Effective Web Graph Representations

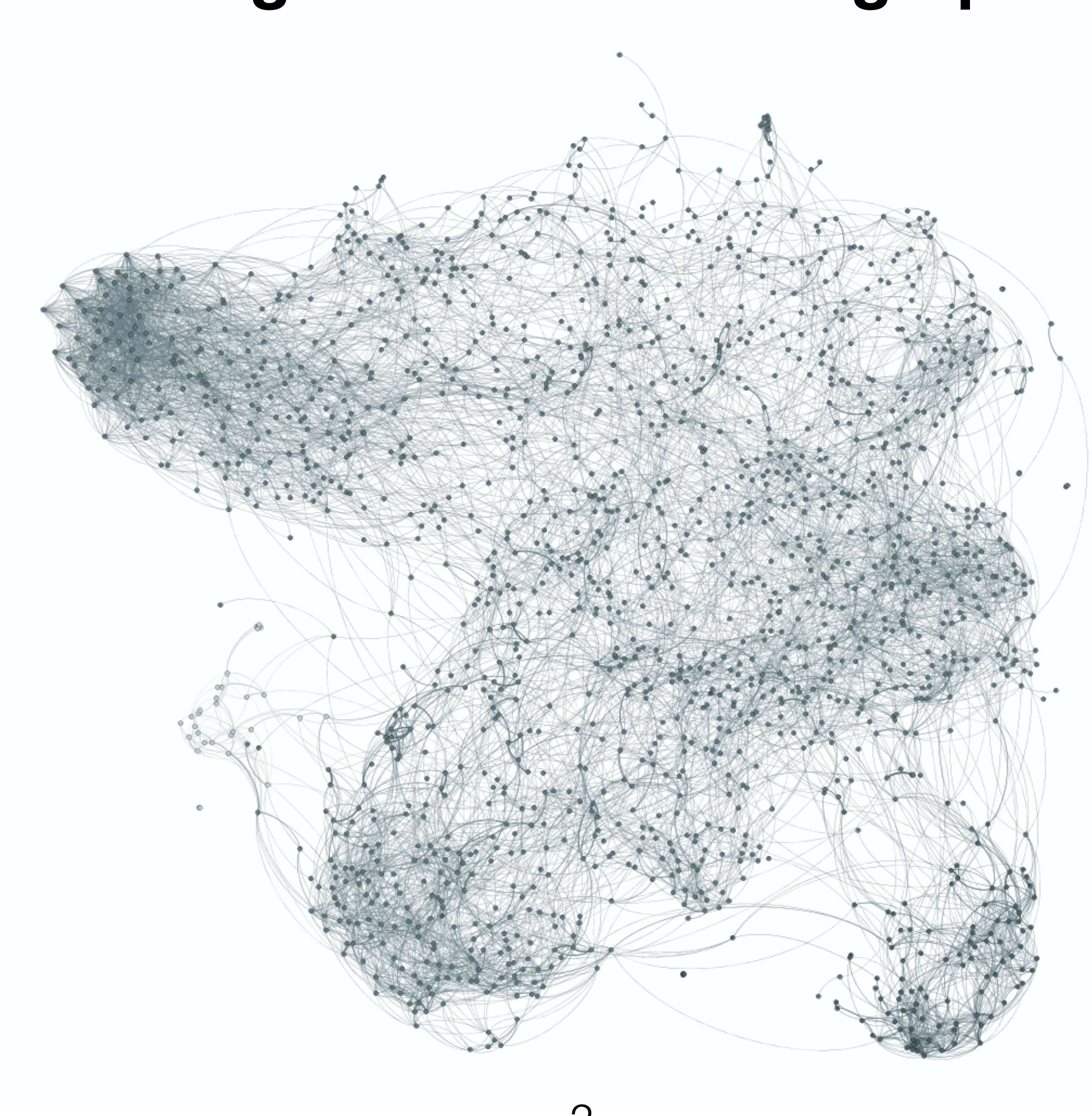
Giulio Ermanno Pibiri
University of Pisa and ISTI-CNR
Pisa, Italy
giulio.pibiri@di.unipi.it

Pisa, 29/10/2018

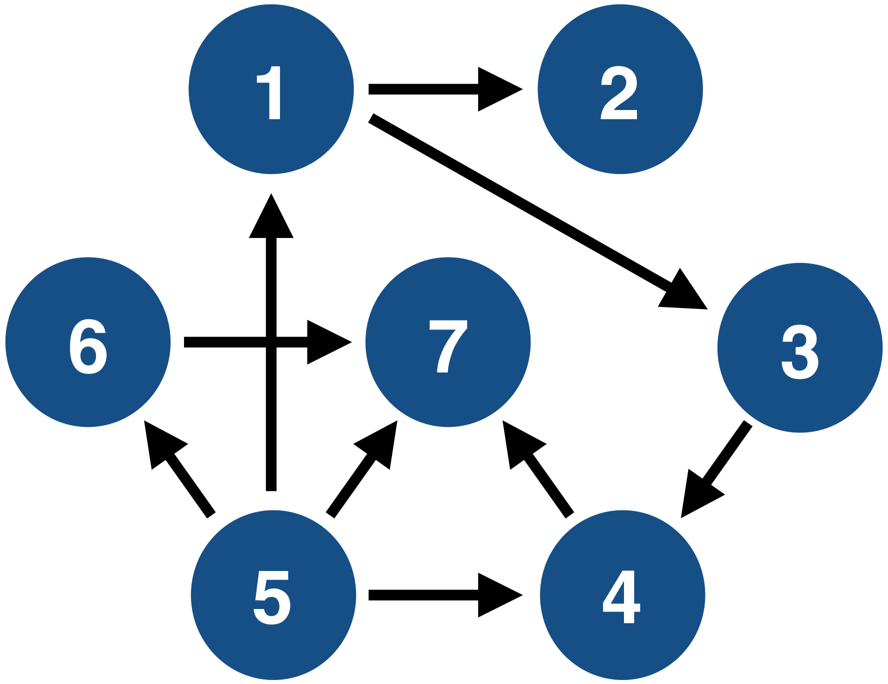
Context - Web Graphs

Web graphs are directed graphs of pages pointing to other pages on the Web.

We focus on compression effectiveness on **large real-world Web graphs**.



Context - Web Graphs



Conceptual graph

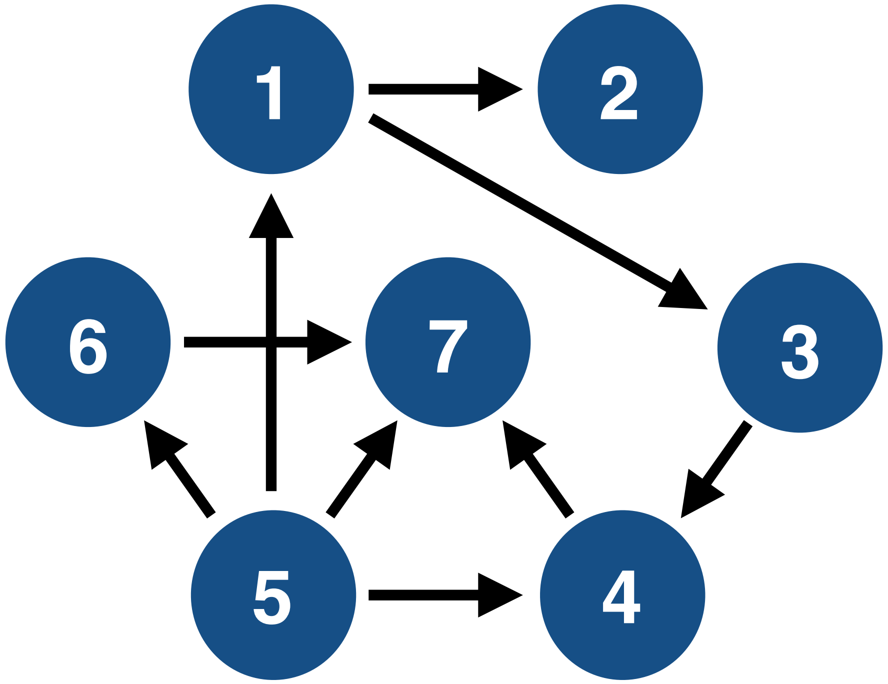
0	1	1	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	1
1	0	0	1	0	1	1
0	0	0	0	0	0	1
0	0	0	0	0	0	0

Adjacency **matrix**

1: 2,3
2: -
3: 4
4: 7
5: 1,4,6,7
6: 7
7: -

Adjacency **lists**

Context - Web Graphs



Conceptual graph

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

Adjacency **matrix**

1: 2,3
2: -
3: 4
4: 7
5: 1,4,6,7
6: 7
7: -

Adjacency **lists**

Many results are known for compressing
integer sequences.

Outline

1. **The WebGraph framework**

2. **k^2-trees**

3. **Block-trees**

4. **2D-Block-trees**

Outline

1. **The WebGraph framework**	2004
2. **k²-trees**	2009
3. **Block-trees**	2014
4. **2D-Block-trees**	2018

The WebGraph Framework

Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

The WebGraph Framework I: Compression Techniques, Boldi-Vigna, WWW 2004

The WebGraph Framework

Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

The WebGraph Framework I: Compression Techniques, Boldi-Vigna, WWW 2004

Locality - pages links to pages whose URL is lexicographically similar. URLs
share long common prefixes.

The WebGraph Framework

Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

The WebGraph Framework I: Compression Techniques, Boldi-Vigna, WWW 2004

Locality - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes.

Use *d-gap* compression.

The WebGraph Framework

Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

The WebGraph Framework I: Compression Techniques, Boldi-Vigna, WWW 2004

Locality - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes.

Use *d-gap* compression.

Similarity - pages that are close together in lexicographic order, tend to have many common successors.

The WebGraph Framework

Java/C++ framework consisting in algorithms and compression codes for managing large Web Graphs.

http://webgraph.di.unimi.it/

The WebGraph Framework I: Compression Techniques, Boldi-Vigna, WWW 2004

Locality - pages links to pages whose URL is lexicographically similar. URLs share long common prefixes.

Use *d-gap* compression.

Similarity - pages that are close together in lexicographic order, tend to have many common successors.

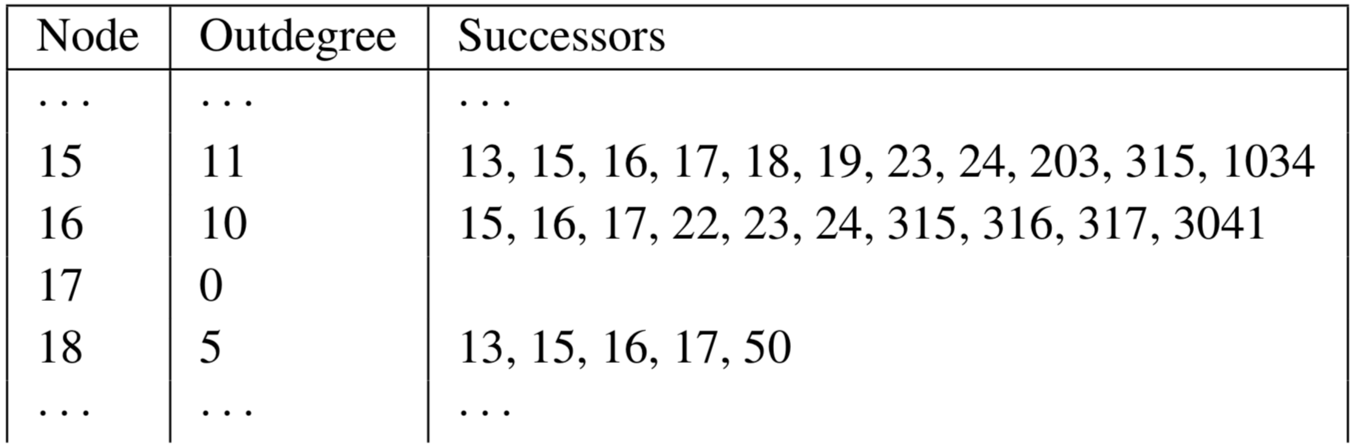
Use *reference* compression.

The WebGraph Framework

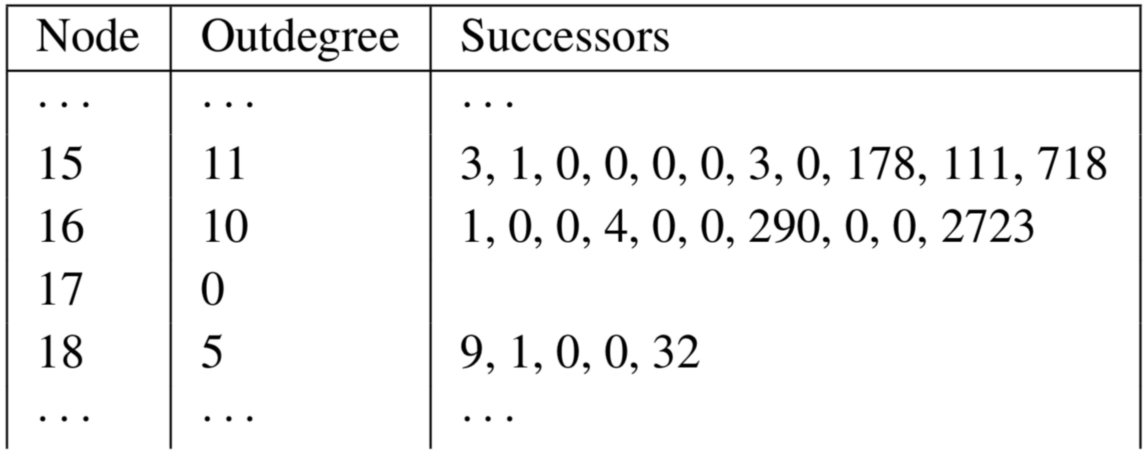
Exploiting **locality**.

If we have: x: $[y_1,\ldots,y_k]$, then we represent
$[y_1 - x, y_2 - y_1 - 1, y_3 - y_2 - 1, \ldots, y_k - y_{k-1} - 1]$

First gap $d = y_1 - x$ is represented as $2d$ if $d \geq 0$ or $2|d|-1$ if $d < 0$



Node	Outdegree	Successors
. . .	. . .	. . .
15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
17	0	
18	5	13, 15, 16, 17, 50
. . .	. . .	. . .



Node	Outdegree	Successors
. . .	. . .	. . .
15	11	3, 1, 0, 0, 0, 0, 3, 0, 178, 111, 718
16	10	1, 0, 0, 4, 0, 0, 290, 0, 0, 2723
17	0	
18	5	9, 1, 0, 0, 32
. . .	. . .	. . .

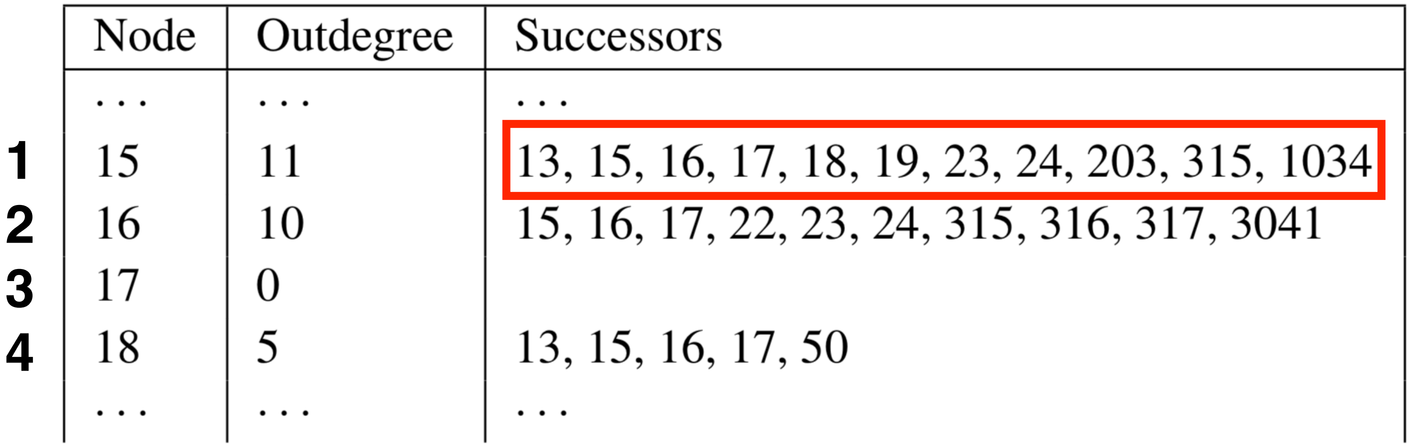
Adjacency lists

***d*-gapped** adjacency lists

The WebGraph Framework

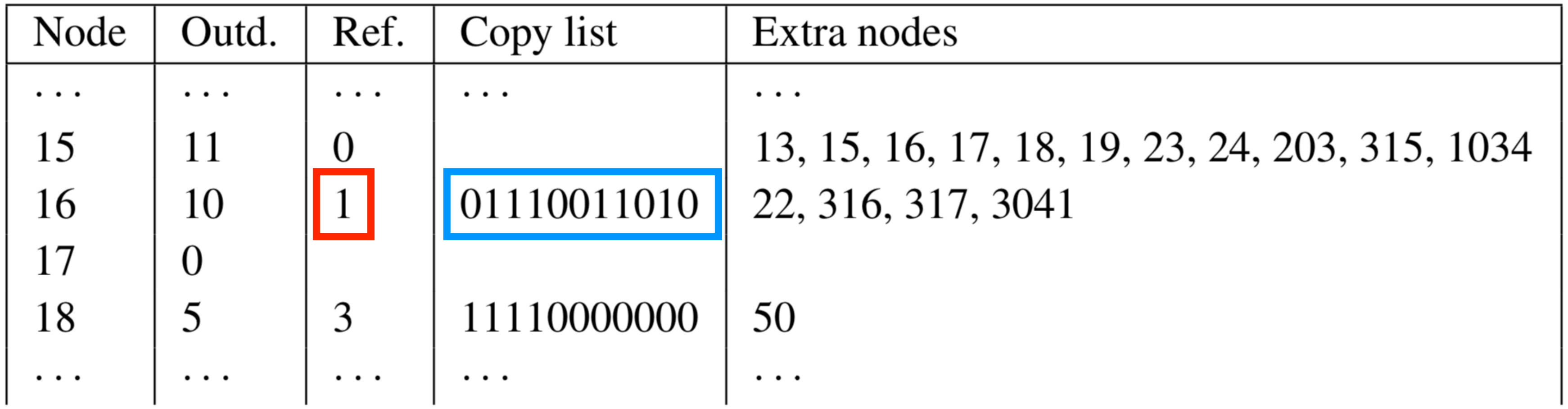
Exploiting **similarity**.

Idea: use reference compression, i.e., represent a list with respect to another one called its **reference list**.



	Node	Outdegree	Successors
	. . .	. . .	. . .
1	15	11	13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
2	16	10	15, 16, 17, 22, 23, 24, 315, 316, 317, 3041
3	17	0	
4	18	5	13, 15, 16, 17, 50
	. . .	. . .	. . .

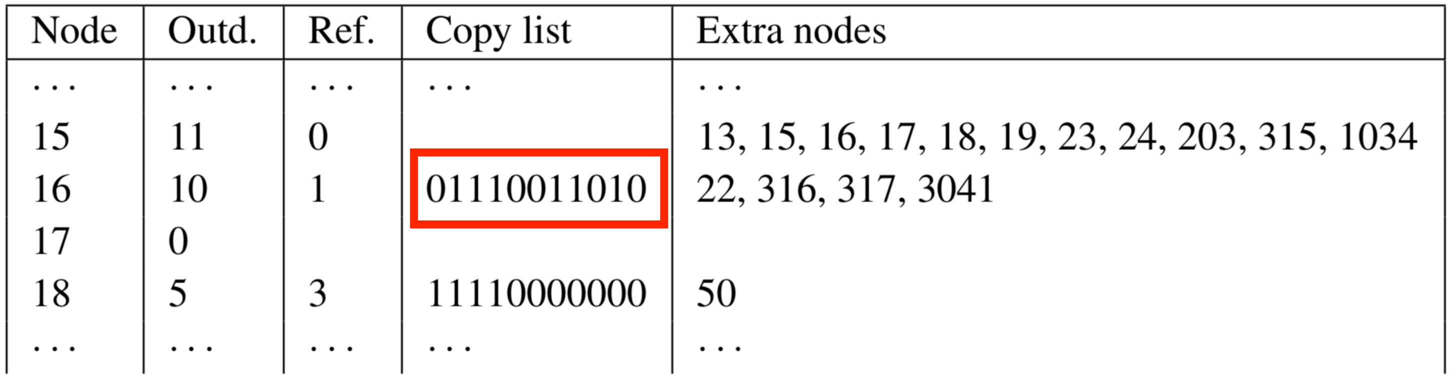
Adjacency lists



Node	Outd.	Ref.	Copy list	Extra nodes
. . .	. . .	. . .	. . .	. . .
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
. . .	. . .	. . .	. . .	. . .

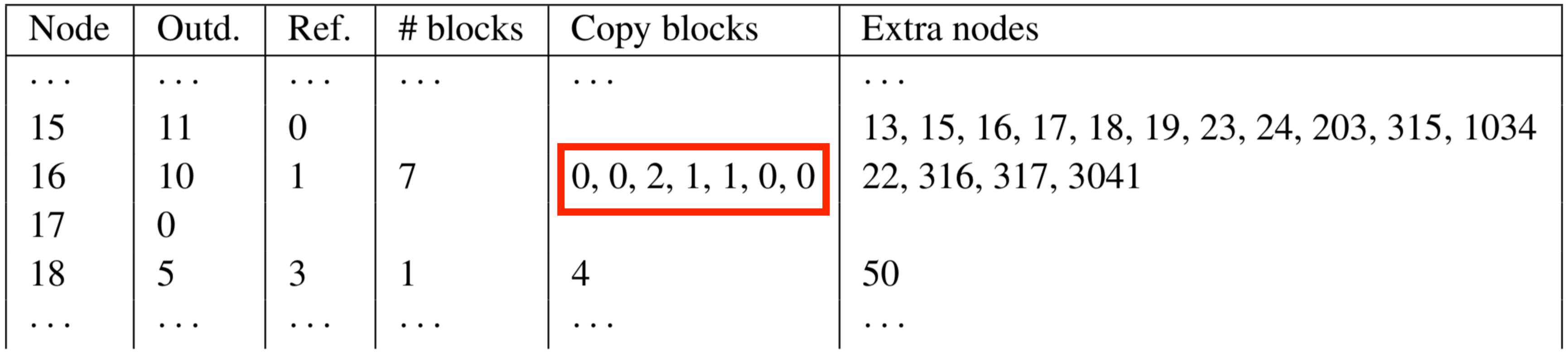
Copy lists

The WebGraph Framework



Node	Outd.	Ref.	Copy list	Extra nodes
. . .	. . .	. . .	. . .	. . .
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
. . .	. . .	. . .	. . .	. . .

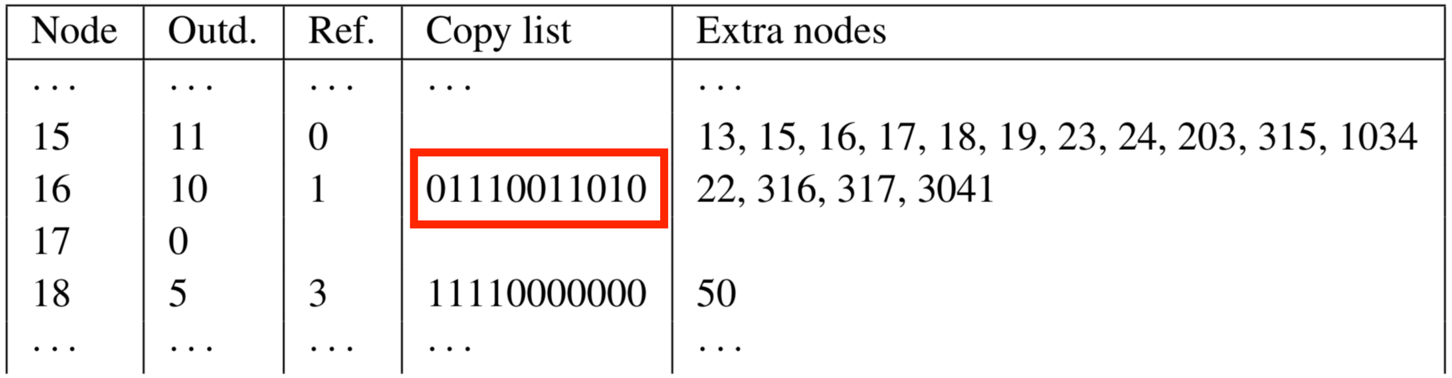
Copy lists



Node	Outd.	Ref.	# blocks	Copy blocks	Extra nodes
. . .	. . .	. . .	. . .	. . .	. . .
15	11	0			13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	7	0, 0, 2, 1, 1, 0, 0	22, 316, 317, 3041
17	0				
18	5	3	1	4	50
. . .	. . .	. . .	. . .	. . .	. . .

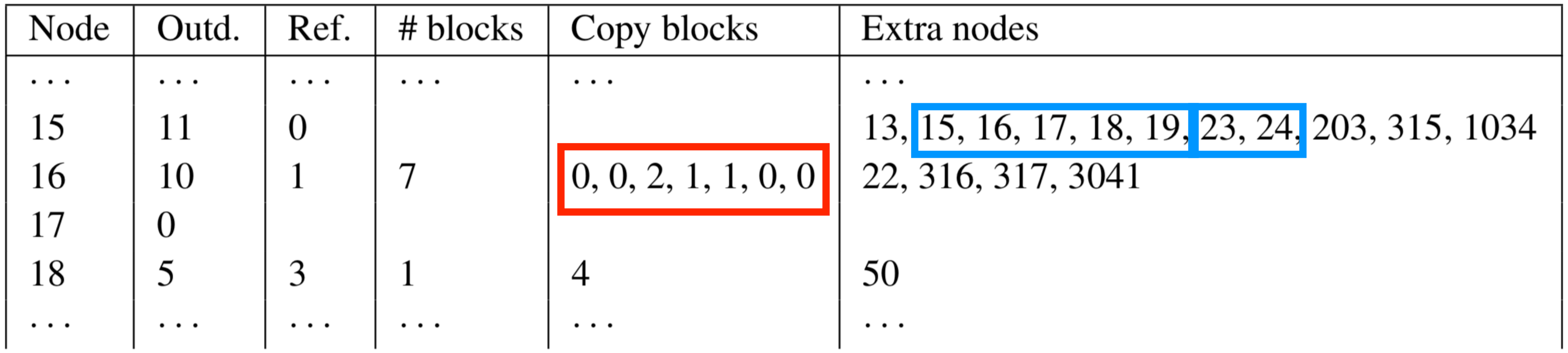
Copy blocks

The WebGraph Framework



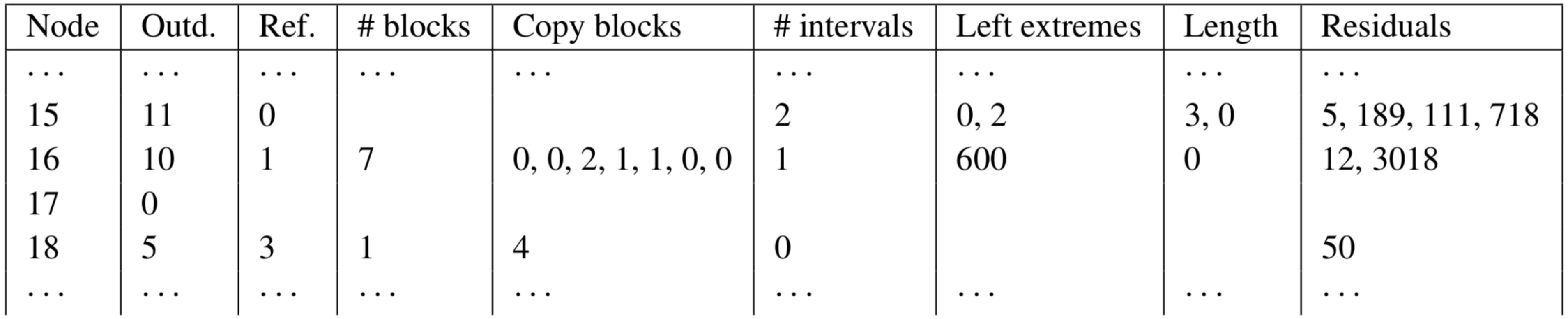
Node	Outd.	Ref.	Copy list	Extra nodes
. . .	. . .	. . .	. . .	. . .
15	11	0		13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	01110011010	22, 316, 317, 3041
17	0			
18	5	3	11110000000	50
. . .	. . .	. . .	. . .	. . .

Copy lists



Node	Outd.	Ref.	# blocks	Copy blocks	Extra nodes
. . .	. . .	. . .	. . .	. . .	. . .
15	11	0			13, 15, 16, 17, 18, 19, 23, 24, 203, 315, 1034
16	10	1	7	0, 0, 2, 1, 1, 0, 0	22, 316, 317, 3041
17	0				
18	5	3	1	4	50
. . .	. . .	. . .	. . .	. . .	. . .

Copy blocks



Node	Outd.	Ref.	# blocks	Copy blocks	# intervals	Left extremes	Length	Residuals
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
15	11	0			2	0, 2	3, 0	5, 189, 111, 718
16	10	1	7	0, 0, 2, 1, 1, 0, 0	1	600	0	12, 3018
17	0							
18	5	3	1	4	0			50
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

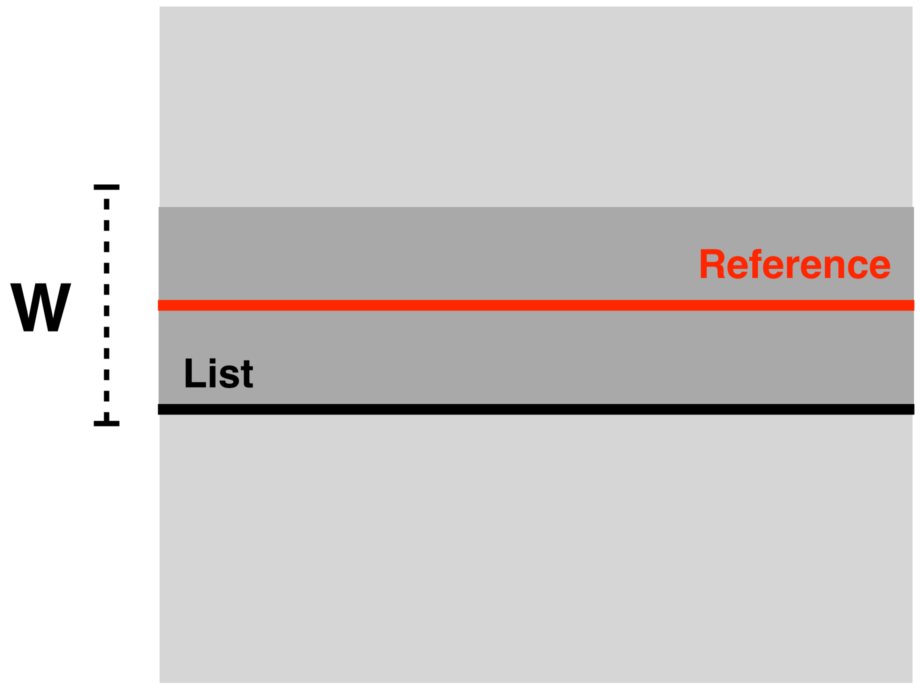
Intervals

An interval is a run, of size ≥ L, of consecutive integers.

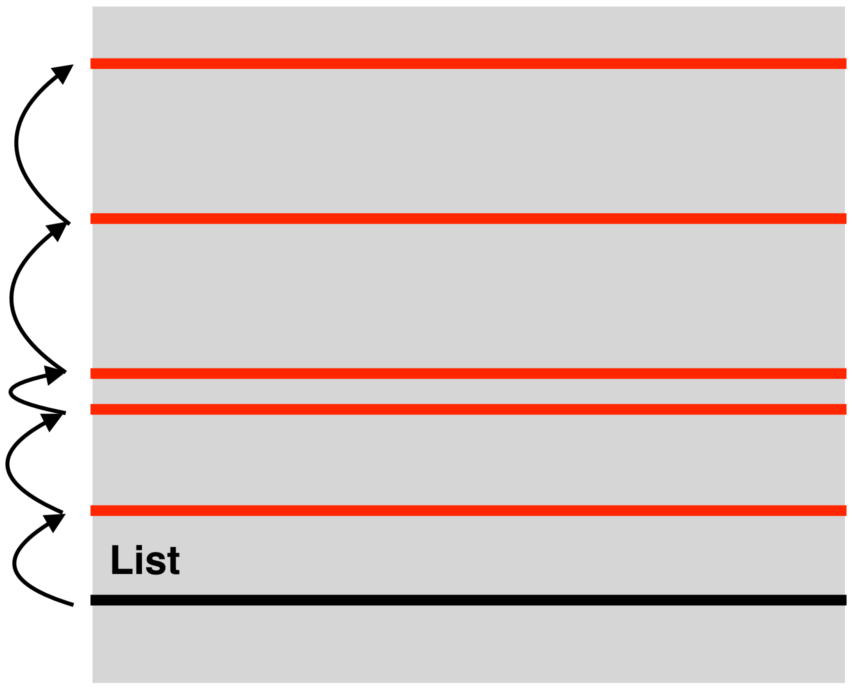
Example for L = 2.

The WebGraph Framework

W - window size



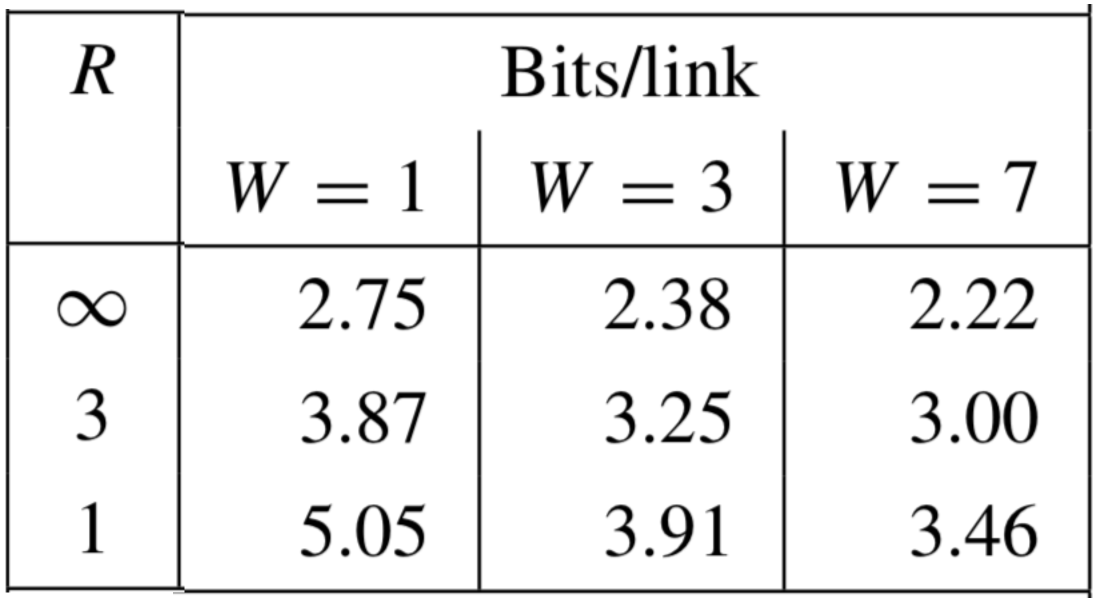
R - maximum reference chain



Tradeoff between compression and decoding time.

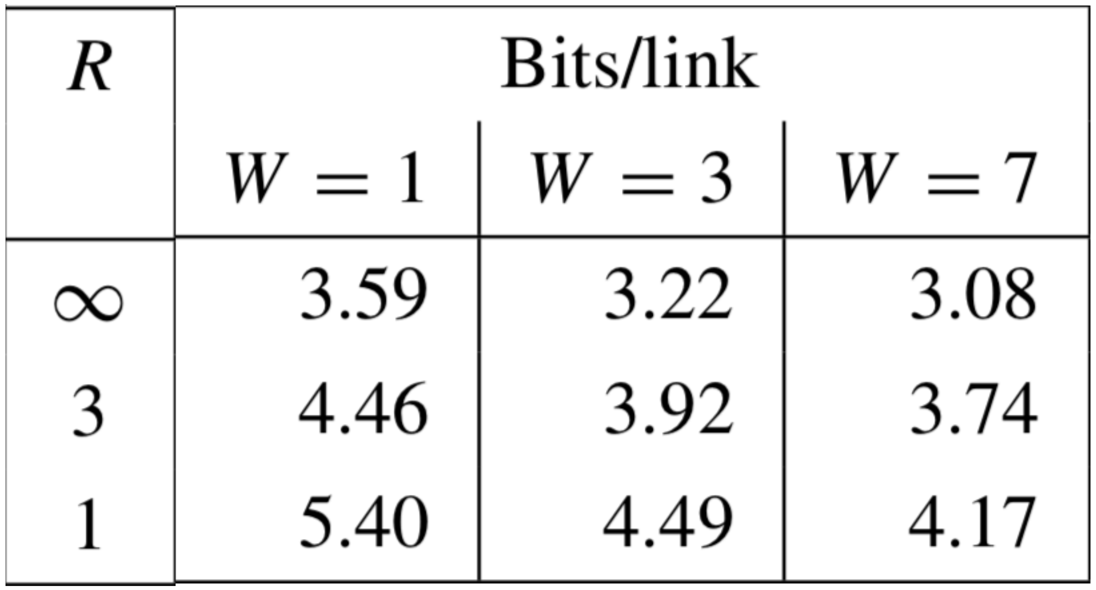
The WebGraph Framework

`.uk`
18.5 million pages
300 million links



R	Bits/link		
	$W = 1$	$W = 3$	$W = 7$
∞	2.75	2.38	2.22
3	3.87	3.25	3.00
1	5.05	3.91	3.46

`WebBase`
118 million pages
1 billion links

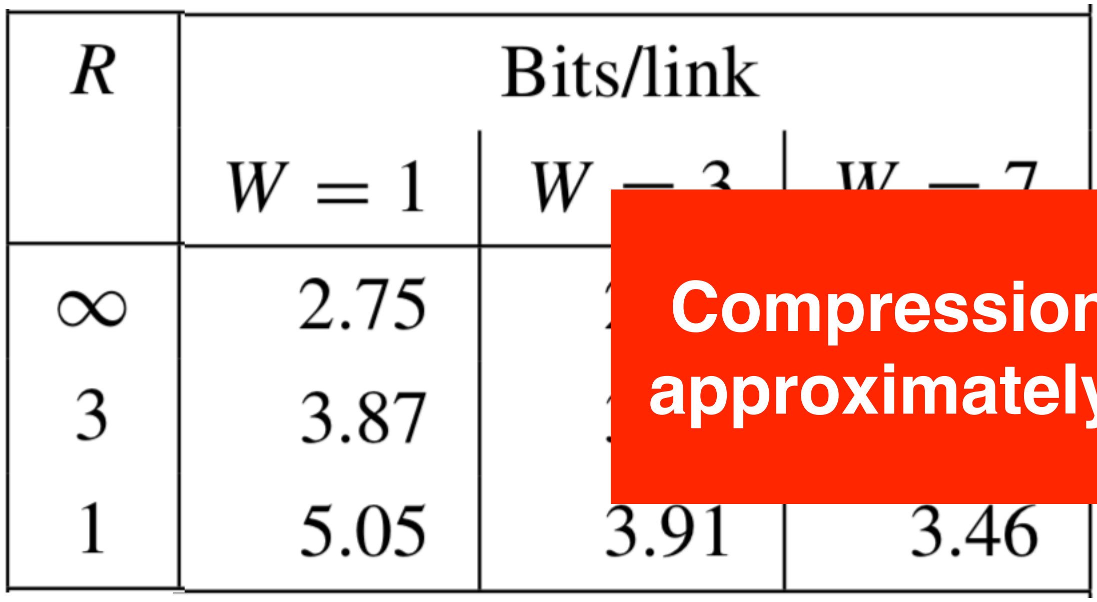


R	Bits/link		
	$W = 1$	$W = 3$	$W = 7$
∞	3.59	3.22	3.08
3	4.46	3.92	3.74
1	5.40	4.49	4.17

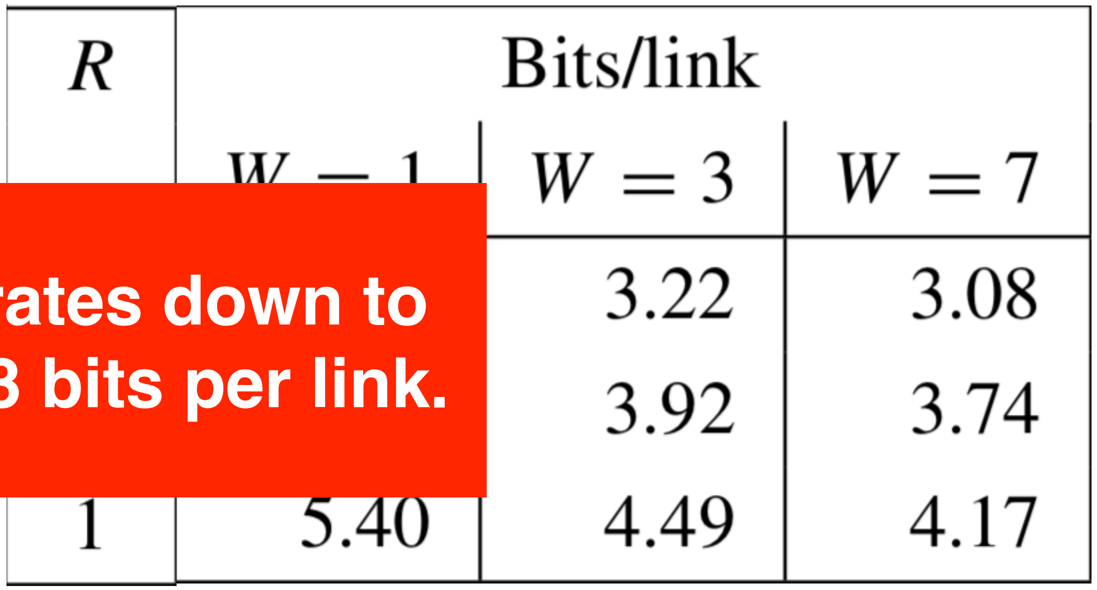
The WebGraph Framework

`.uk`
18.5 million pages
300 million links

`WebBase`
118 million pages
1 billion links



R	Bits/link		
	$W = 1$	$W = 3$	$W = 7$
∞	2.75		
3	3.87		
1	5.05	3.91	3.46

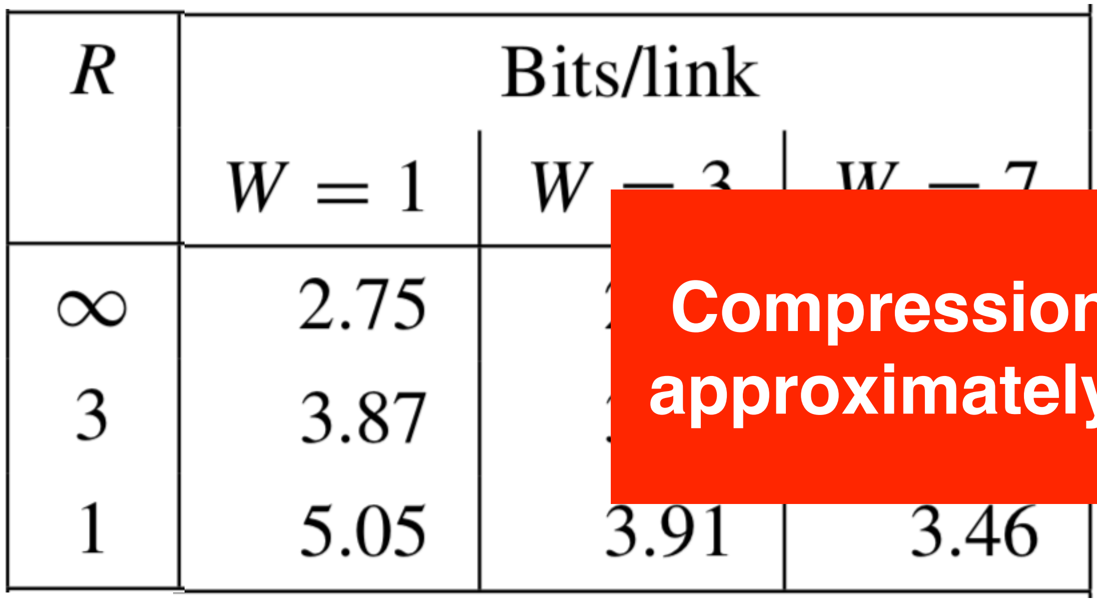


R	Bits/link		
	$W = 1$	$W = 3$	$W = 7$
		3.22	3.08
		3.92	3.74
1	5.40	4.49	4.17

Compression rates down to approximately 3 bits per link.

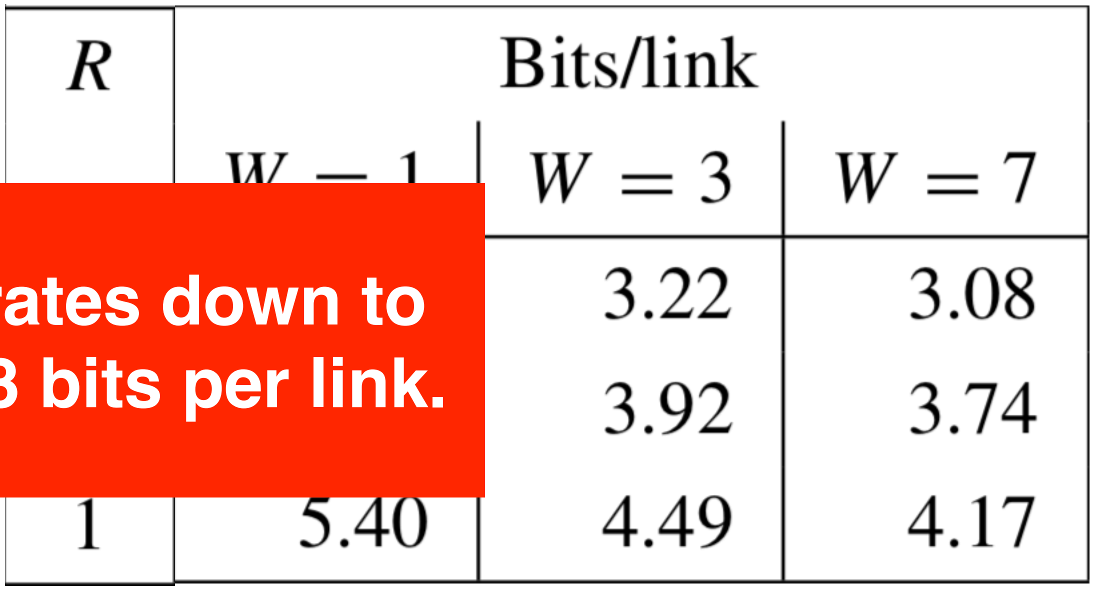
The WebGraph Framework

`.uk`
18.5 million pages
300 million links



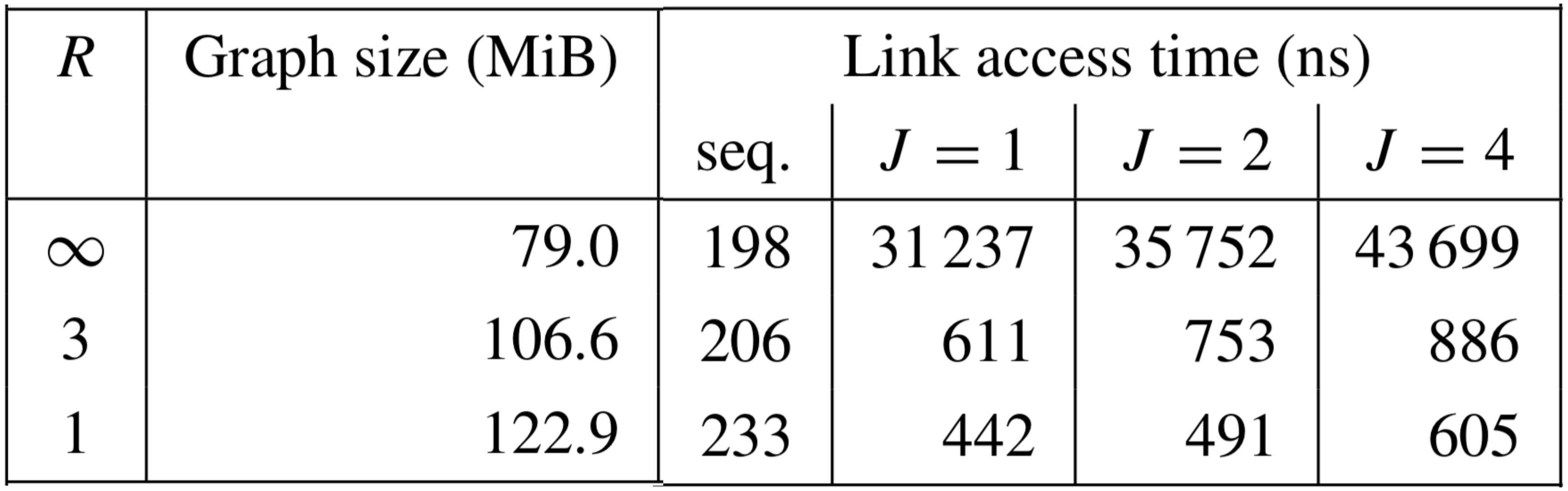
R	Bits/link		
	$W = 1$	$W = 3$	$W = 7$
∞	2.75	[illegible]	[illegible]
3	3.87	[illegible]	[illegible]
1	5.05	3.91	3.46

`WebBase`
118 million pages
1 billion links



R	Bits/link		
	$W = 1$	$W = 3$	$W = 7$
[illegible]	[illegible]	3.22	3.08
[illegible]	[illegible]	3.92	3.74
1	5.40	4.49	4.17

Compression rates down to approximately 3 bits per link.



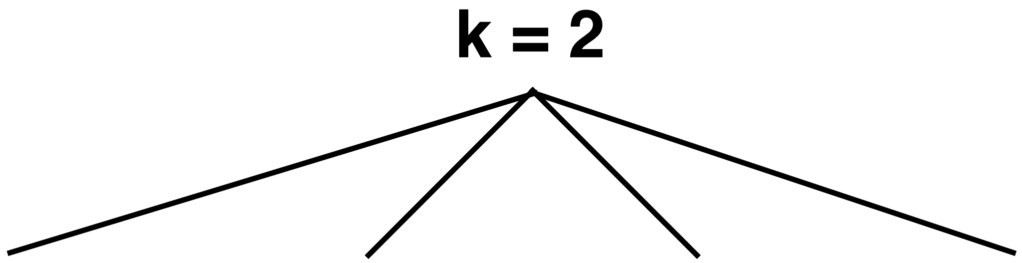
R	Graph size (MiB)	Link access time (ns)			
		seq.	$J = 1$	$J = 2$	$J = 4$
∞	79.0	198	31 237	35 752	43 699
3	106.6	206	611	753	886
1	122.9	233	442	491	605

k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009

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

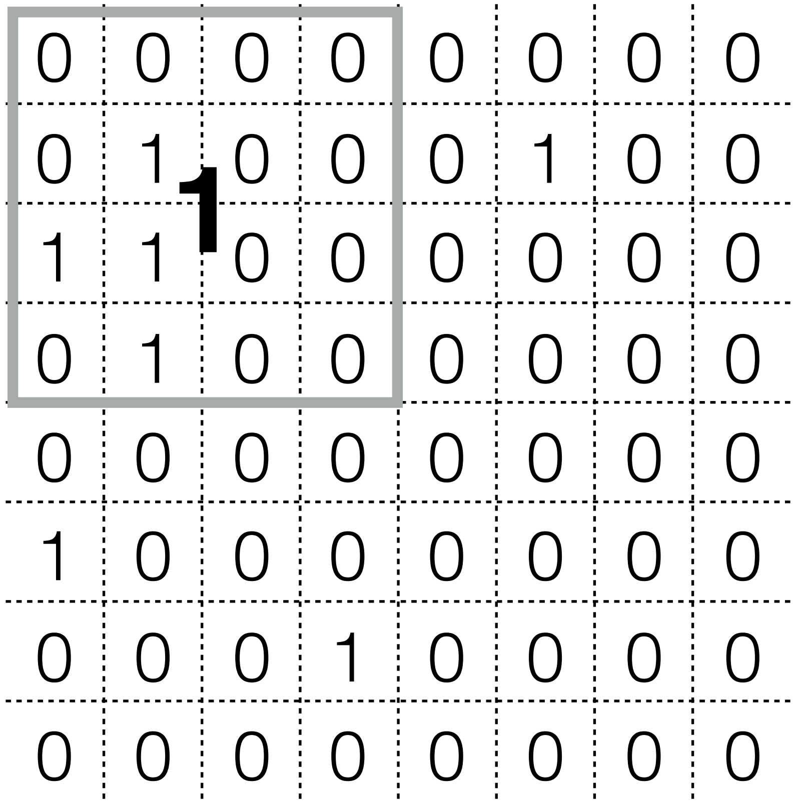


Credits to A. Gómez-Brandón

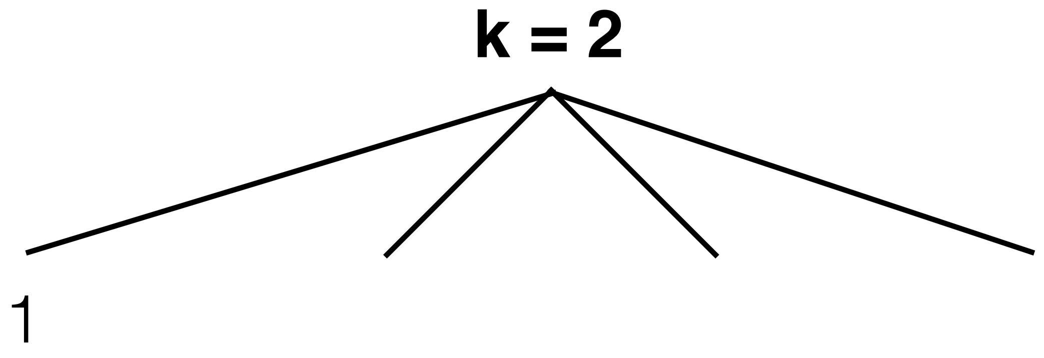
k^2-trees

k^2-ary tree
representation of the
adjacency matrix.

k^2-trees for Compact Web Graph Representation,
Brisaboa-Ladra-Navarro, SPIRE 2009



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

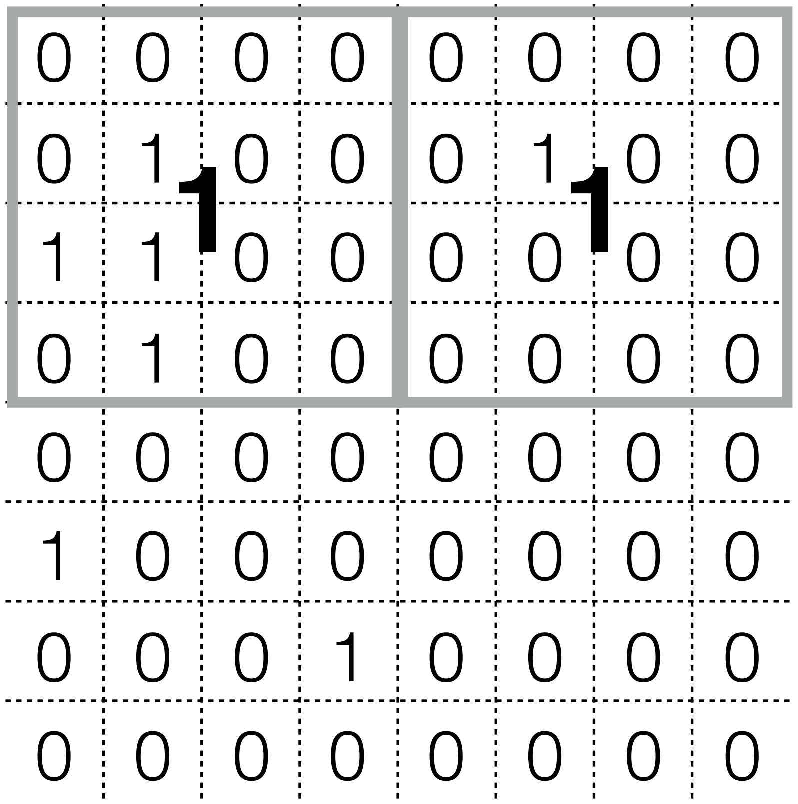


Credits to A. Gómez-Brandón

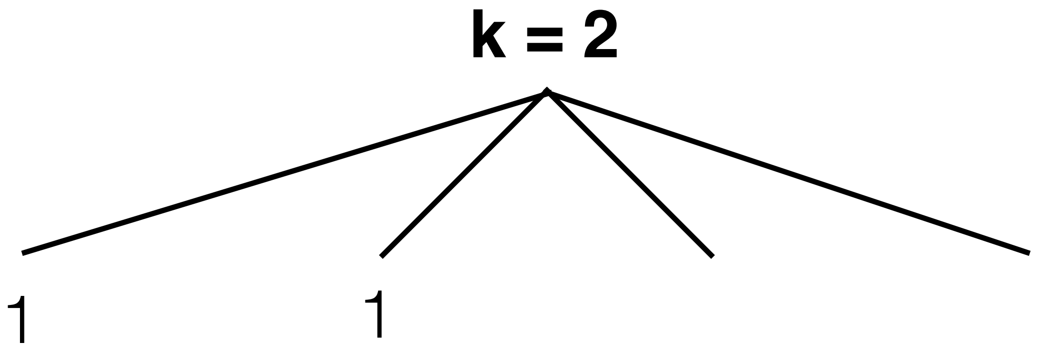
k²-trees

k²-ary tree representation of the adjacency matrix.

k²-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

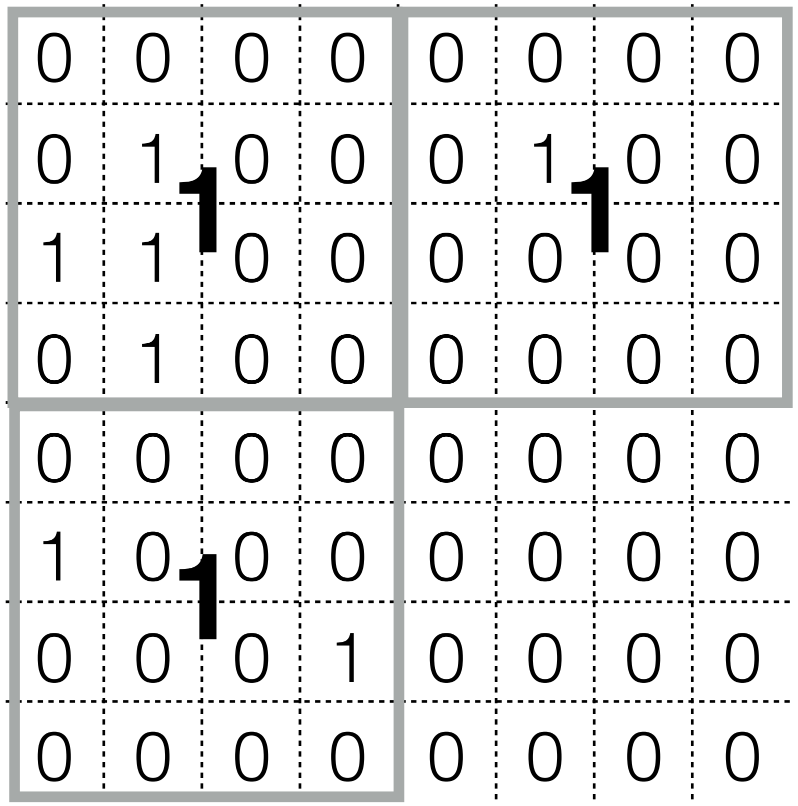


Credits to A. Gómez-Brandón

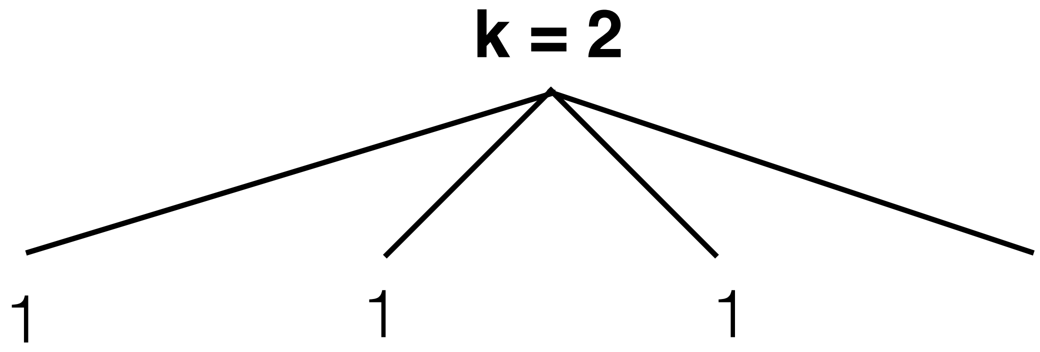
k²-trees

k²-ary tree representation of the adjacency matrix.

k²-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

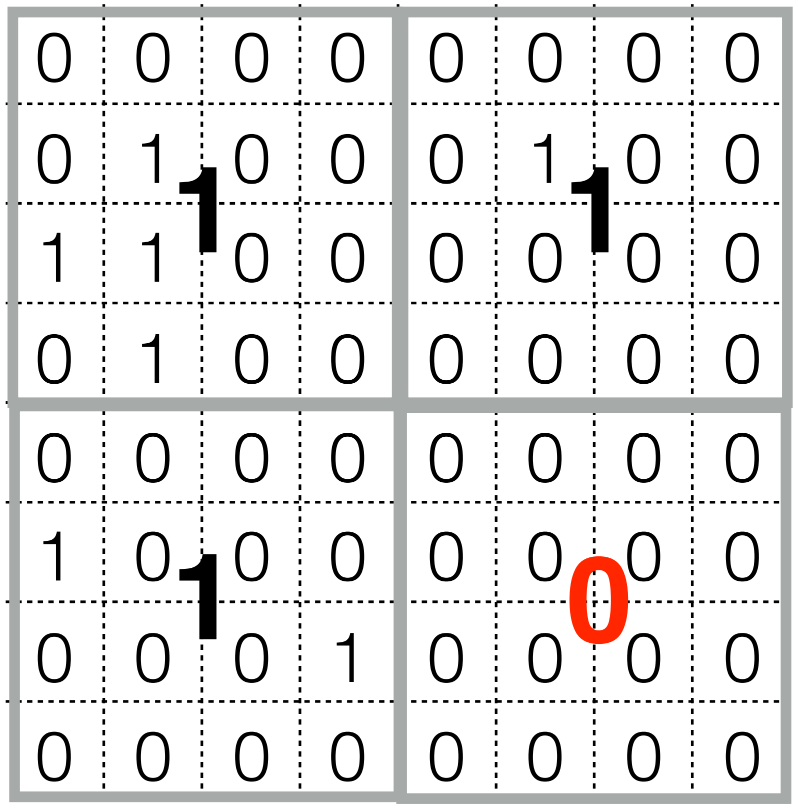


Credits to A. Gómez-Brandón

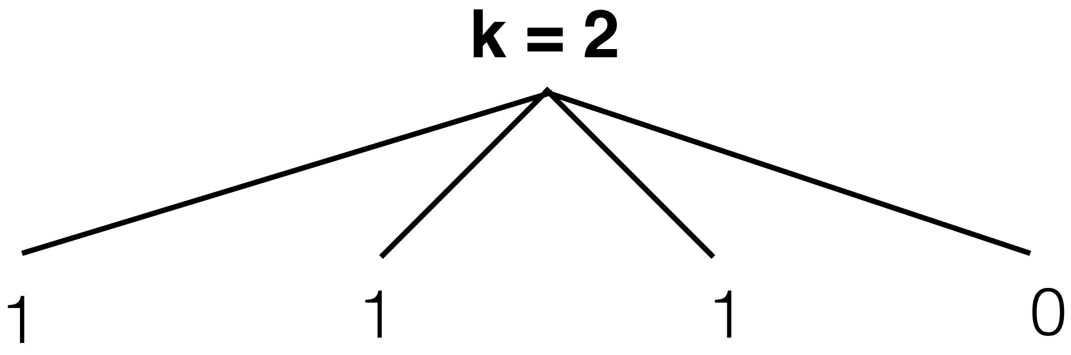
k²-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

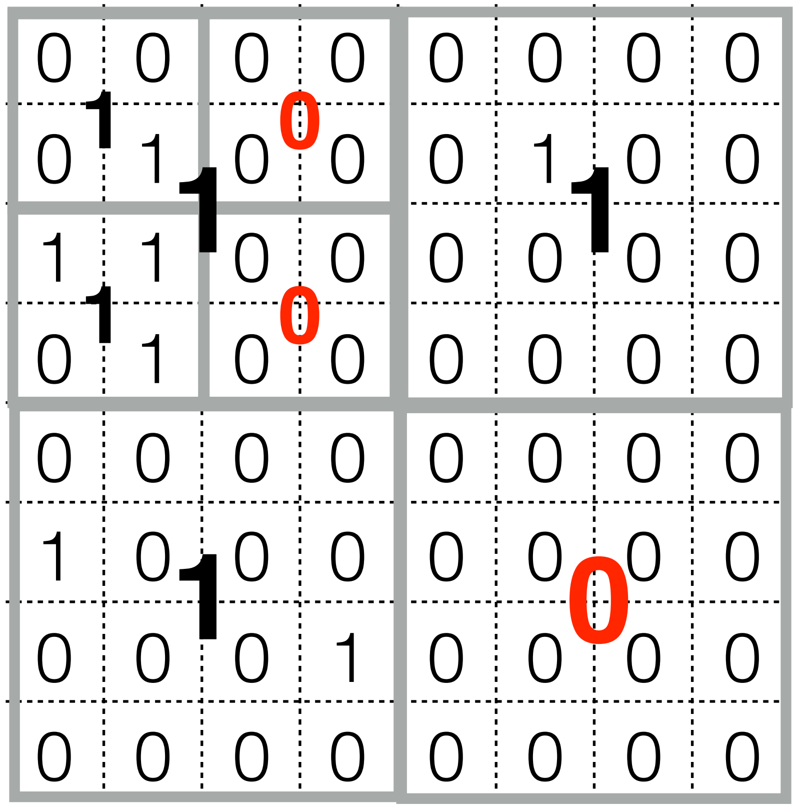


Credits to A. Gómez-Brandón

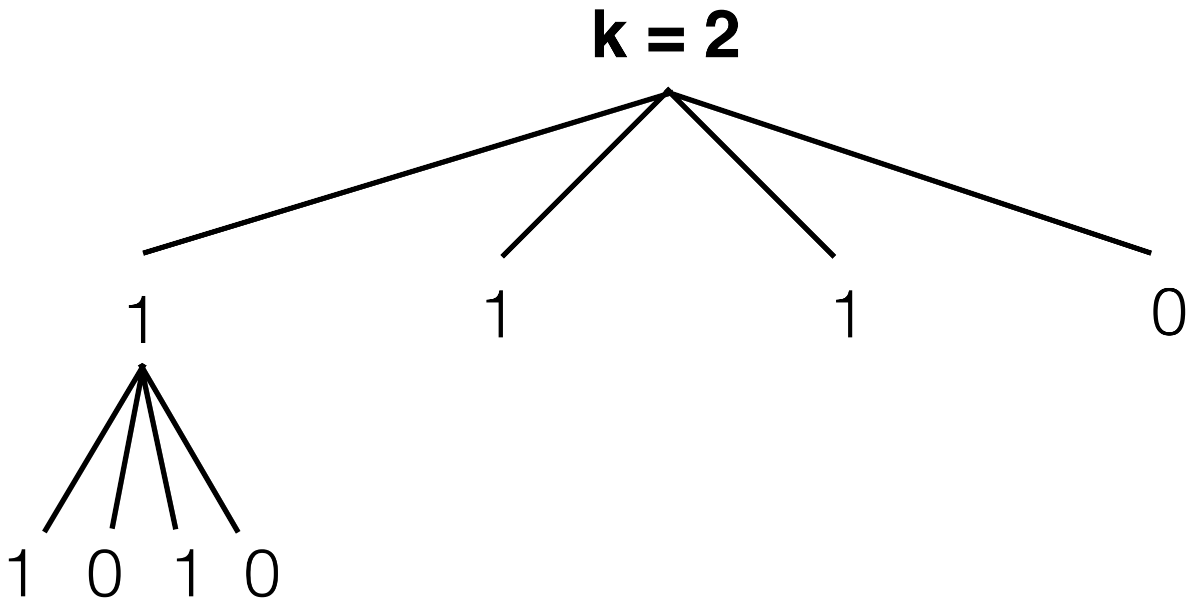
k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

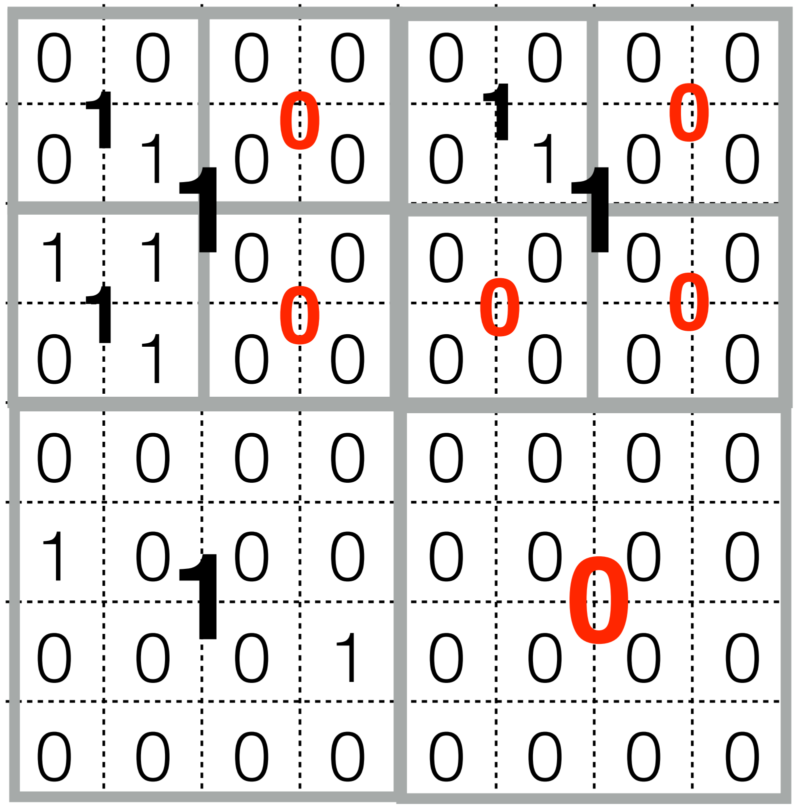


Credits to A. Gómez-Brandón

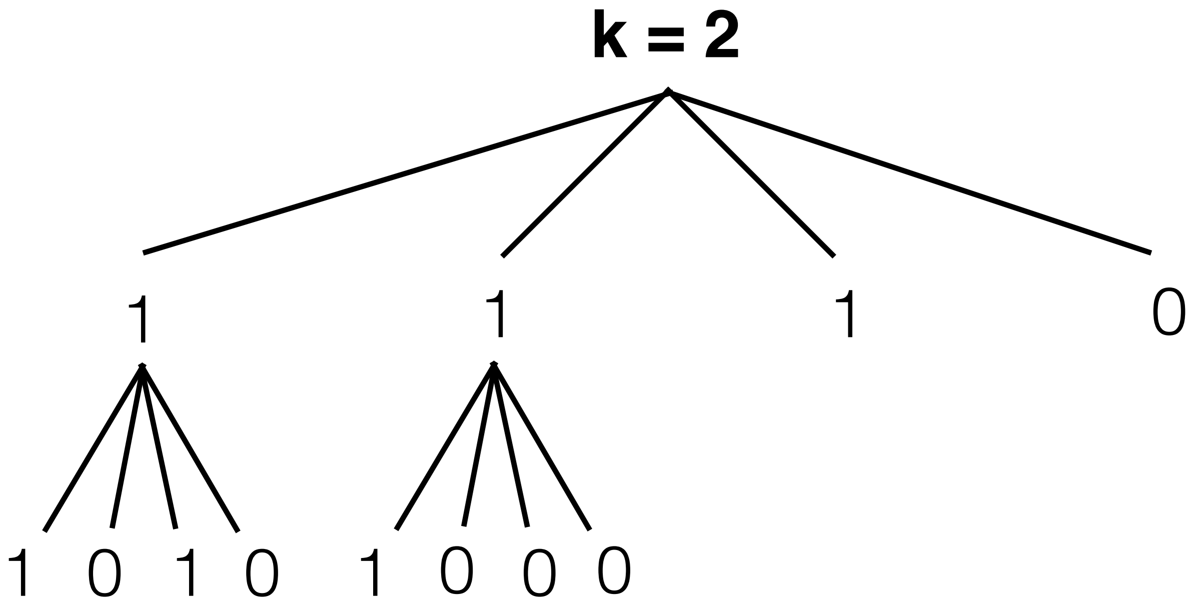
k^2-trees

k^2-ary tree
representation of the
adjacency matrix.

k^2-trees for Compact Web Graph Representation,
Brisaboa-Ladra-Navarro, SPIRE 2009



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

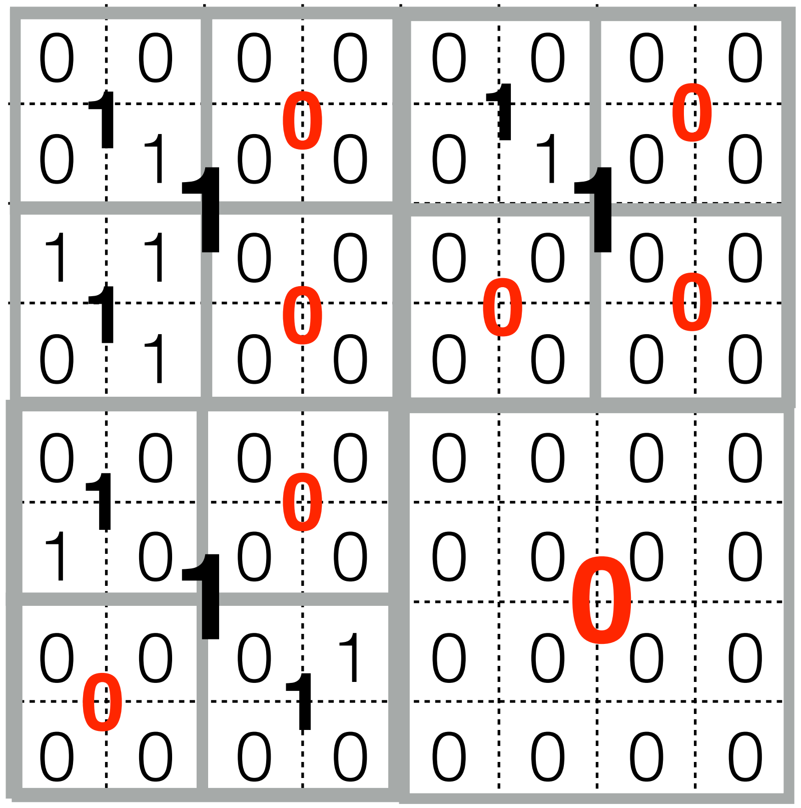


Credits to A. Gómez-Brandón

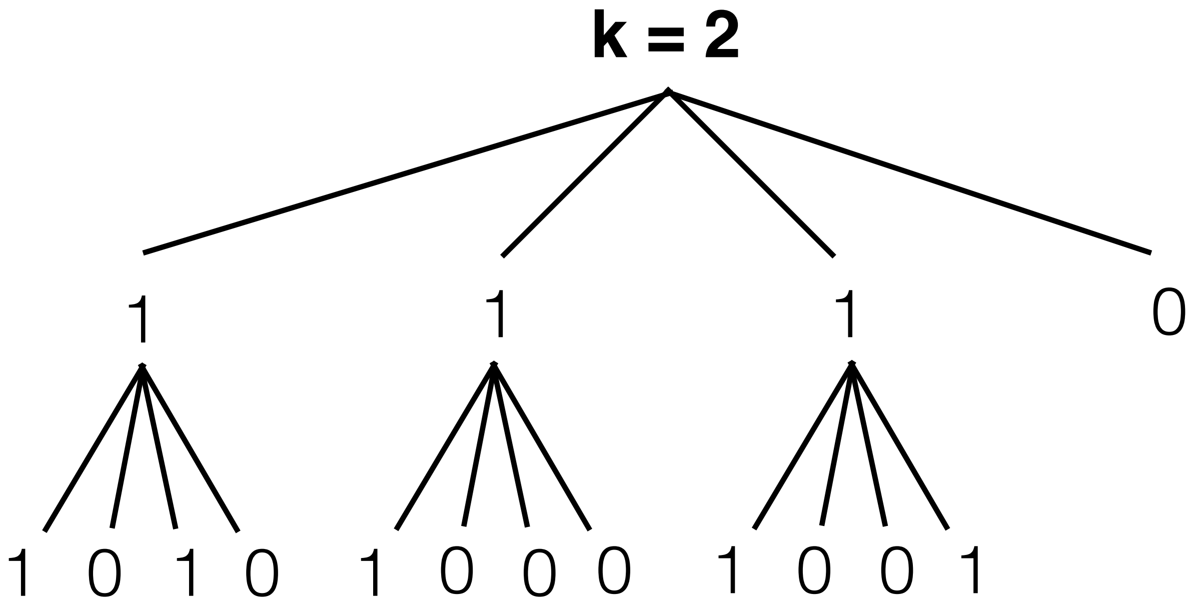
k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

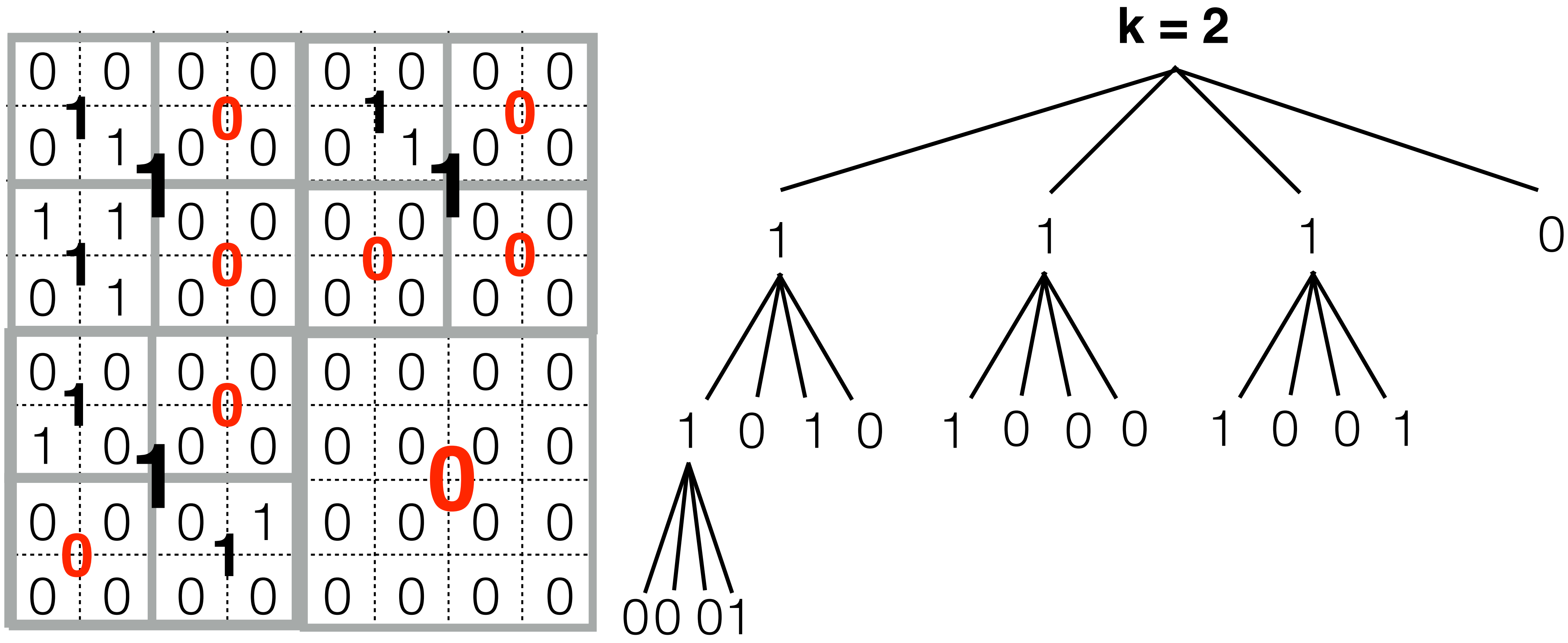


Credits to A. Gómez-Brandón

k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



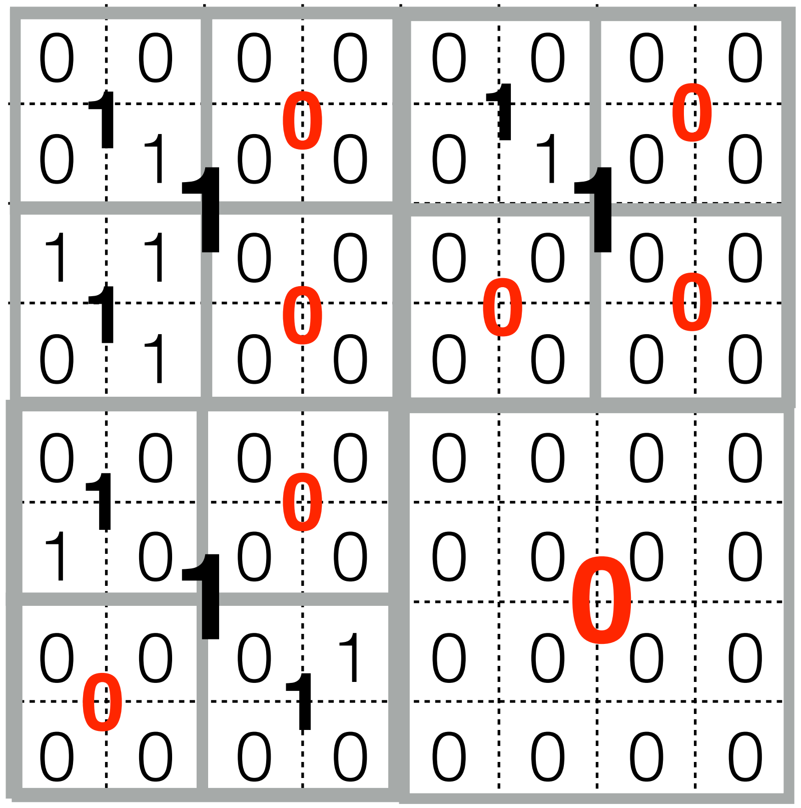
0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0
1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0

Credits to A. Gómez-Brandón

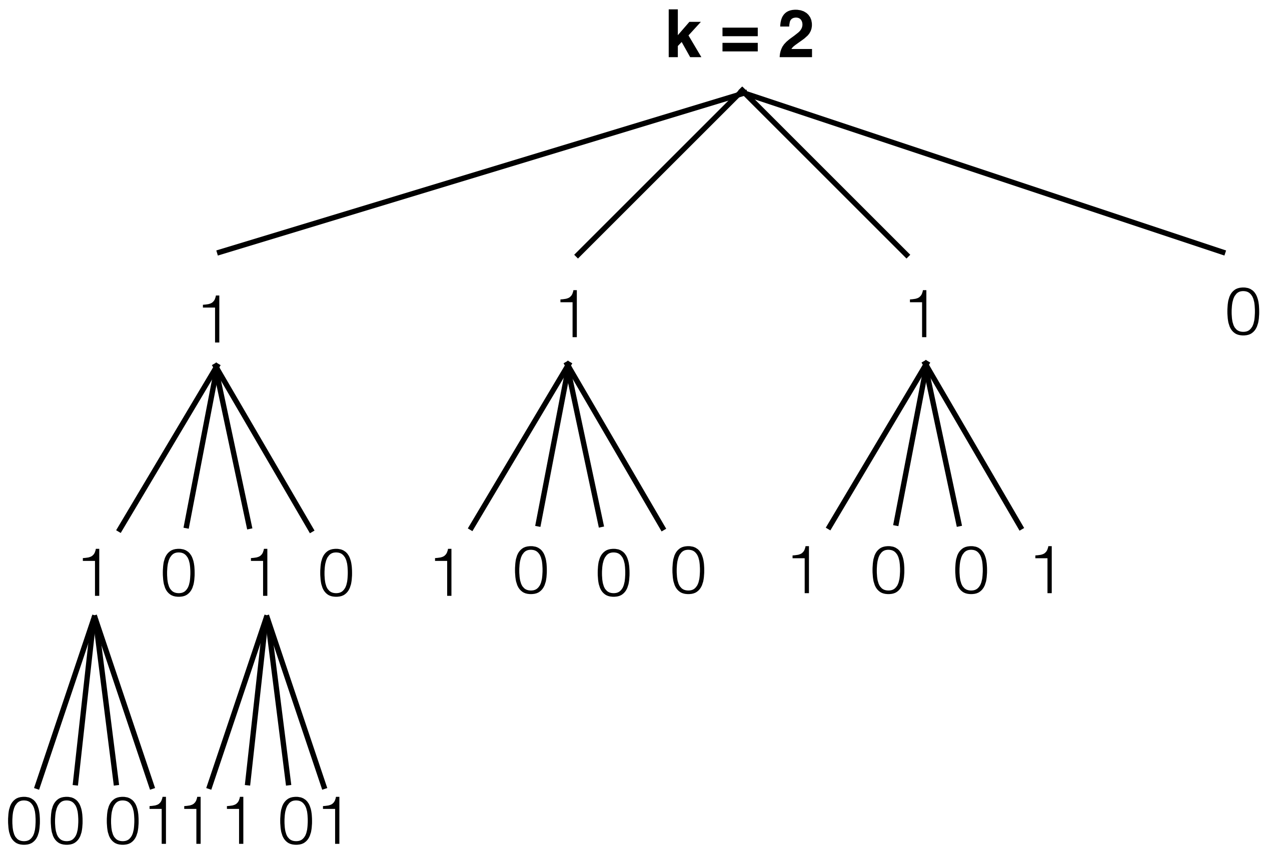
k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

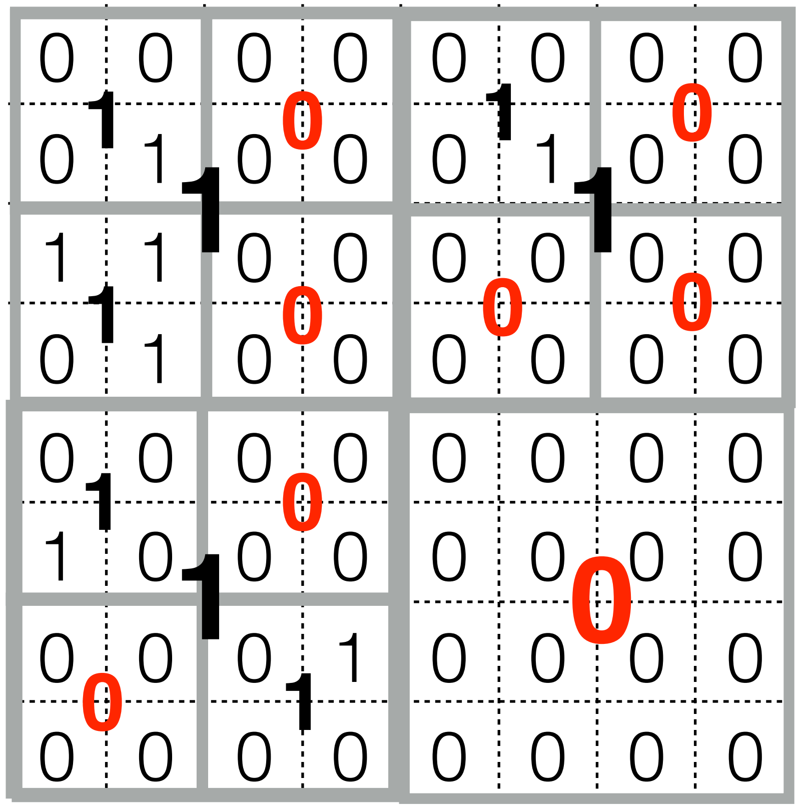


Credits to A. Gómez-Brandón

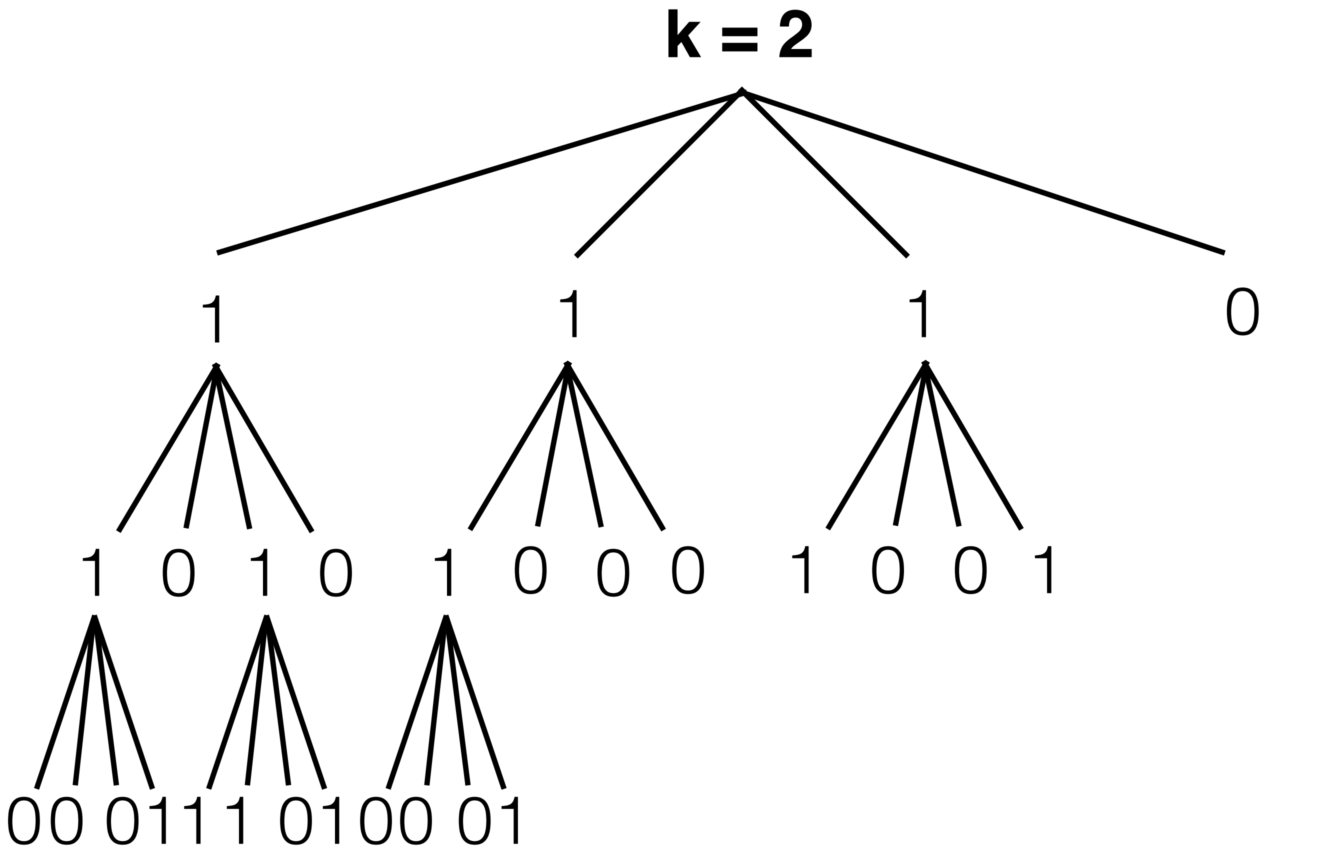
k²-trees

k²-ary tree representation of the adjacency matrix.

k²-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

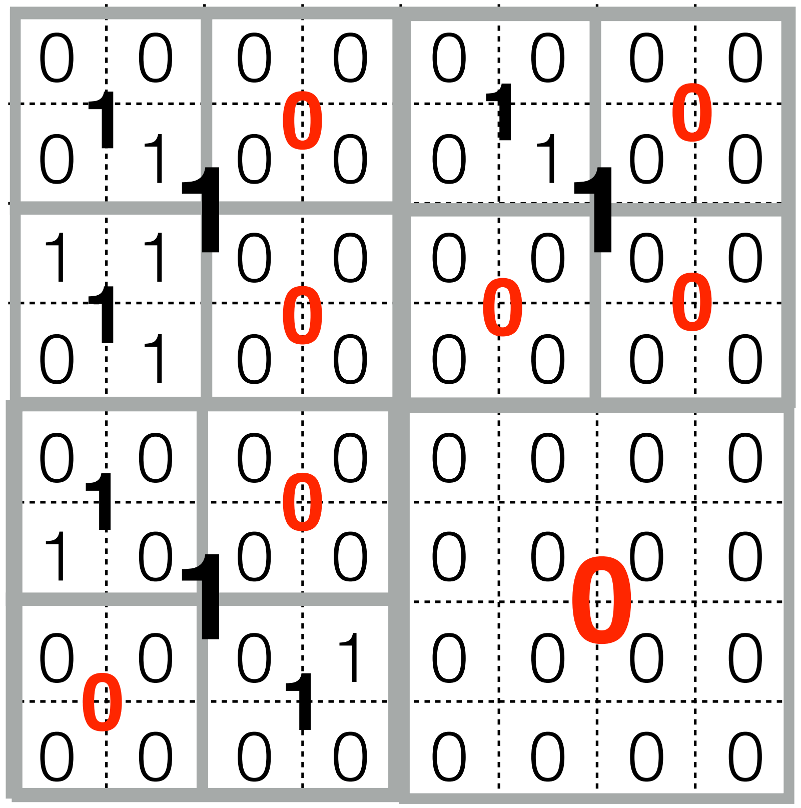


Credits to A. Gómez-Brandón

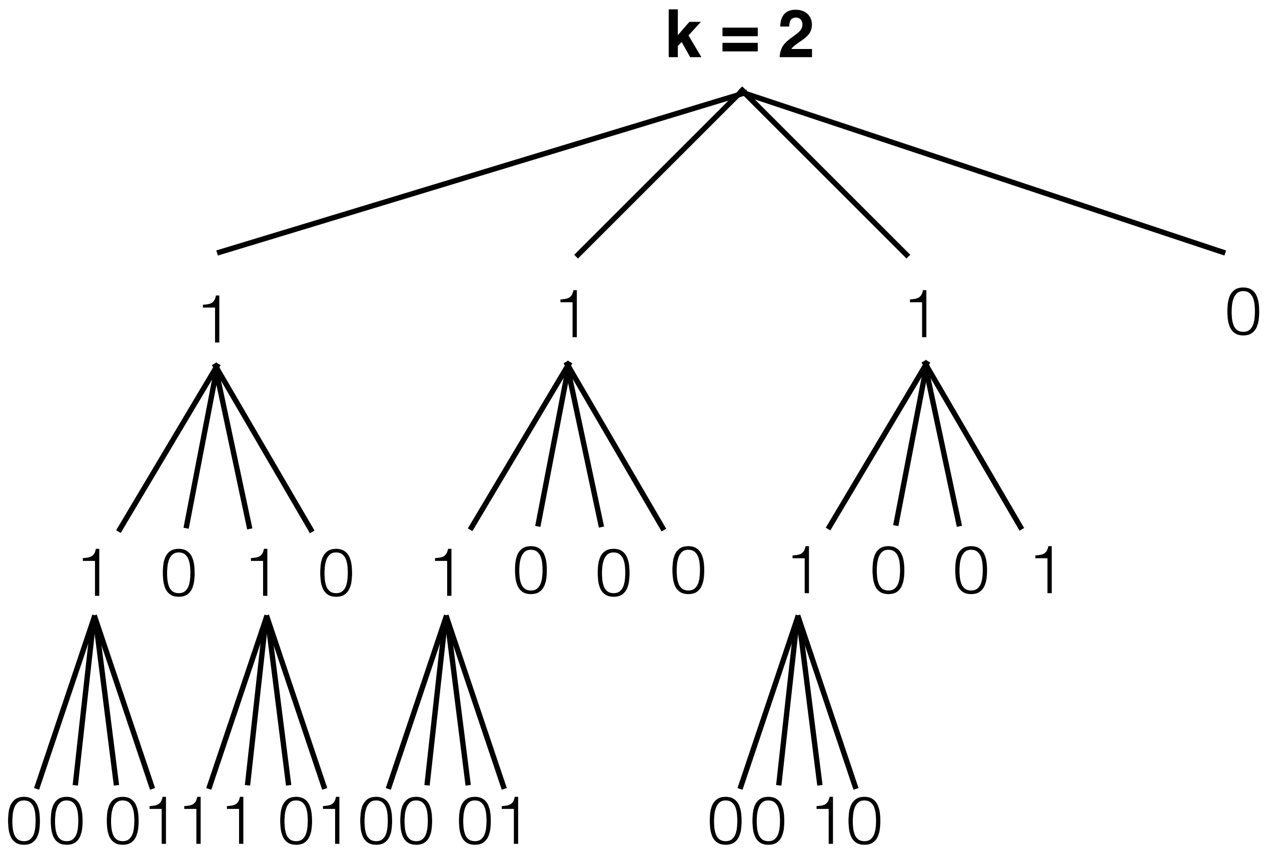
k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

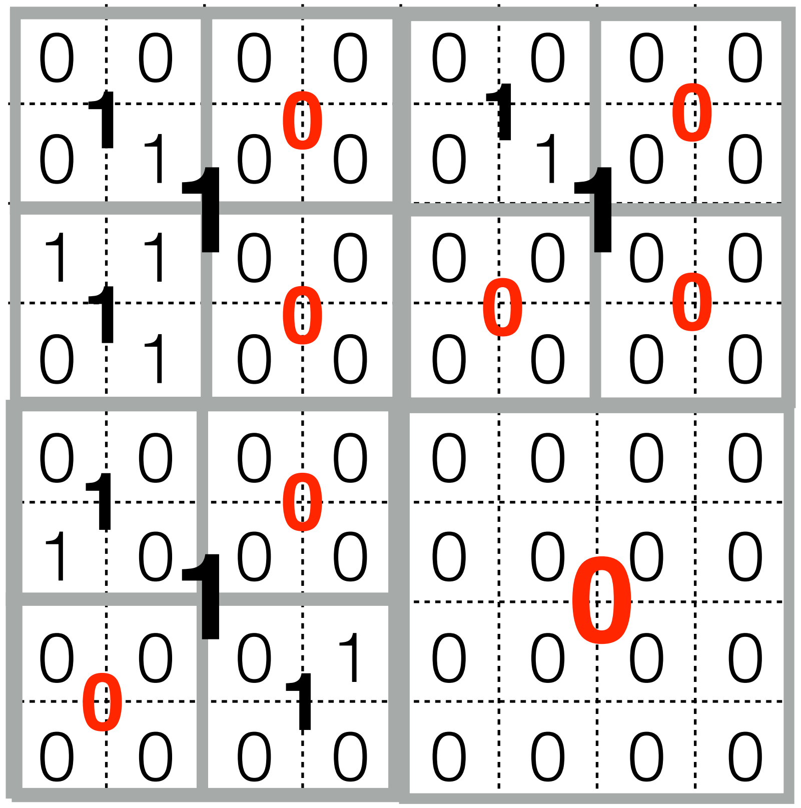


Credits to A. Gómez-Brandón

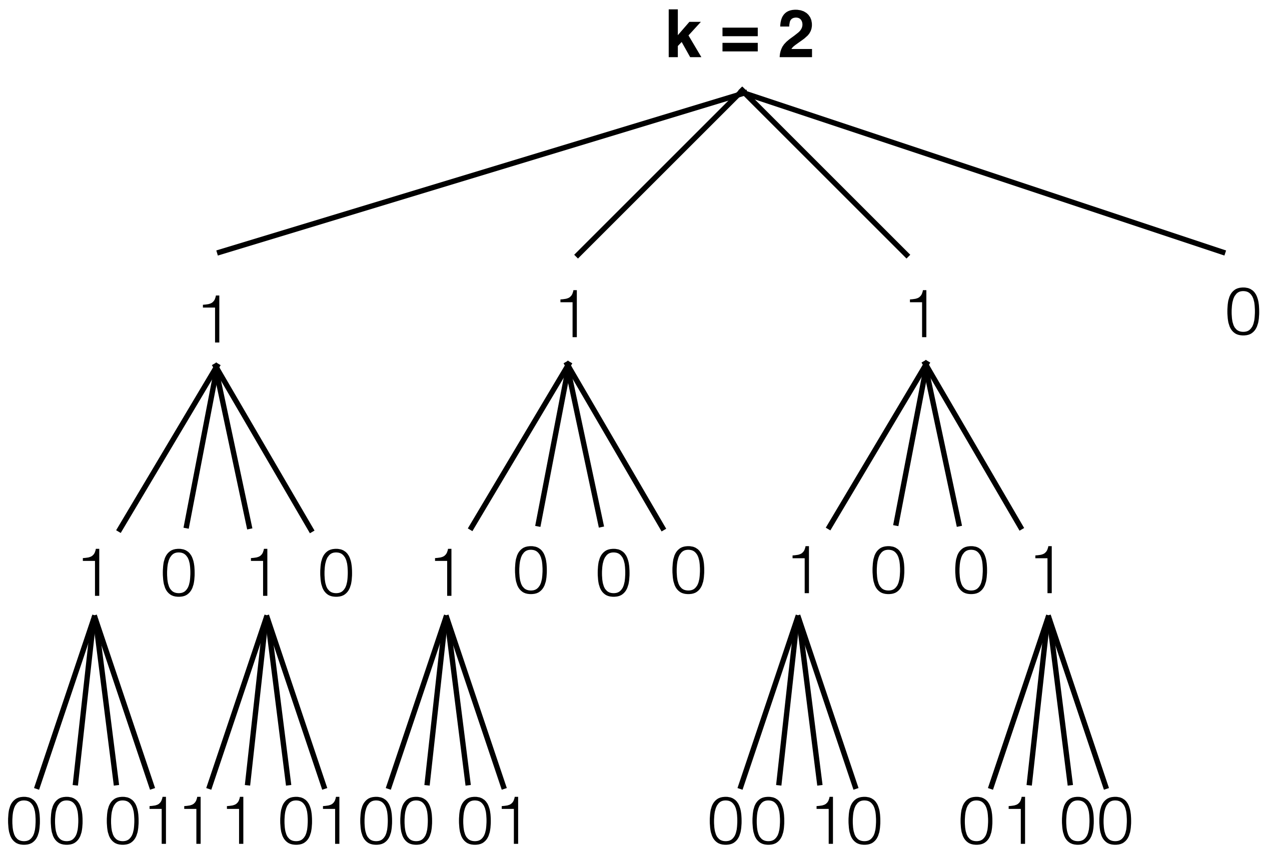
k^2-trees

k^2-ary tree
representation of the
adjacency matrix.

k^2-trees for Compact Web Graph Representation,
Brisaboa-Ladra-Navarro, SPIRE 2009



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

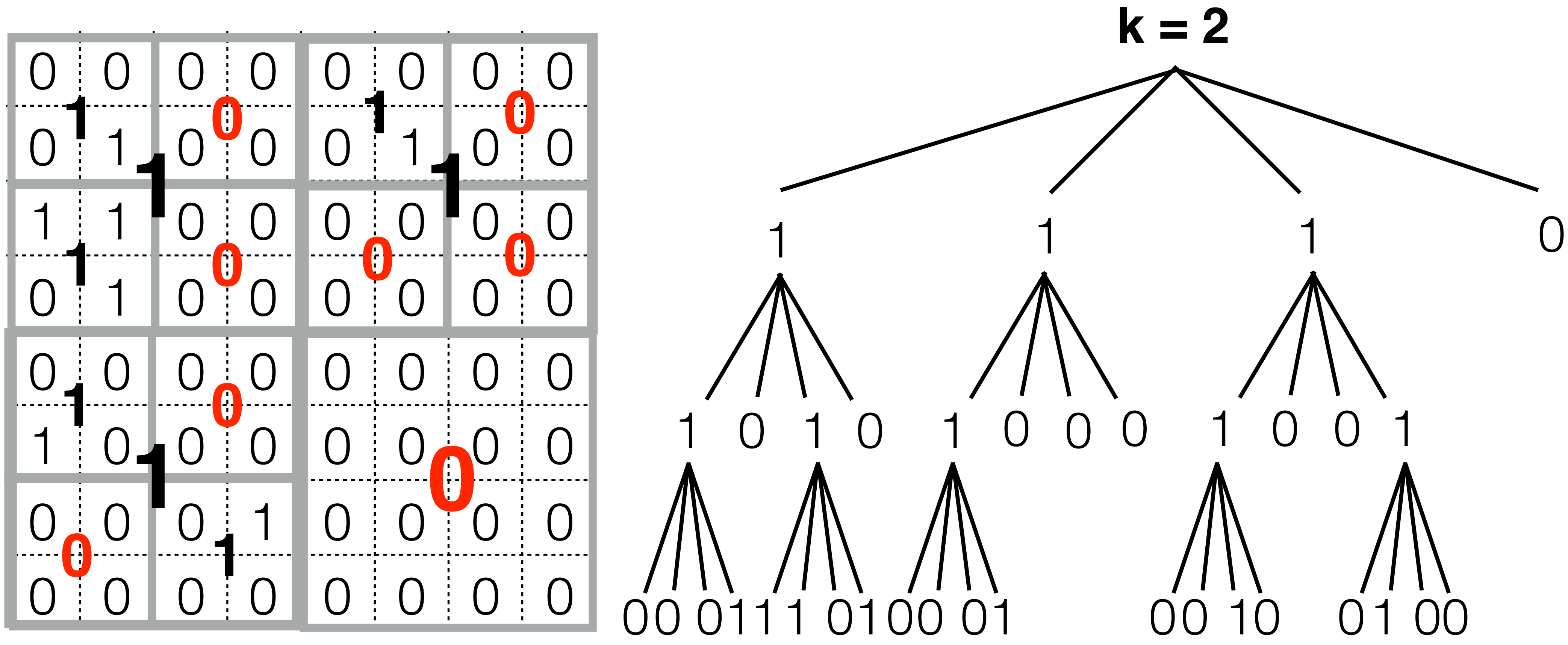


Credits to A. Gómez-Brandón

k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

$$11101010001001 \quad 000111010001001000100$$

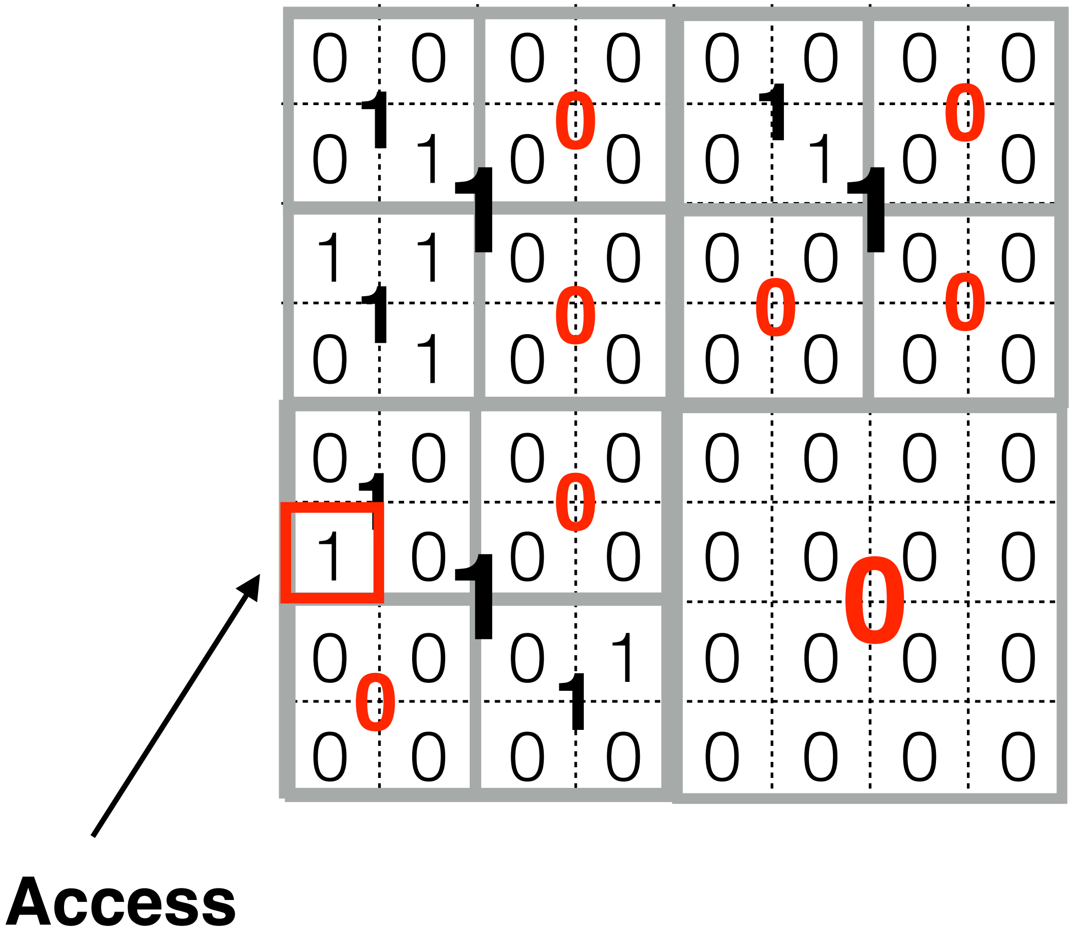
T **L**

Credits to A. Gómez-Brandón

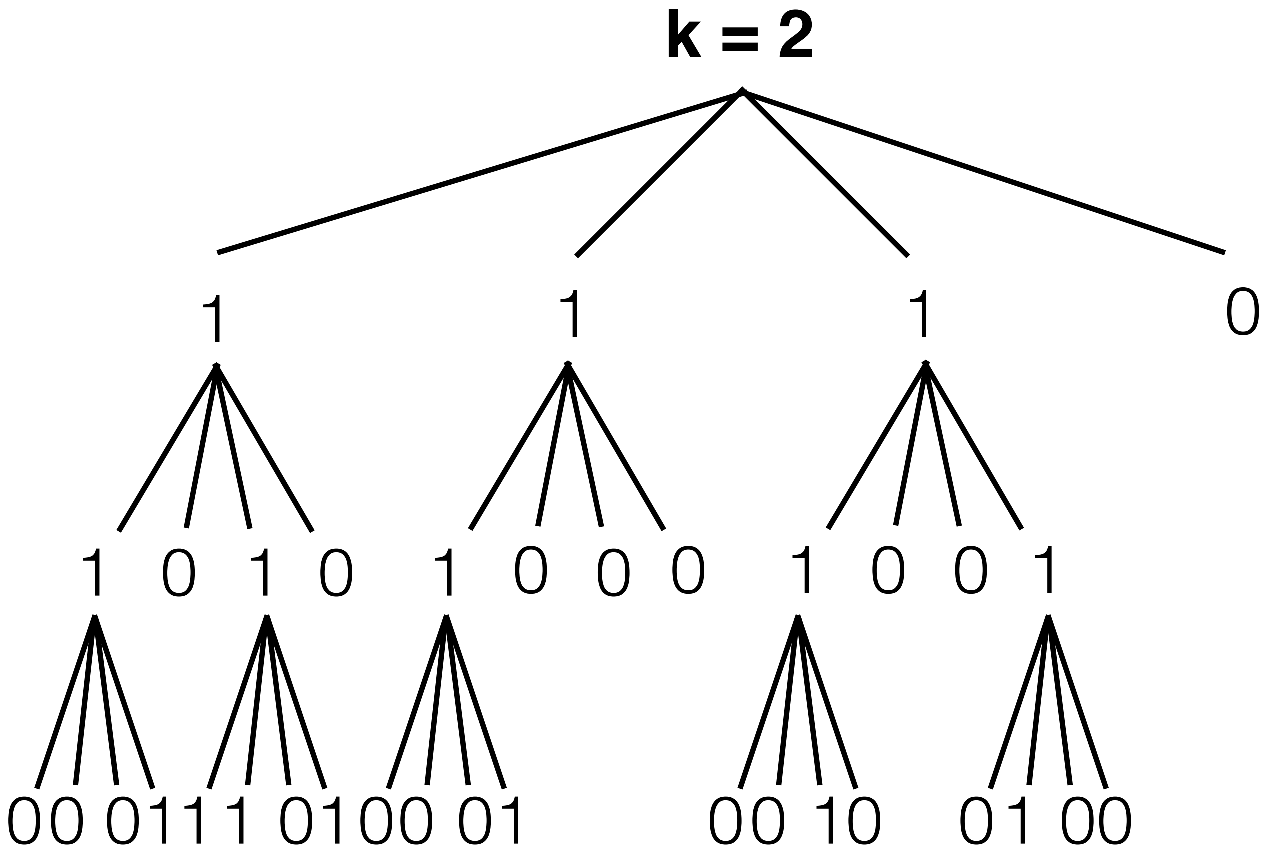
k^2-trees

k^2-ary tree
representation of the
adjacency matrix.

k^2-trees for Compact Web Graph Representation,
Brisaboa-Ladra-Navarro, SPIRE 2009



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



1110101010001001 0001110100010010100

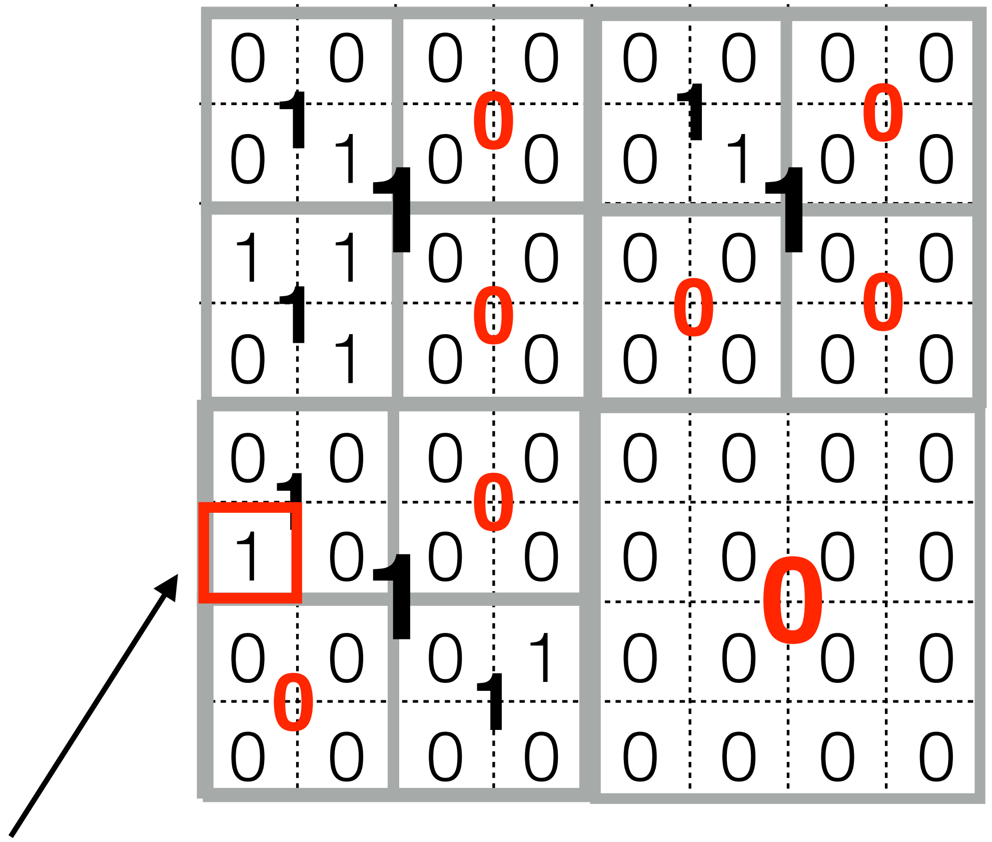
T **L**

Credits to A. Gómez-Brandón

k²-trees

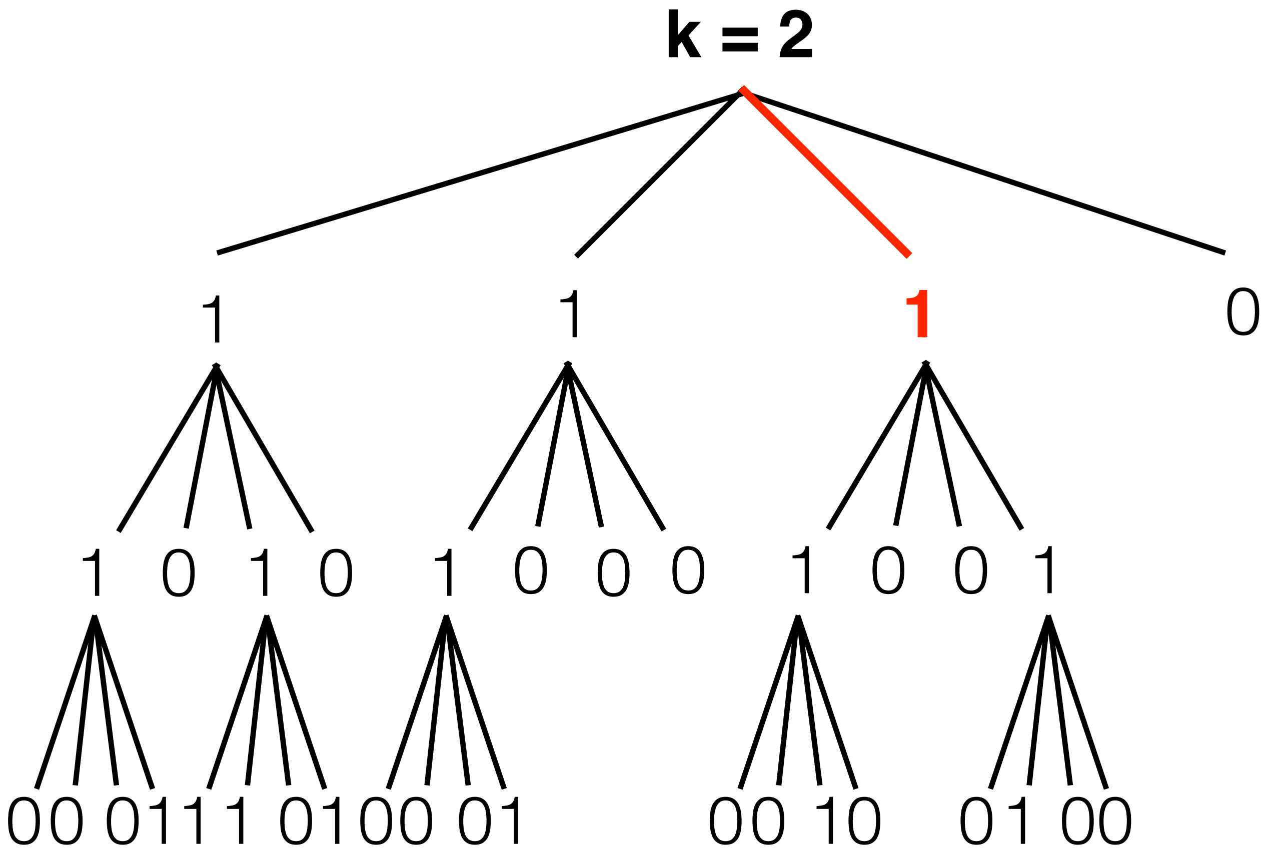
k²-ary tree
representation of the
adjacency matrix.

k²-trees for Compact Web Graph Representation,
Brisaboa-Ladra-Navarro, SPIRE 2009



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

Access



111010101000100 1 0001110100010010 0

11101010100010 01 is shown as: 1110101010001001   0001110100010010 0100

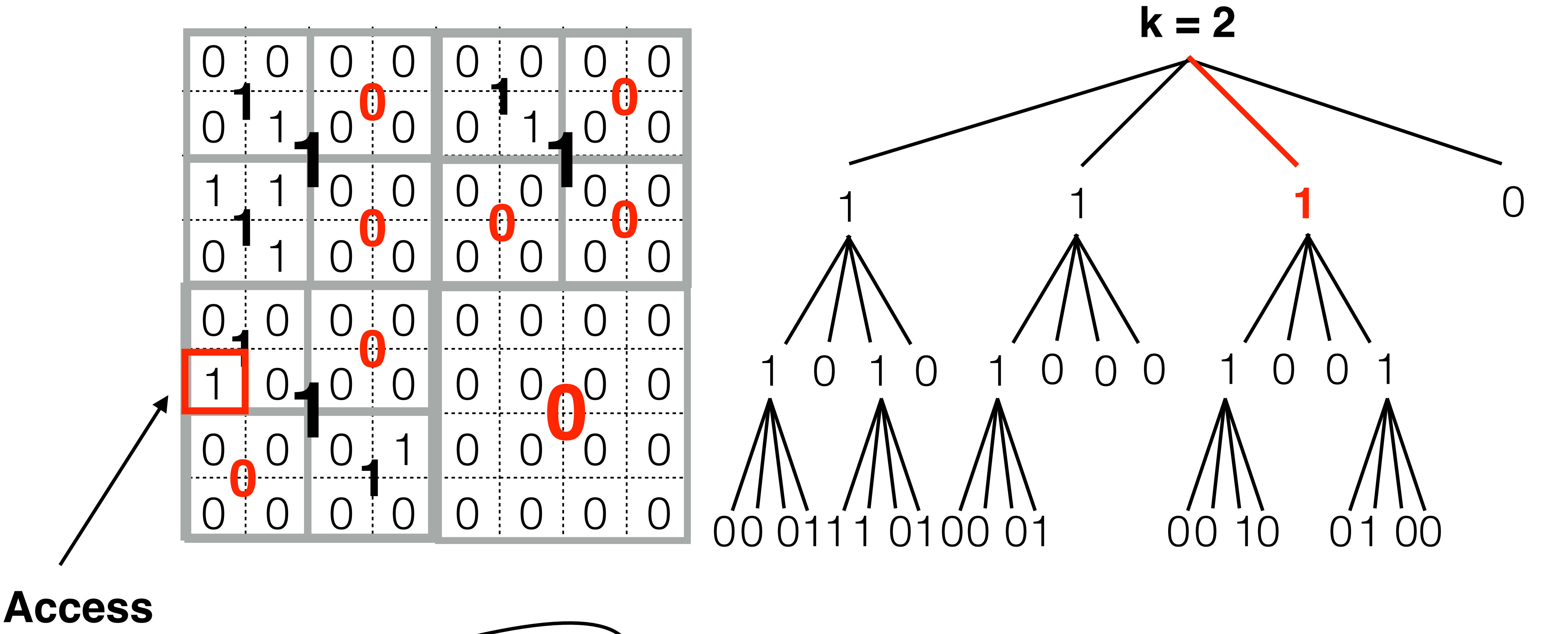
T **L**

Credits to A. Gómez-Brandón

k^2-trees

k^2-ary tree
representation of the
adjacency matrix.

k^2-trees for Compact Web Graph Representation,
Brisaboa-Ladra-Navarro, SPIRE 2009



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

11**10**101010001001 00011101000100100100

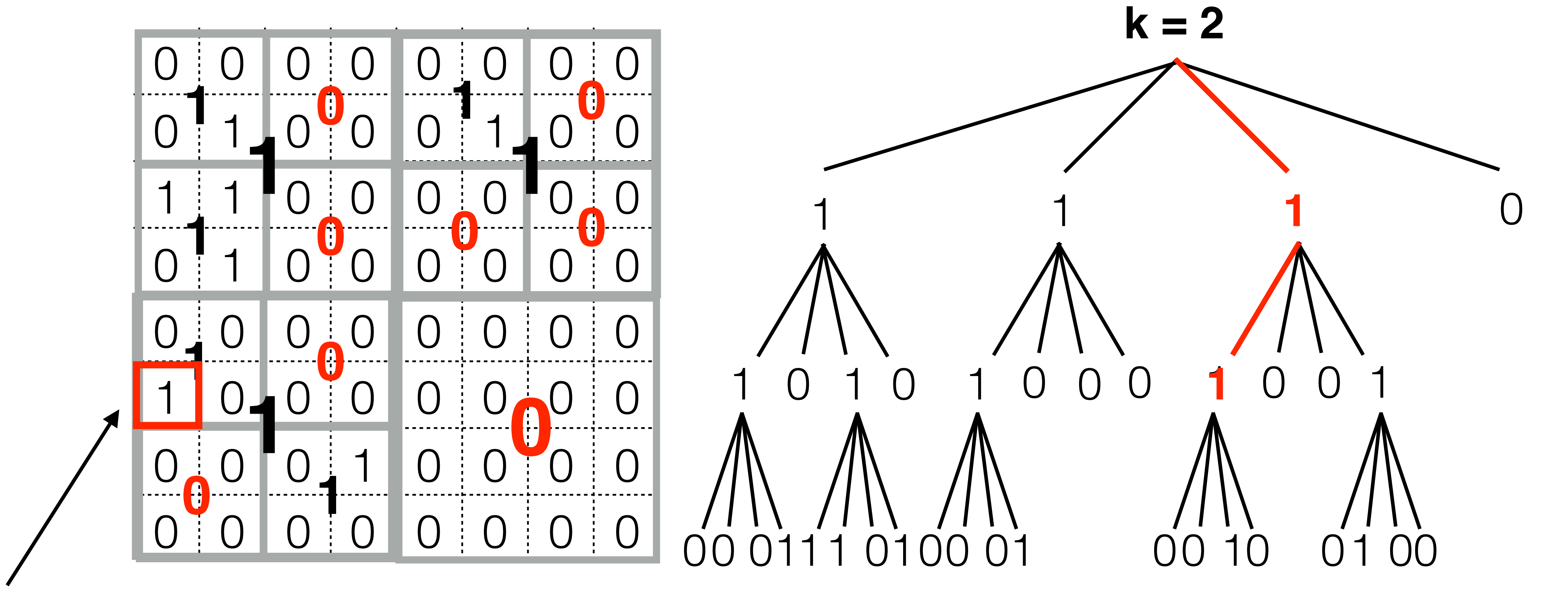
T **L**

Credits to A. Gómez-Brandón

k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

Access

11**1**0101010001001 00011101000100100100

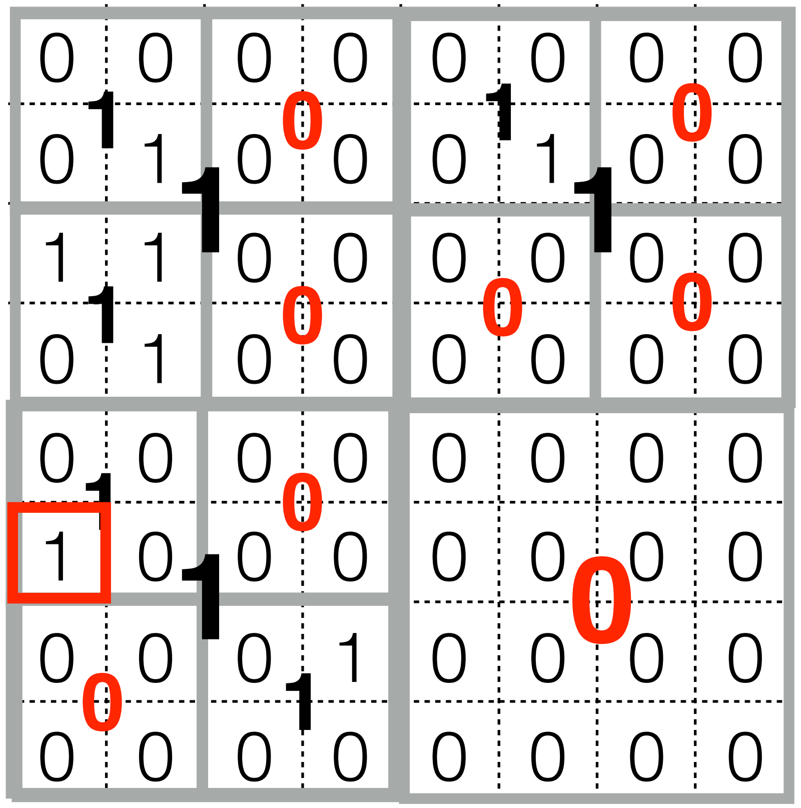
T **L**

Credits to A. Gómez-Brandón

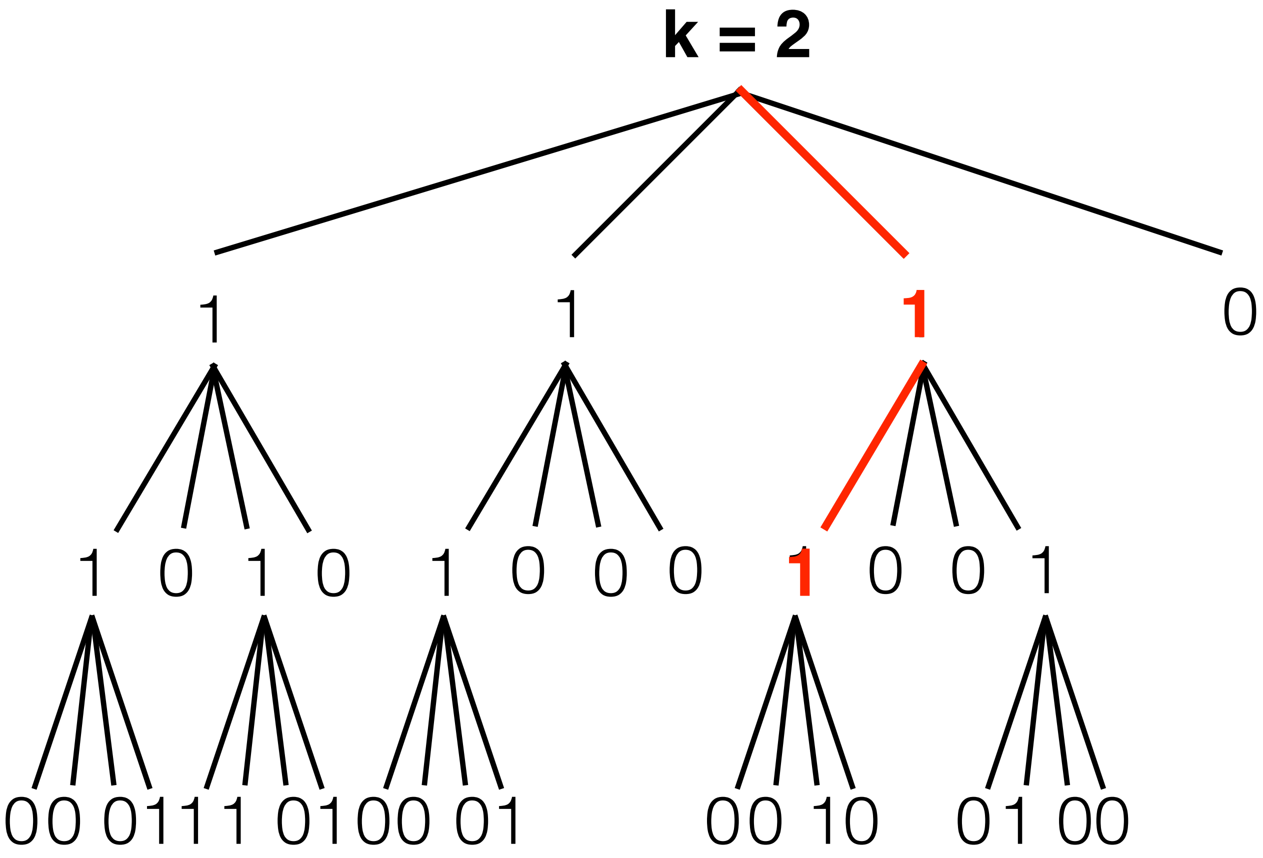
k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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



Access

11**10**10101000**1**00**1** 0001110100010**0**0100100

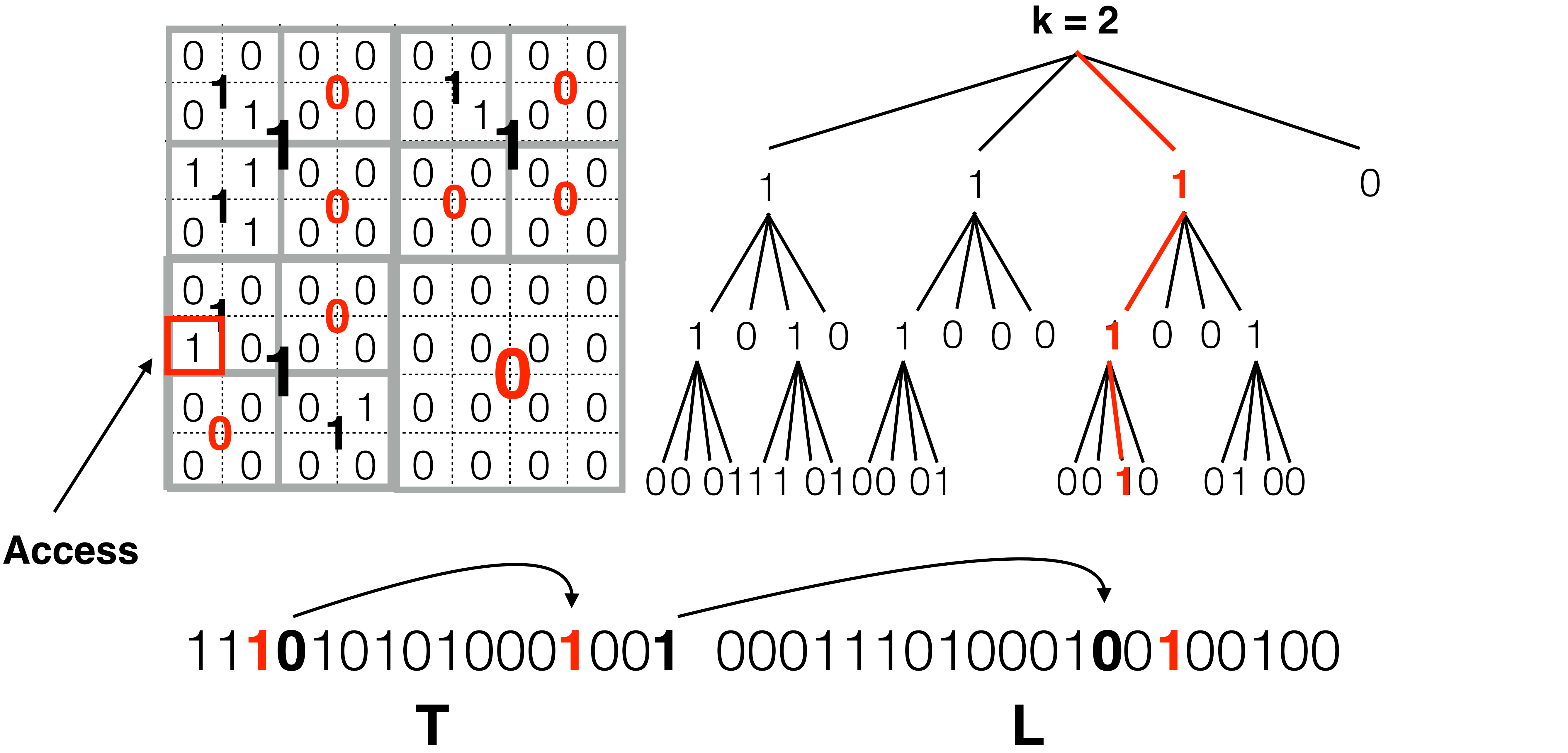
T **L**

Credits to A. Gómez-Brandón

k²-trees

k²-ary tree representation of the adjacency matrix.

k²-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009



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

Access

1110101010001001 00011101000100100100

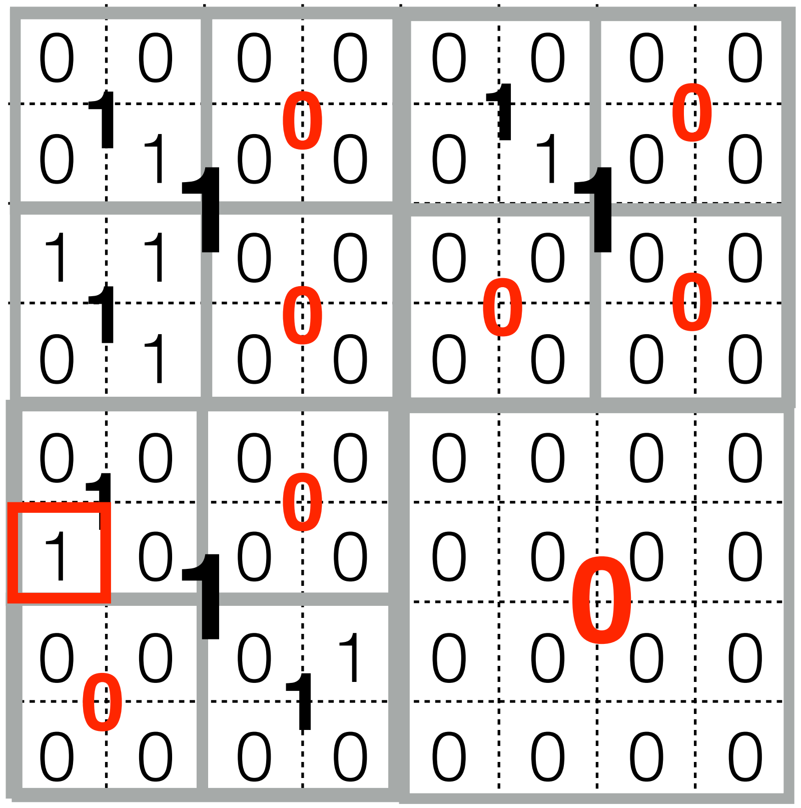
T **L**

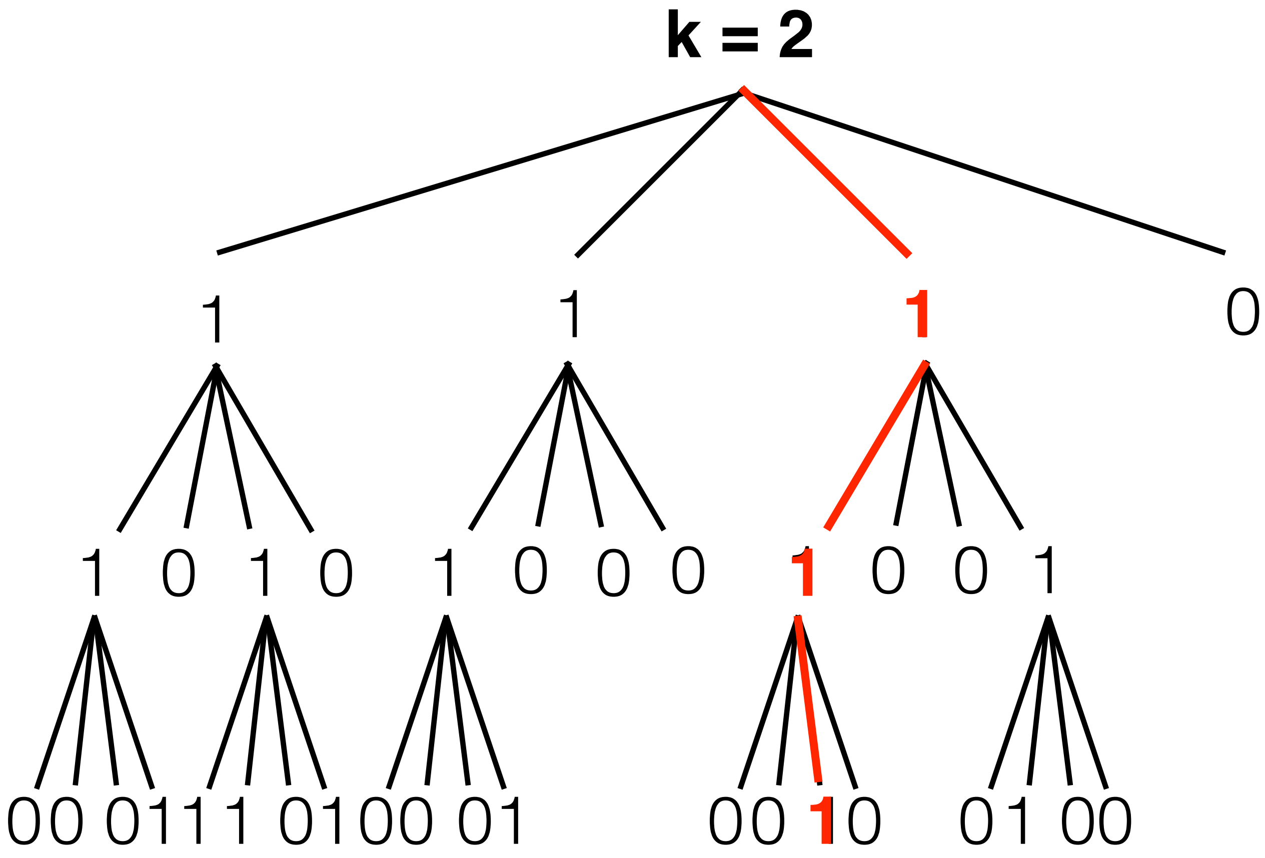
Credits to A. Gómez-Brandón

k^2-trees

k^2-ary tree representation of the adjacency matrix.

k^2-trees for Compact Web Graph Representation, Brisaboa-Ladra-Navarro, SPIRE 2009





Access

11**10**10101000**1**00**1** 000111010001**0**0**1**00100

T

L

rank$_1$ x k^2 on **T** in O(1)

Credits to A. Gómez-Brandón

k²-trees

n pages m links

$$h = \lceil \log_{k^2} n^2 \rceil$$

k²-trees

n pages m links

$h = \lceil \log_{k^2} n^2 \rceil$ So the total space is $mk^2 \lceil \log_{k^2} n^2 \rceil$ bits

k2-trees

n pages m links

$h = \lceil \log_{k^2} n^2 \rceil$ So the total space is $mk^2 \lceil \log_{k^2} n^2 \rceil$ bits

$$k^2 \sum_{i=0}^{\lfloor \log_{k^2} m \rfloor - 1} k^{2i} + mk^2 (\lceil \log_{k^2} n \rceil - \lfloor \log_{k^2} m \rfloor)$$

$$= k^2 \frac{m-1}{k^2-1} + mk^2 \log_{k^2} \frac{n^2}{m}$$

$$= mk^2 \left(\log_{k^2} \frac{n^2}{m} + O(1) \right) \text{ bits}$$

k²-trees

n pages m links

$h = \lceil \log_{k^2} n^2 \rceil$ So the total space is $mk^2 \lceil \log_{k^2} n^2 \rceil$ bits

$$k^2 \sum_{i=0}^{\lfloor \log_{k^2} m \rfloor - 1} k^{2i} + mk^2 (\lceil \log_{k^2} n \rceil - \lfloor \log_{k^2} m \rfloor)$$

$$= k^2 \frac{m-1}{k^2-1} + mk^2 \log_{k^2} \frac{n^2}{m}$$

$$= mk^2 \left(\log_{k^2} \frac{n^2}{m} + O(1) \right) \text{ bits}$$

Information theoretic
lower bound

$$\log \binom{n^2}{m} \approx m \log \frac{n^2}{m} + O(m)$$

k²-trees

n pages m links

$h = \lceil \log_{k^2} n^2 \rceil$ So the total space is $mk^2 \lceil \log_{k^2} n^2 \rceil$ bits

$$k^2 \sum_{i=0}^{\lfloor \log_{k^2} m \rfloor - 1} k^{2i} + mk^2(\lceil \log_{k^2} n \rceil - \lfloor \log_{k^2} m \rfloor)$$

$$= k^2 \frac{m-1}{k^2-1} + mk^2 \log_{k^2} \frac{n^2}{m}$$

$$= mk^2 \left(\log_{k^2} \frac{n^2}{m} + O(1) \right) \text{ bits}$$

$k = 2$

Information theoretic
lower bound

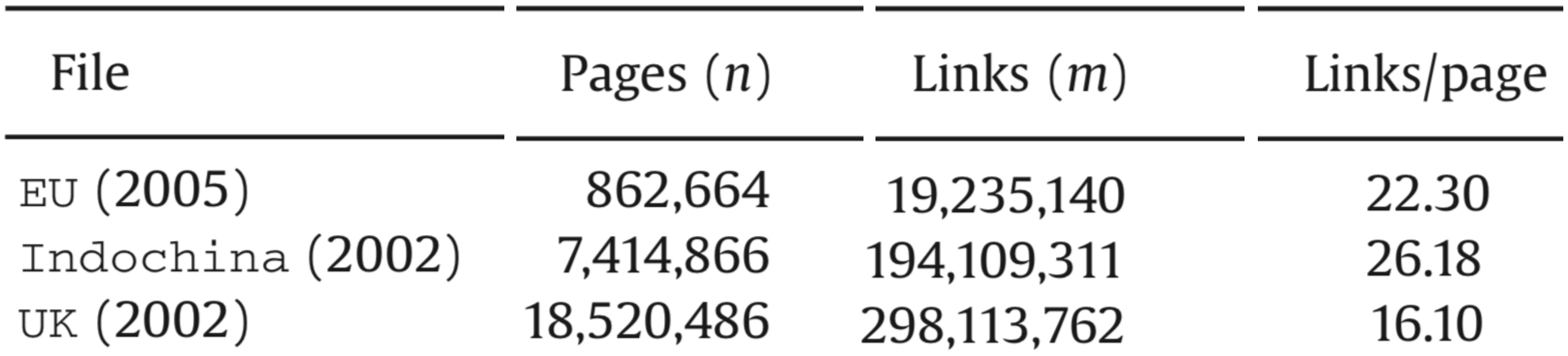
↔

$2m \log \frac{n^2}{m} + O(m)$

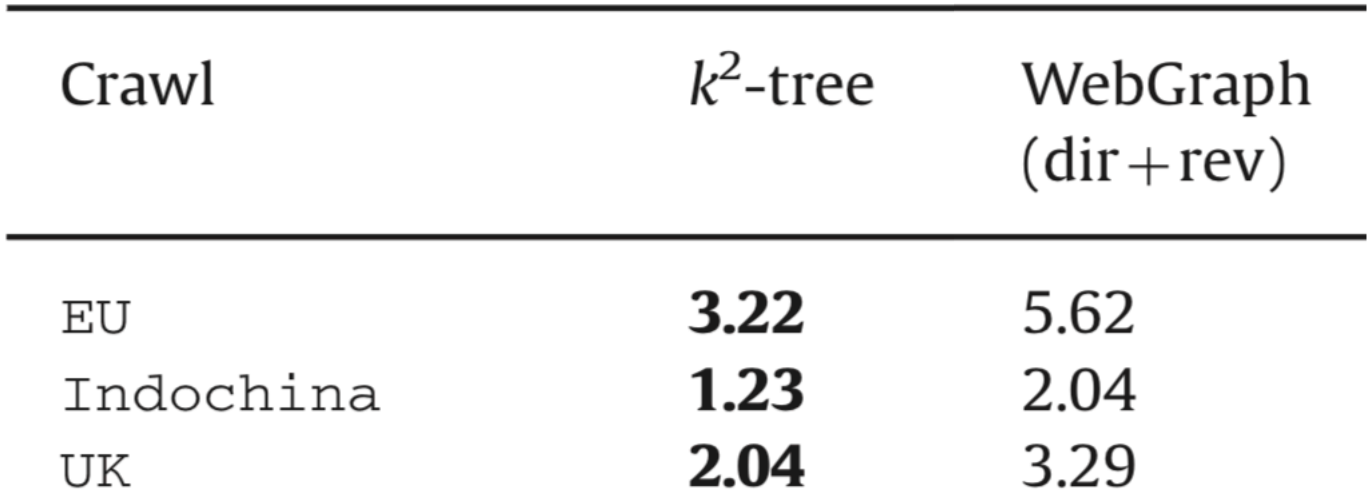
$\log \binom{n^2}{m} \approx m \log \frac{n^2}{m} + O(m)$

2X away

k^2-trees



File	Pages (n)	Links (m)	Links/page
EU (2005)	862,664	19,235,140	22.30
Indochina (2002)	7,414,866	194,109,311	26.18
UK (2002)	18,520,486	298,113,762	16.10



Crawl	k^2-tree	WebGraph (dir+rev)
EU	**3.22**	5.62
Indochina	**1.23**	2.04
UK	**2.04**	3.29

Space results in bits x link.

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings,
Belazzougui *et al.*, DCC 2014

Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings,
Belazzougui *et al.*, DCC 2014

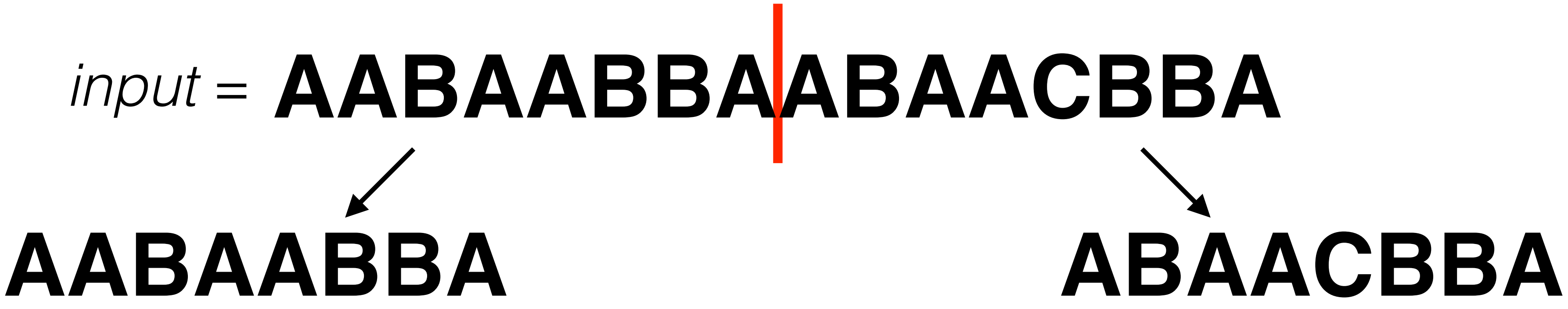
input = **AABAABBAABAACBBA**

Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014



input = **AABAABBAABAACBBA**

AABAABBA

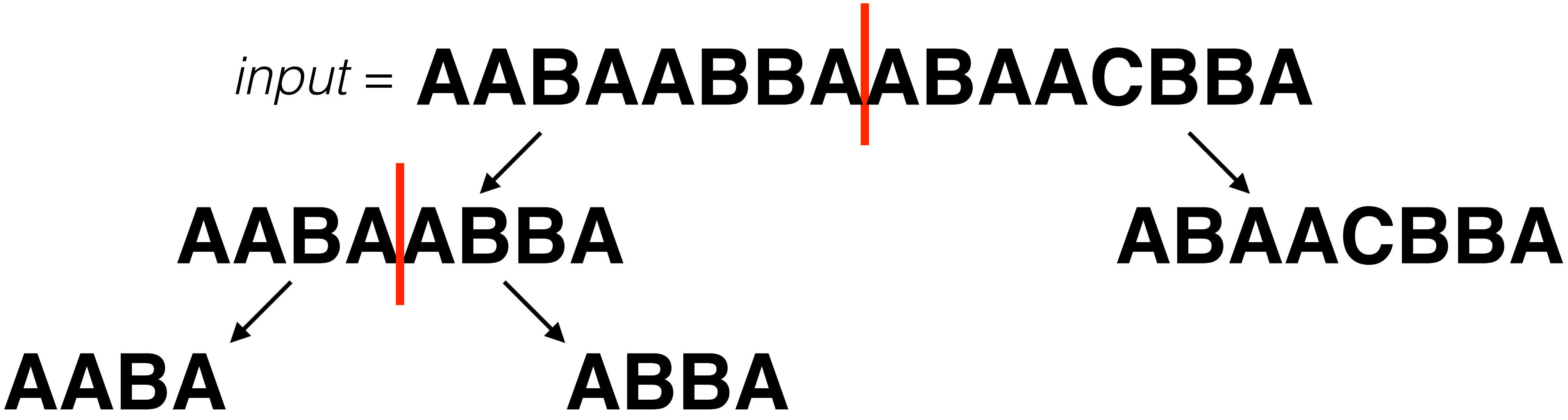
ABAACBBA

Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings,
Belazzougui *et al.*, DCC 2014



input = AABAABBAABAACBBA

AABAABBA

ABAACBBA

AABA

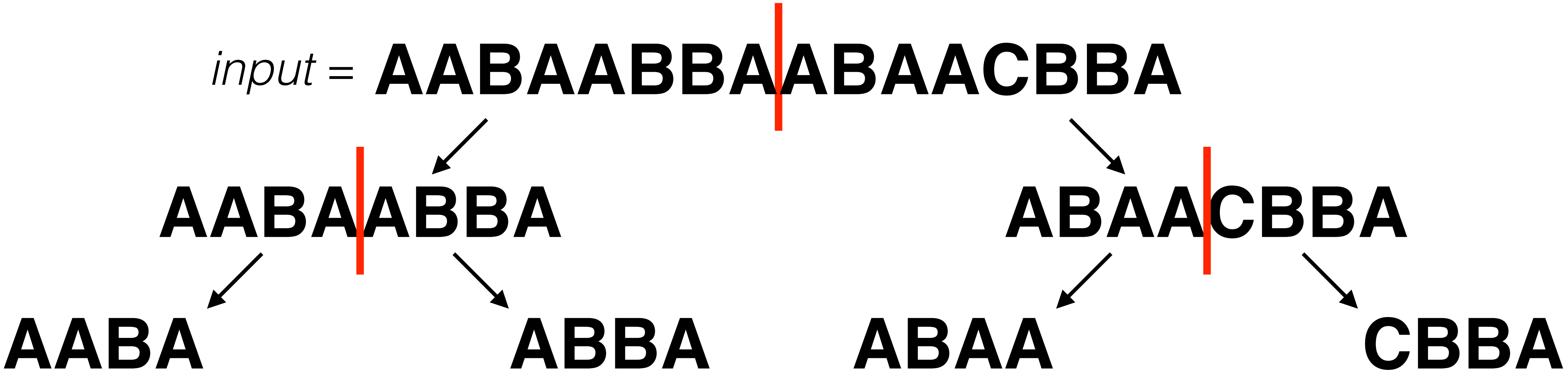
ABBA

Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queies on LZ-Bounded Encodings,
Belazzougui *et al.*, DCC 2014



input = AABAABBAABAACBBA

AABAABBA

ABAACBBA

AABA

ABBA

ABAA

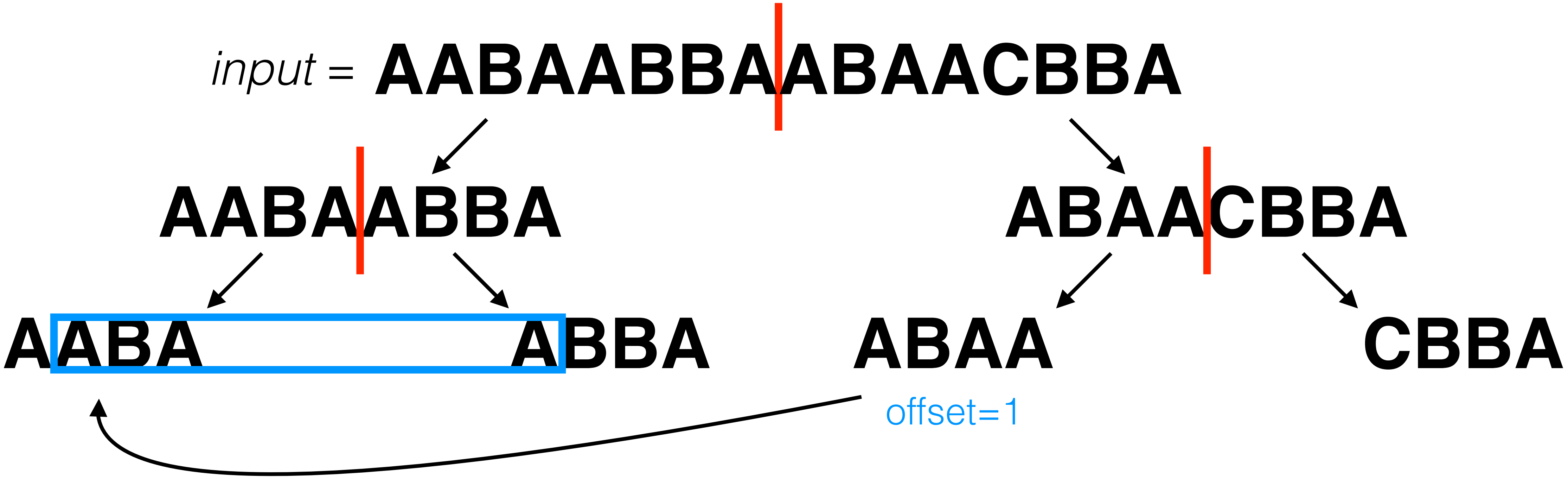
CBBA

Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

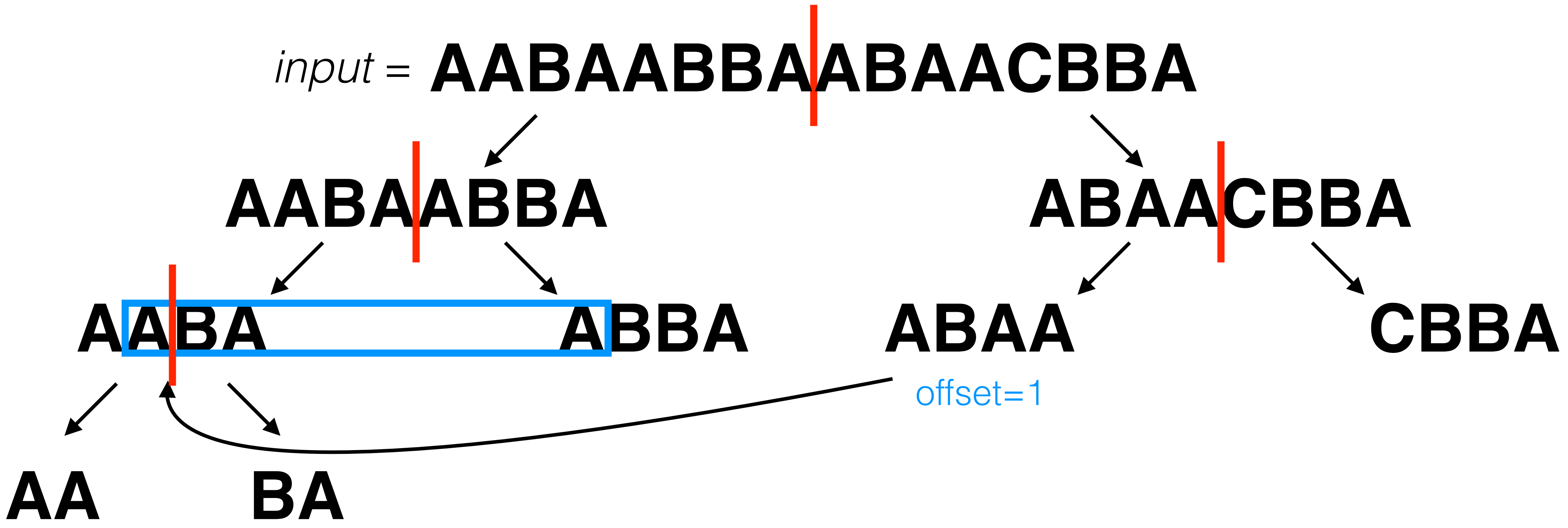


Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

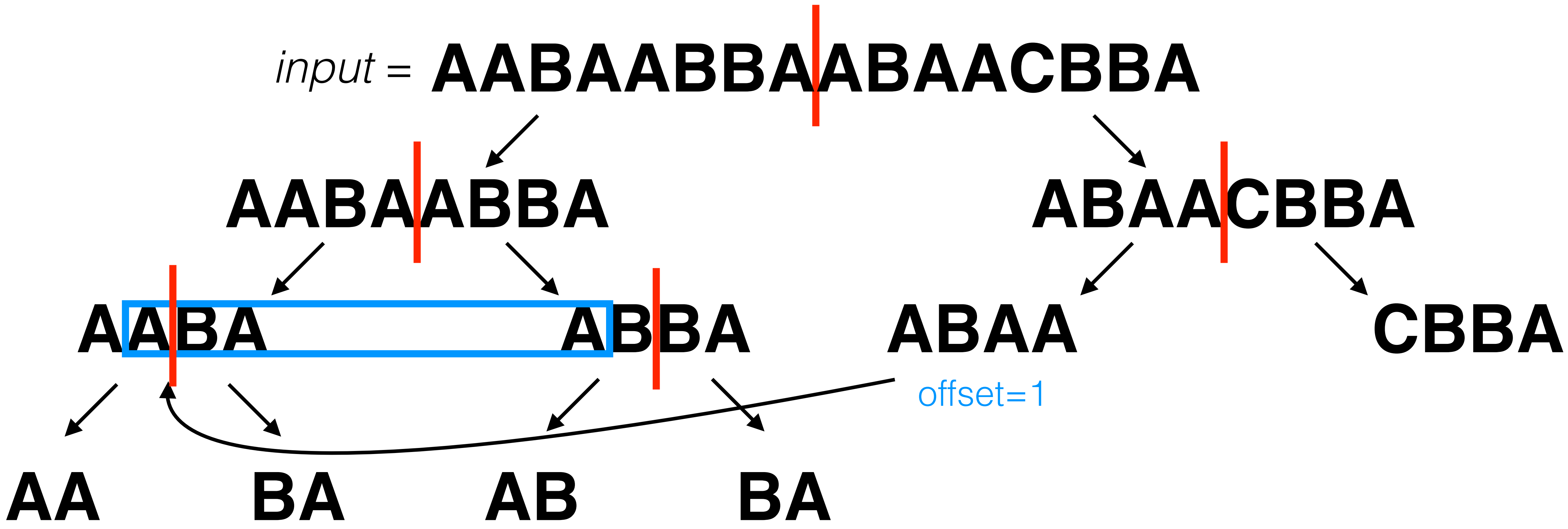


Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

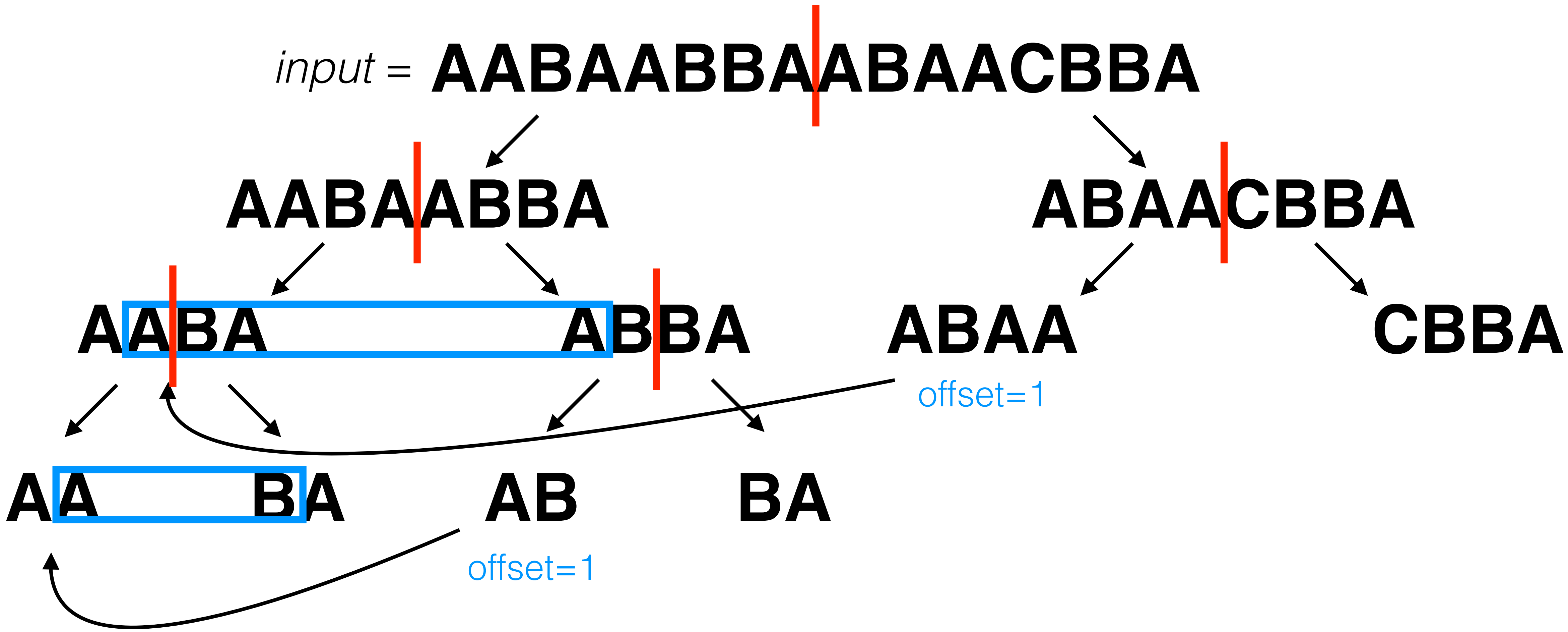


Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

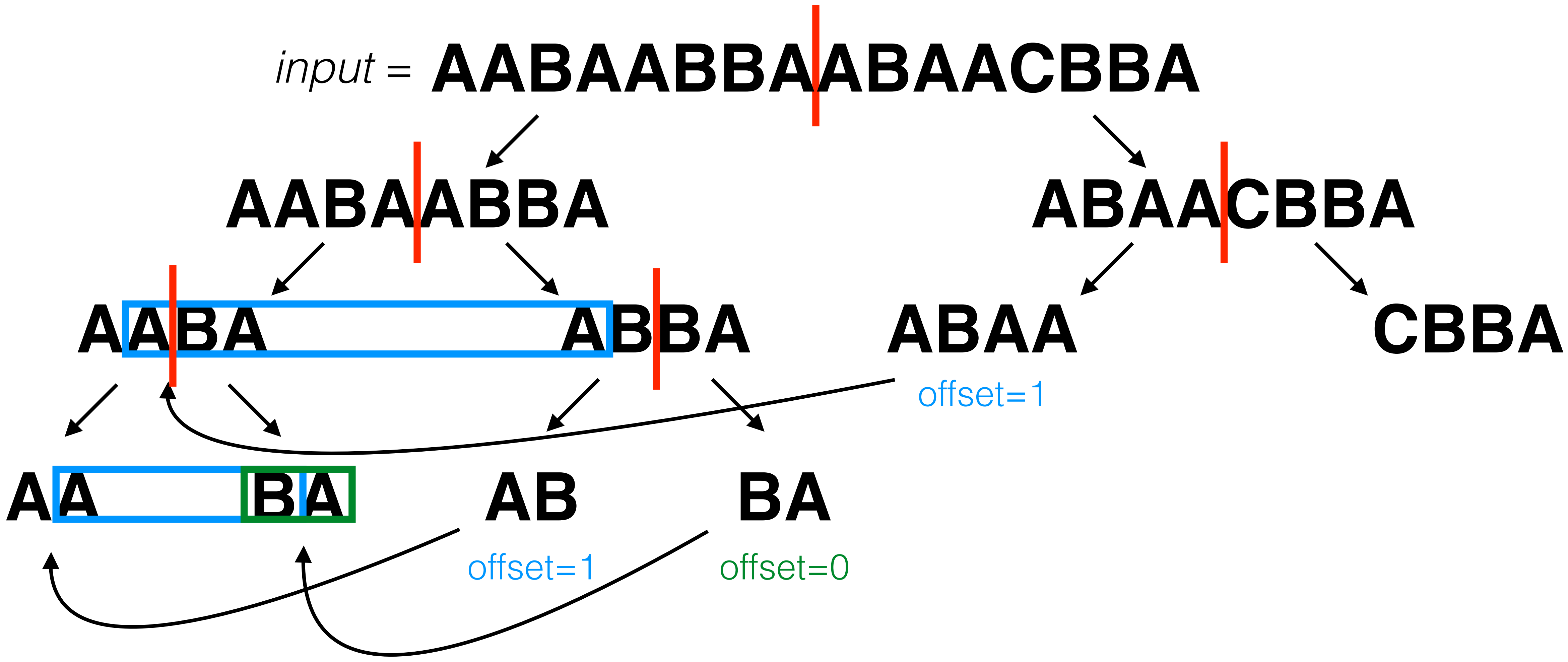


Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

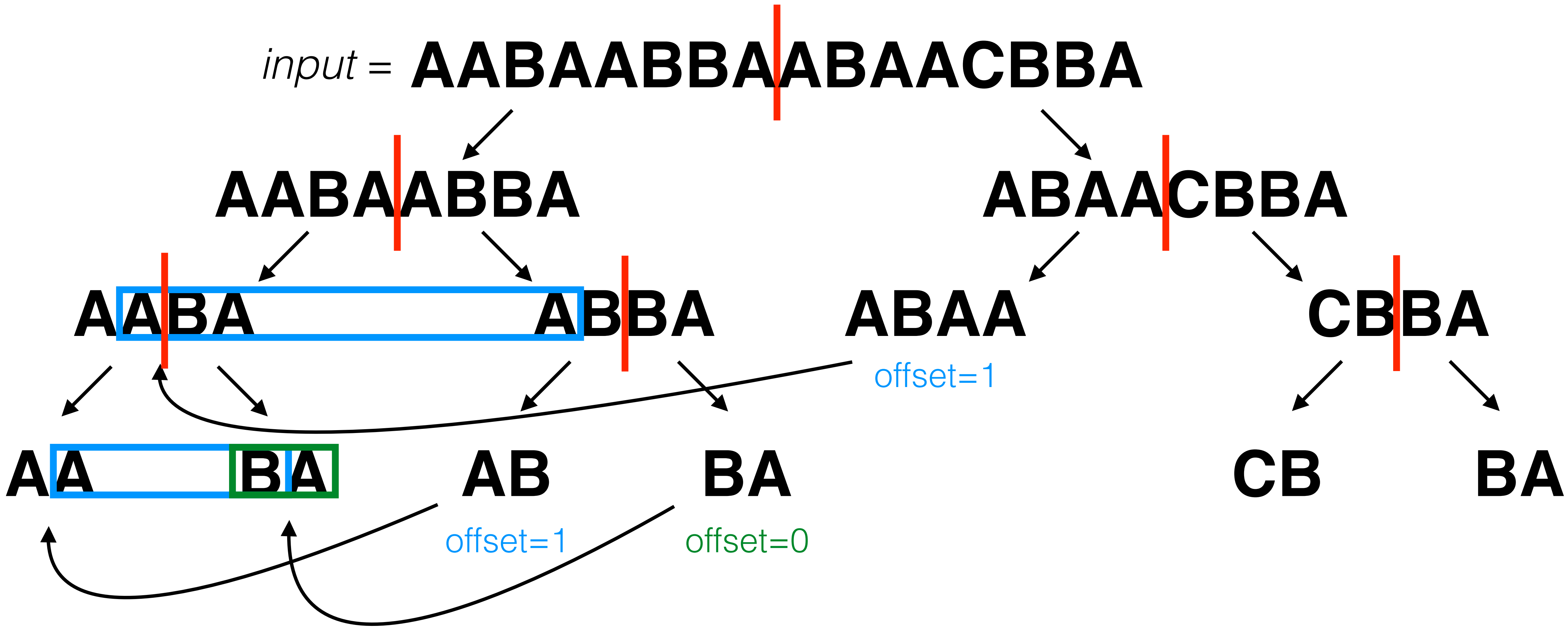


Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

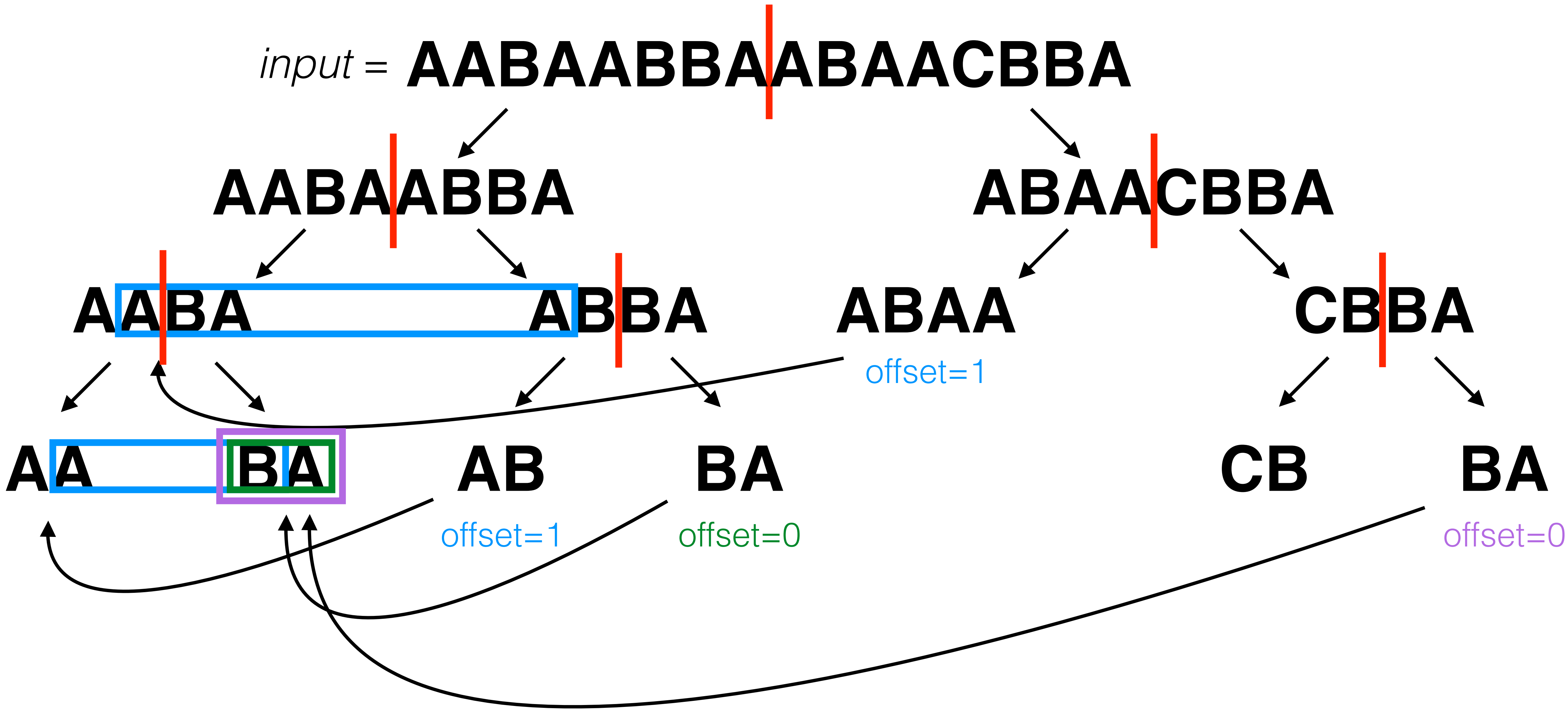


Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014



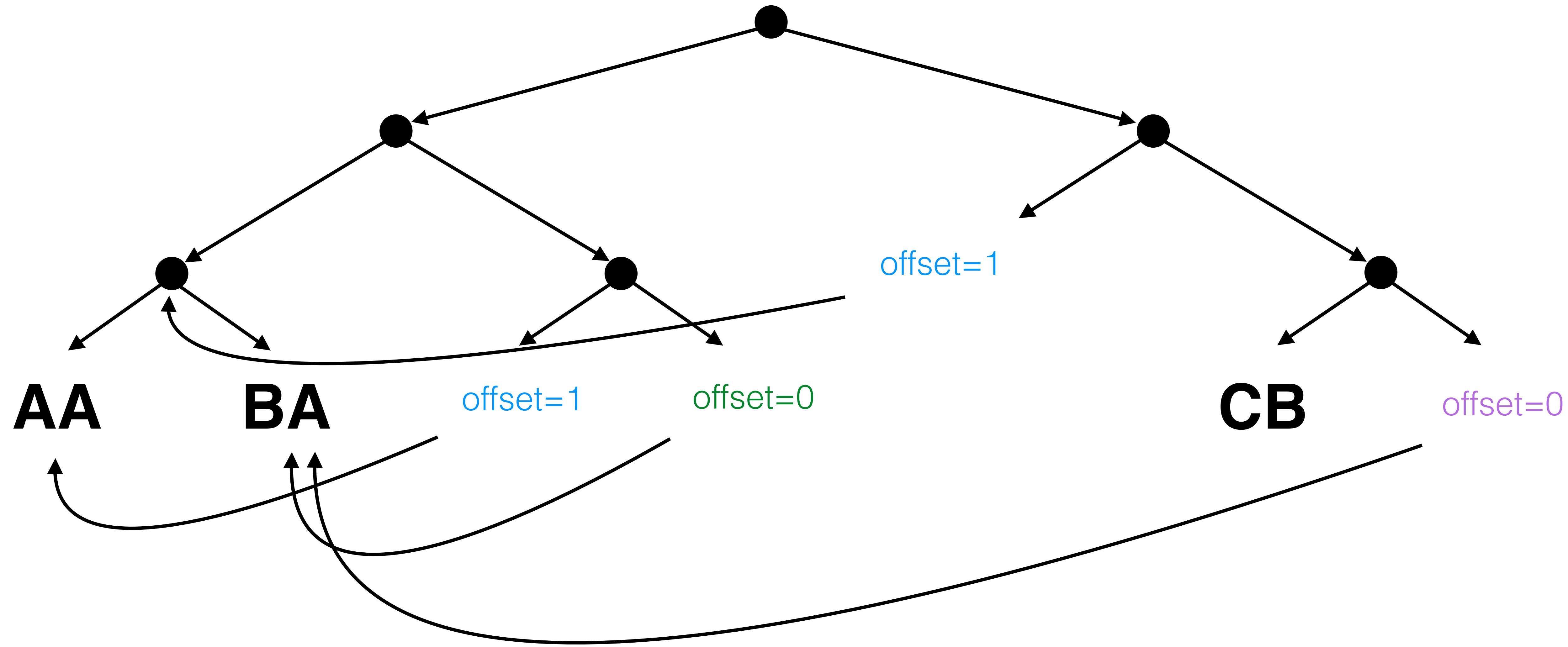
Credits to A. Gómez-Brandón

Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**



Credits to A. Gómez-Brandón

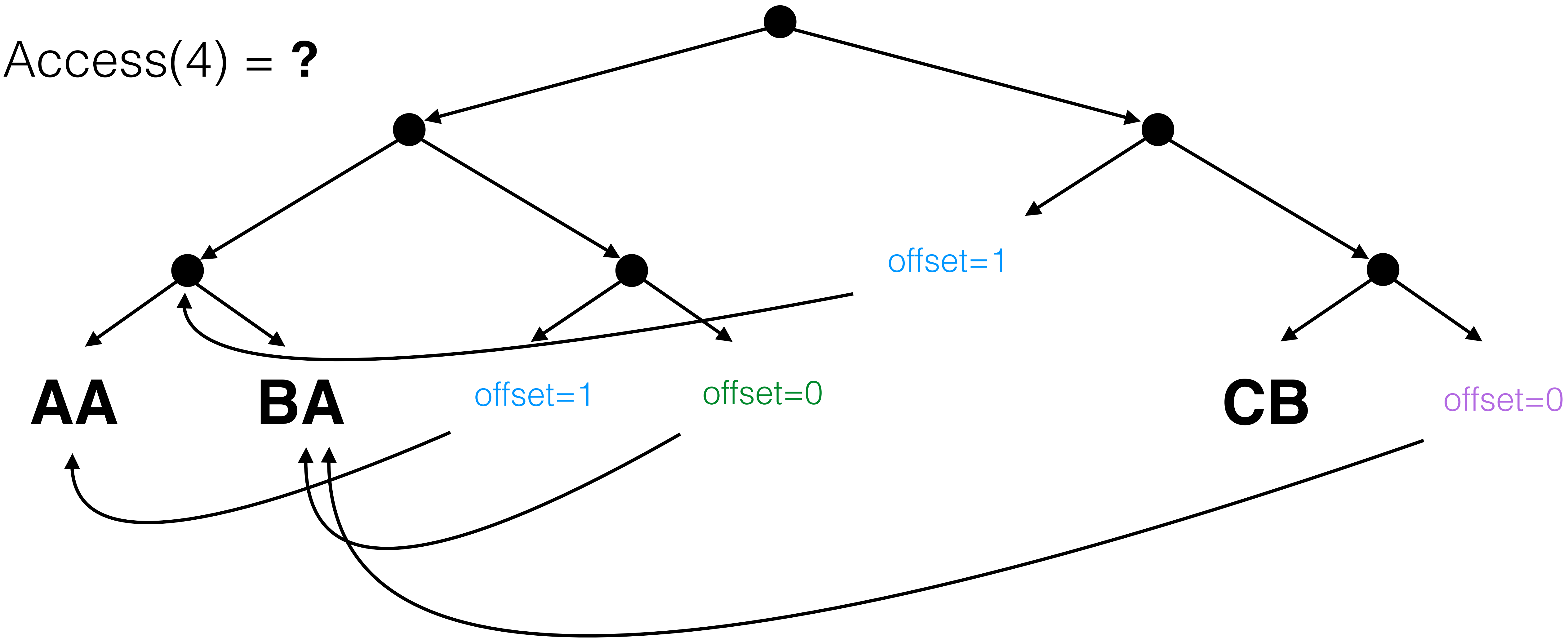
Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**

Access(4) = **?**



Credits to A. Gómez-Brandón

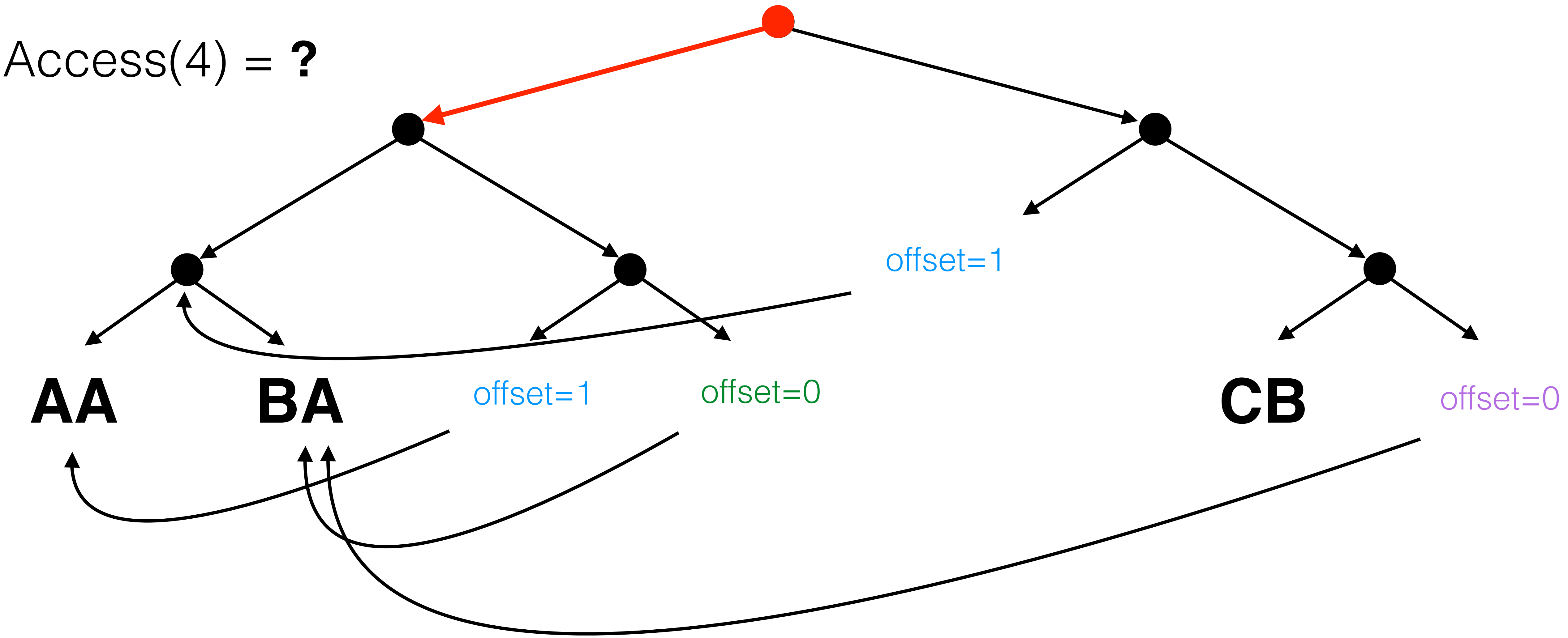
Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings,
Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**

Access(4) = **?**



Credits to A. Gómez-Brandón

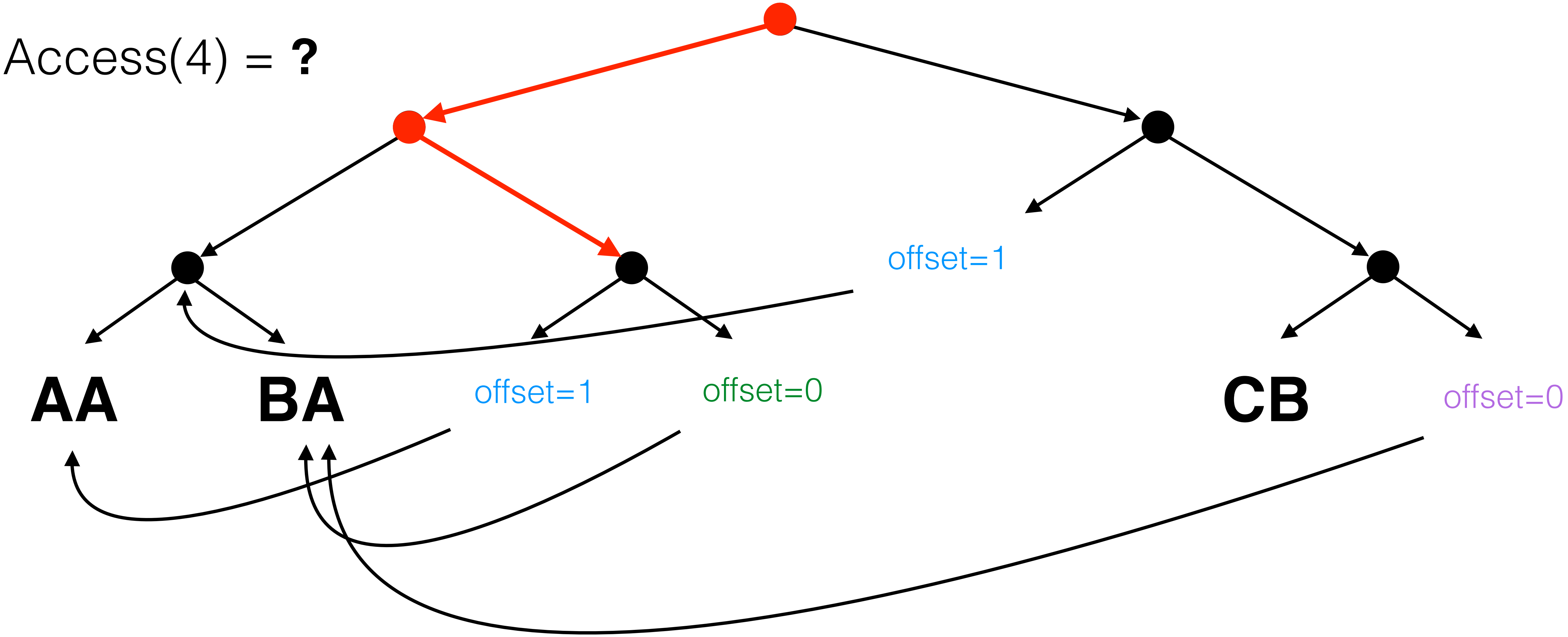
Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**

Access(4) = **?**



Credits to A. Gómez-Brandón

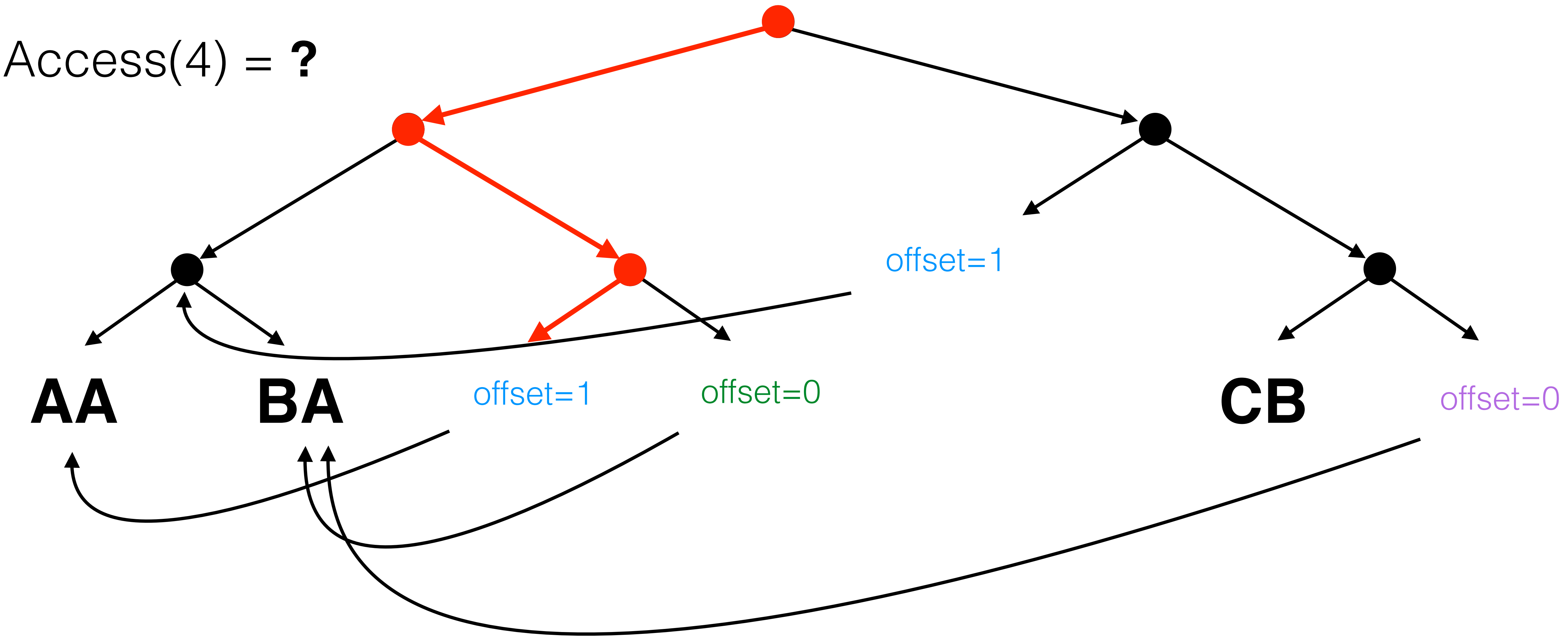
Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**

Access(4) = **?**



Credits to A. Gómez-Brandón

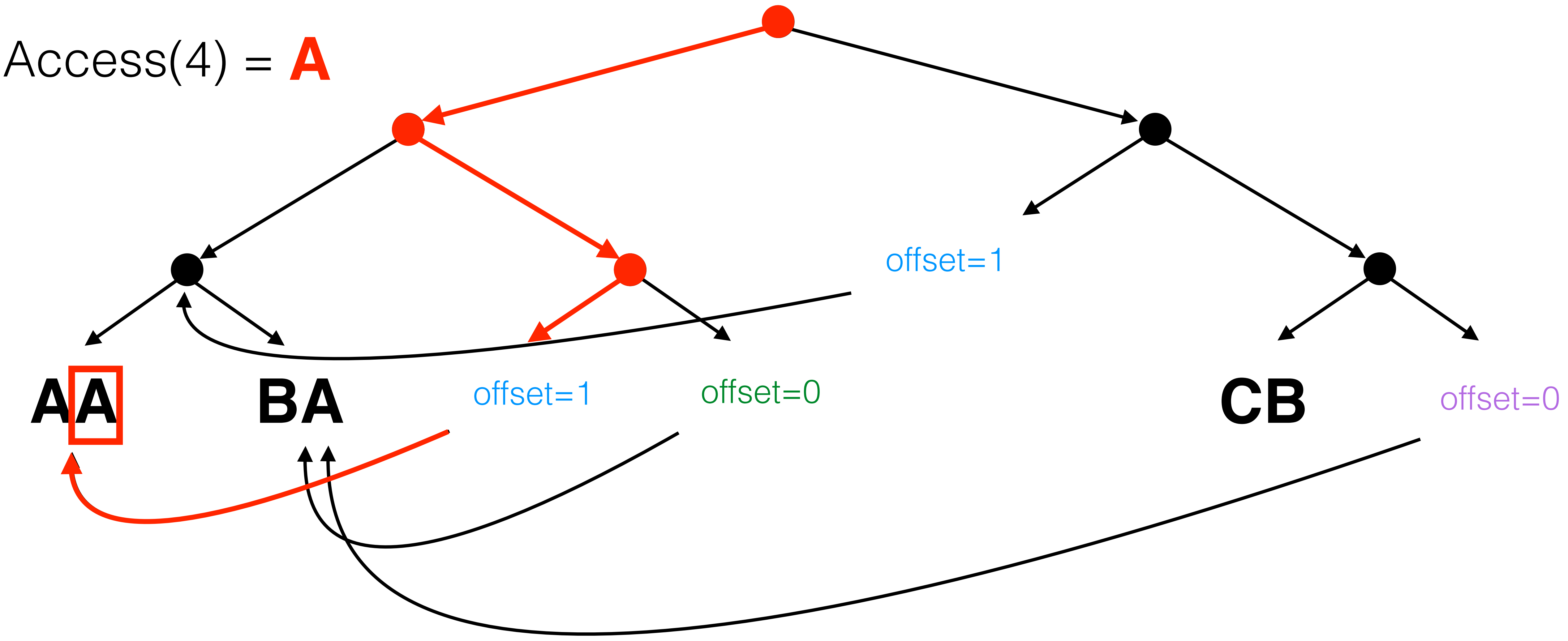
Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**

Access(4) = **A**



Credits to A. Gómez-Brandón

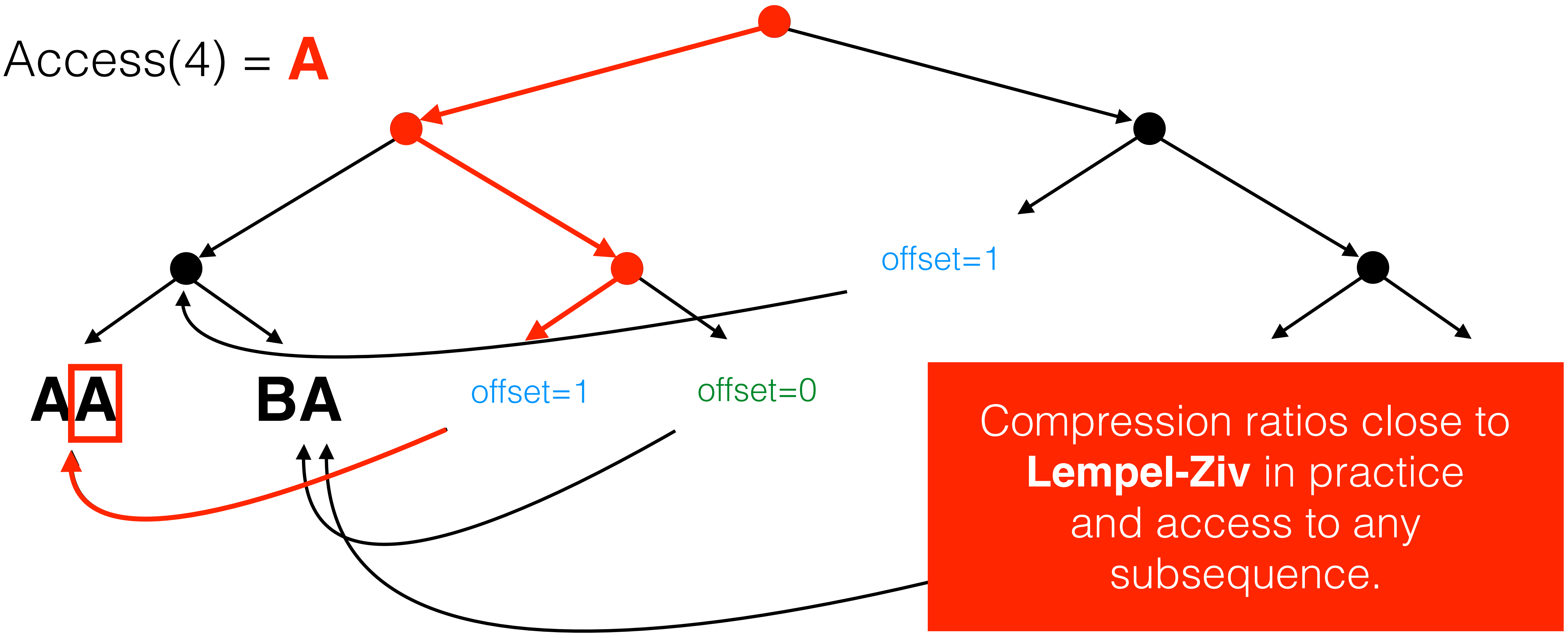
Block-trees

A block-tree divides a *string* into fixed-size blocks and those appearing earlier are represented with pointers.

Queries on LZ-Bounded Encodings, Belazzougui *et al.*, DCC 2014

input = **AABAABBAABAACBBA**

Access(4) = **A**



Compression ratios close to **Lempel-Ziv** in practice and access to any subsequence.

Credits to A. Gómez-Brandón

Block-trees

n size of the string

σ alphabet size

Stop recursion when

$$B\lceil \log \sigma \rceil > \lceil \log n \rceil$$

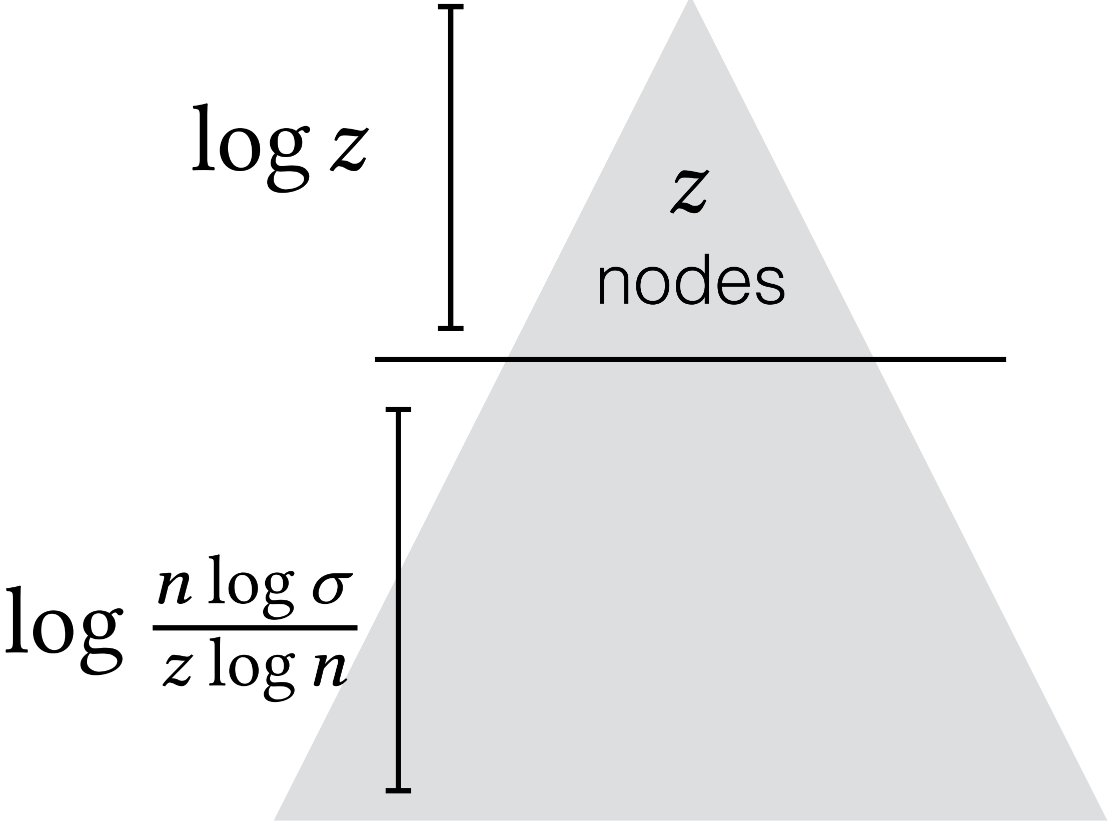
$$B = O(\log n / \log \sigma)$$

$$h = \log \frac{n \log \sigma}{\log n}$$

At level i

t_i nodes take

$O(t_i \log n)$ bits



$$z \log n \; + \; \log \frac{n \log \sigma}{z \log n} O(z \log n)$$

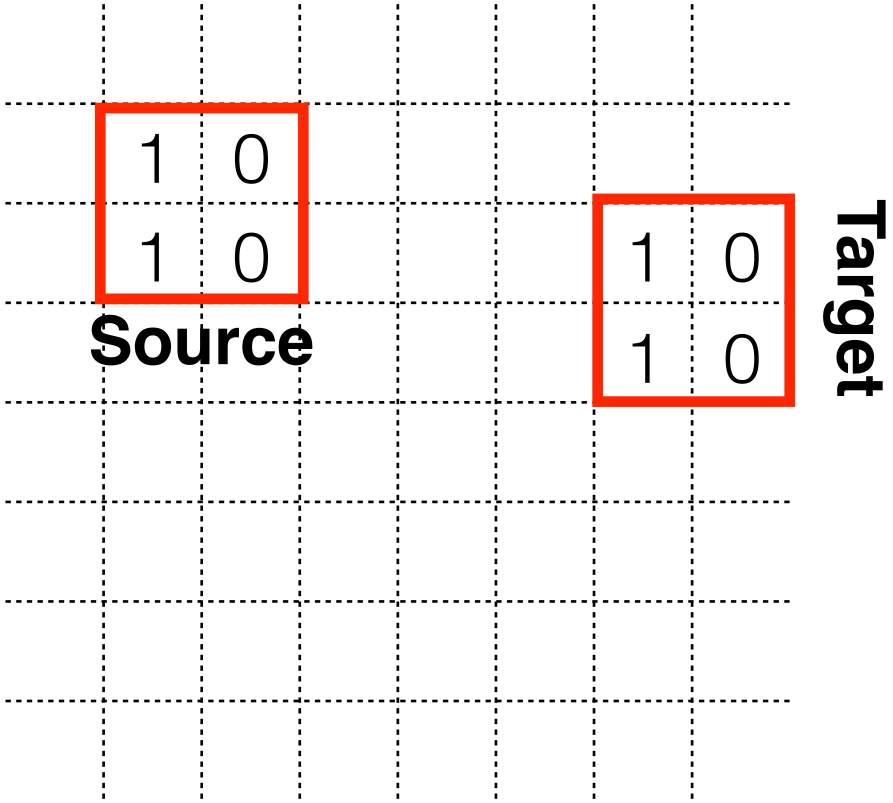
$$= O(z \log n \log \frac{n \log \sigma}{z \log n}) \text{ bits}$$

2D Block-trees

A *hybrid* between the k^2-tree to exploit the clustering of the 0s and the block tree to exploit the repetitiveness of the adjacency matrix.

Two-Dimensional Block Trees, Brisaboa, Gagie, Gómez-Brandón, Navarro, DCC 2018

A **source** block may overlap up to **4** adjacent blocks.



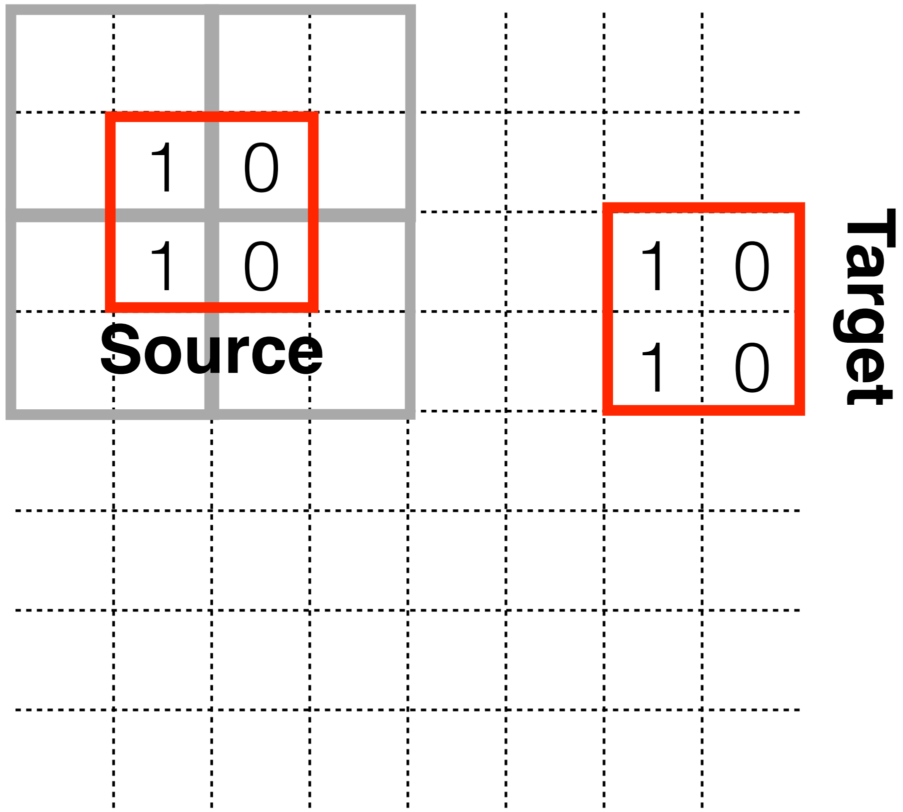
Credits to A. Gómez-Brandón

2D Block-trees

A *hybrid* between the k^2-tree to exploit the clustering of the 0s and the block tree to exploit the repetitiveness of the adjacency matrix.

Two-Dimensional Block Trees, Brisaboa, Gagie, Gómez-Brandón, Navarro, DCC 2018

A **source** block may overlap up to **4** adjacent blocks.



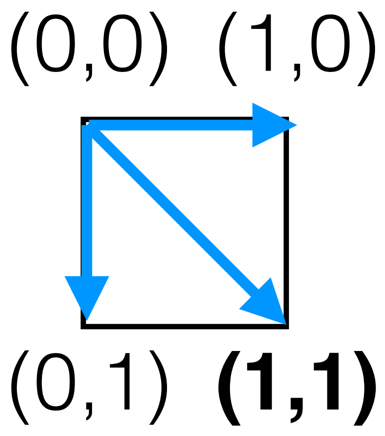
Credits to A. Gómez-Brandón

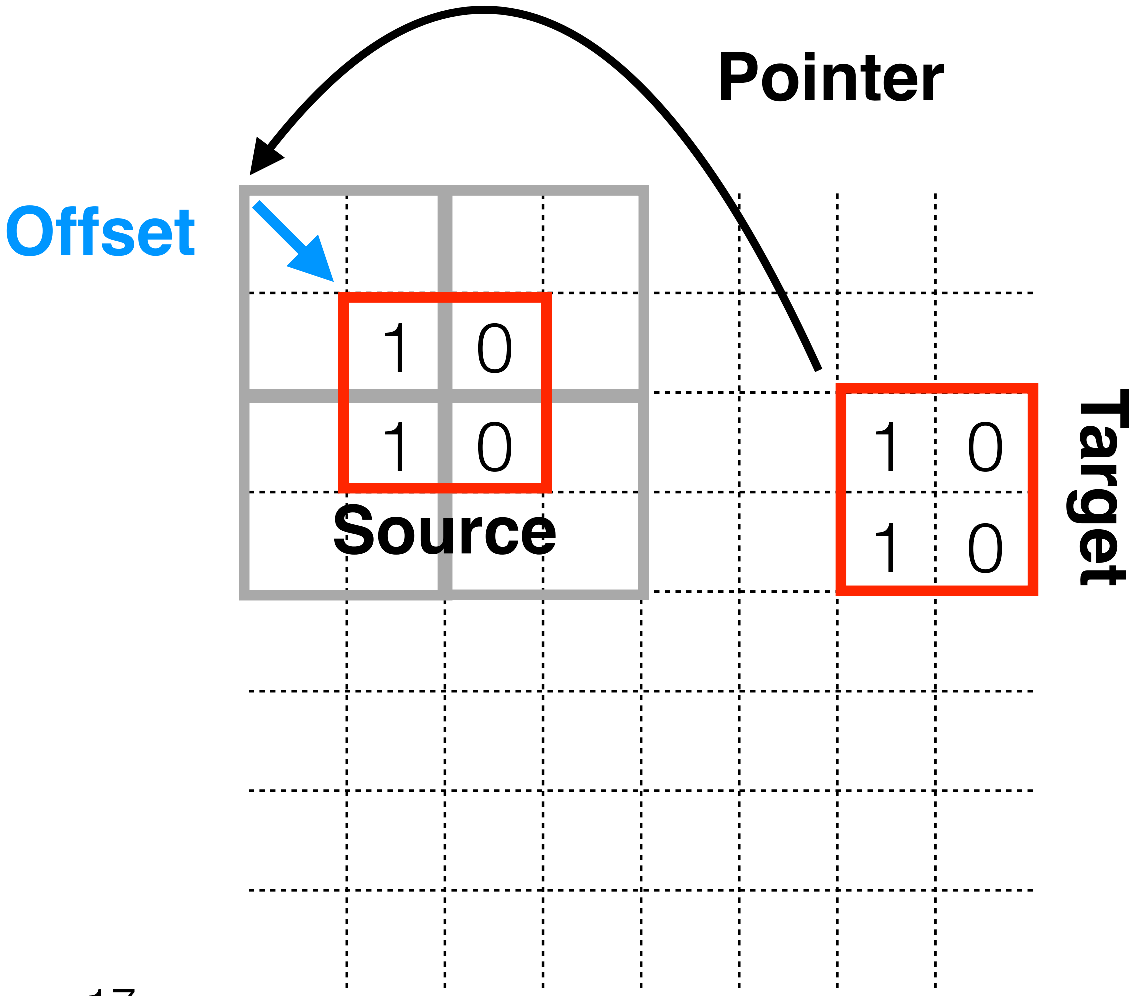
2D Block-trees

A *hybrid* between the k^2-tree to exploit the clustering of the 0s and the block tree to exploit the repetitiveness of the adjacency matrix.

Two-Dimensional Block Trees, Brisaboa, Gagie, Gómez-Brandón, Navarro, DCC 2018

A **source** block may overlap up to **4** adjacent blocks.



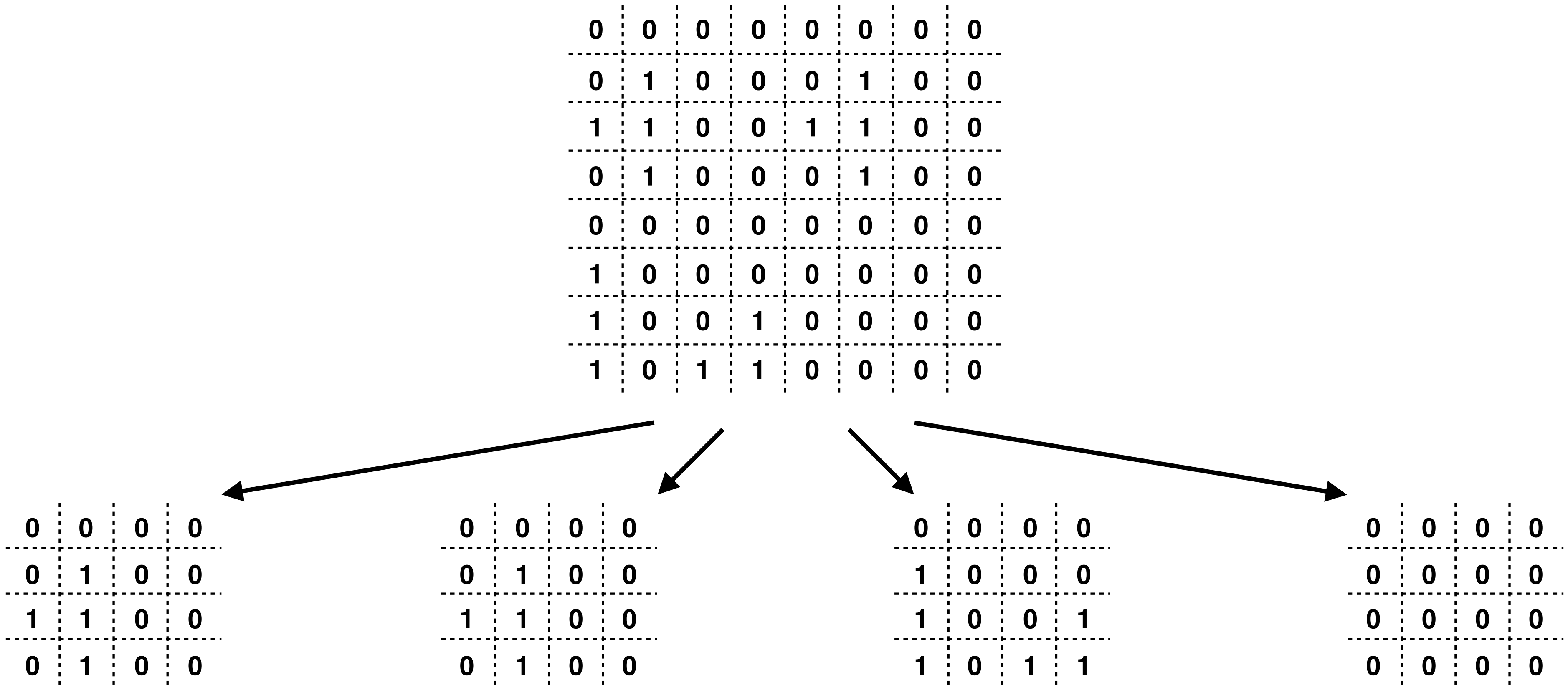


Credits to A. Gómez-Brandón

2D Block-trees

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

2D Block-trees



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

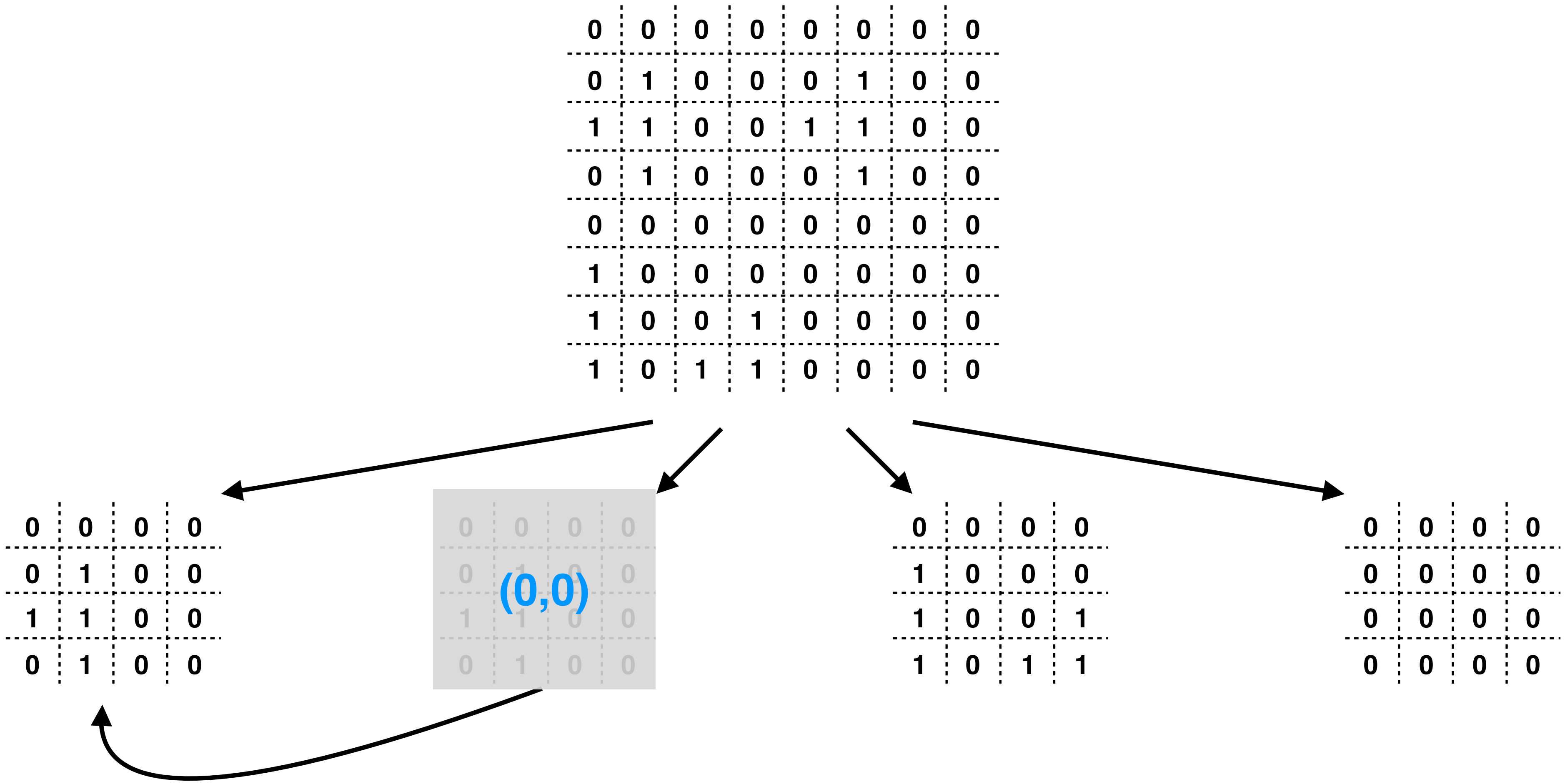
0	0	0	0
0	1	0	0
1	1	0	0
0	1	0	0

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

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

2D Block-trees



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

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

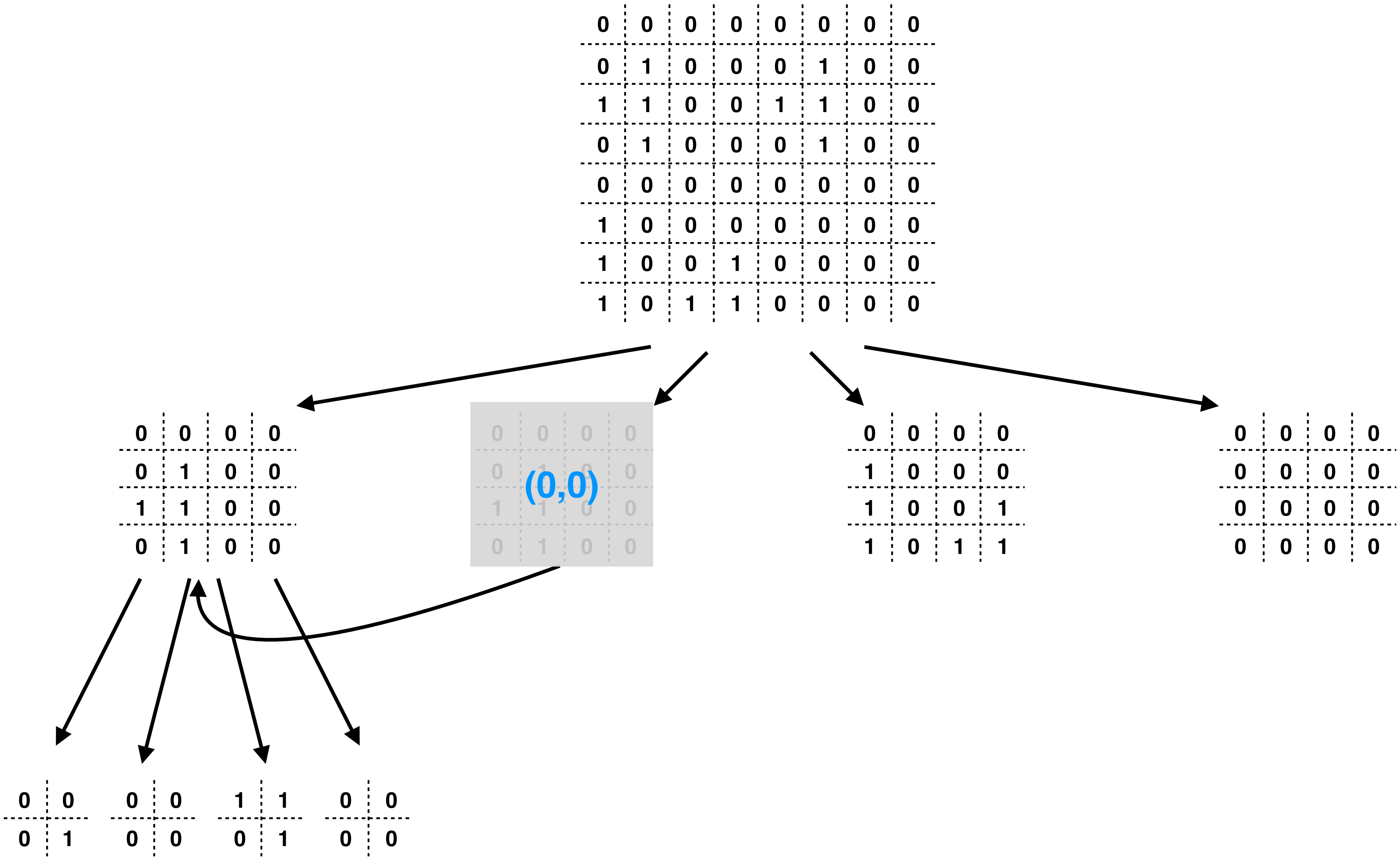
0	0	0	0
0	1	0	0
1	1	0	0
0	1	0	0

(0,0)

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

2D Block-trees



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

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

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

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

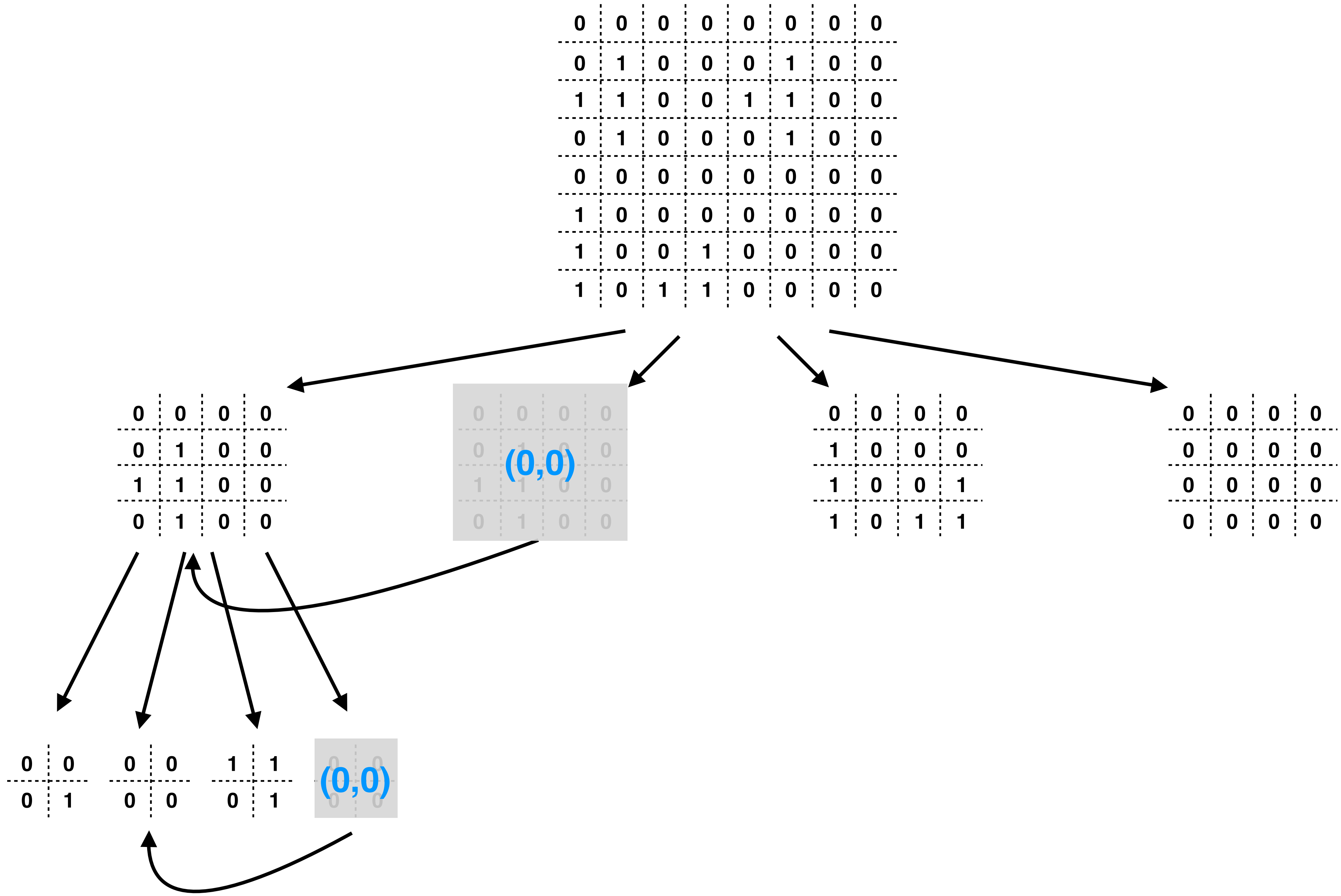
0	0
0	1

0	0
0	0

1	1
0	1

0	0
0	0

2D Block-trees



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

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

(0,0)

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

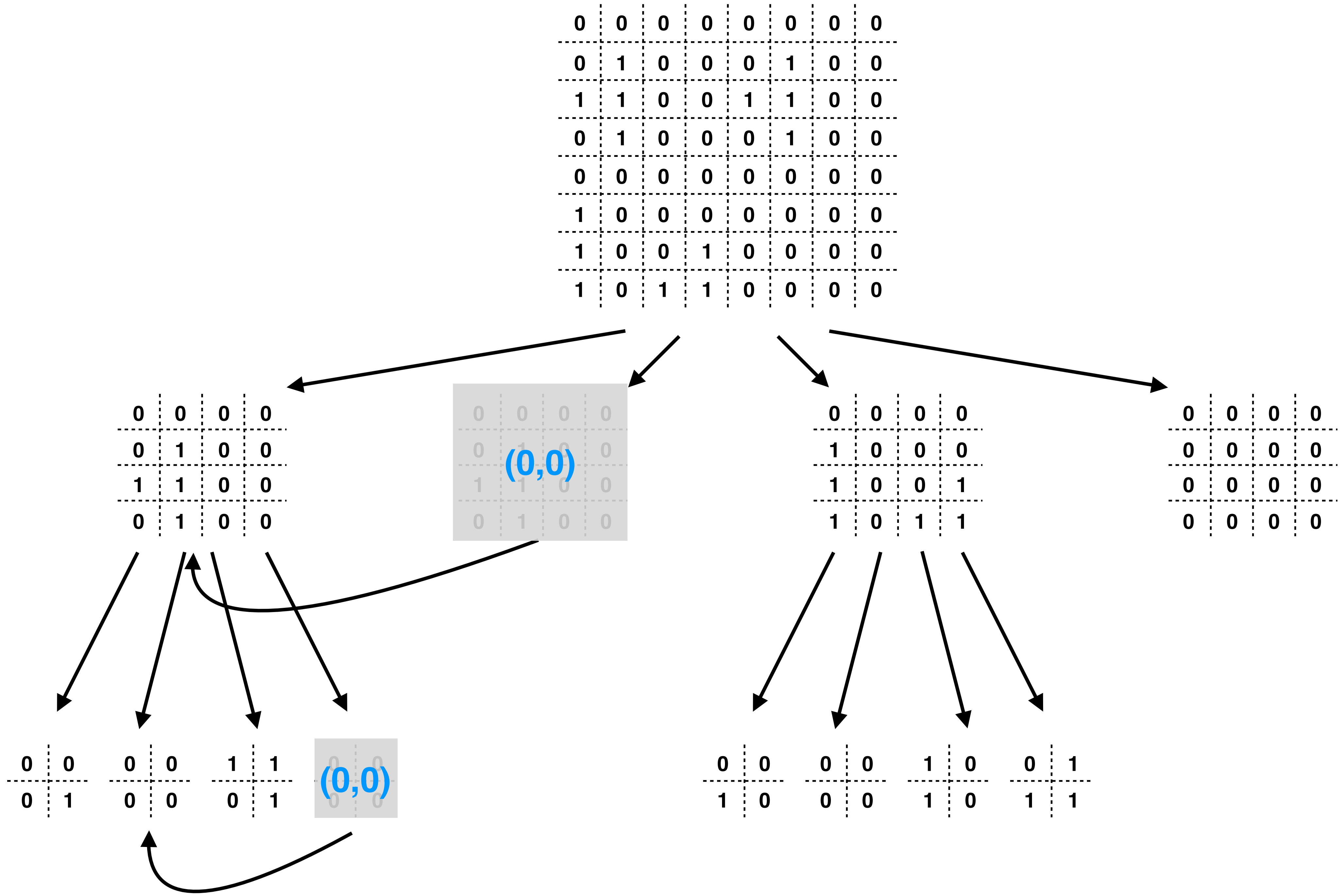
0	0
0	1

0	0
0	0

1	1
0	1

(0,0)

2D Block-trees



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

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

(0,0)

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

1	1
0	1

(0,0)

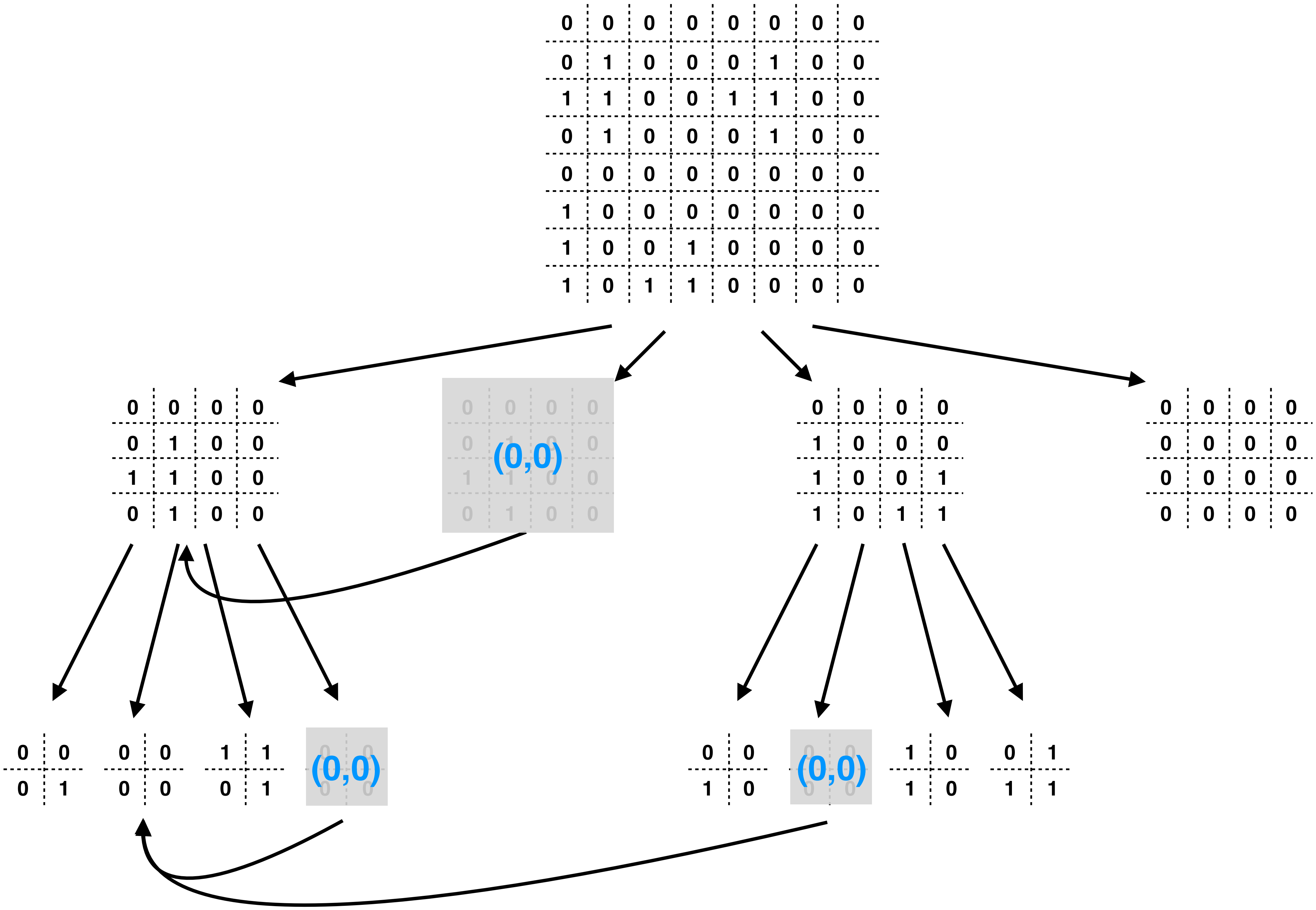
0	0
1	0

0	0
0	0

1	0
1	0

0	1
1	1

2D Block-trees



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

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

(0,0)

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

1	1
0	1

(0,0)

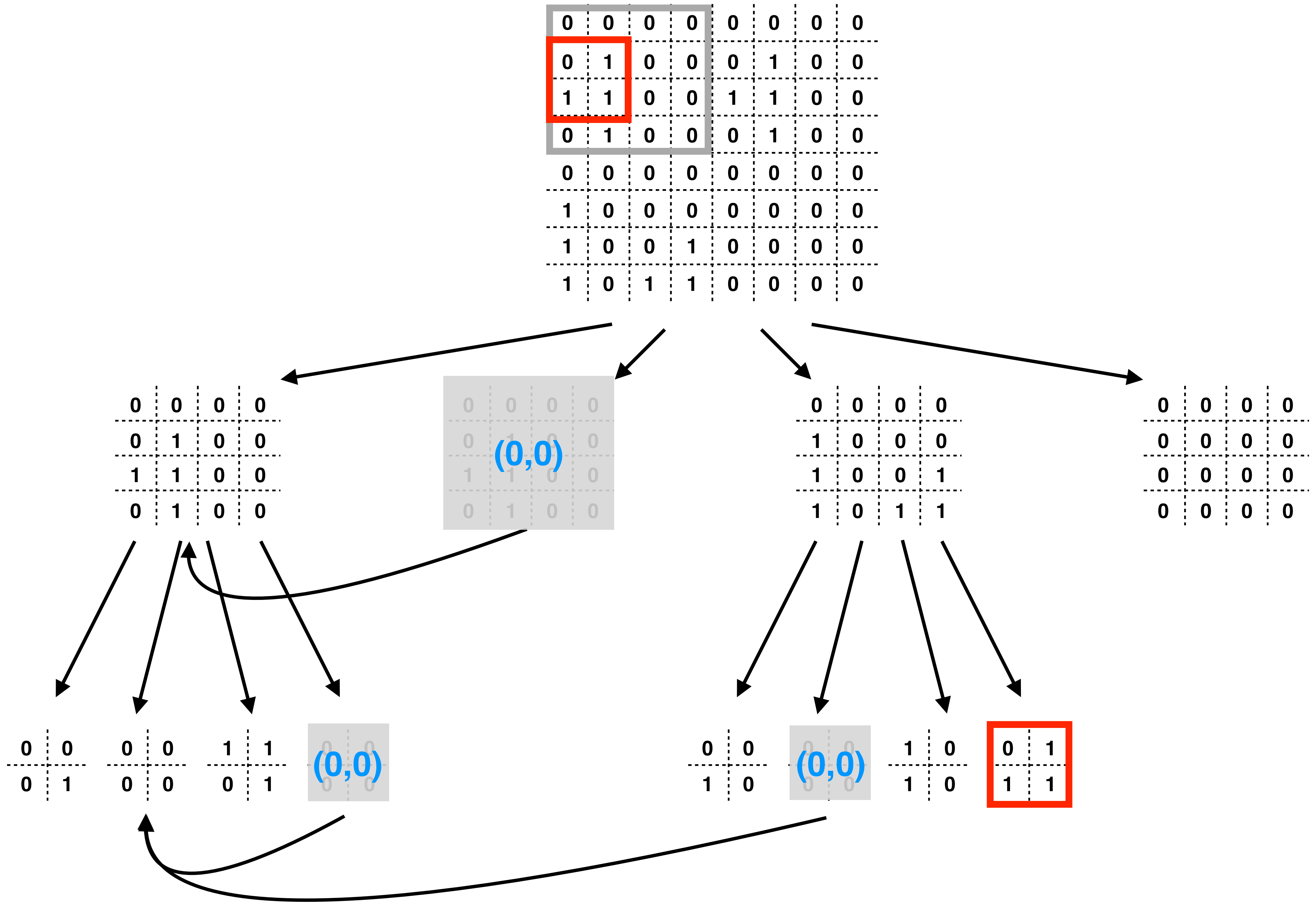
0	0
1	0

(0,0)

1	0
1	0

0	1
1	1

2D Block-trees



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

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

(0,0)

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

1	1
0	1

(0,0)

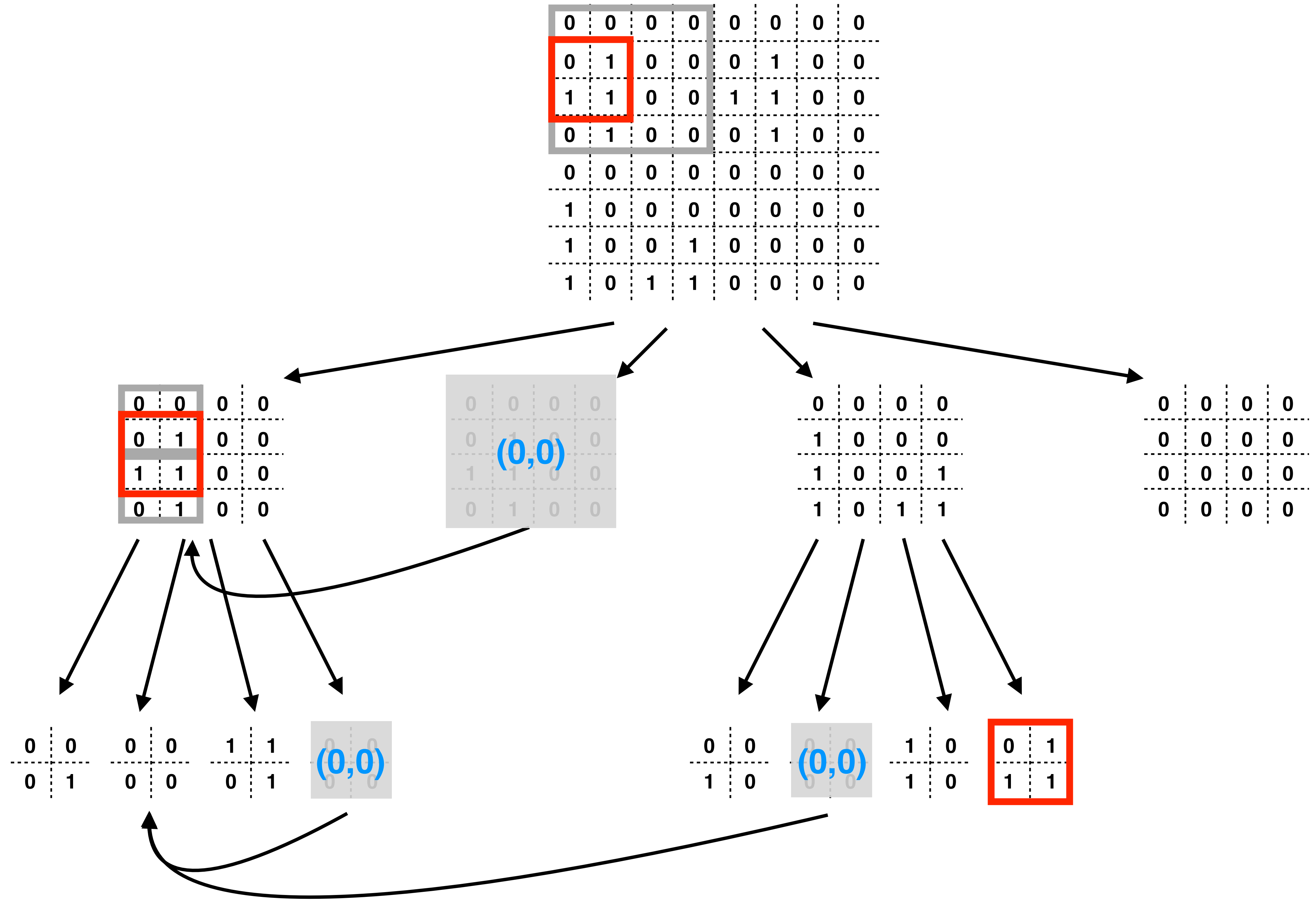
0	0
1	0

(0,0)

1	0
1	0

0	1
1	1

2D Block-trees



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

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

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

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

1	1
0	1

0	0
0	0

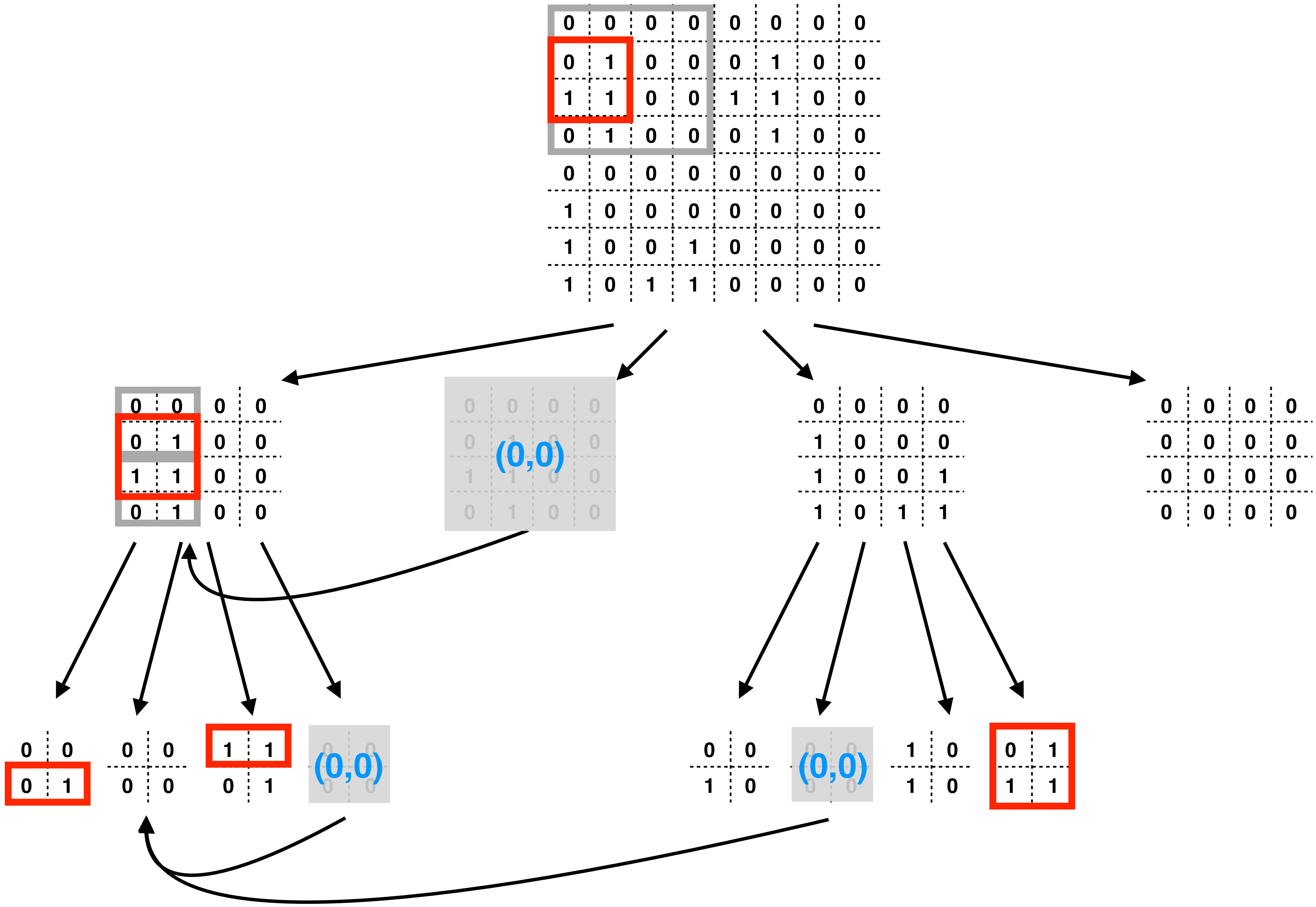
0	0
1	0

0	0
0	0

1	0
1	0

0	1
1	1

2D Block-trees



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

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

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

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

1	1
0	1

0	0
0	0

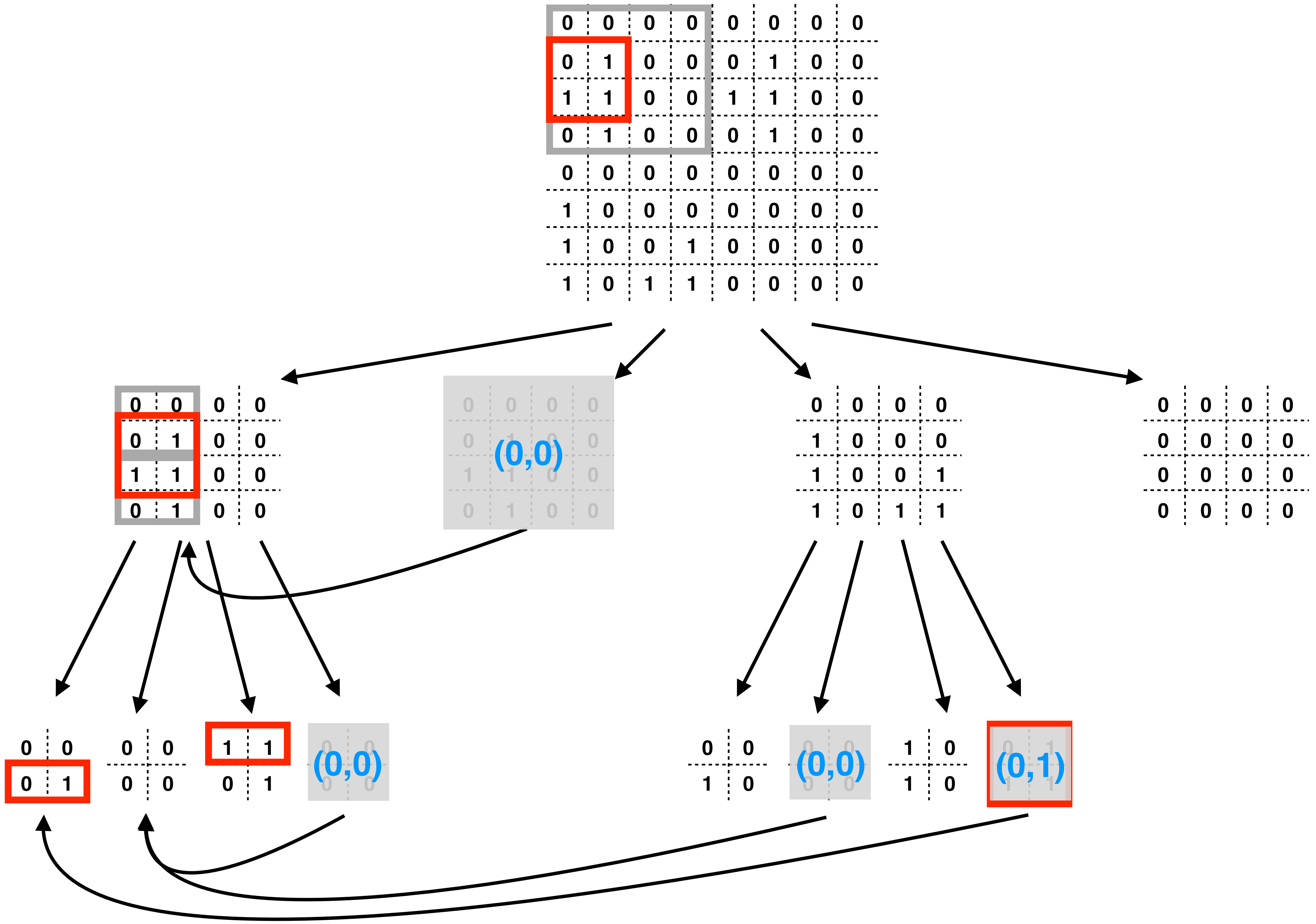
0	0
1	0

0	0
0	0

1	0
1	0

0	1
1	1

2D Block-trees



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

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

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

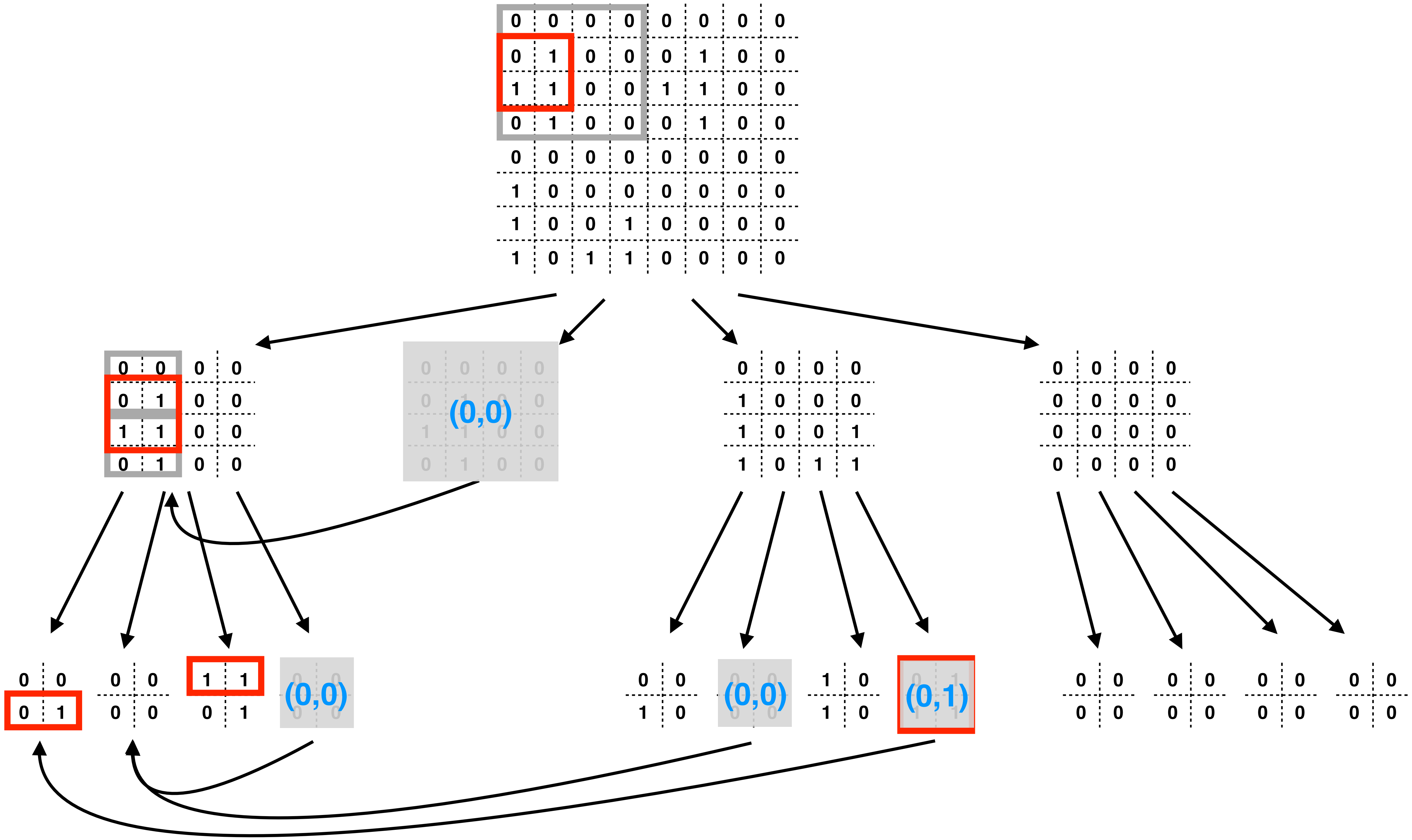
0	0
0	0

1	1
0	1

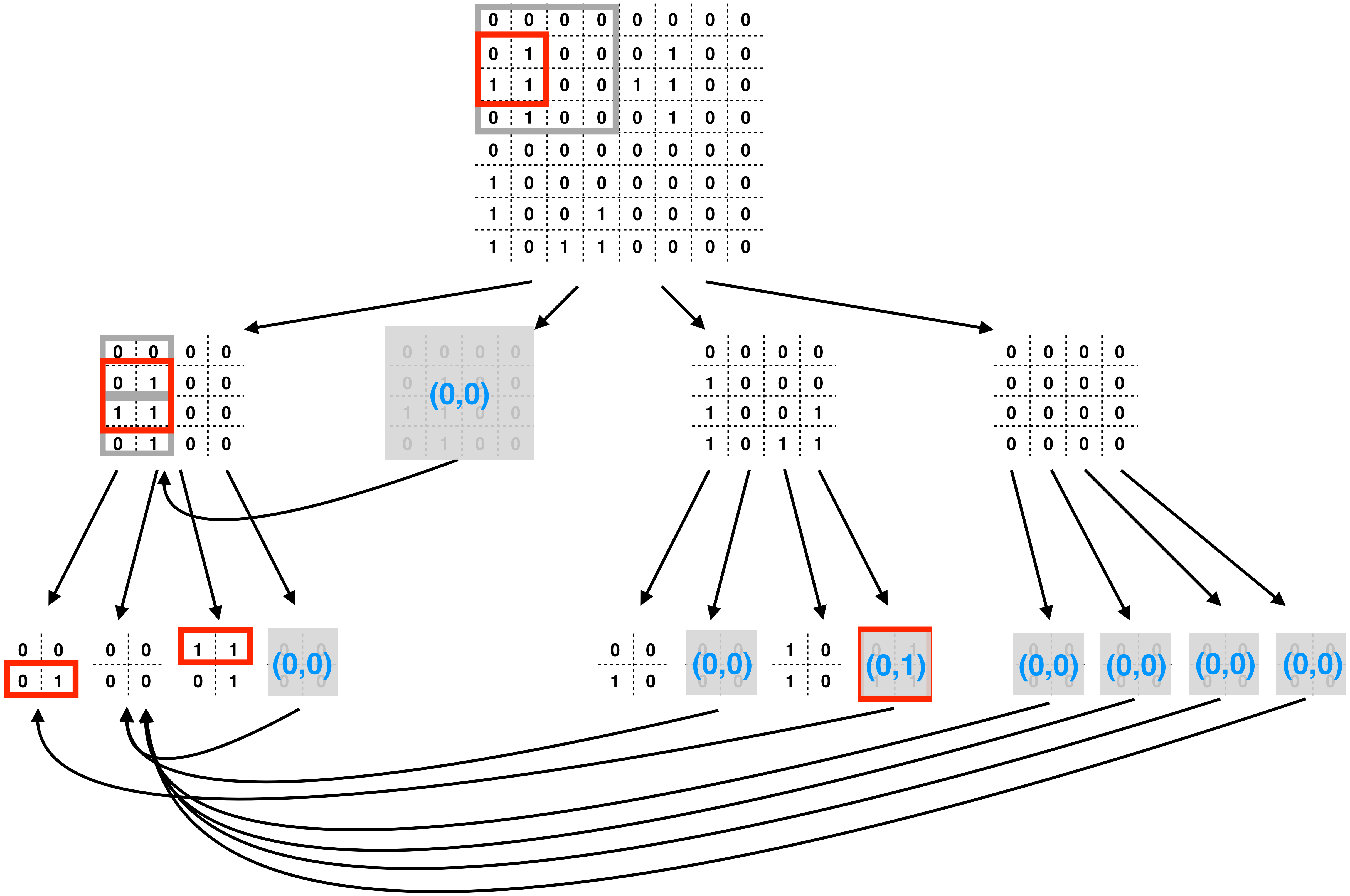
0	0
1	0

1	0
1	0

2D Block-trees

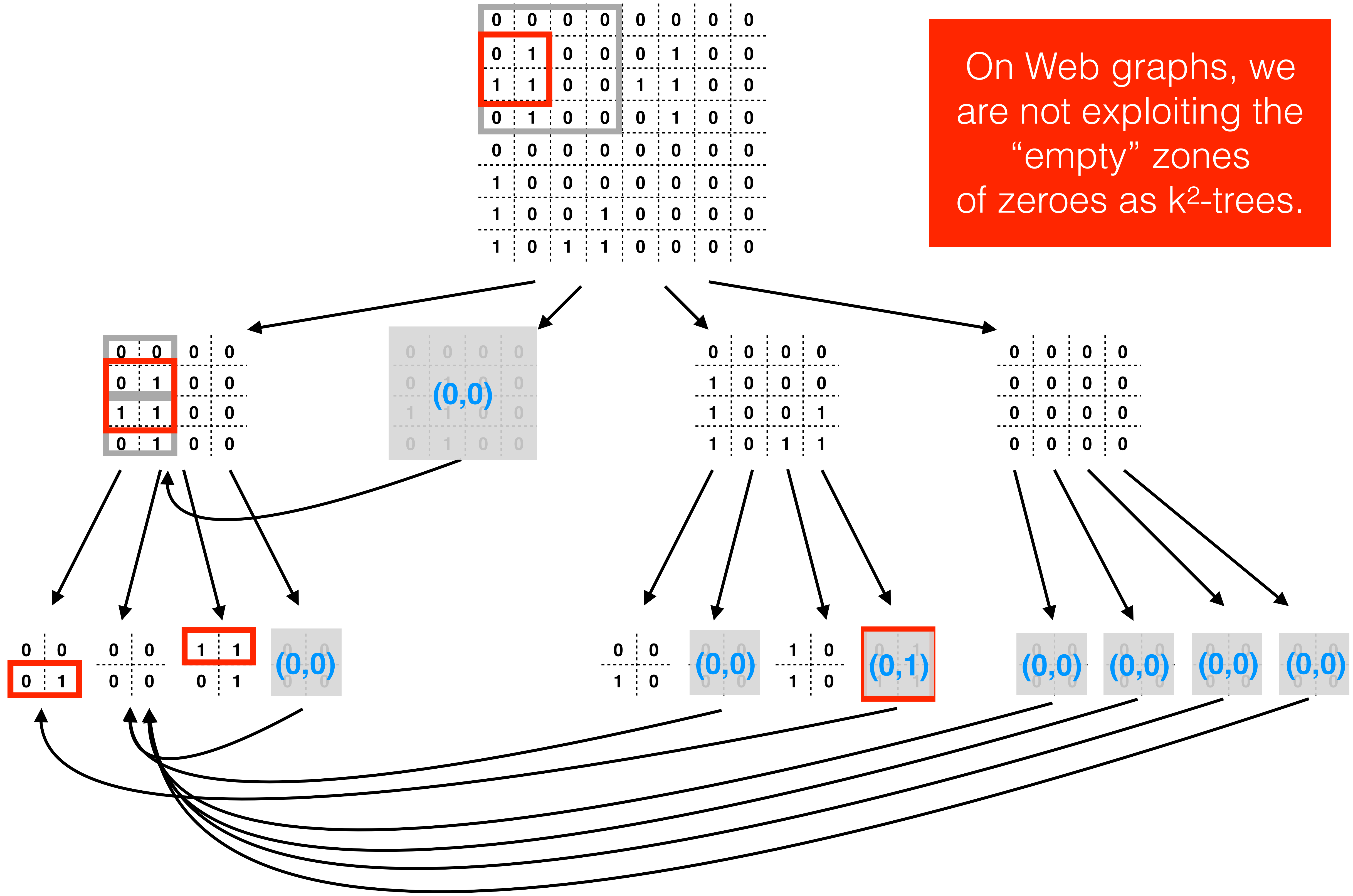


2D Block-trees



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

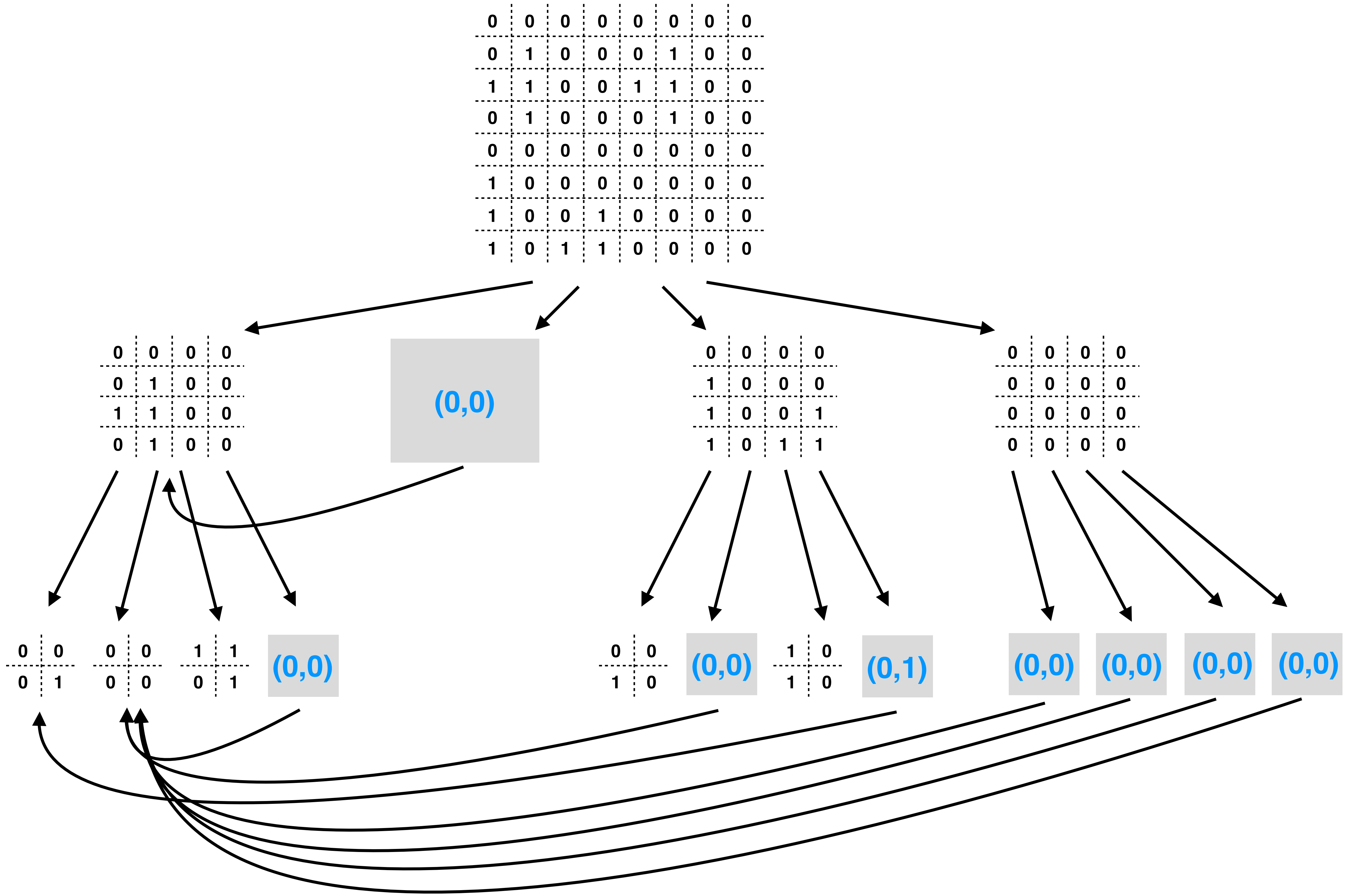
2D Block-trees



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

On Web graphs, we are not exploiting the “empty” zones of zeroes as k^2-trees.

2D Block-trees on Web graphs



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

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

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

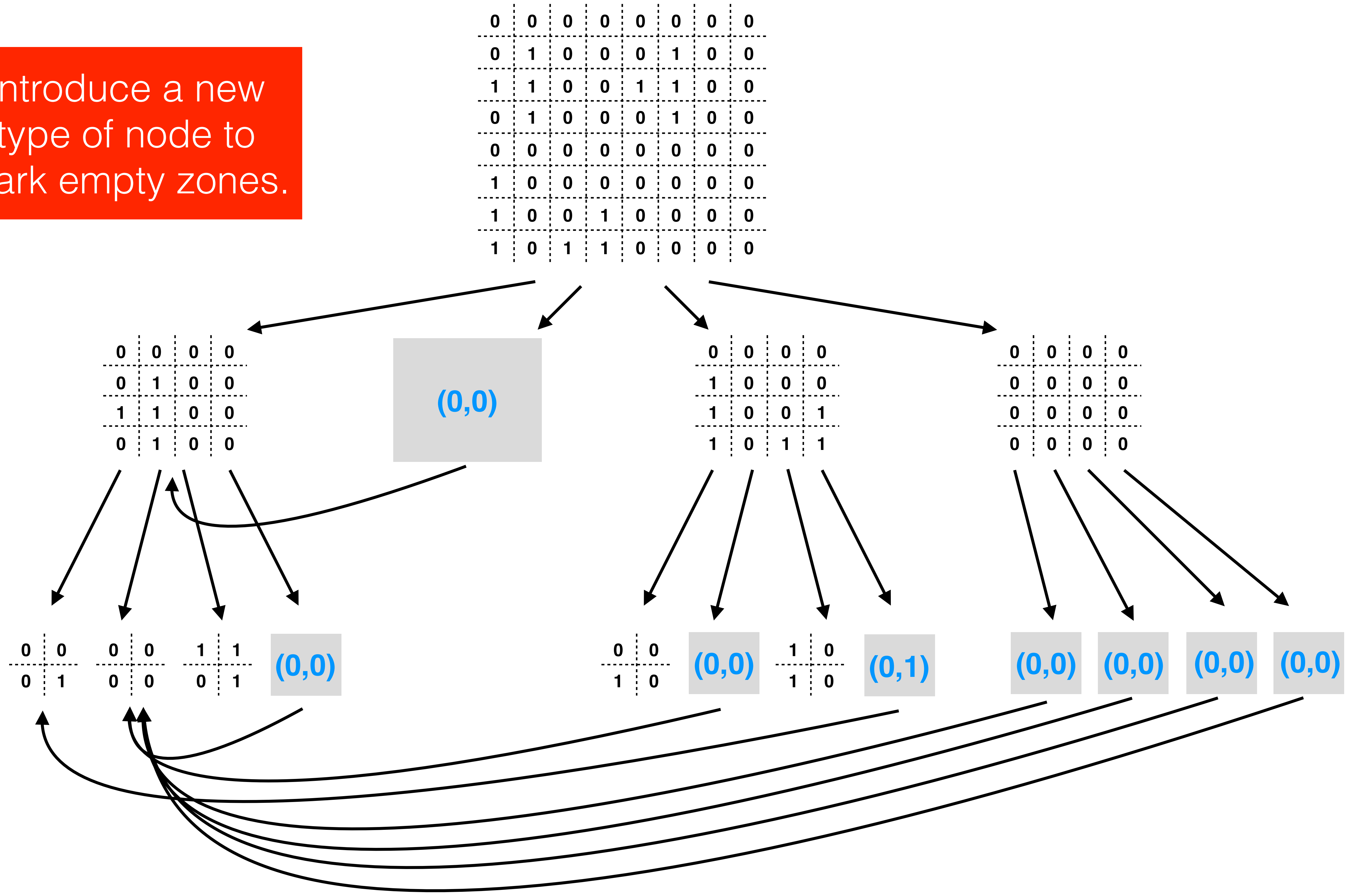
1	1
0	1

0	0
1	0

1	0
1	0

2D Block-trees on Web graphs

Introduce a new type of node to mark empty zones.



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

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

(0,0)

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

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

0	0
0	1

0	0
0	0

1	1
0	1

(0,0)

0	0
1	0

(0,0)

1	0
1	0

(0,1)

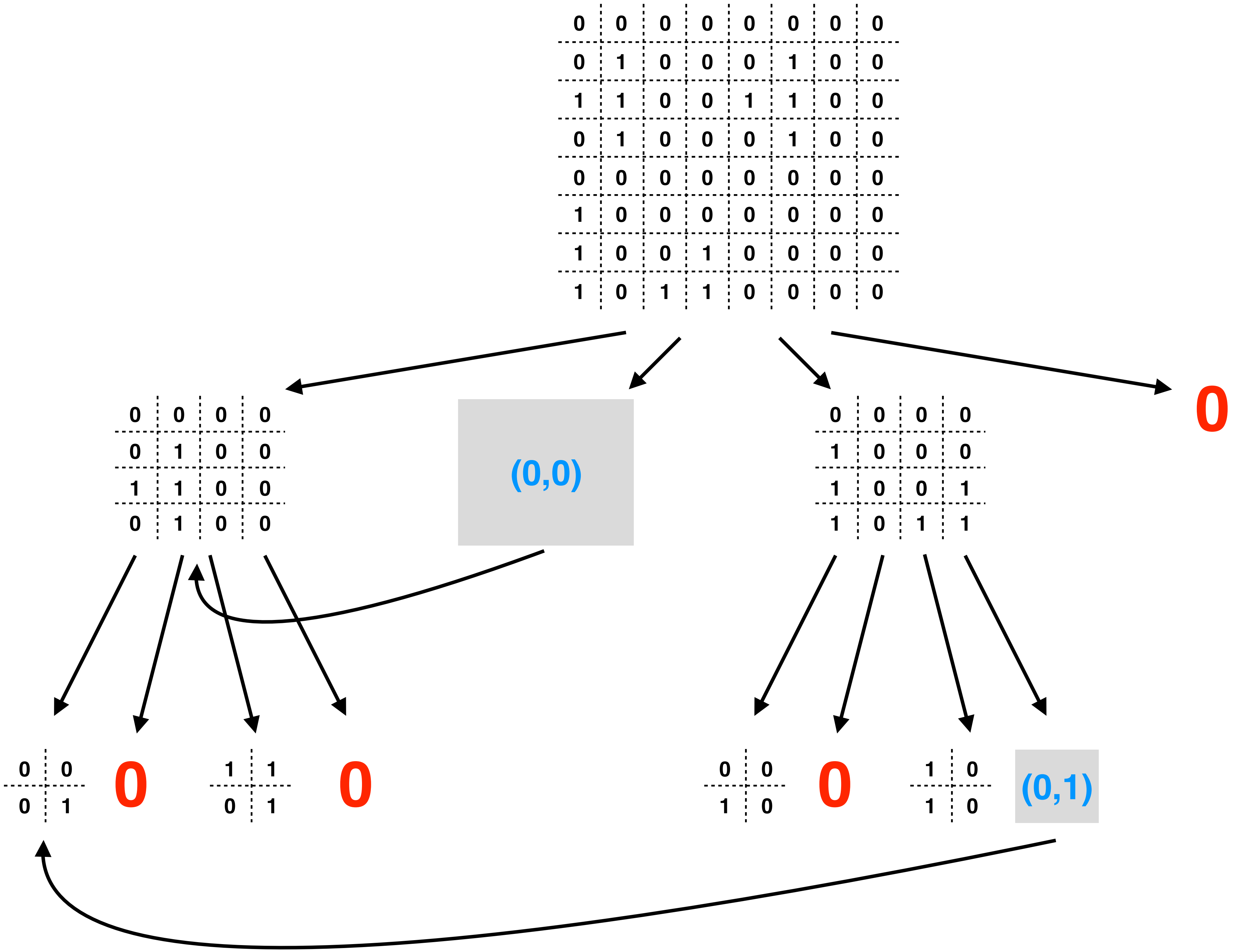
(0,0)

(0,0)

(0,0)

(0,0)

2D Block-trees on Web graphs



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

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

(0,0)

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

0

0	0
0	1

0

1	1
0	1

0

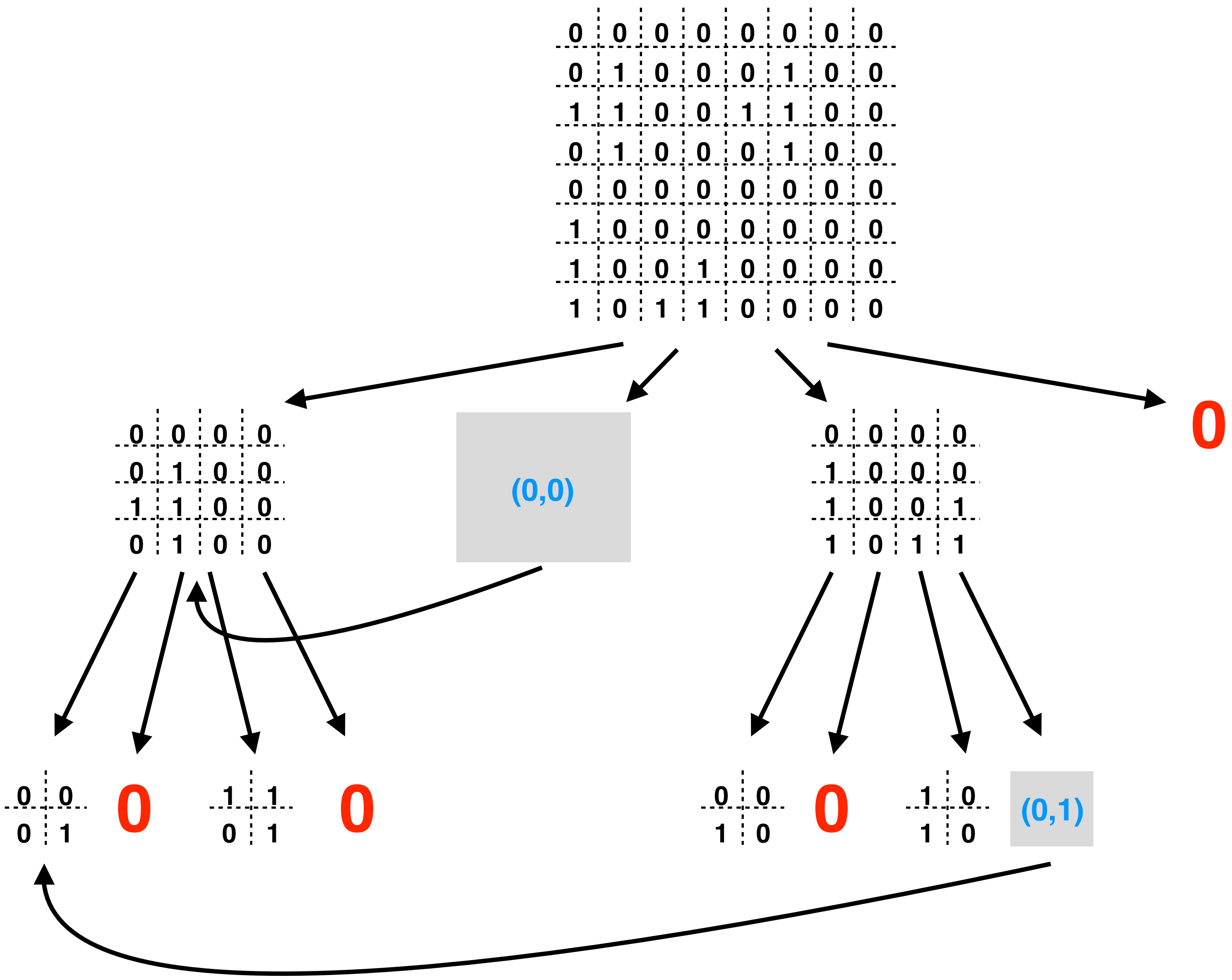
0	0
1	0

0

1	0
1	0

(0,1)

2D Block-trees on Web graphs



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

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

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

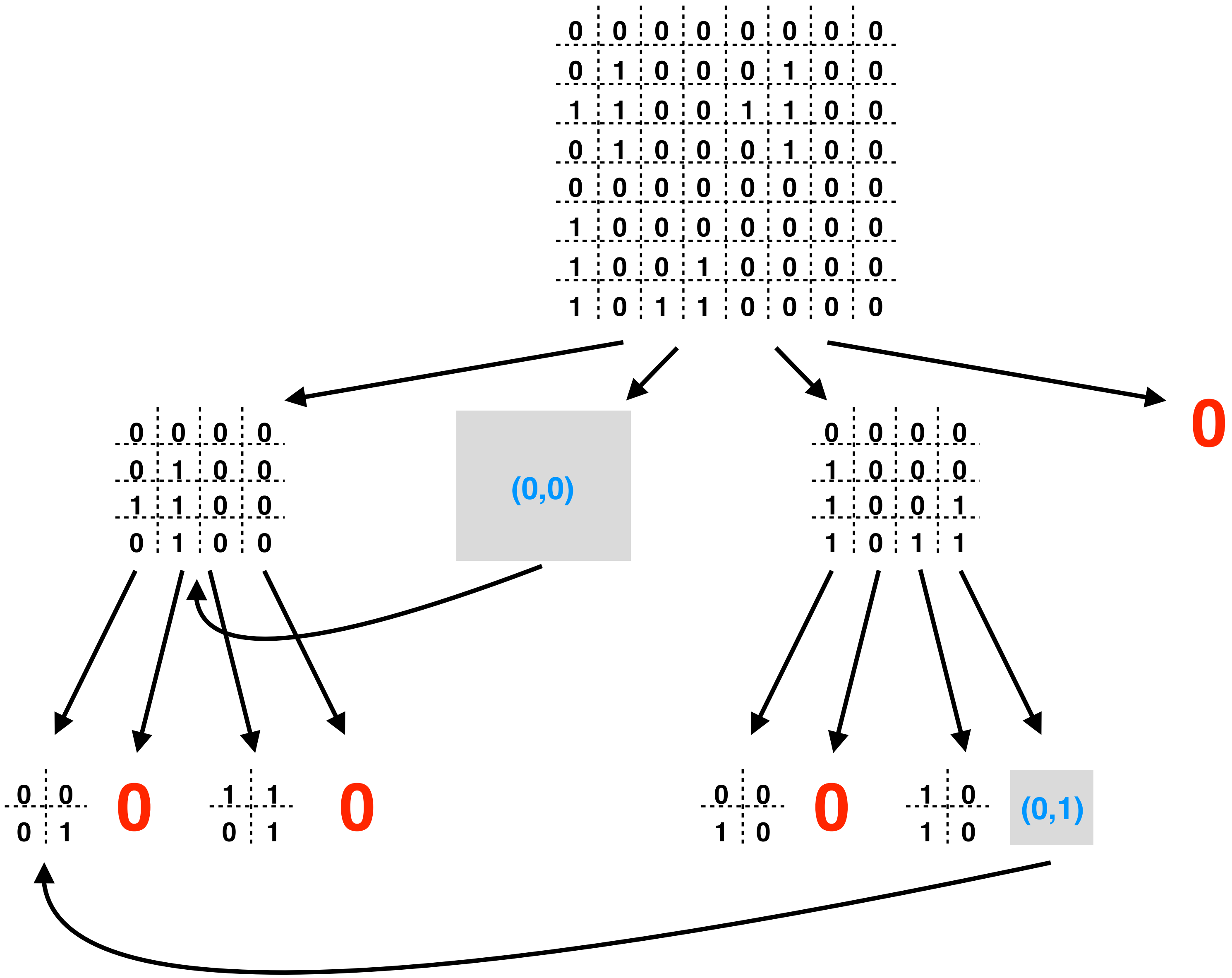
0	0
0	1

1	1
0	1

0	0
1	0

1	0
1	0

2D Block-trees on Web graphs



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

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

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

0	0
0	1

1	1
0	1

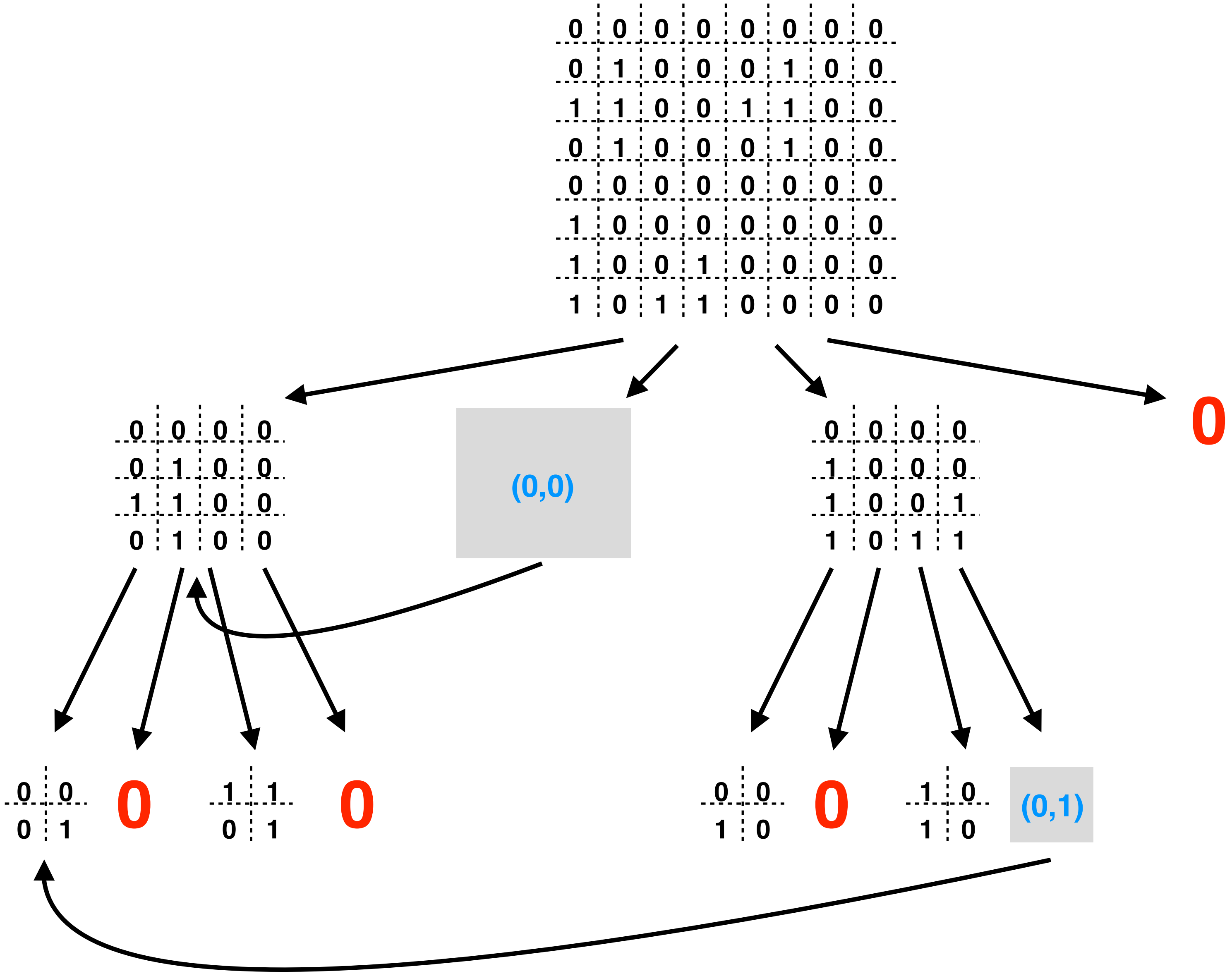
0	0
1	0

1	0
1	0

T 1010 1010 1010

L 0001 1101 0010 1010

2D Block-trees on Web graphs



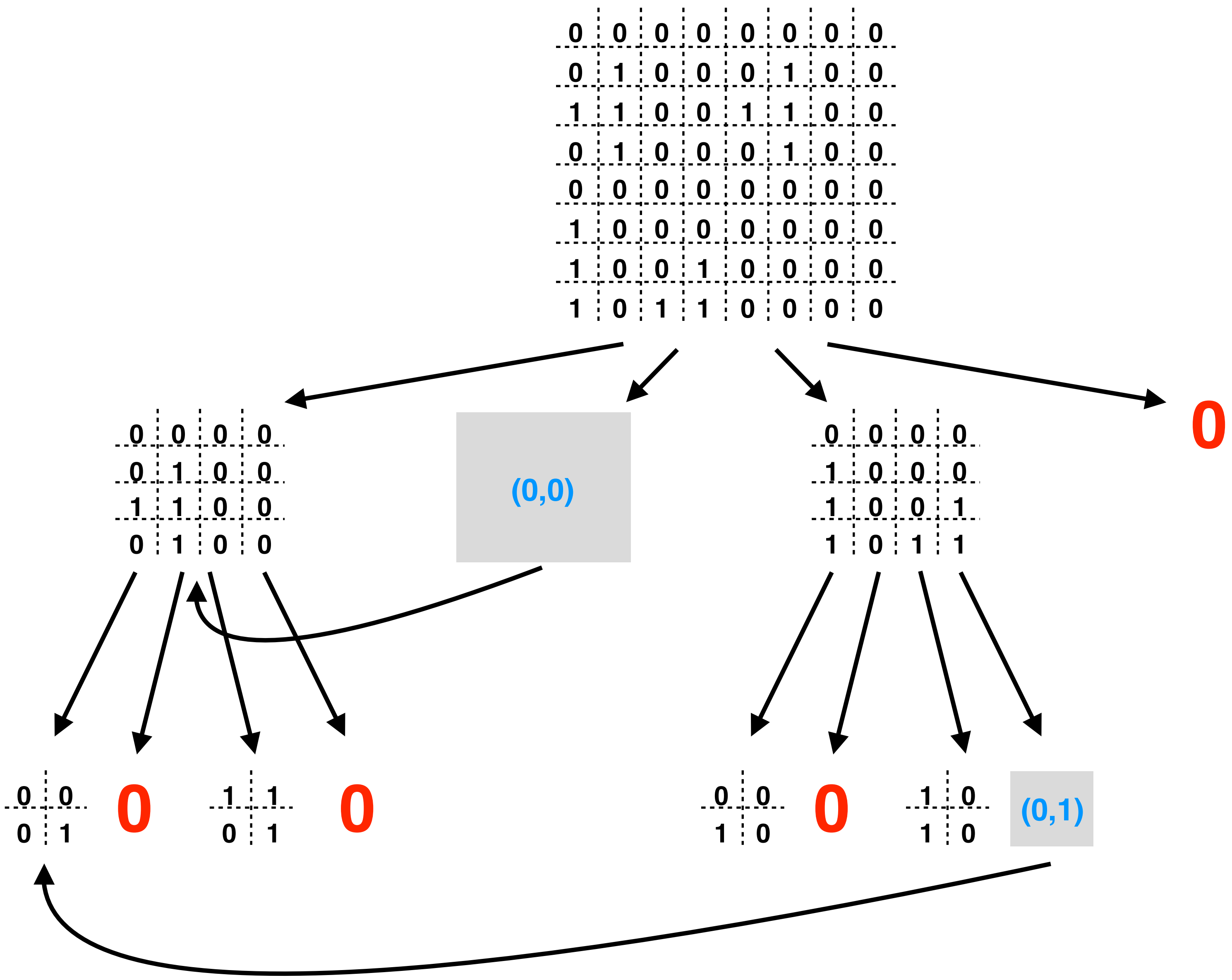
0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0
1	1	0	0	1	1	0	0
0	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
1	0	1	1	0	0	0	0

T 1010 1010 1010

L 0001 1101 0010 1010

N 100001

2D Block-trees on Web graphs



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

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

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

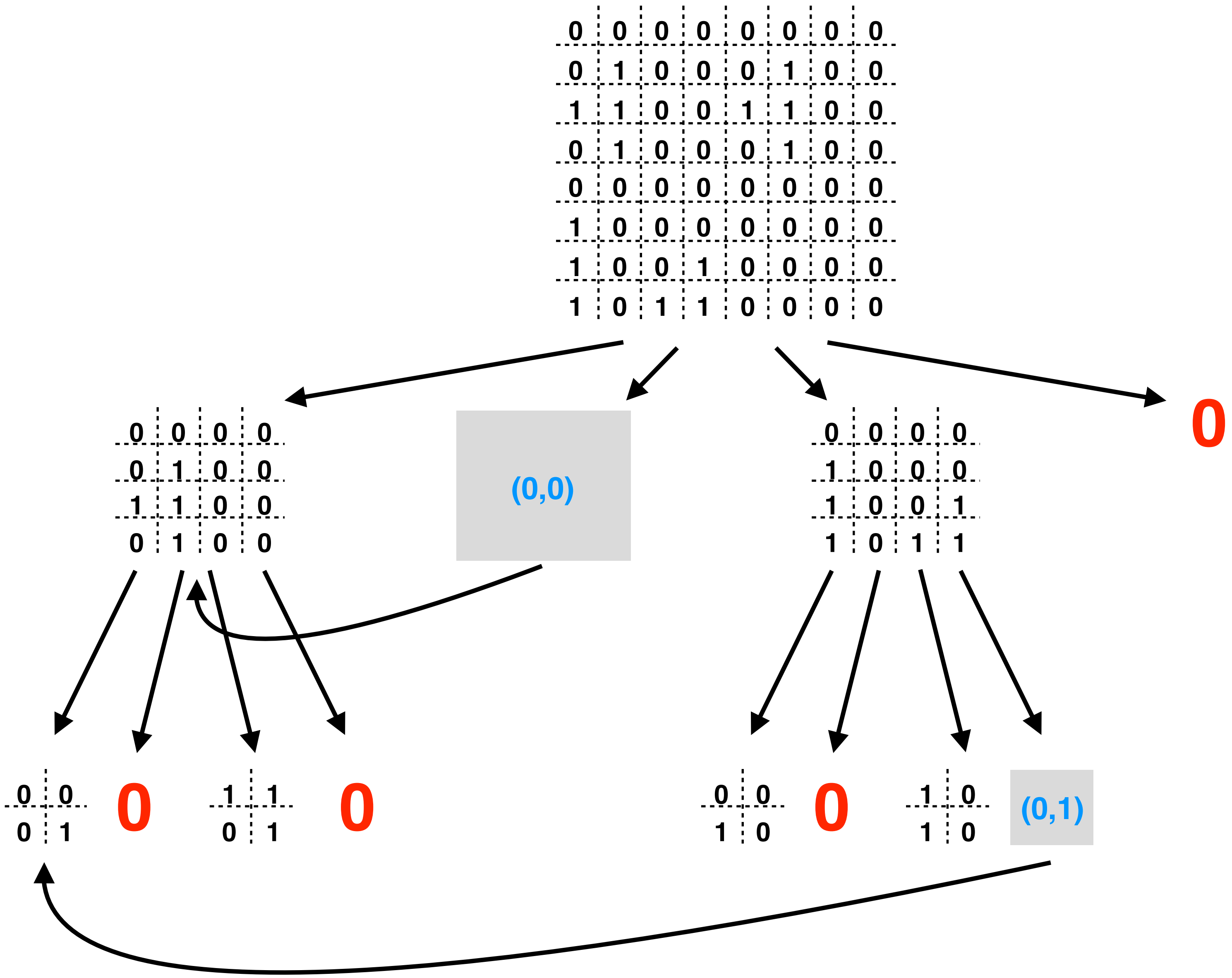
T 1010 1010 1010

L 0001 1101 0010 1010

N 100001

P$_1$ 1 **O**$_1$ 00

2D Block-trees on Web graphs



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

$$\begin{matrix} 0&0&0&0\\ 0&1&0&0\\ 1&1&0&0\\ 0&1&0&0 \end{matrix} \qquad (0,0) \qquad \begin{matrix} 0&0&0&0\\ 1&0&0&0\\ 1&0&0&1\\ 1&0&1&1 \end{matrix} \qquad 0$$

$$\begin{matrix} 0&0\\ 0&1 \end{matrix} \quad 0 \quad \begin{matrix} 1&1\\ 0&1 \end{matrix} \quad 0 \qquad \begin{matrix} 0&0\\ 1&0 \end{matrix} \quad 0 \quad \begin{matrix} 1&0\\ 1&0 \end{matrix} \quad (0,1)$$

T 1010 1010 1010

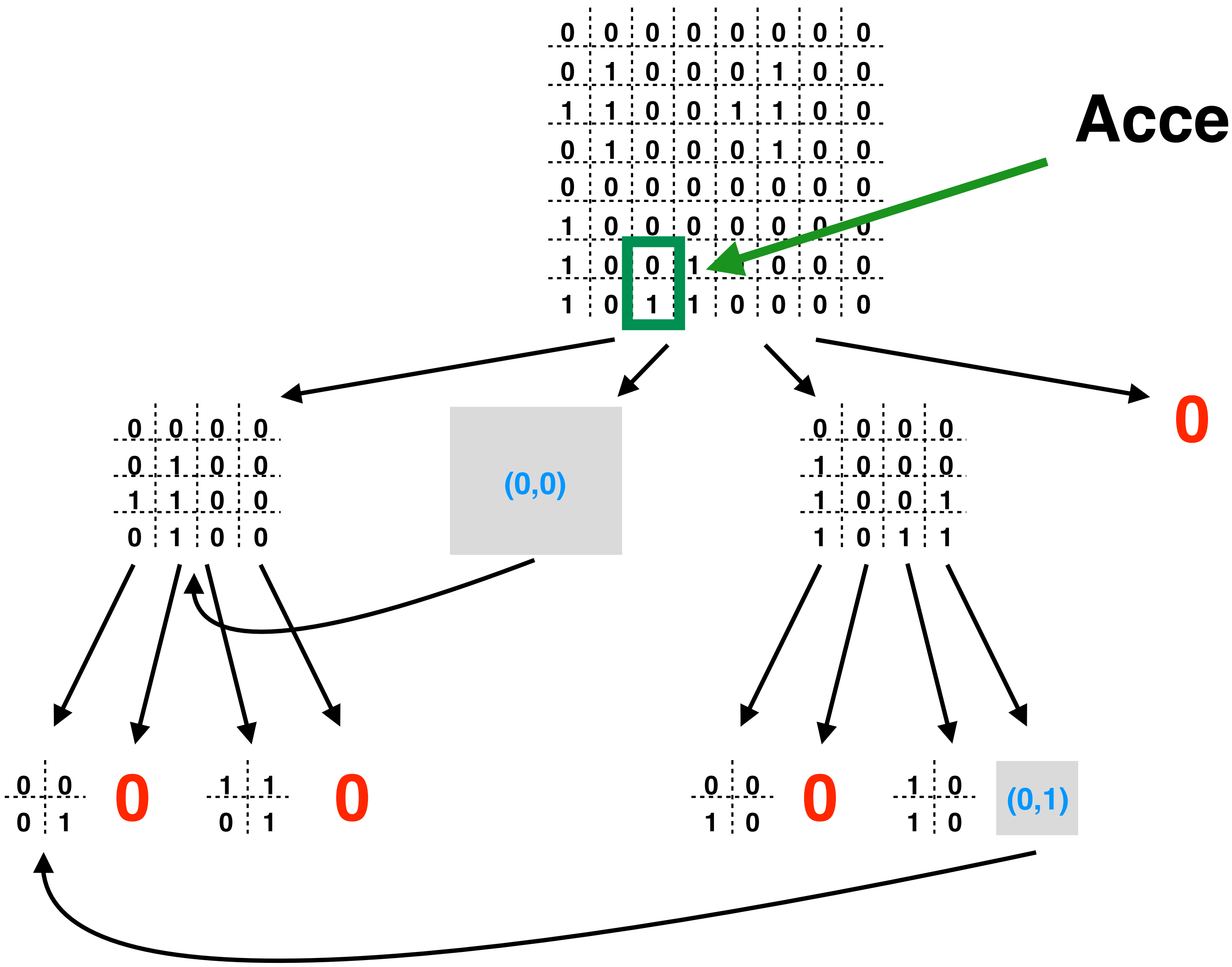
L 0001 1101 0010 1010

N 100001

P_1 1 **O_1** 00

P_2 7 **O_2** 01

2D Block-trees on Web graphs



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

T 1010 1010 1010

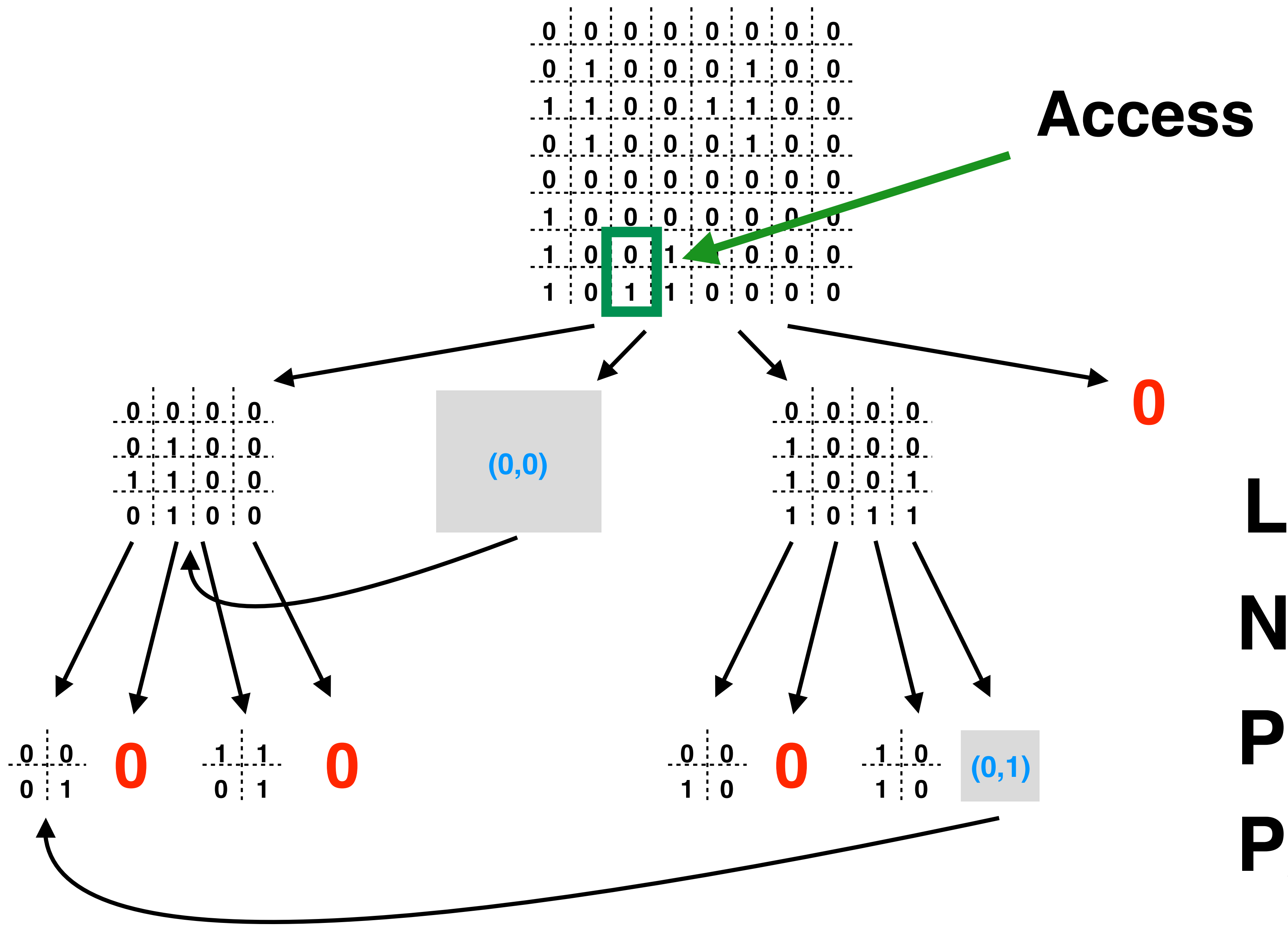
L 0001 1101 0010 1010

N 100001

P$_1$ 1 **O$_1$** 00

P$_2$ 7 **O$_2$** 01

2D Block-trees on Web graphs



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

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

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

0	0
0	1

1	1
0	1

0	0
1	0

1	0
1	0

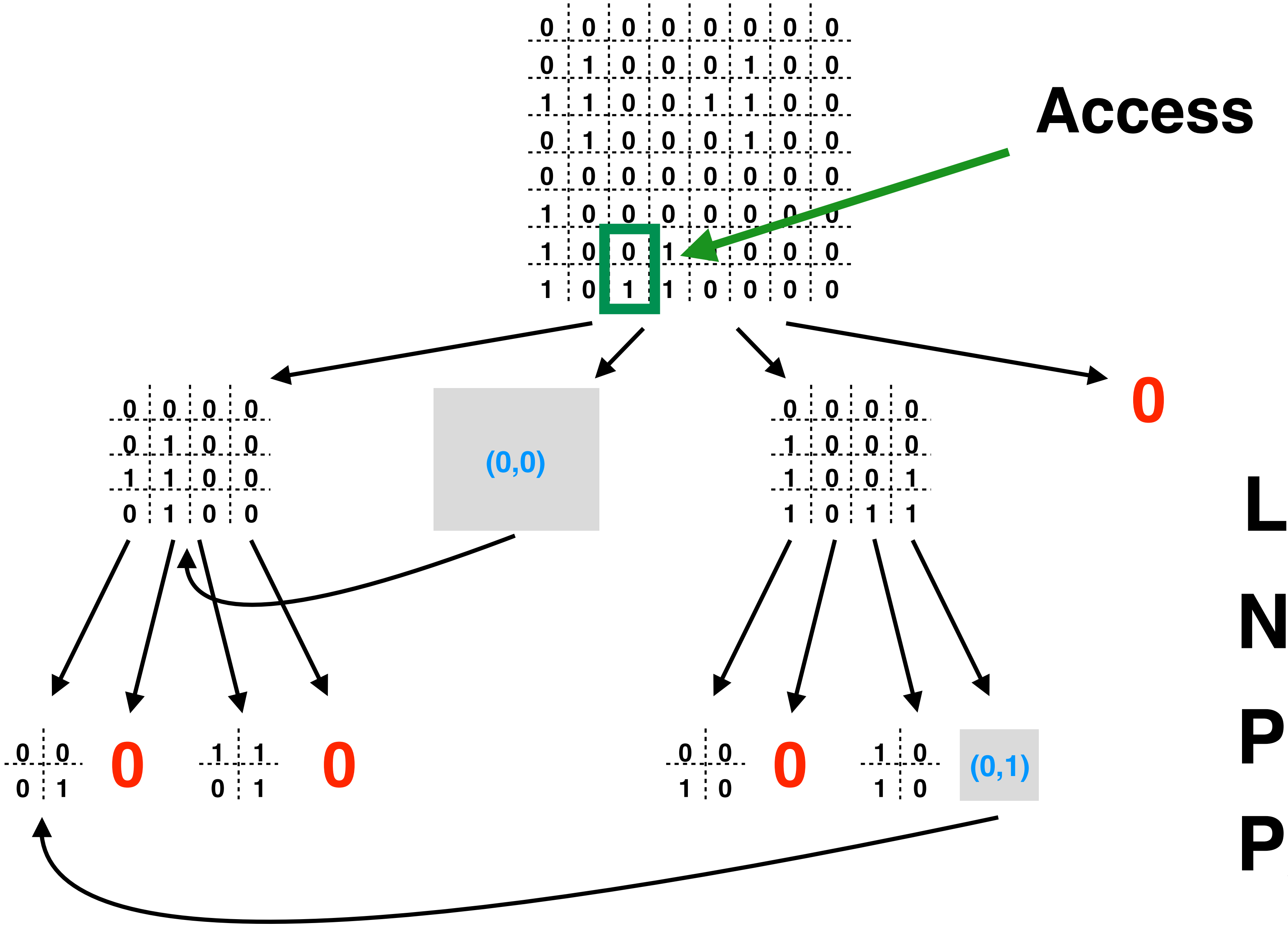
T 1010 1010 1010

L 0001 1101 0010 1010

N 100001

$\mathbf{P_1}$	1	$\mathbf{O_1}$	00
$\mathbf{P_2}$	7	$\mathbf{O_2}$	01

2D Block-trees on Web graphs



T 1010 1010 1010

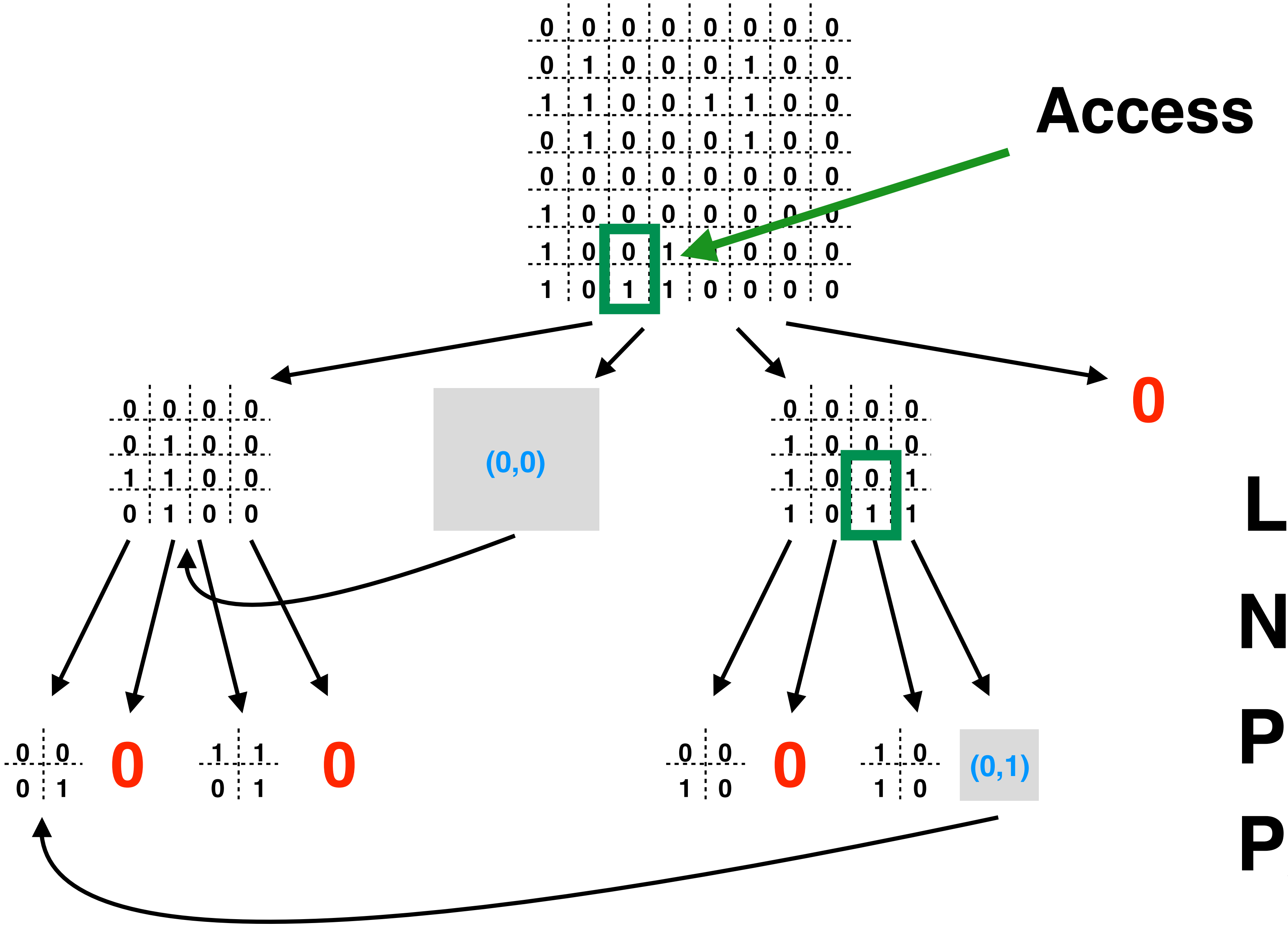
L 0001 1101 0010 1010

N 100001

P$_1$ 1 **O$_1$** 00

P$_2$ 7 **O$_2$** 01

2D Block-trees on Web graphs



T 1010 1010 1010

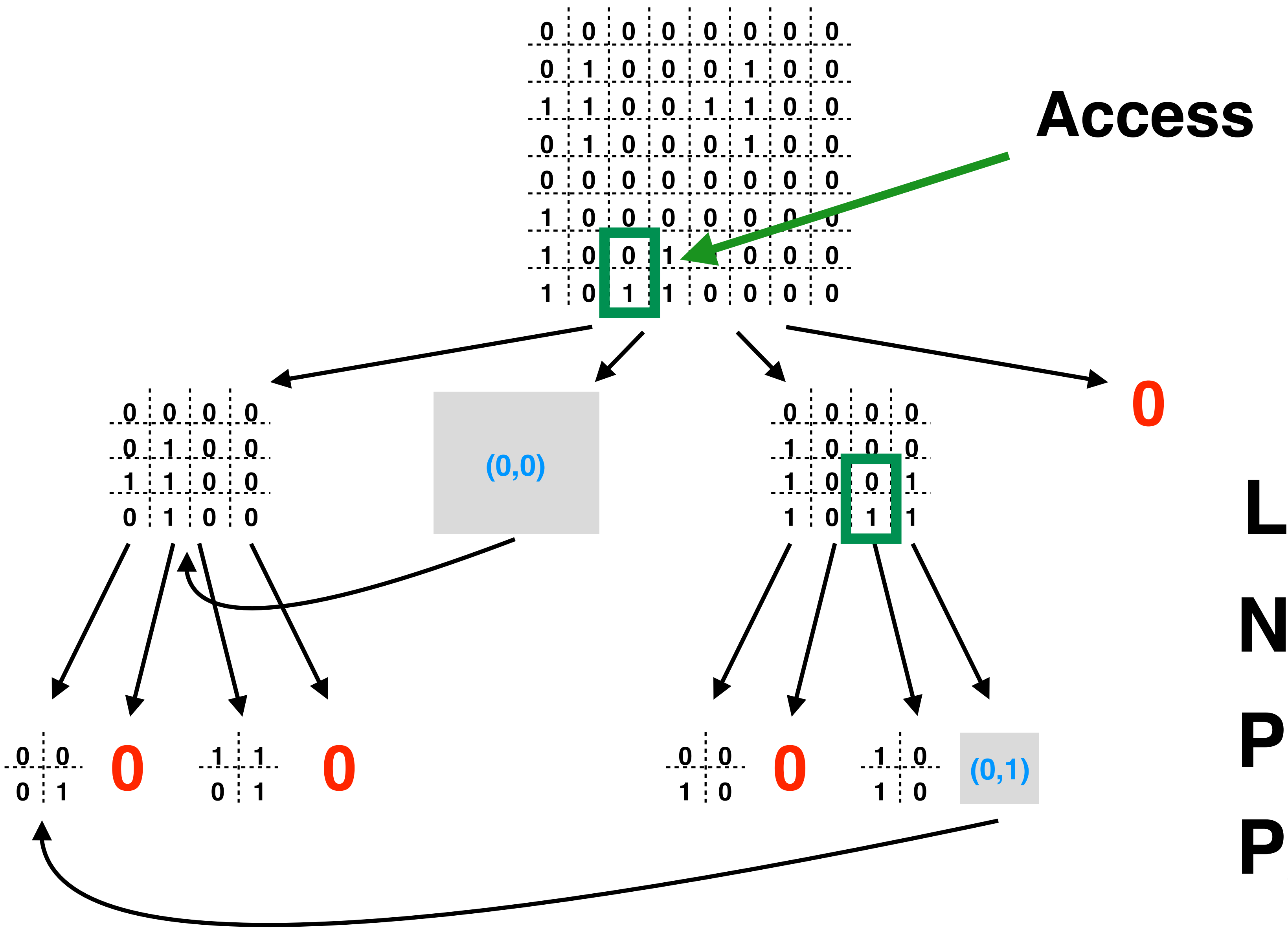
L 0001 1101 0010 1010

N 100001

P$_1$ 1 **O**$_1$ 00

P$_2$ 7 **O**$_2$ 01

2D Block-trees on Web graphs



T 1010 1010 1010

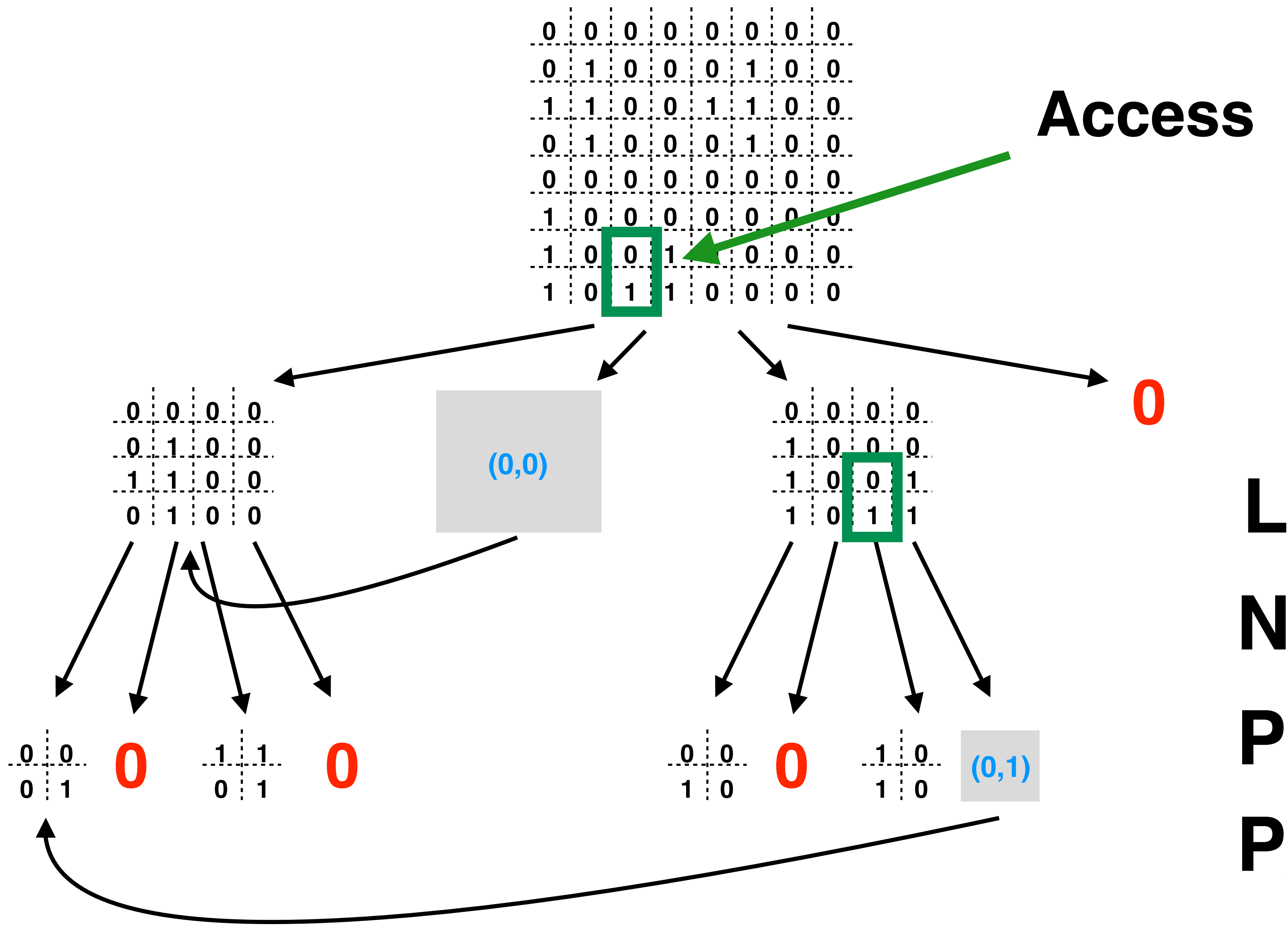
L 0001 1101 0010 1010

N 100001

P$_1$ 1 **O**$_1$ 00

P$_2$ 7 **O**$_2$ 01

2D Block-trees on Web graphs



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

T 1010 1010 1010

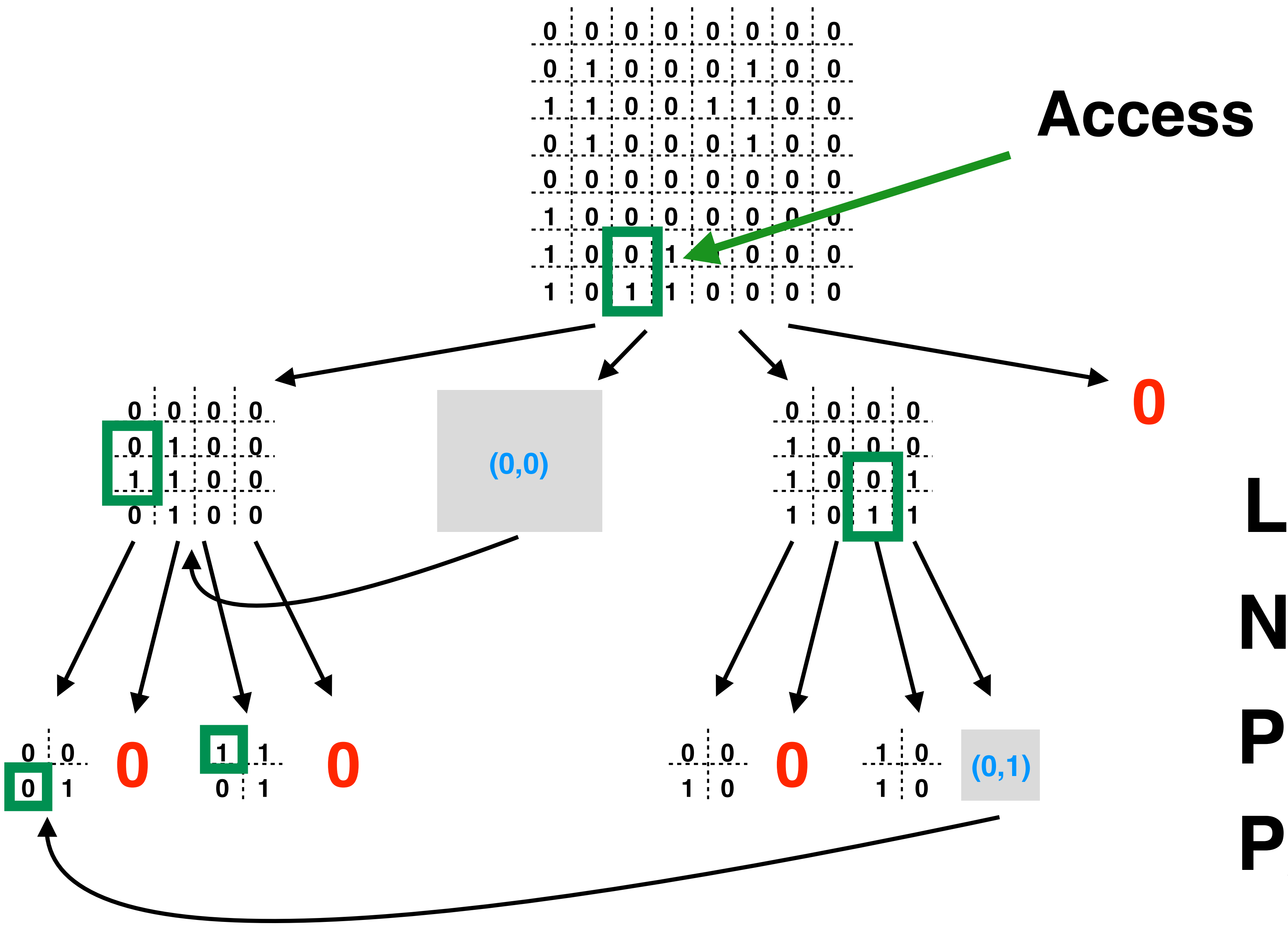
L 0001 1101 0010 1010

N 100001

P_1 1 **O_1** 00

P_2 7 **O_2** 01

2D Block-trees on Web graphs



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

T 1010 1010 1010

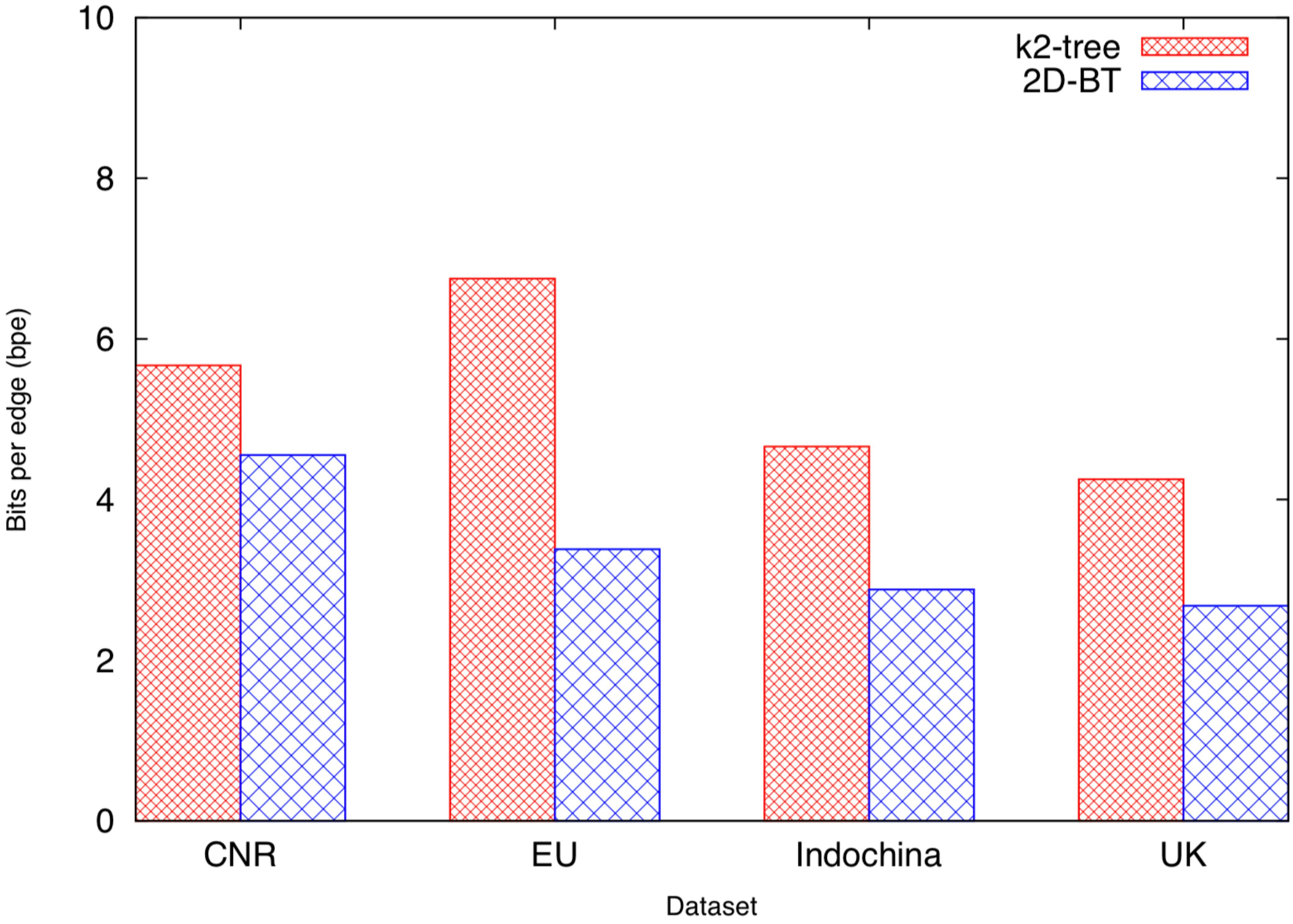
L 0001 1101 0010 1010

N 100001

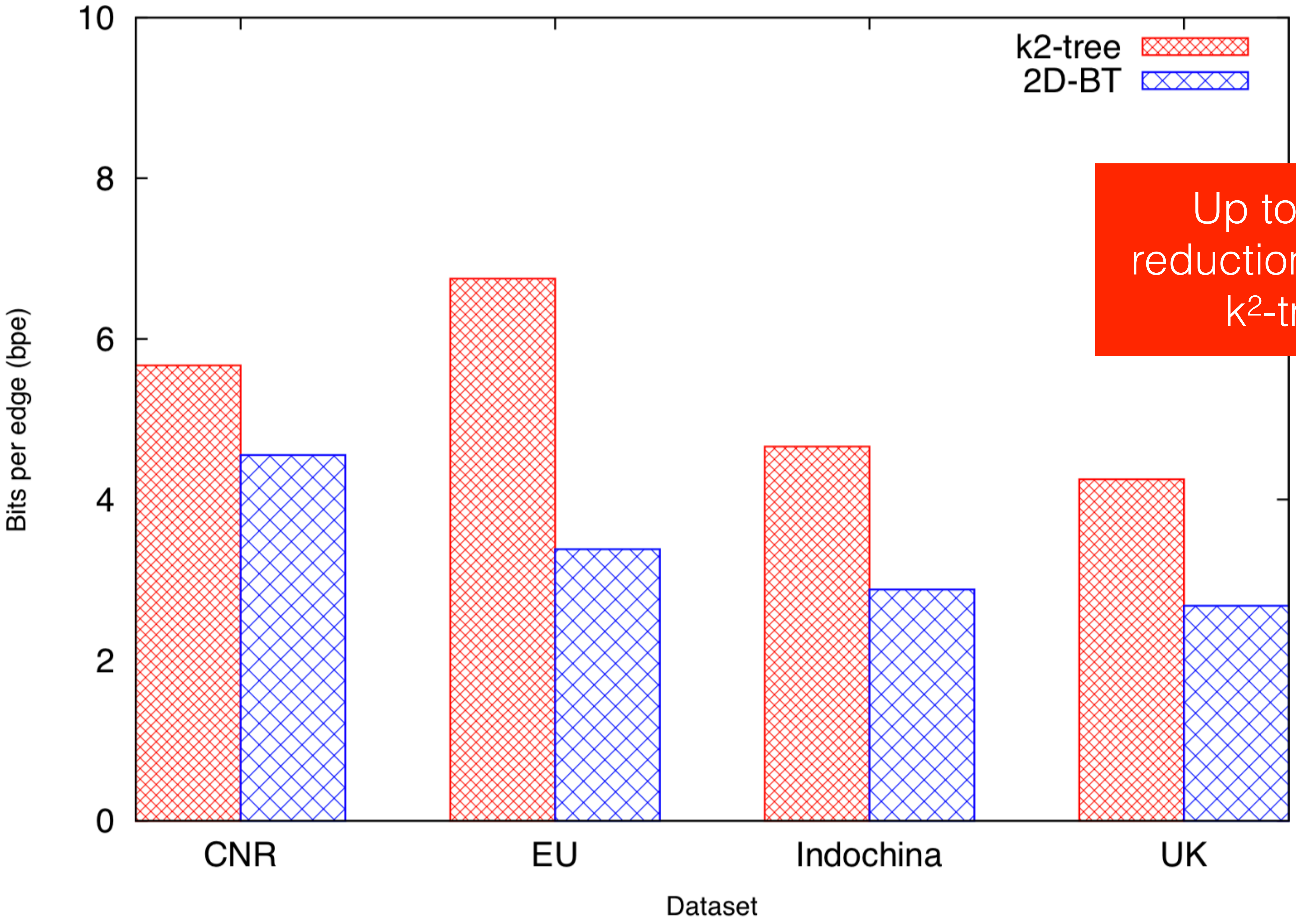
P_1 1 **O_1** 00

P_2 7 **O_2** 01

2D Block-trees on Web graphs

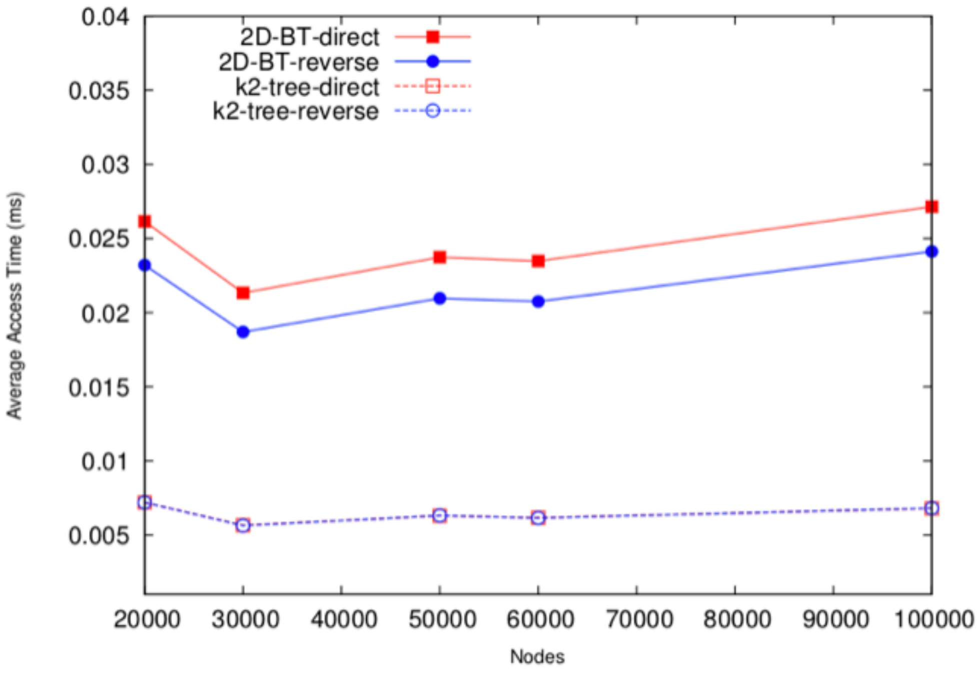


2D Block-trees on Web graphs

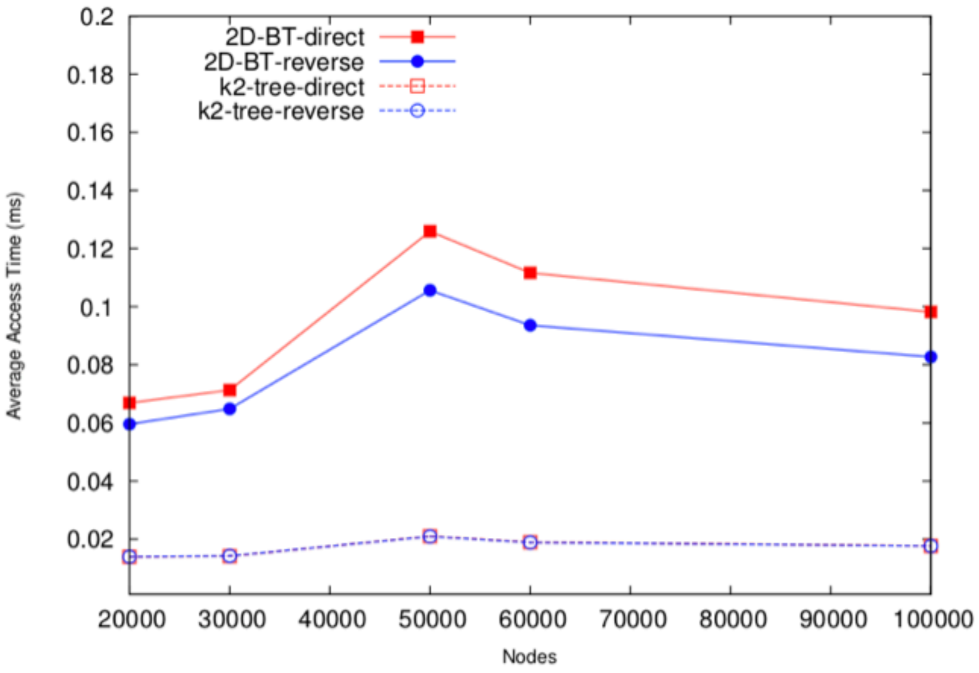


Up to **50%** reduction wrt the k^2-tree.

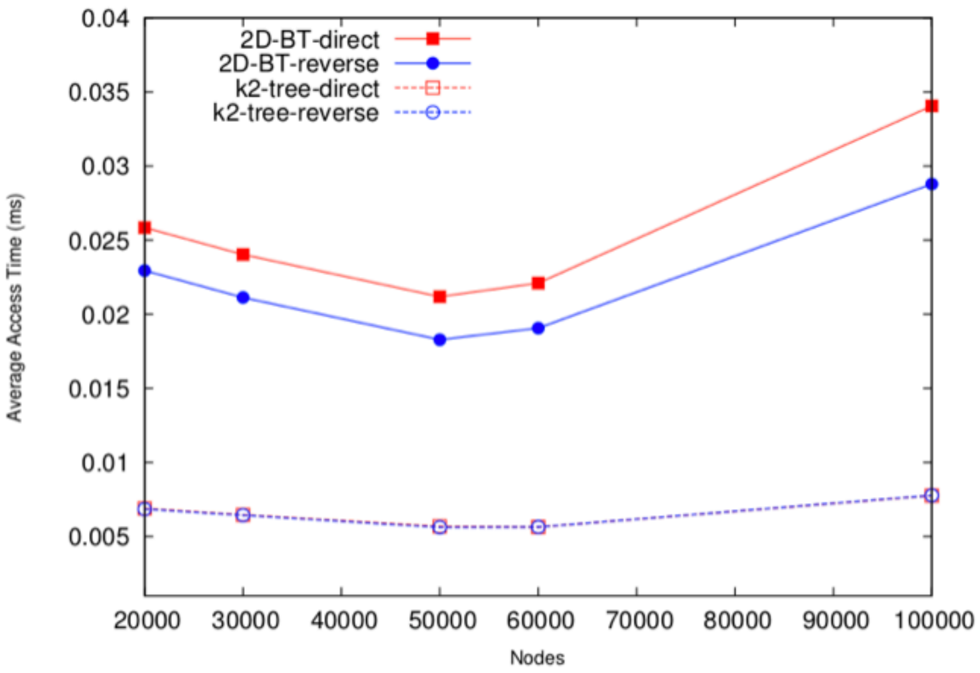
2D Block-trees on Web graphs



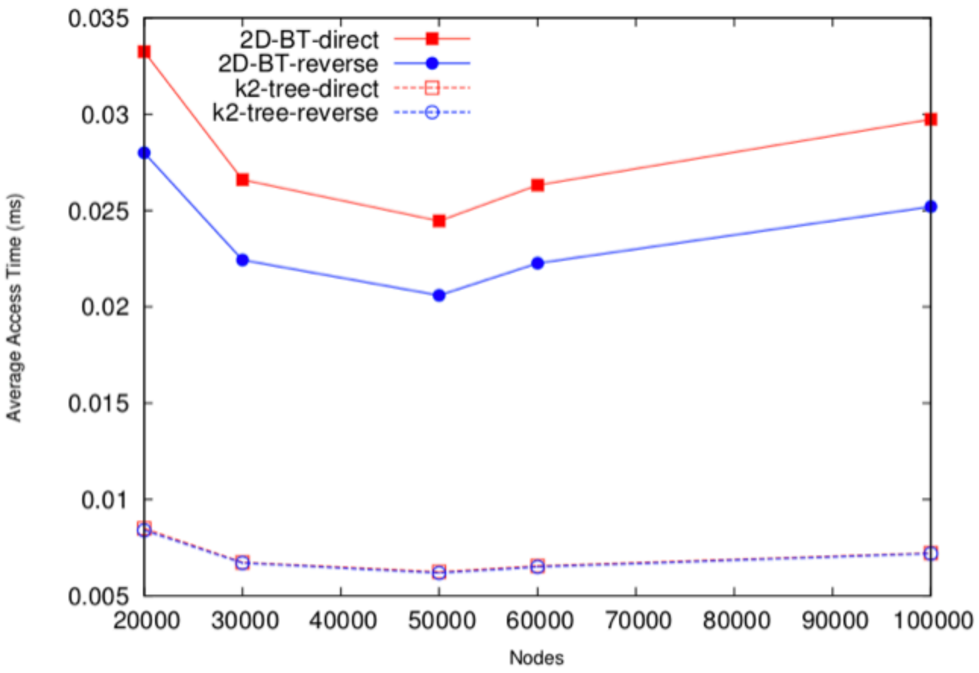
(a) Dataset CNR



(b) Dataset EU

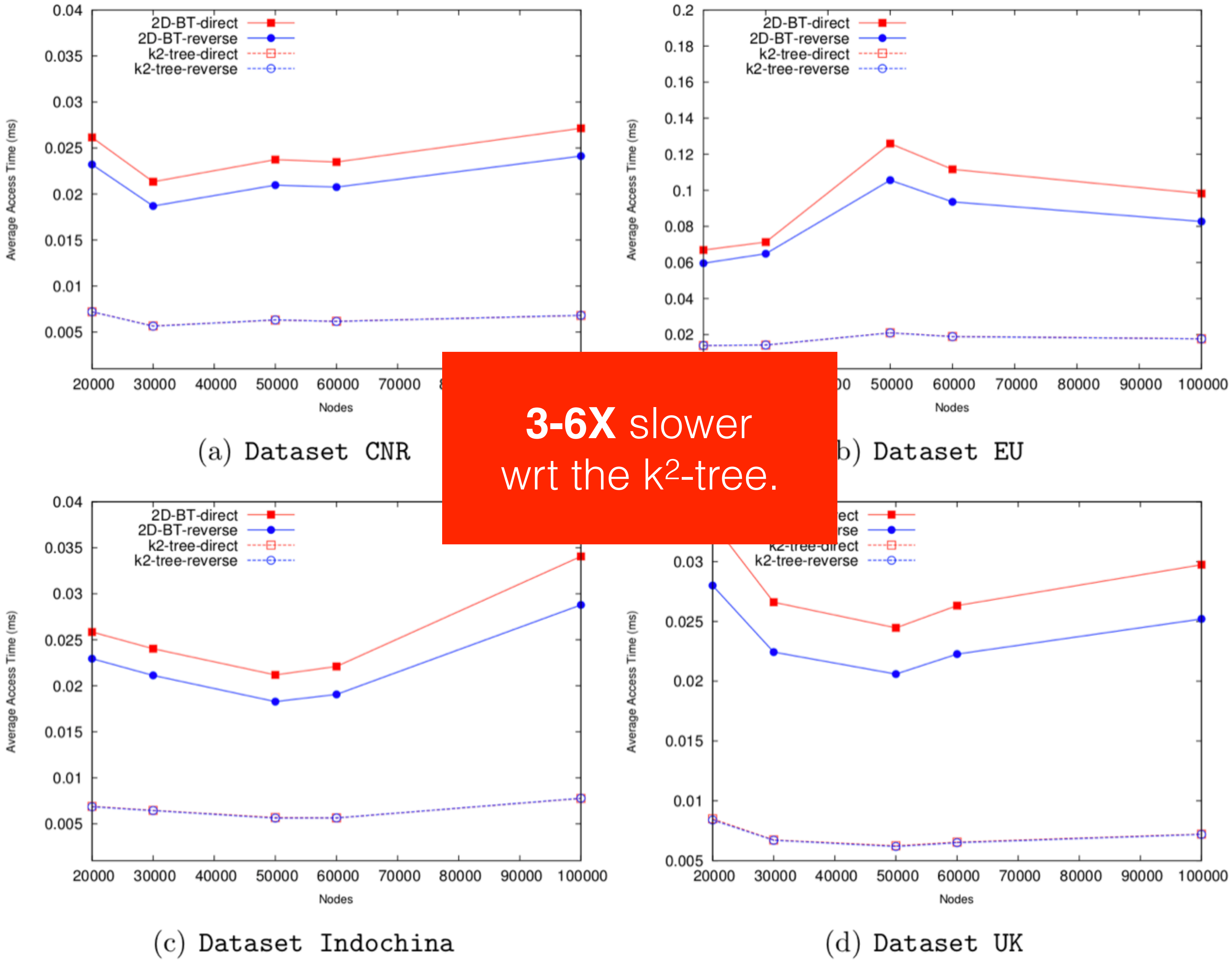


(c) Dataset Indochina



(d) Dataset UK

2D Block-trees on Web graphs



(a) Dataset CNR

(b) Dataset EU

(c) Dataset Indochina

(d) Dataset UK

3-6X slower wrt the k^2-tree.

Summing up

WebGraph supports efficient extraction of direct neighbours in excellent compressed space.

k^2-trees support direct and reverse navigation;
good trade-off between space and time;
do not exploit repetitiveness.

Block-trees compress (1D) strings to compression ratios close to LZ and support efficient random access to any substring.

2D-block-trees combines the capturing-spareness behaviour of k^2-trees with the capturing-repetitiveness of block-trees.
Up to 50% reduction in space, but 3-6X slower than k^2-trees.

Take home messages

WebGraph is still the most compact representation *if* only direct navigation is allowed.

k^2-trees achieve a good trade-off between space and time *when* both direct and reverse navigation is needed.

2D-block-trees are even smaller than k^2-trees *but* much slower.

Thanks for your attention, time, patience!

Any questions?