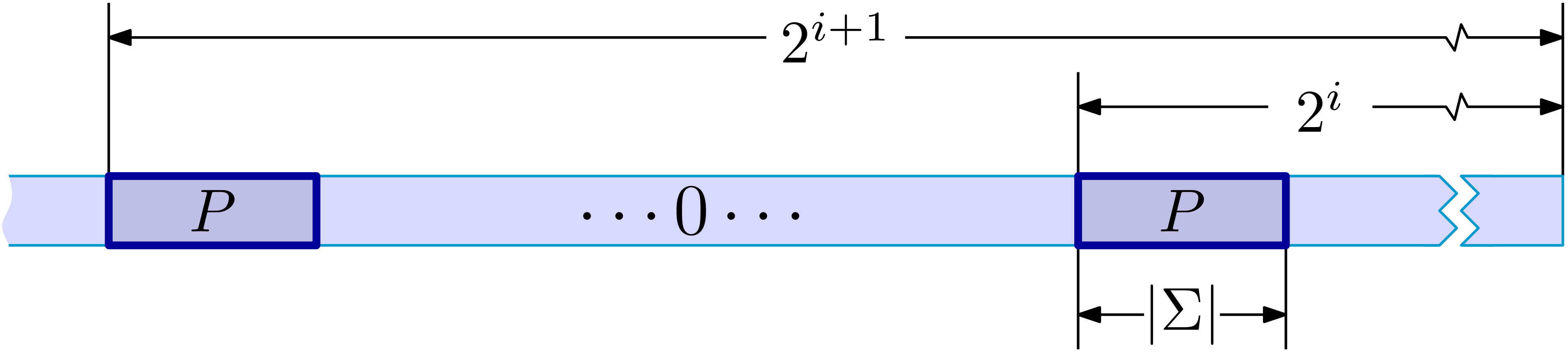
We can only infer whether **R** is the symbol 1 or not, i.e. only one bit of information.

Hamming distance

More difficult than convolution:

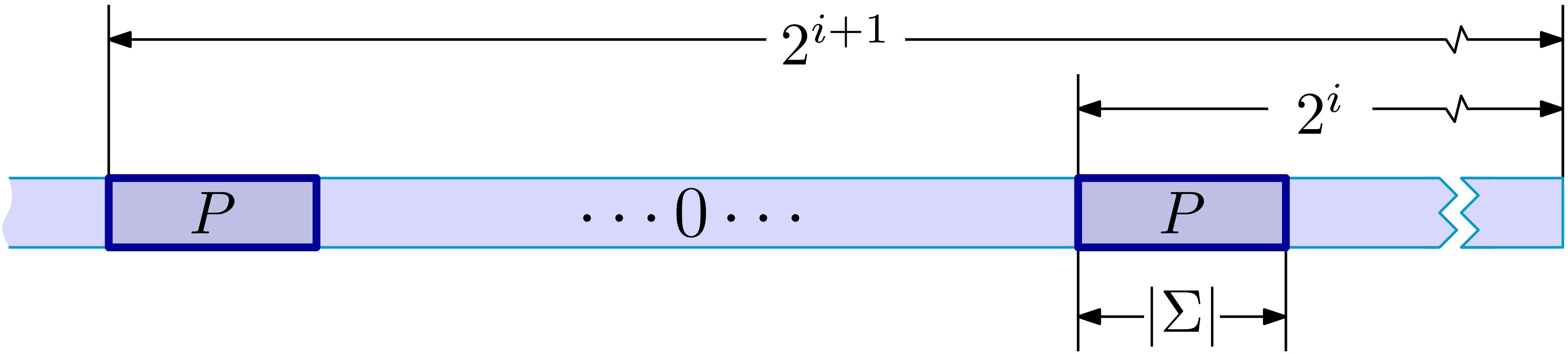
- Appear to get at most 1 bit of information per symbol.
- Too large alphabet implies large Hamming distance (on random input), i.e. low entropy.
- Too small an alphabet implies low entropy per symbol.
- No obvious worst case pattern.

A harder instance



Substring P at every power of two position, and 0 elsewhere (a symbol that does not occur in the stream).

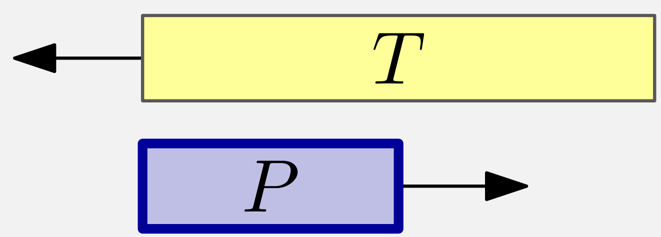
A harder instance



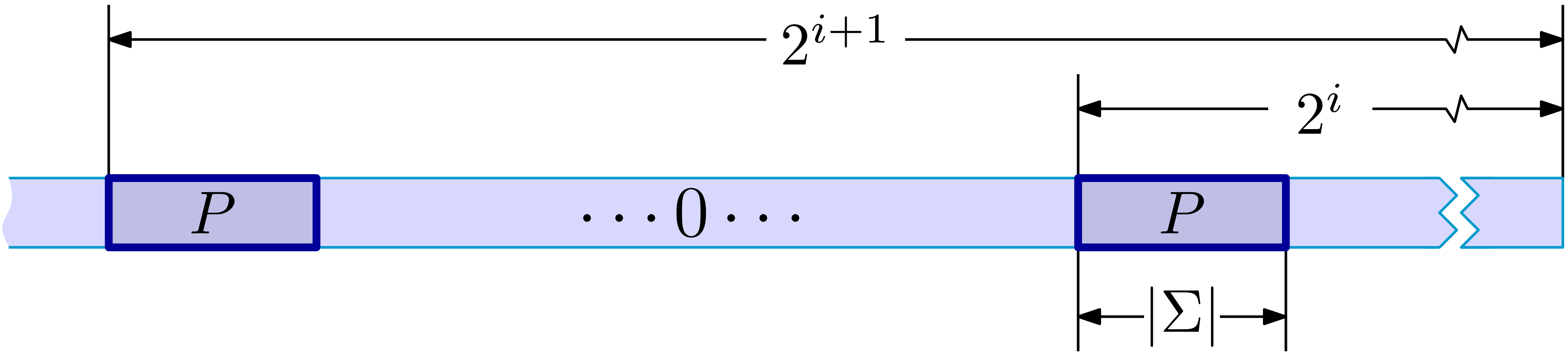
Substring P at every power of two position, and 0 elsewhere (a symbol that does not occur in the stream).

Lemma

There is a P s.t. sliding it over all $2|P|$ length strings T (over alphabet $\Sigma \setminus \{0\}$) generates $|\Sigma|^{\Theta(|\Sigma|)}$ *distinct* Hamming array ouputs.



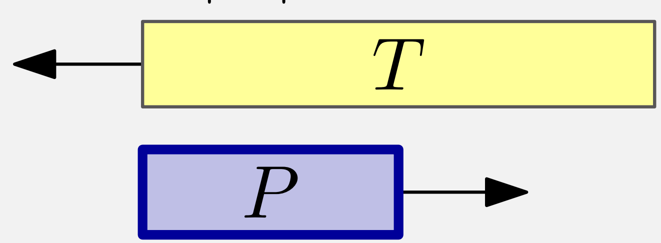
A harder instance



Substring P at every power of two position, and 0 elsewhere (a symbol that does not occur in the stream).

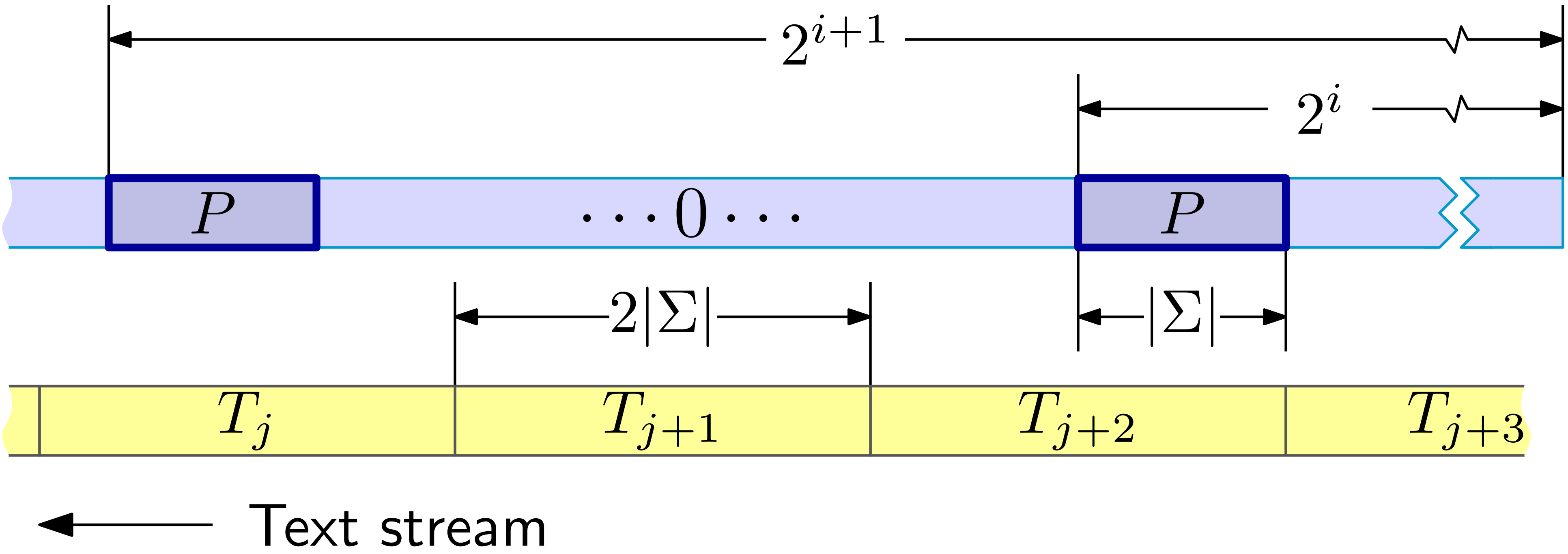
Lemma

There is a P s.t. sliding it over all $2|P|$ length strings T (over alphabet $\Sigma \setminus \{0\}$) generates $|\Sigma|^{\Theta(|\Sigma|)}$ *distinct* Hamming array ouputs.



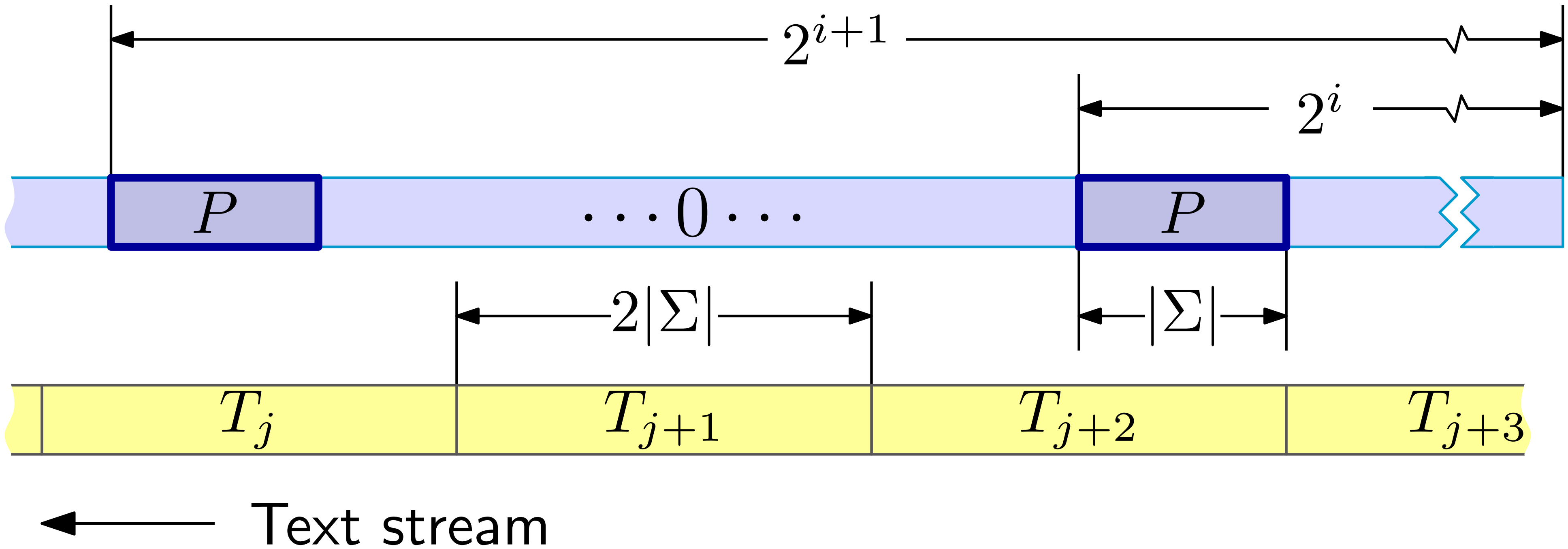
Great news! Highest entropy we can hope for.

The hard instance



Each T_j is drawn uniformly from a set $\mathcal{T}$ of size $|\Sigma|^{\Theta(|\Sigma|)}$.
Any two strings in $\mathcal{T}$ give distinct Hamming outputs with P.

The hard instance



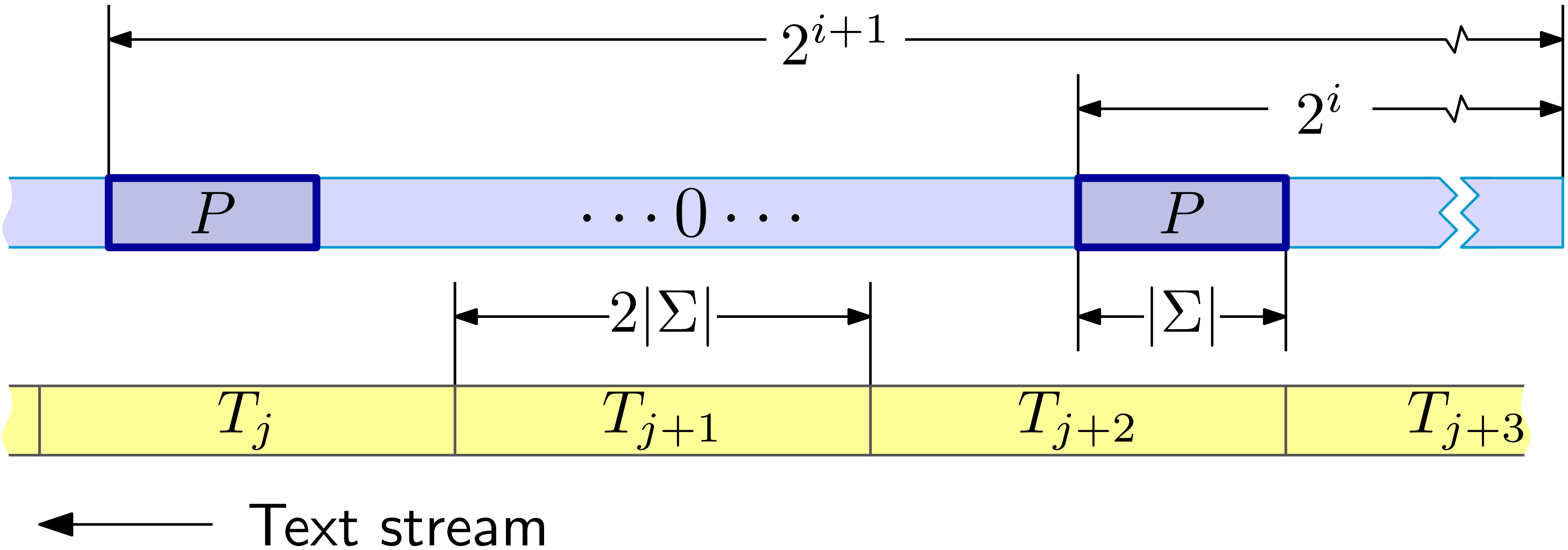
Each T_j is drawn uniformly from a set $\mathcal{T}$ of size $|\Sigma|^{\Theta(|\Sigma|)}$.
Any two strings in $\mathcal{T}$ give distinct Hamming outputs with P.

Recover $\Theta(\ell)$ symbols from a window of ℓ unknown input symbols. Entropy:

$$\Theta\left(\frac{\ell}{2|\Sigma|} \cdot \log |\Sigma|^{\Theta(|\Sigma|)}\right) = \Theta(\ell \cdot \log |\Sigma|) = \Theta(\ell\delta)$$

$\delta = \log |\Sigma|$

The hard instance



Each T_j is drawn uniformly from a set $\mathcal{T}$ of size $|\Sigma|^{\Theta(|\Sigma|)}$.

Hence lower bound $\Omega\left(\frac{\delta}{w}\log n\right)$

Recover $\Theta(\ell)$ symbols from a window of ℓ unknown input symbols. Entropy:

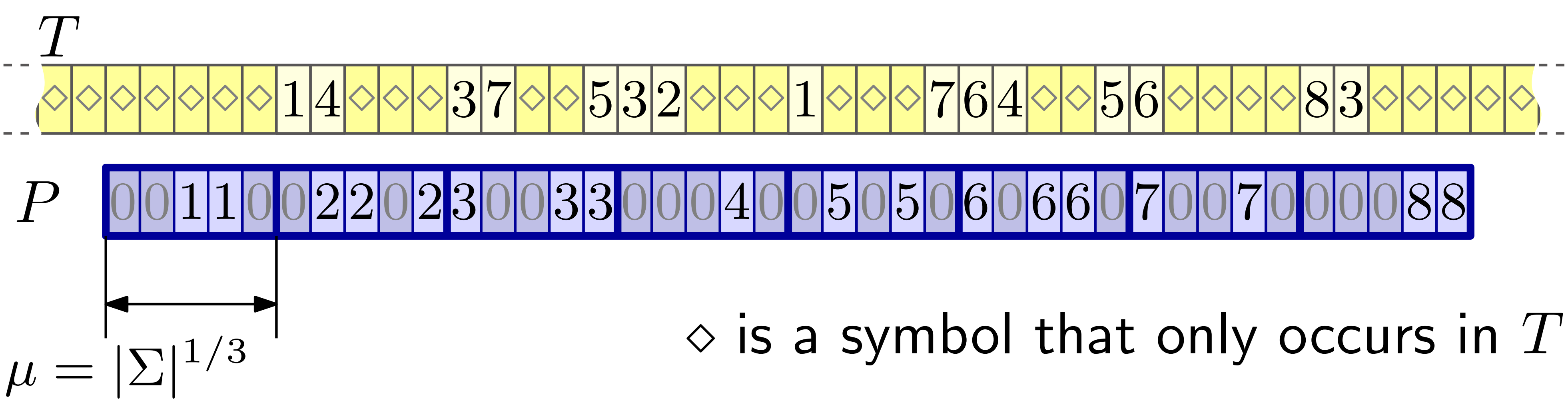
$$\Theta\left(\frac{\ell}{2|\Sigma|} \cdot \log |\Sigma|^{\Theta(|\Sigma|)}\right) = \Theta(\ell \cdot \log |\Sigma|) = \Theta(\ell\delta)$$

$\delta = \log |\Sigma|$

The string P

Proof overview of the lemma.

- Partition P into blocks, each using a unique symbol.

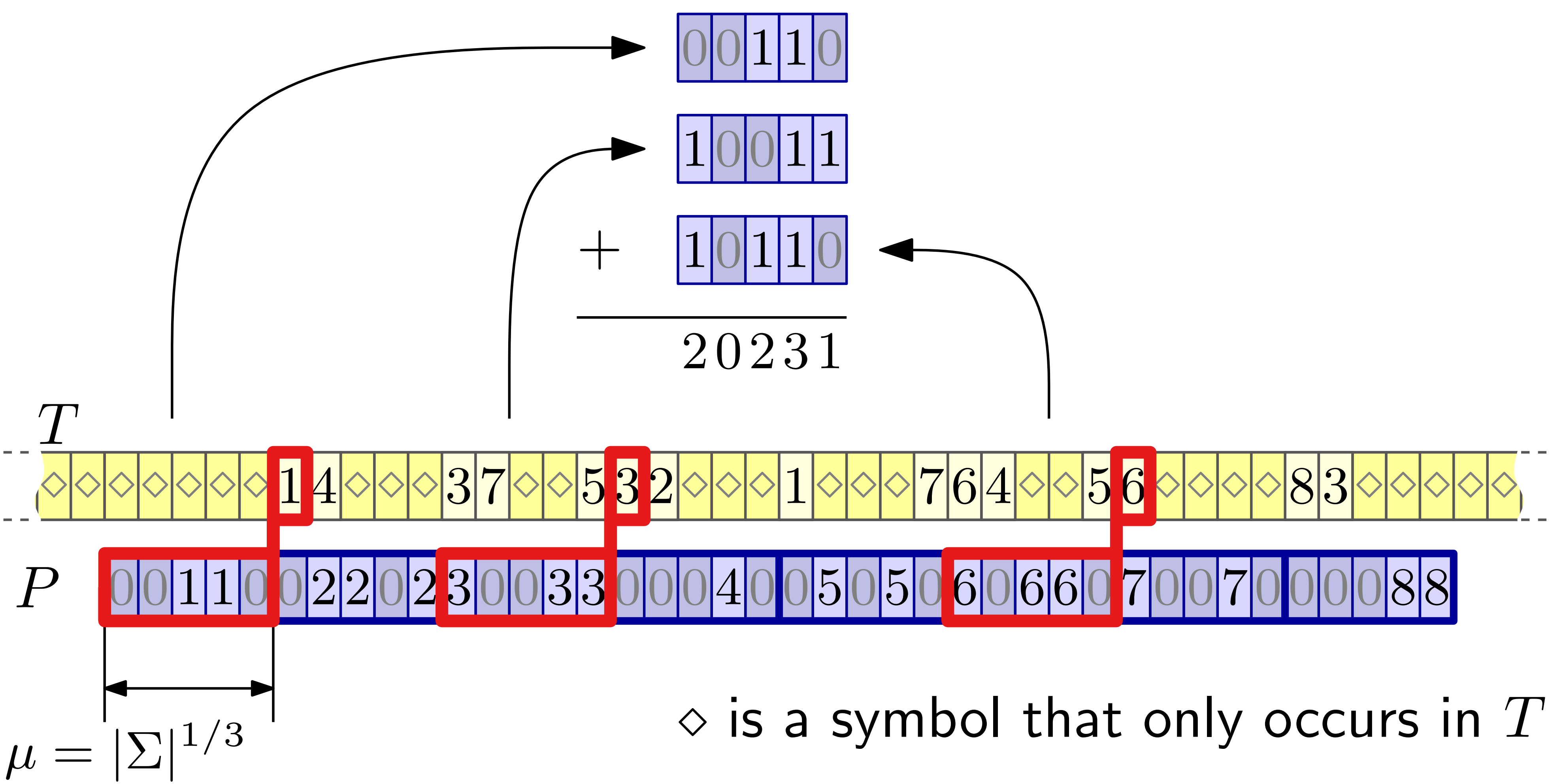


$\diamond$ is a symbol that only occurs in T

The string P

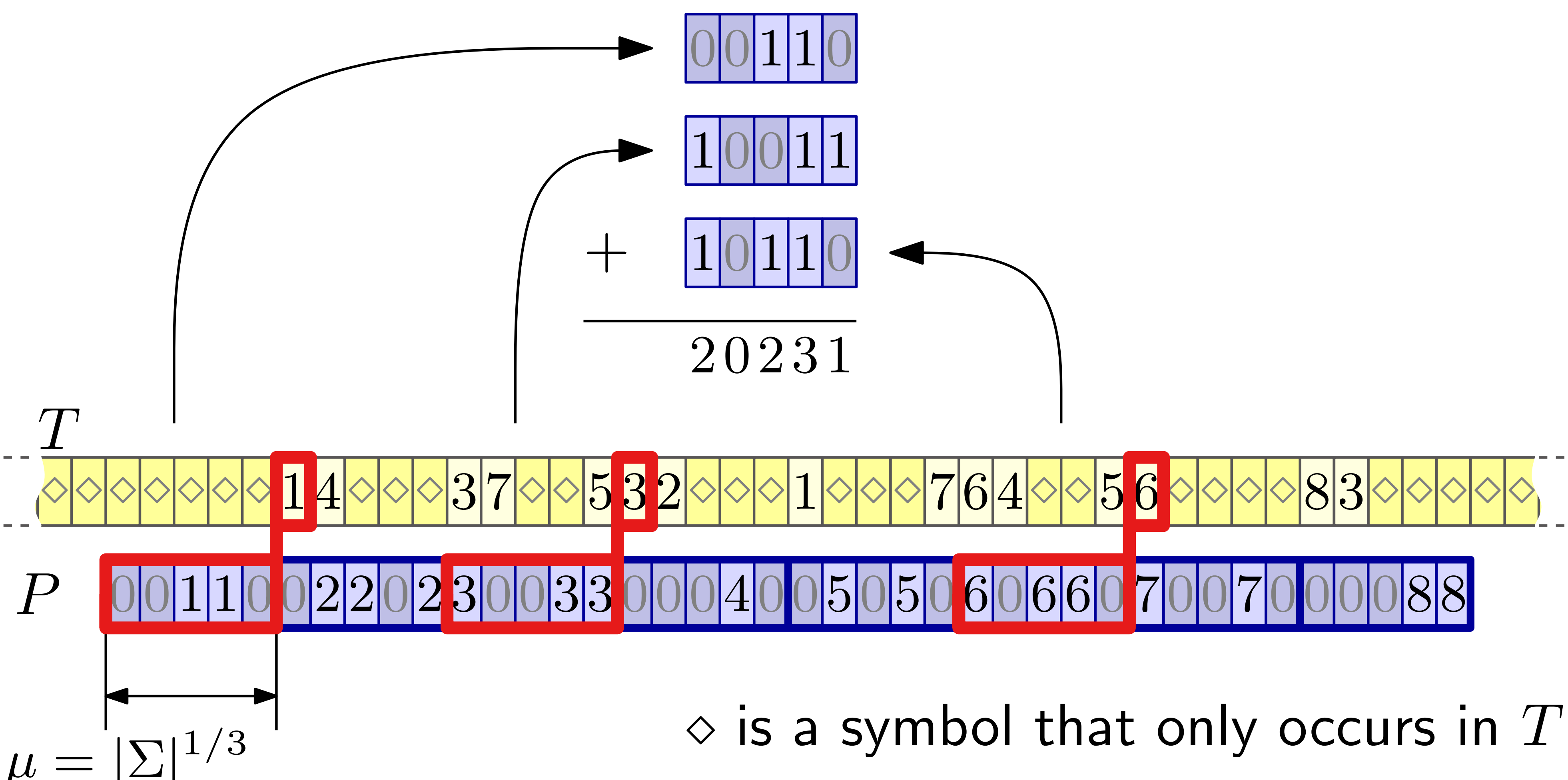
Proof overview of the lemma.

- Partition P into blocks, each using a unique symbol.
- Symbols of T will slide over P, and match sums will correspond to sums of binary vectors.



The string P

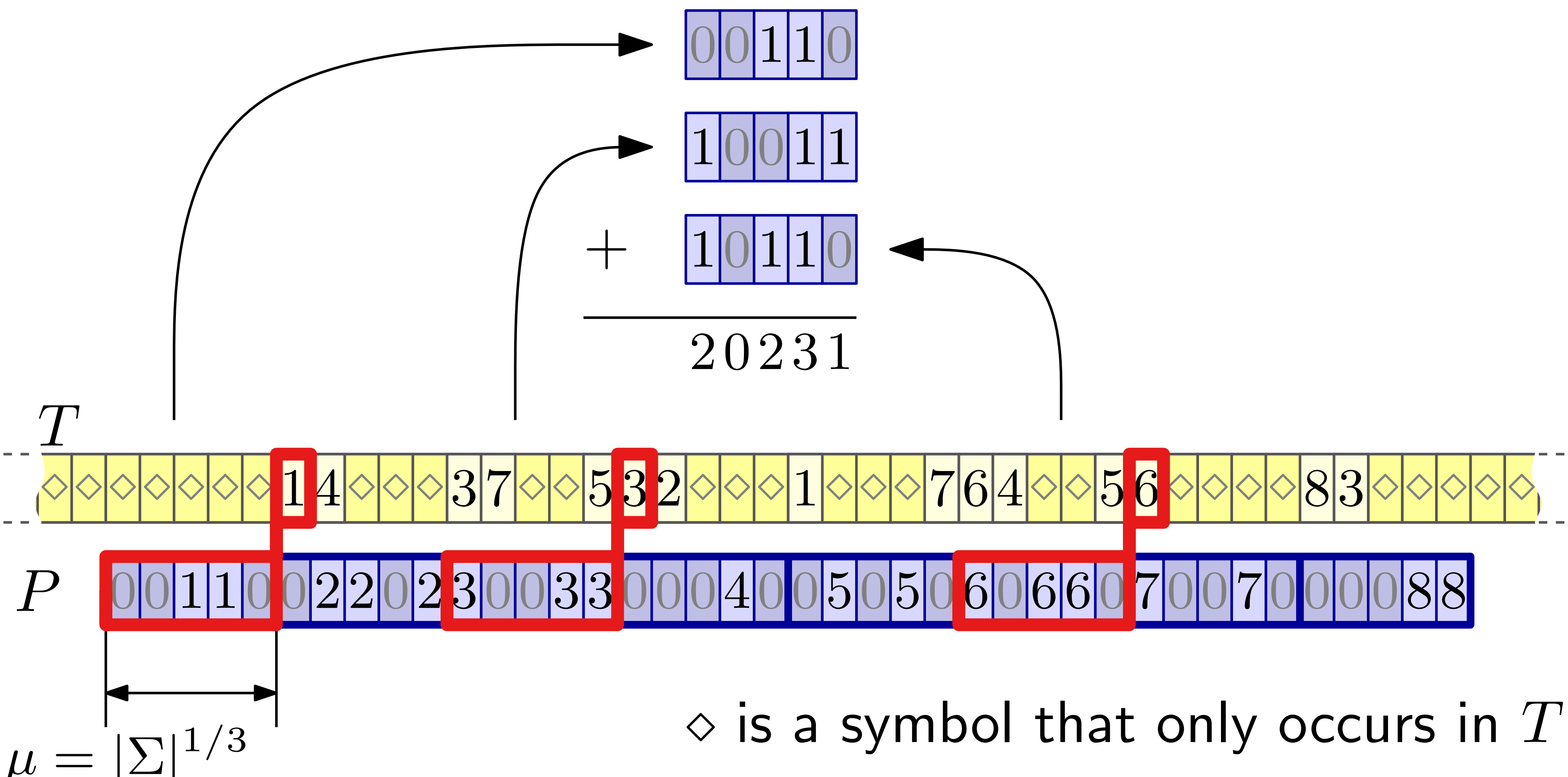
- For each window of μ outputs, one can obtain $\mu^{\Theta(\mu)}$ distinct vector sums. (Proof involves cyclic binary codes.)



$\diamond$ is a symbol that only occurs in T

The string P

- For each window of μ outputs, one can obtain $\mu^{\Theta(\mu)}$ distinct vector sums. (Proof involves cyclic binary codes.)
- Thus, over the whole of T there are $|\Sigma|^{\Theta(|\Sigma|)}$ possible distinct Hamming array ouputs.



◇ is a symbol that only occurs in T

What next?

Entirely new techniques appear to be needed again for seemingly related problems. For example:

- Edit distance (outputs can be encoded in $O(n)$ bits)
- Decision problems (entropy is very low)

What next?

Entirely new techniques appear to be needed again for seemingly related problems. For example:

- Edit distance (outputs can be encoded in $O(n)$ bits)
- Decision problems (entropy is very low)

Thank you!