Parsing Unary Boolean Grammars Using Online Convolution

Alexander Okhotin[1] Christian Reitwießner[2]

[1]University of Turku, Finland

[2]University of Würzburg, Germany

Dagstuhl Seminar, December 2010

Boolean Grammars

- Context-free grammars:

$$A \to \alpha$$

"If w is generated by α, then w is generated by A".
Multiple rules for A: disjunction.

Boolean Grammars

- Context-free grammars:

$$A \to \alpha$$

"If w is generated by α, then w is generated by A".
Multiple rules for A: disjunction.

- Conjunctive grammars (Okhotin, 2000):

$$A \to \alpha_1 \& \ldots \& \alpha_m$$

"If w is generated by each α_i, then w is generated by A".

Boolean Grammars

- Context-free grammars:

$$A \to \alpha$$

"If w is generated by α, then w is generated by A".
Multiple rules for A: disjunction.

- Conjunctive grammars (Okhotin, 2000):

$$A \to \alpha_1 \& \ldots \& \alpha_m$$

"If w is generated by each α_i, then w is generated by A".

- Boolean grammars (Okhotin, 2003):

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n$$

"If w is generated by each α_i and by none of β_j, then w is generated by A".

Examples and Definitions

Conjunctive grammar for $\{a^n b^n c^n \mid n \geq 0\}$

$S \to AB\&DC$
$A \to aA \mid \varepsilon$
$B \to bBc \mid \varepsilon$
$C \to cC \mid \varepsilon$
$D \to aDb \mid \varepsilon$

Examples and Definitions

Conjunctive grammar for $\{a^n b^n c^n \mid n \geq 0\}$

$S \to AB\&DC$
$A \to aA \mid \varepsilon$
$B \to bBc \mid \varepsilon$
$C \to cC \mid \varepsilon$
$D \to aDb \mid \varepsilon$

Boolean grammar for $\{a^m b^n c^n \mid m \neq n\}$

$S \to AB\&\neg DC$
$A \to aA \mid \varepsilon$
$B \to bBc \mid \varepsilon$
$C \to cC \mid \varepsilon$
$D \to aDb \mid \varepsilon$

Examples and Definitions

Conjunctive grammar for $\{a^n b^n c^n \mid n \geq 0\}$

$S \to AB\&DC$
$A \to aA \mid \varepsilon$
$B \to bBc \mid \varepsilon$
$C \to cC \mid \varepsilon$
$D \to aDb \mid \varepsilon$

Boolean grammar for $\{a^m b^n c^n \mid m \neq n\}$

$S \to AB\&\neg DC$
$A \to aA \mid \varepsilon$
$B \to bBc \mid \varepsilon$
$C \to cC \mid \varepsilon$
$D \to aDb \mid \varepsilon$

Quadruple $G = (\Sigma, N, P, S)$, with rules of the form

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n, \quad \text{with } A \in N,\ \alpha_i, \beta_i \in (\Sigma \cup N)^*$$

Generated language defined by (for example) unique solutions of language equations.

Examples and Definitions

Conjunctive grammar for $\{a^n b^n c^n \mid n \geq 0\}$

$$S \to AB\&DC \qquad S = AB \cap DC$$
$$A \to aA \mid \varepsilon \qquad A = aA \cup \varepsilon$$
$$B \to bBc \mid \varepsilon \qquad B = bBc \cup \varepsilon$$
$$C \to cC \mid \varepsilon \qquad C = cC \cup \varepsilon$$
$$D \to aDb \mid \varepsilon \qquad D = aDb \cup \varepsilon$$

Boolean grammar for $\{a^m b^n c^n \mid m \neq n\}$

$$S \to AB\&\neg DC \qquad S = AB \cap \overline{DC}$$
$$A \to aA \mid \varepsilon \qquad A = aA \cup \varepsilon$$
$$B \to bBc \mid \varepsilon \qquad B = bBc \cup \varepsilon$$
$$C \to cC \mid \varepsilon \qquad C = cC \cup \varepsilon$$
$$D \to aDb \mid \varepsilon \qquad D = aDb \cup \varepsilon$$

Quadruple $G = (\Sigma, N, P, S)$, with rules of the form

$$A \to \alpha_1 \& \ldots \& \alpha_m \& \neg\beta_1 \& \ldots \& \neg\beta_n, \quad \text{with } A \in N,\ \alpha_i, \beta_i \in (\Sigma \cup N)^*$$

Generated language defined by (for example) unique solutions of language equations.

Power and Parsability of Boolean Grammars

Expressibility

Boolean grammars can...

- use all boolean operations
- express "all variables must be declared before use"
- generate $\{wcw \mid w \in \{a, b\}^*\}$ and $\{ww \mid w \in \{a, b\}^*\}$

Power and Parsability of Boolean Grammars

Expressibility

Boolean grammars can...

- use all boolean operations
- express "all variables must be declared before use"
- generate $\{wcw \mid w \in \{a, b\}^*\}$ and $\{ww \mid w \in \{a, b\}^*\}$

Parsability

Okhotin, 2010: Boolean grammars can be parsed by Boolean matrix multiplication (similar to Valiant's algorithm) in $O(n^{2.376})$.

Power and Parsability of Boolean Grammars

Expressibility

Boolean grammars can...

- use all boolean operations
- express "all variables must be declared before use"
- generate $\{wcw \mid w \in \{a, b\}^*\}$ and $\{ww \mid w \in \{a, b\}^*\}$

Parsability

Okhotin, 2010: Boolean grammars can be parsed by Boolean matrix multiplication (similar to Valiant's algorithm) in $O(n^{2.376})$.

Open Questions

- Conjunctive and Boolean grammars different?
- Exact power of conjunctive/Boolean grammars?

Unary Boolean Grammars

Definition

Grammar/language is unary if its alphabet has only one symbol.

$\rightarrow$ Unary context-free languages are regular!

Unary Boolean Grammars

Definition

Grammar/language is unary if its alphabet has only one symbol.

$\rightarrow$ Unary context-free languages are regular!

Theorem (Jeż, 2007)

Unary Boolean (even conjunctive) grammars can generate the non-regular language $\{a^{4^n} \mid n \geq 0\}$.

Unary Boolean Grammars

Definition

Grammar/language is unary if its alphabet has only one symbol.

$\rightarrow$ Unary context-free languages are regular!

Theorem (Jeż, 2007)

Unary Boolean (even conjunctive) grammars can generate the non-regular language $\{a^{4^n} \mid n \geq 0\}$.

Theorem (Jeż, Okhotin, 2008)

There is some unary Boolean (even Conjunctive) language outside of logarithmic space, unless PSPACE = EXPTIME.

Connection to Equations over Sets of Natural Numbers

Different Interpretation

unary Boolean grammars or language equations	equations over sets of natural numbers with operations $\cap, \cup, \neg, +$
$A, B \subseteq \{a\}^*$	$A, B \subseteq \mathbb{N}$
$A \cdot B = \{a^{i+j} \mid a^i \in A, a^j \in B\}$	$A + B = \{i + j \mid i \in A, j \in B\}$

Naïve Parsing Algorithm

Reminder: Rules (Binary Normal Form)

$$A \to B_1 C_1 \& \ldots \& B_n C_n \& \neg D_1 E_1 \& \ldots \& \neg D_m E_m$$

$$A \to a$$

Input: Grammar G, a^n

reserve one bitvector 1...n per nonterminal (NTs) and concatenation pair (CPs)

initialize bitvectors for NTs from rules $A \to a$

for i := 2 to n:

  CPs: set bit i from bits 1...(i-1) in NTs

$$a^i \in A \cdot B \iff \bigvee_{k=1}^{i-1} (a^k \in A \wedge a^{i-k} \in B)$$

  NTs: set bit i from bit i in CPs according to rules

Running time: $O(|G| \cdot n^2)$

Improving the Algorithm

What can be improved?

- Boolean operations not problematic, but concatenation is done inefficiently
- $\forall i\colon a^i \in A \cdot B \iff \bigvee_{k=1}^{i-1}(a^k \in A \wedge a^{i-k} \in B)$
- $\rightarrow$ Boolean convolution of bitvectors
- since $a^k \in A$ is only known after we know $a^k \in C \cdot D$ for all CPs C, D, we need fast online convolution

Definition

Online convolution of bitvectors: Read inputs bit by bit (low order first) and print ith bit of output before reading ith bit of input

Improved Algorithm

Definition

Online convolution of bitvectors: Read inputs bit by bit (low order first) and print ith bit of output before reading ith bit of input

Input: Grammar G, a^n

reserve one bitvector 1...n per nonterminal (NTs) and concatenation pair (CPs)

initialize bitvectors for NTs from rules $A \to a$

run parallel instances of online conv. for each CP

for i := 2 to n:

  CPs: set bit i using online convolution

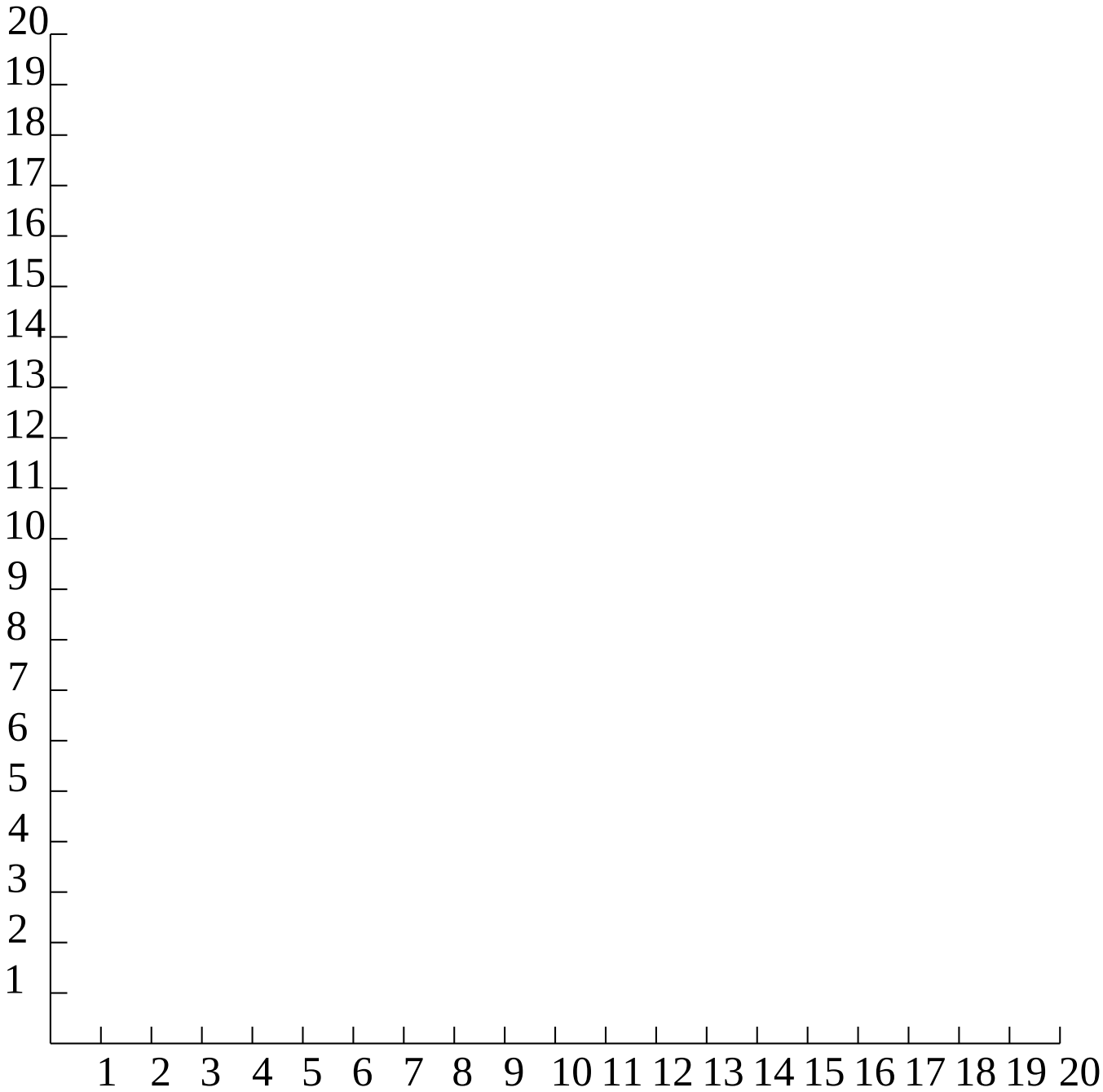
$$a^i \in A \cdot B \iff \bigvee_{k=1}^{i-1} (a^k \in A \wedge a^{i-k} \in B)$$

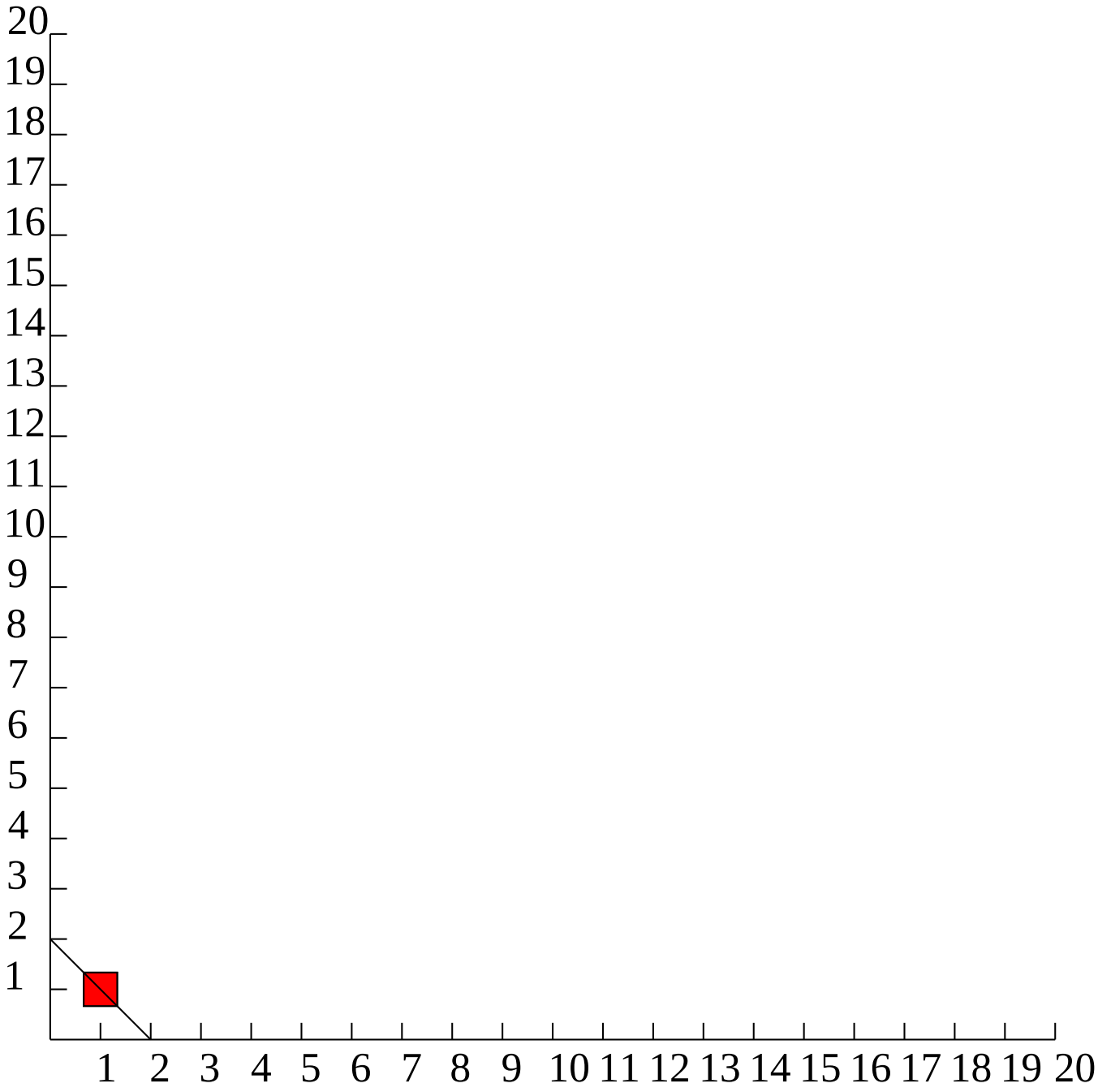
  NTs: set bit i from bit i in CPs according to rules

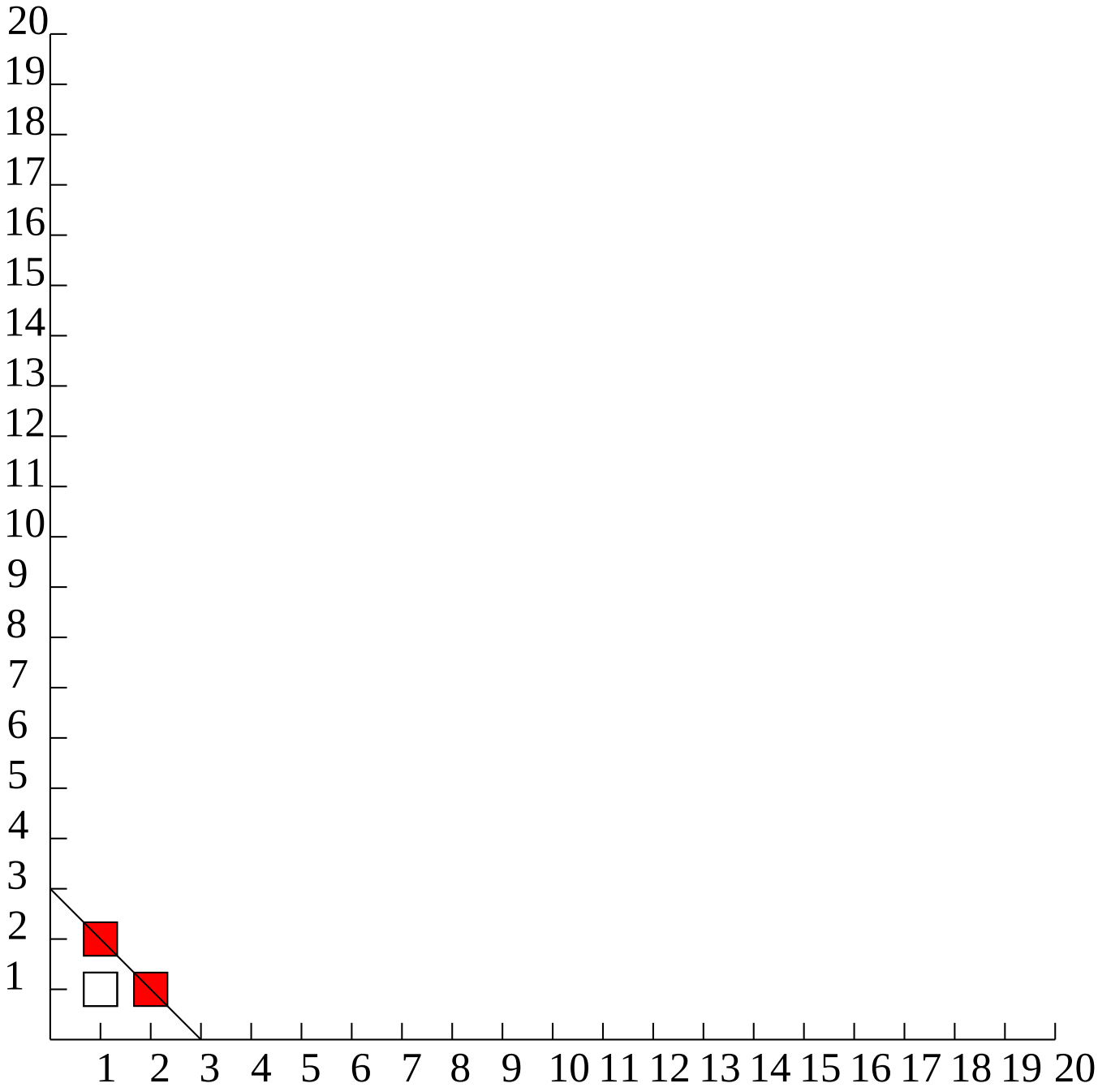
Offline to Online

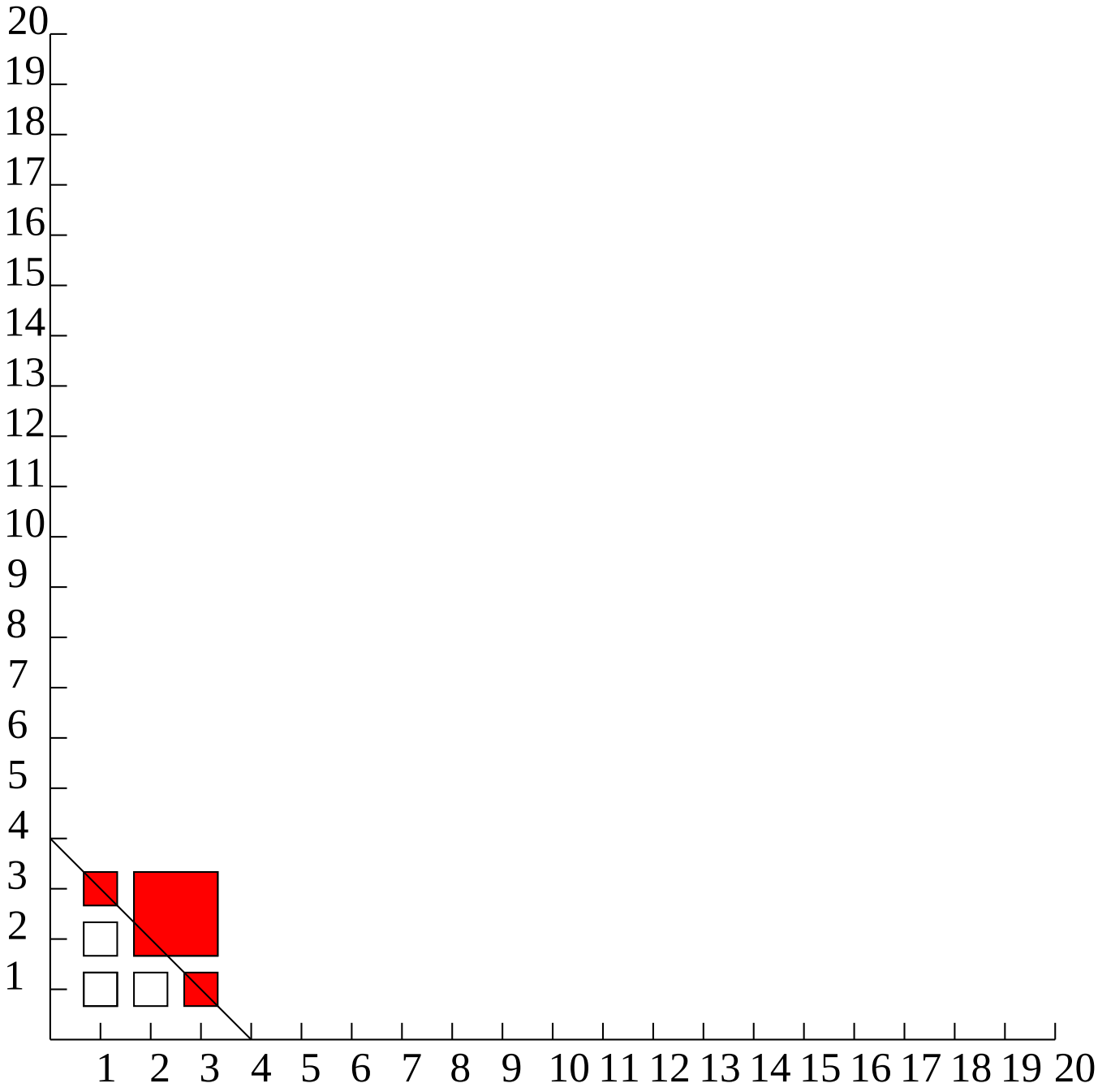
Theorem (Fischer, Stockmeyer, 1974)

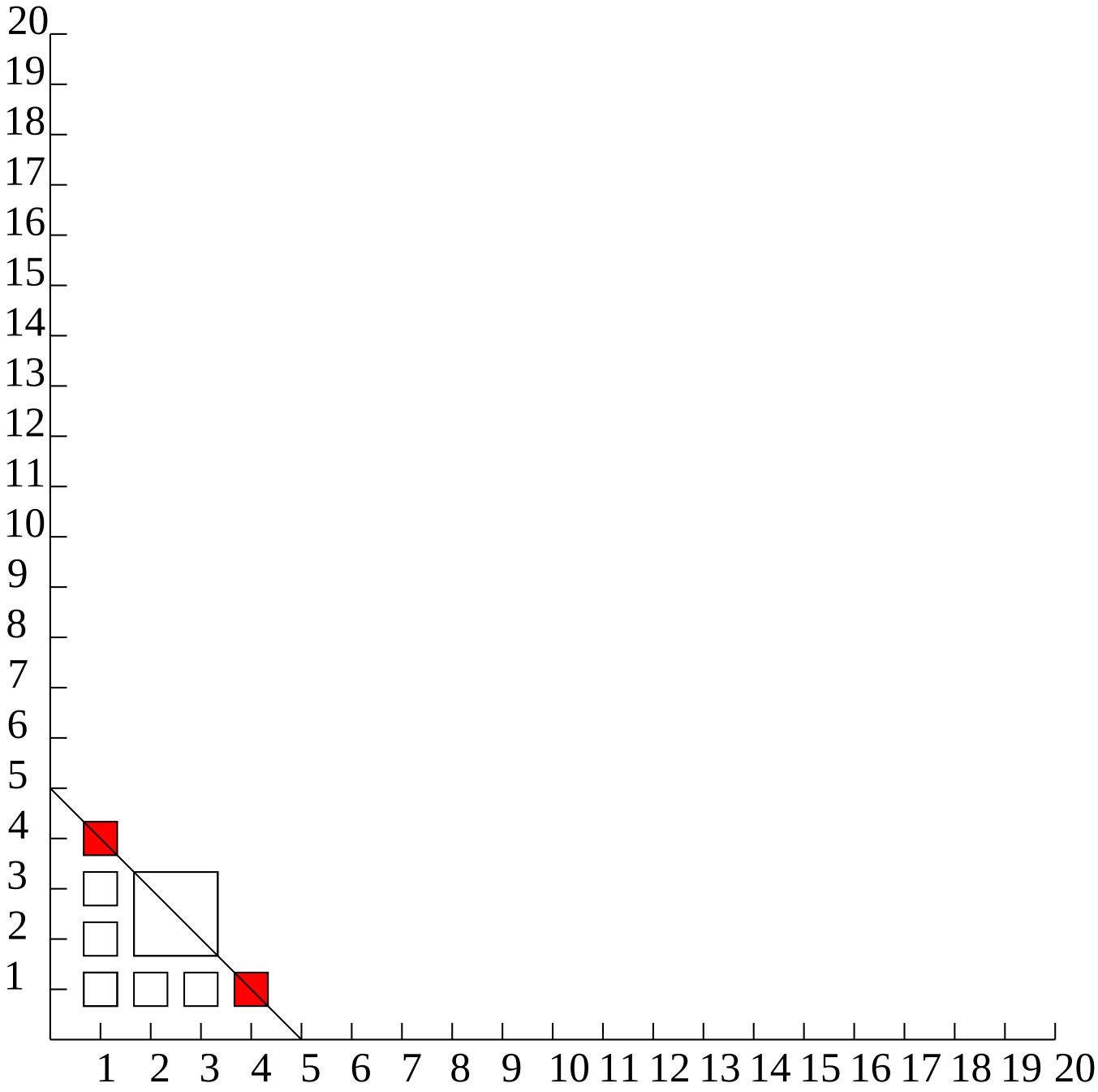
Any $t(n)$ offline algorithm for integer multiplication can be transformed to an $O(t(n) \log n)$ online algorithm for integer multiplication (for reasonable t).
The same holds true for Boolean convolution.

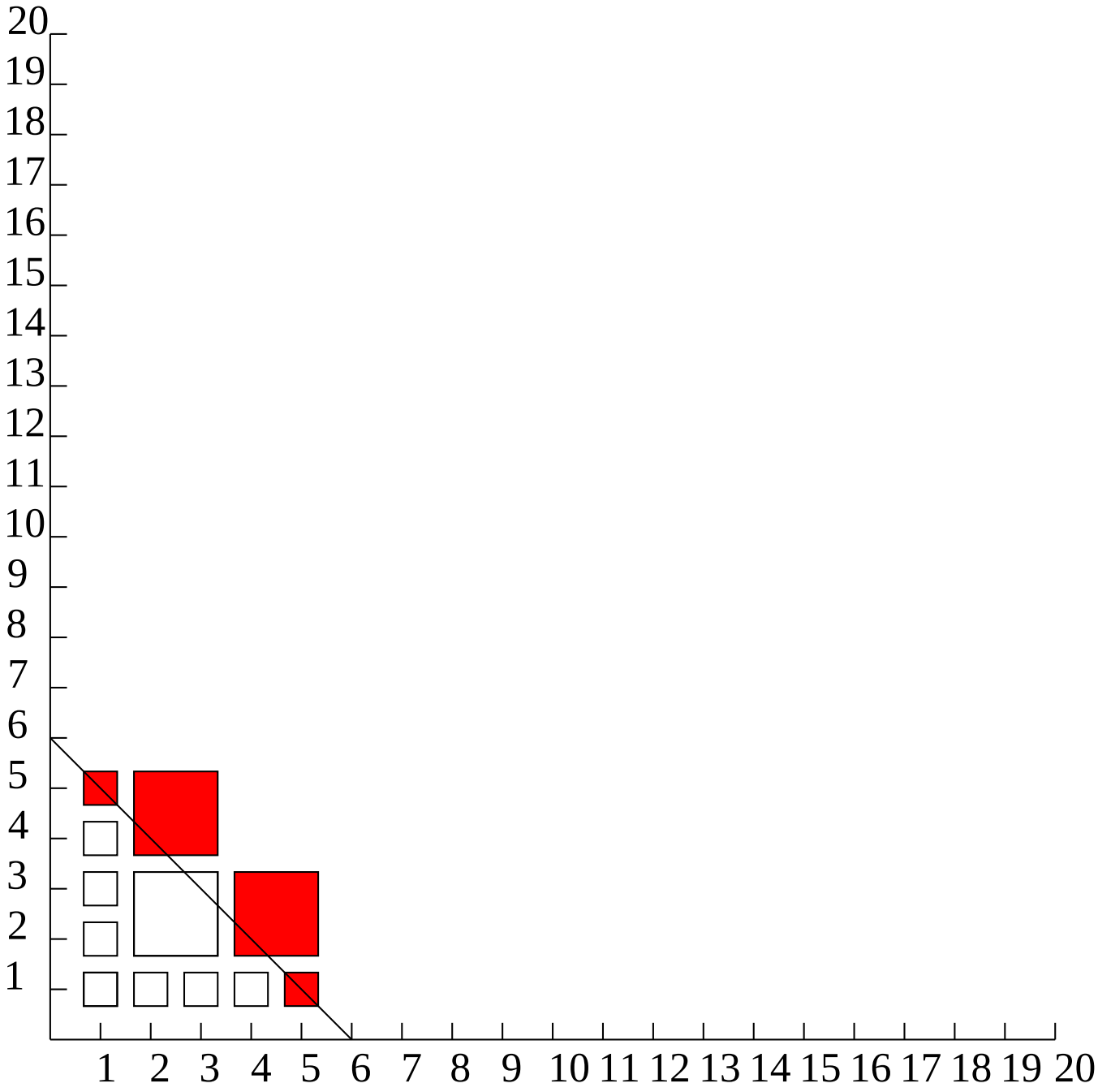

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

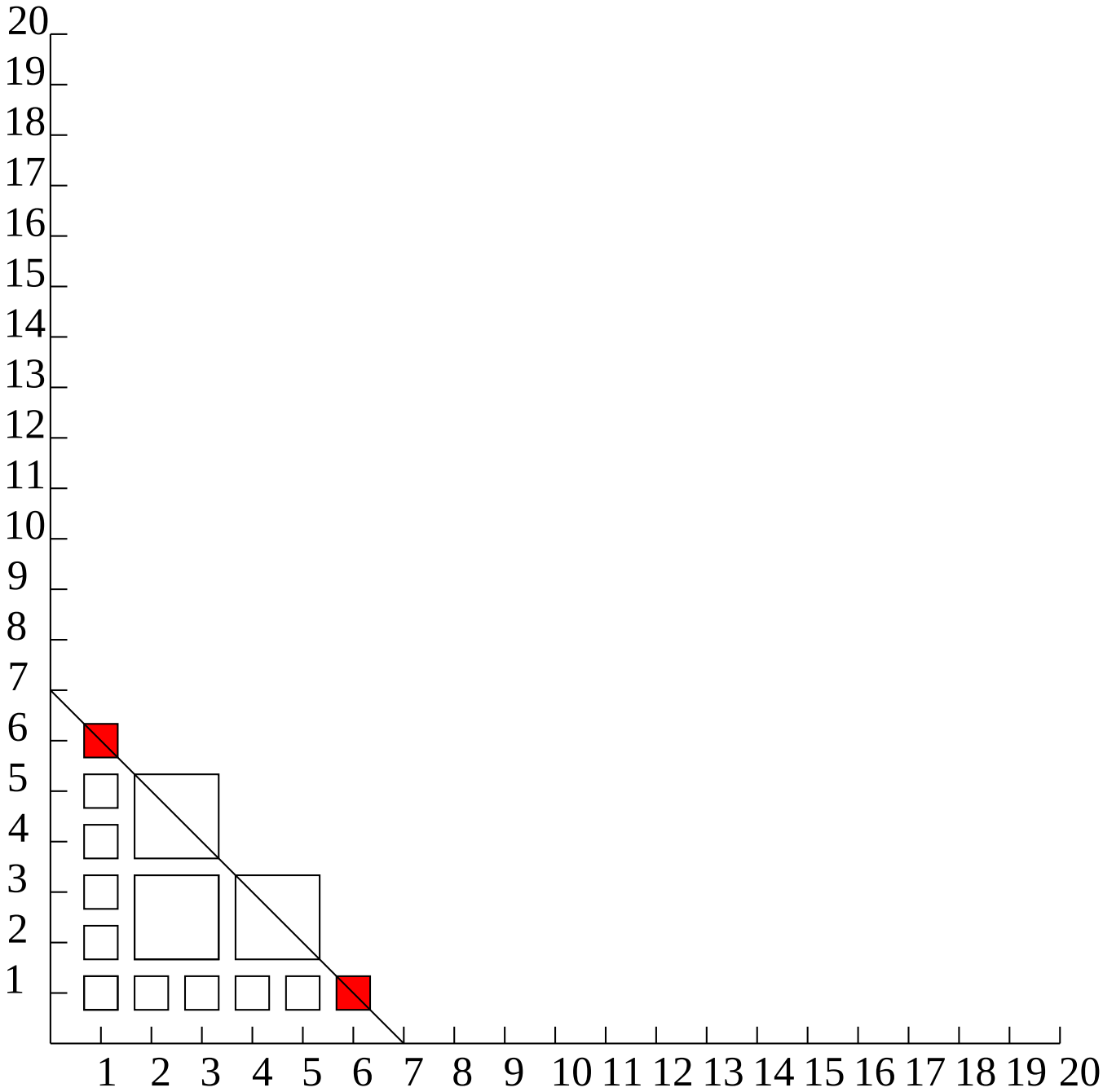

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

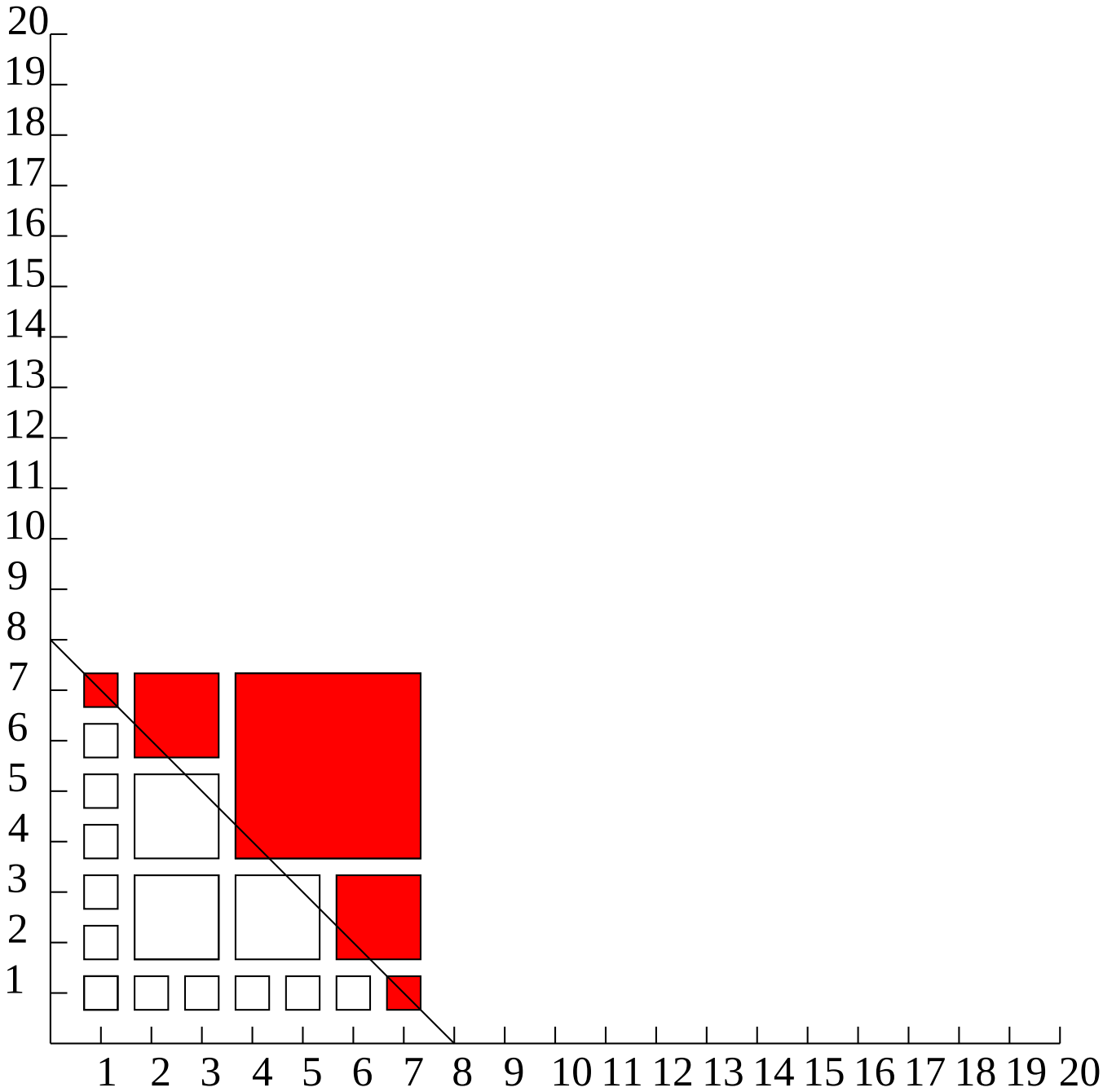

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

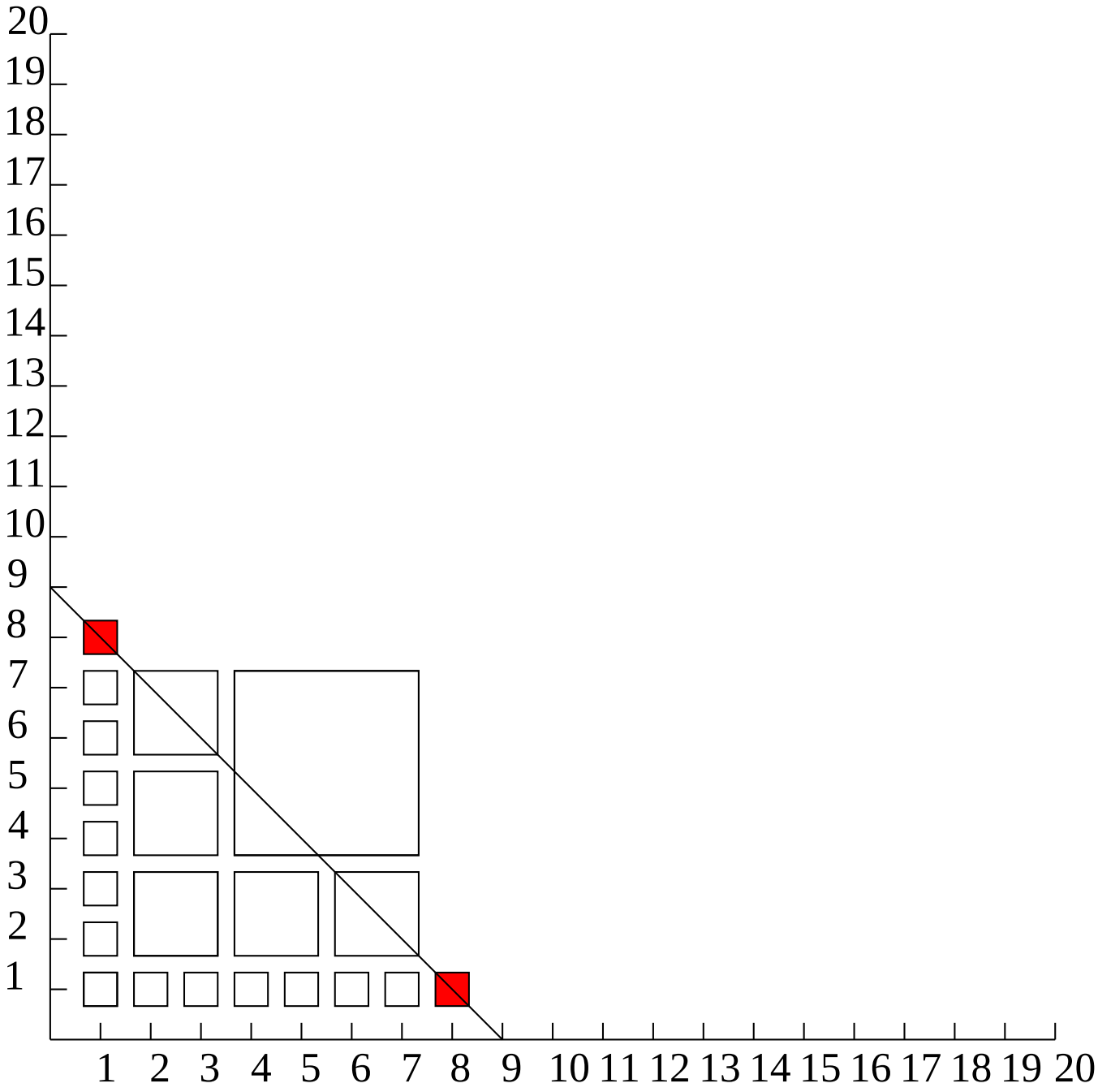

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

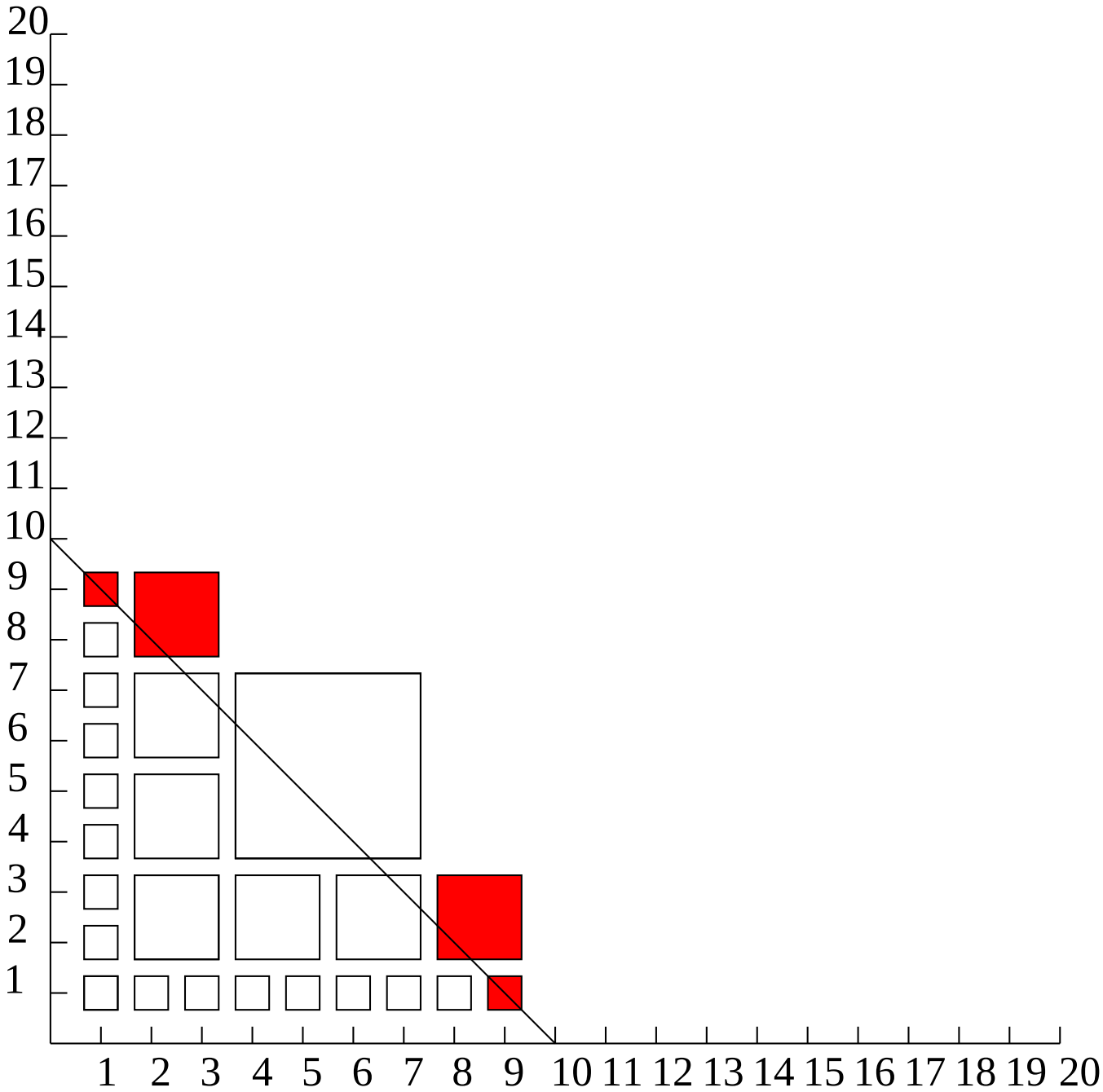

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

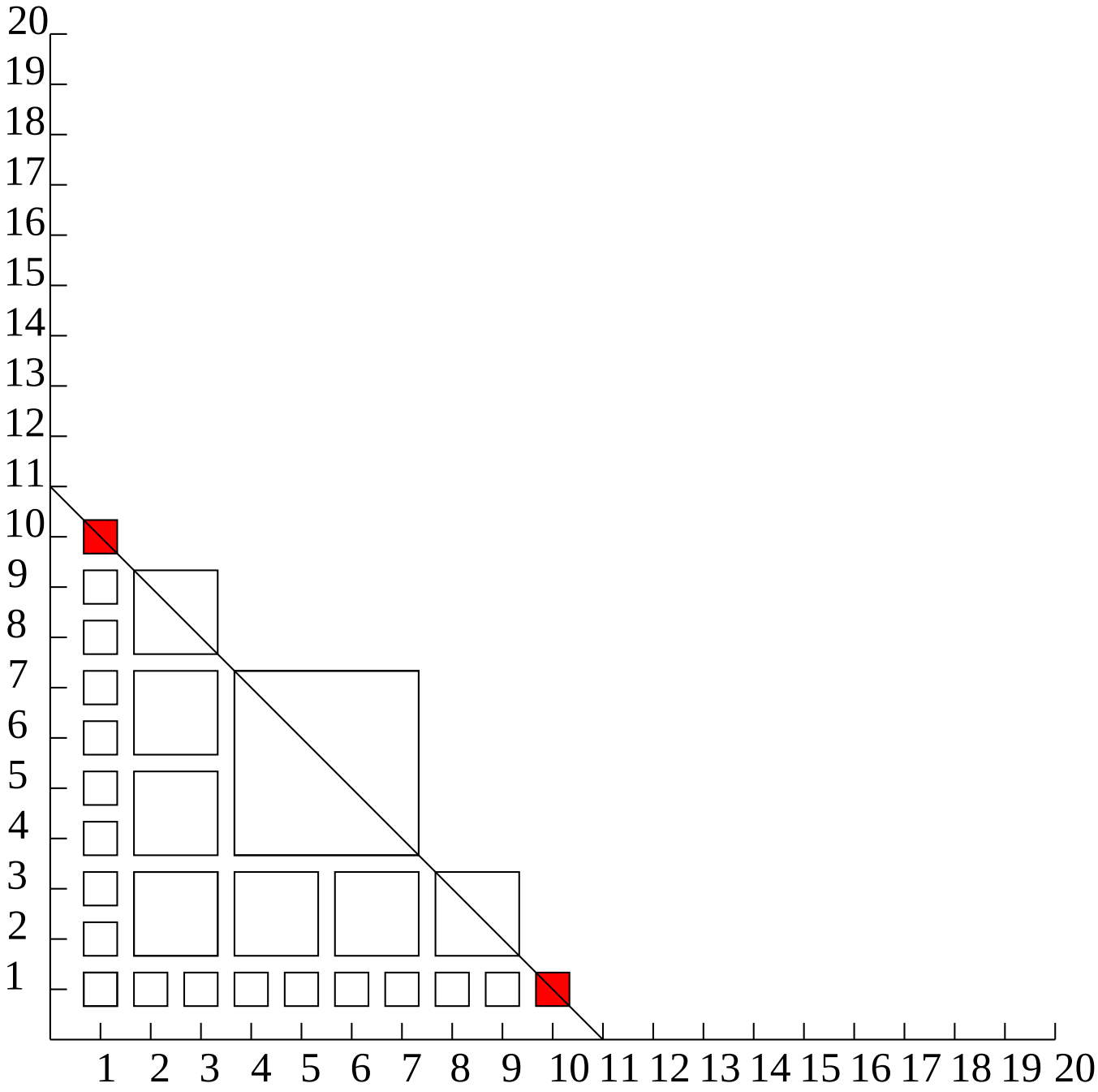

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

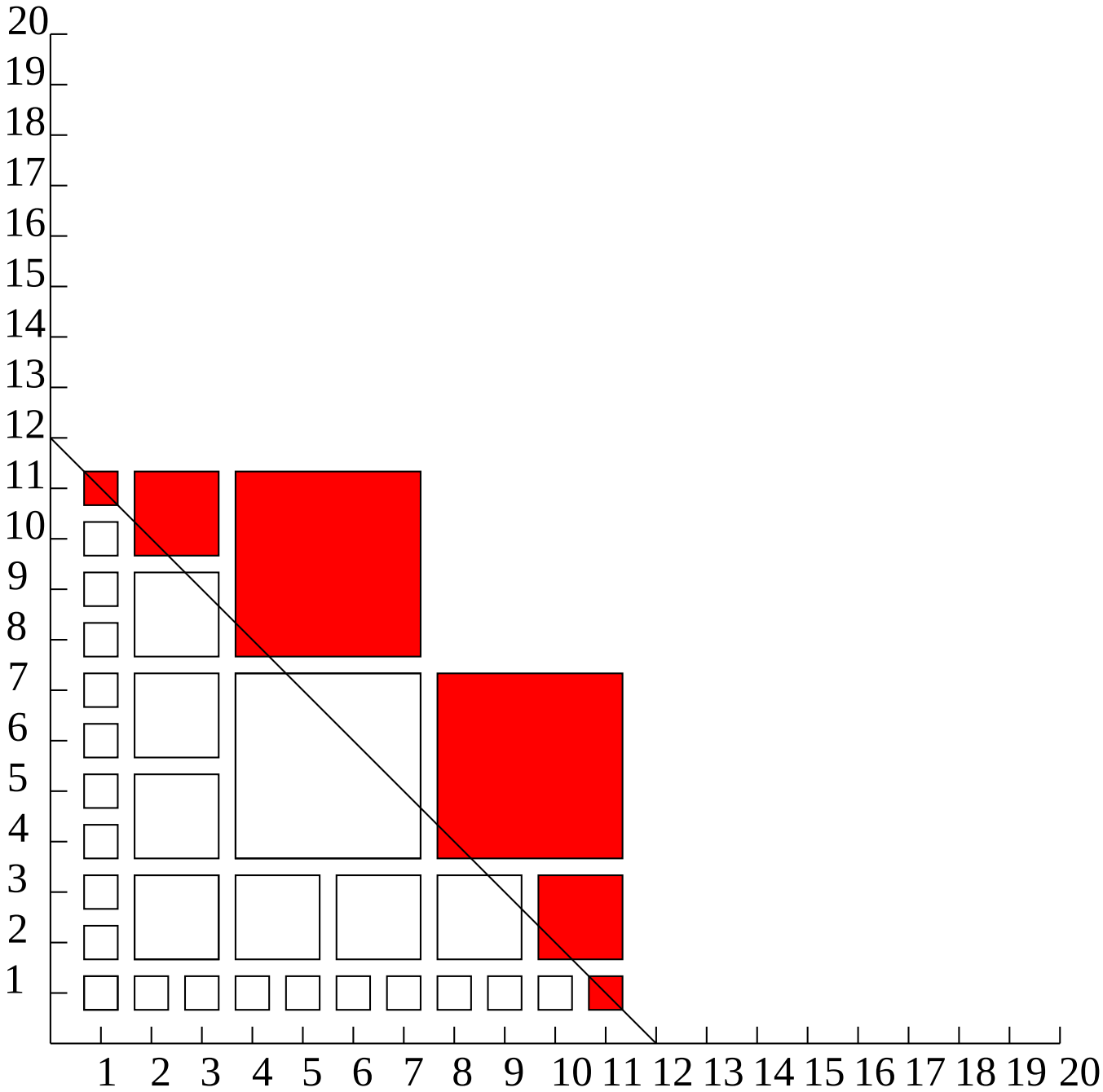

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

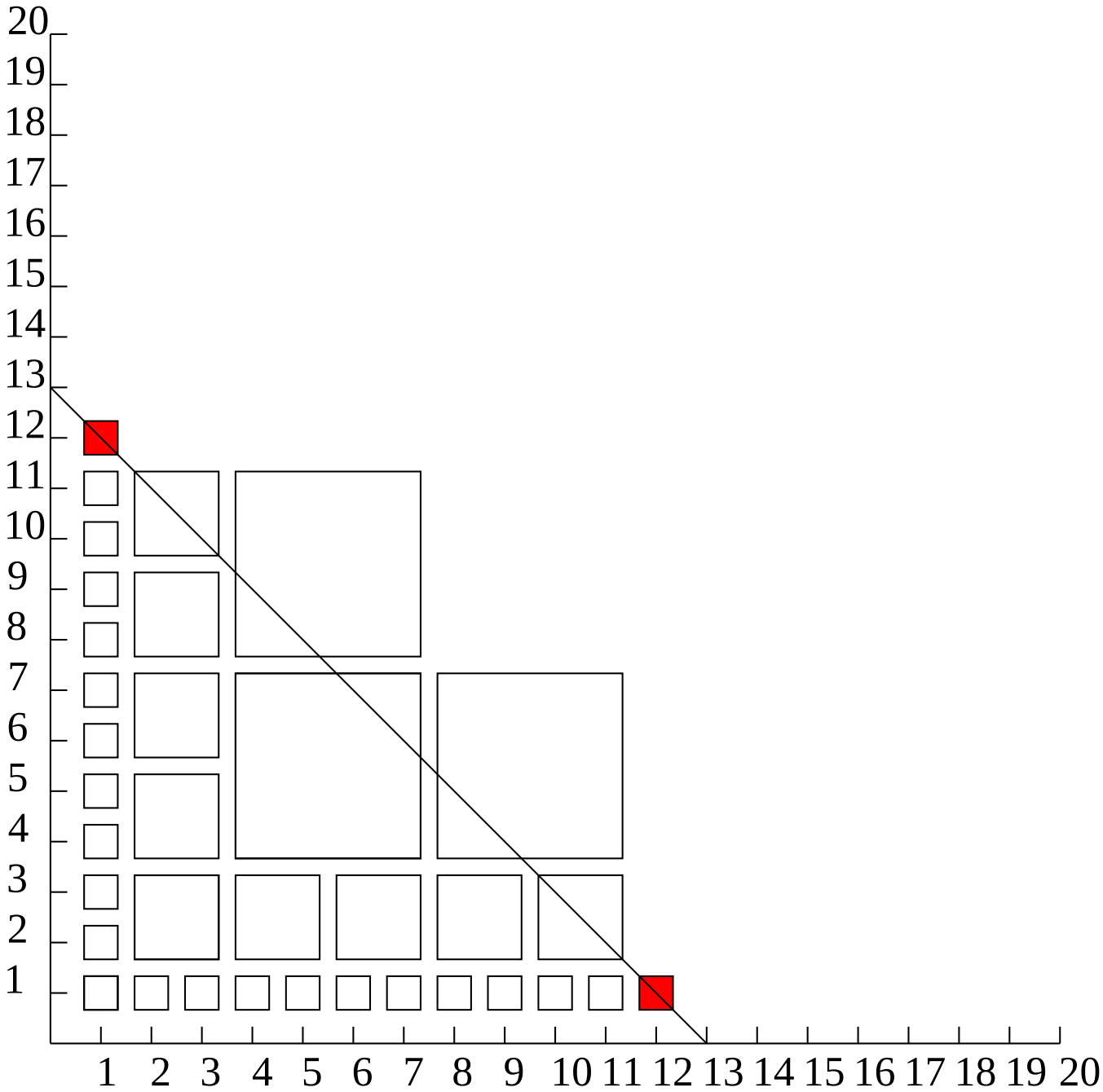

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

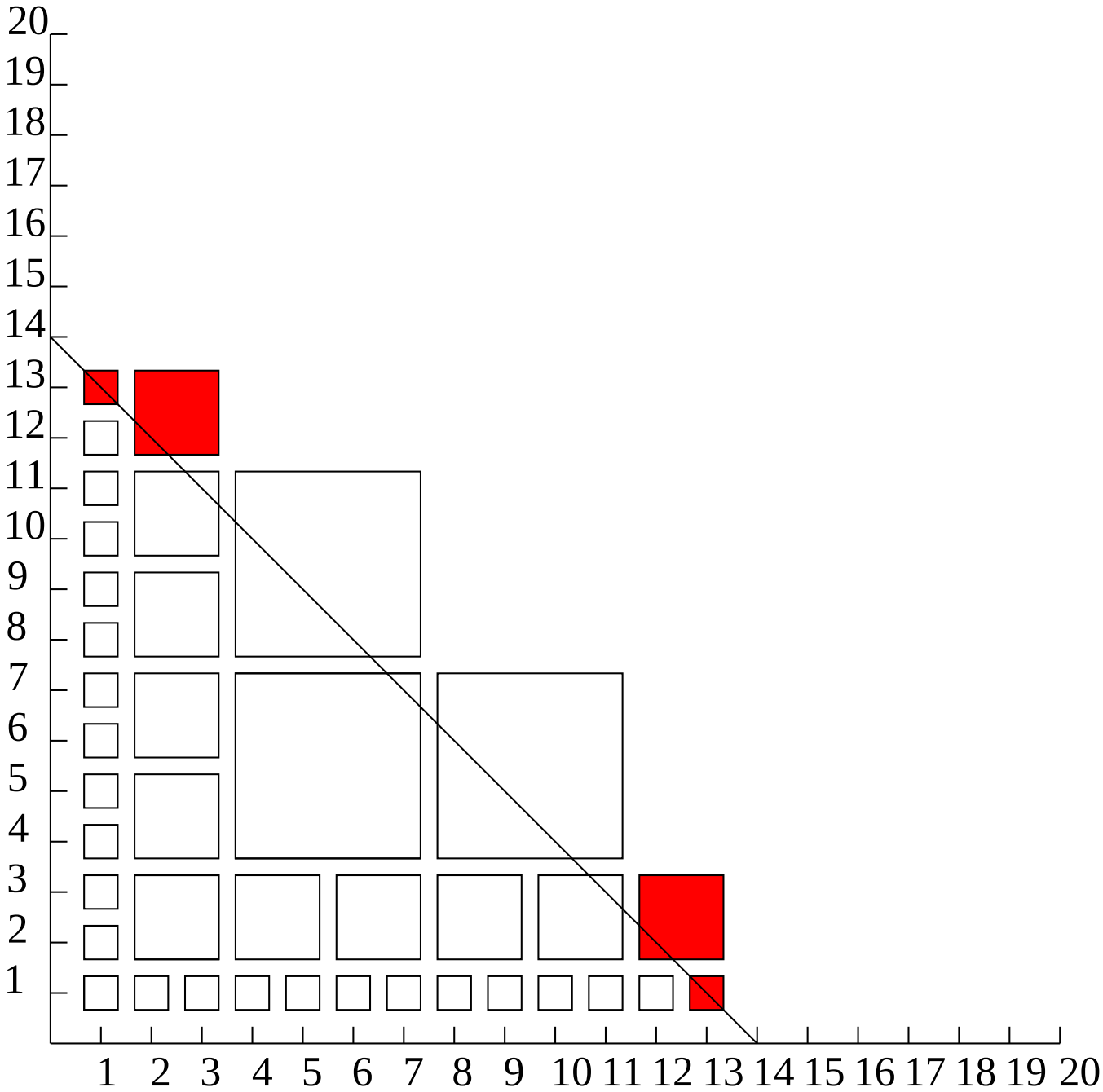

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

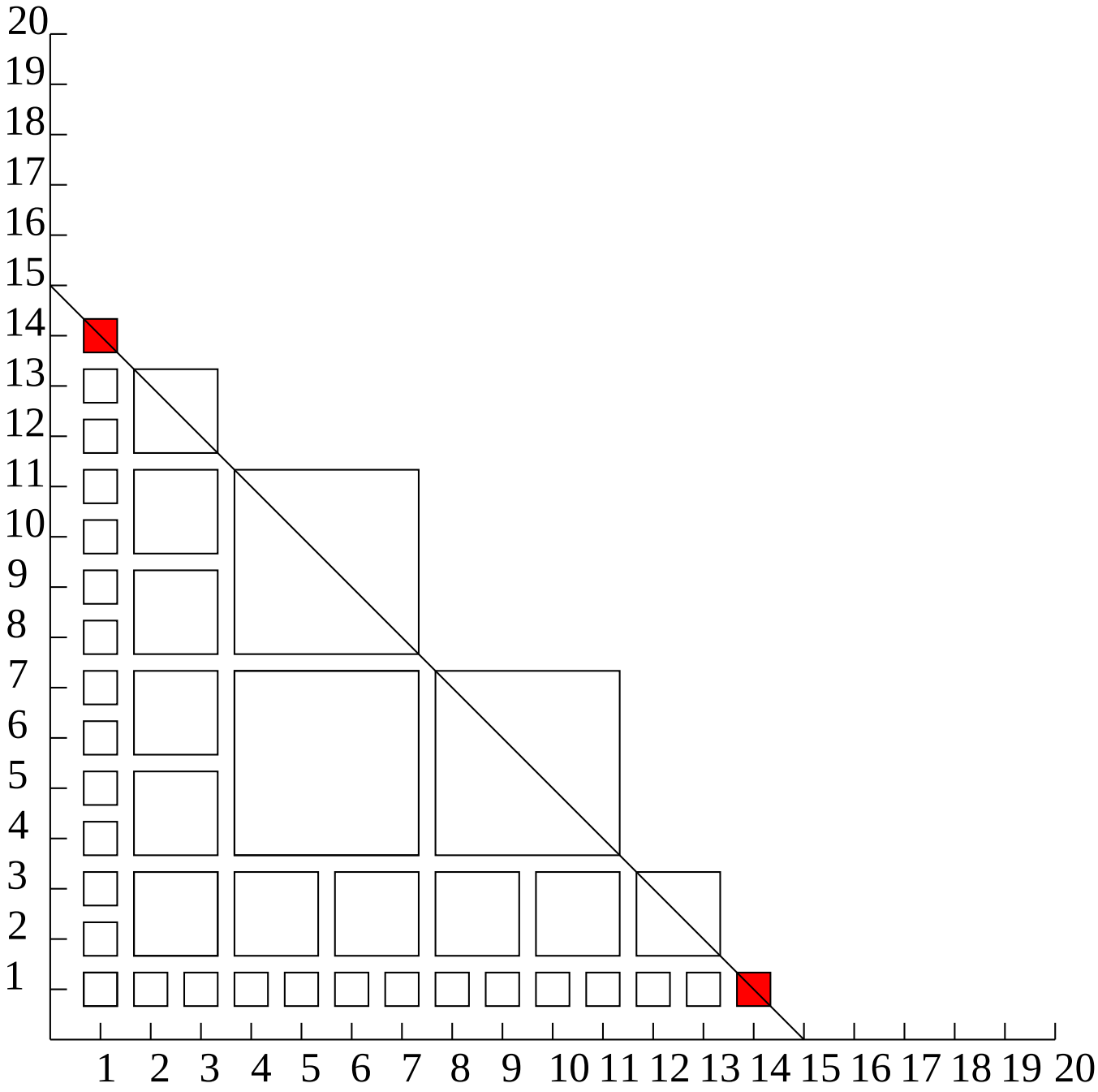

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

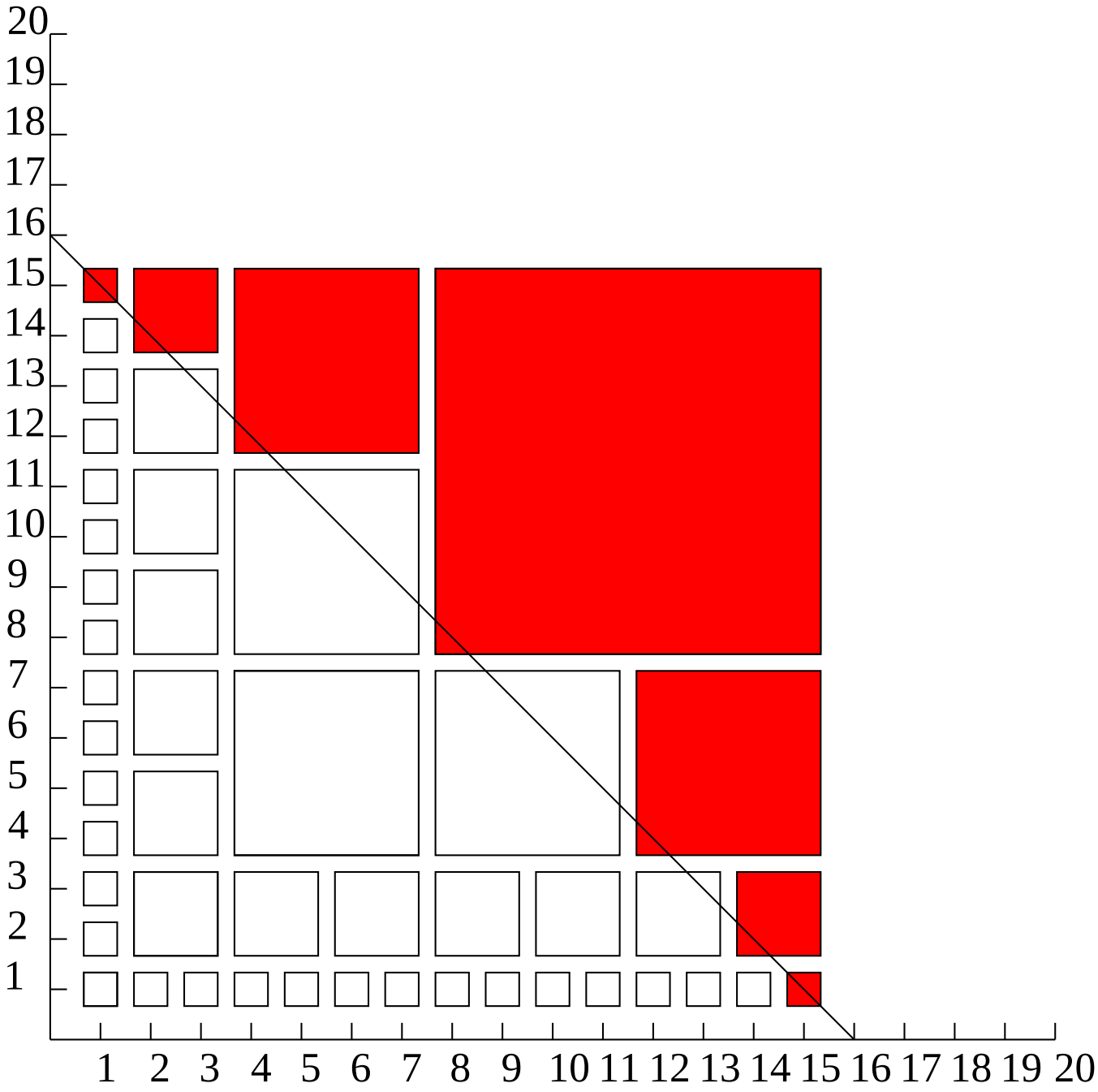

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

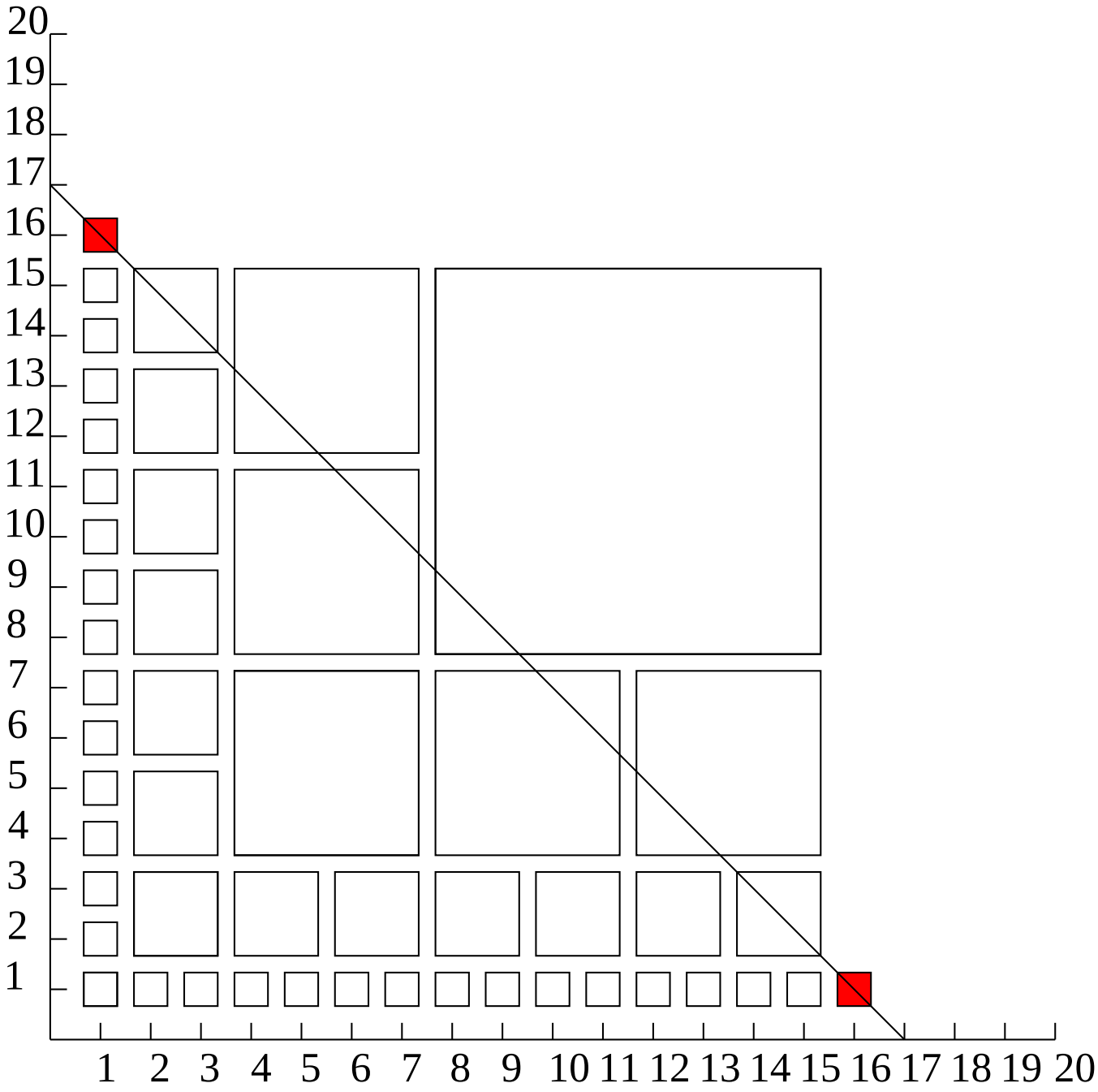

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

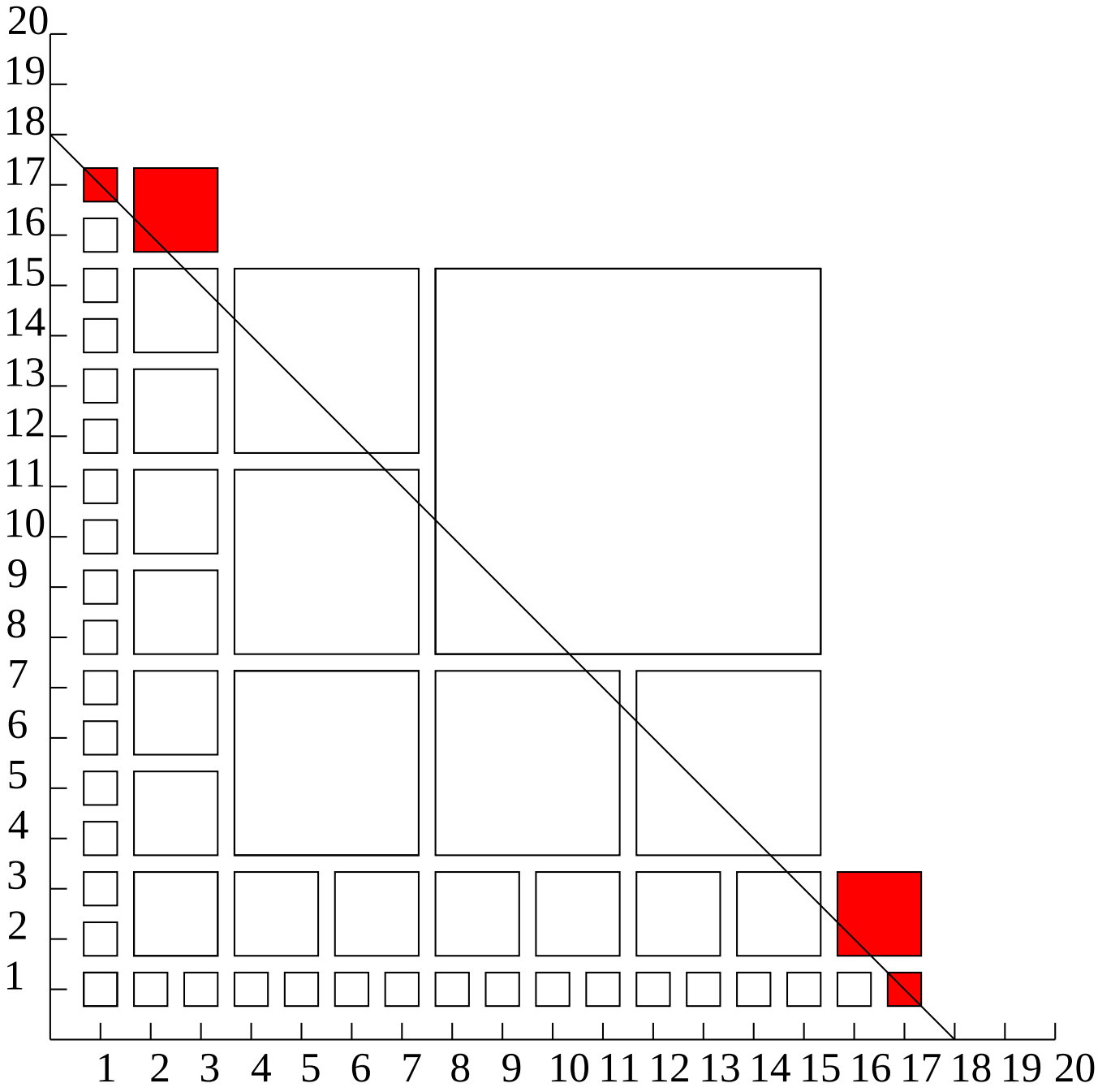

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

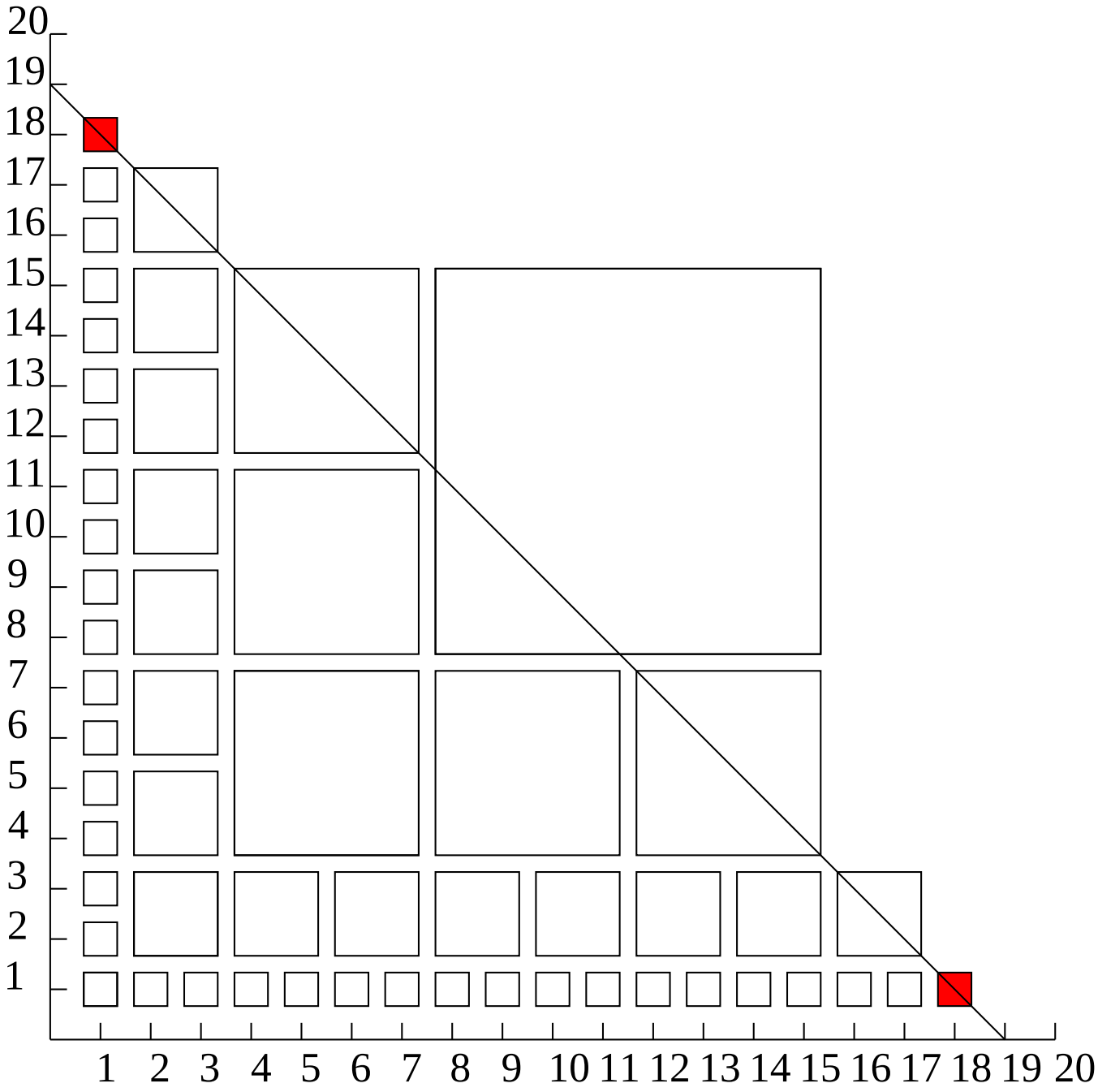

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

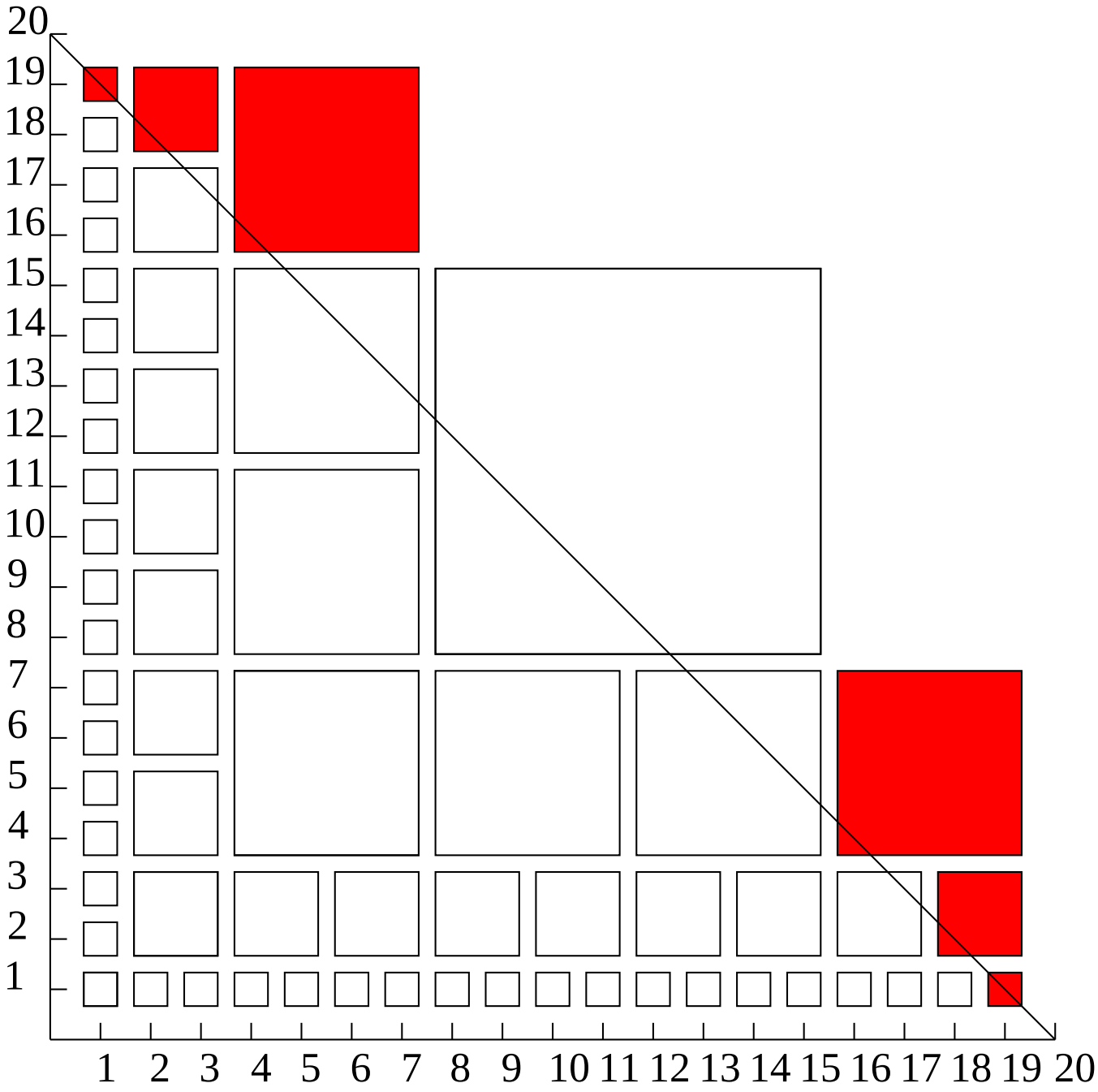

20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20


20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Fast Offline Boolean Convolution

Offline convolution of $(a_0, a_1, \ldots, a_{n-1})$ and $(b_0, b_1, \ldots, b_{n-1})$ by integer multiplication:

For $$a := \sum_{i=0}^{n-1} a_i \cdot 2^i \qquad b := \sum_{j=0}^{n-1} b_j \cdot 2^j$$

we get

$$a \cdot b = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i \cdot b_j \cdot 2^{(i+j)} \qquad = \sum_{k=0}^{2n-2} 2^k \qquad \cdot \sum_{i+j=k} a_i \cdot b_j.$$

Spread the bits (leave large enough gaps) to avoid interfering carries!

Fast Offline Boolean Convolution

Offline convolution of $(a_0, a_1, \ldots, a_{n-1})$ and $(b_0, b_1, \ldots, b_{n-1})$ by integer multiplication:

For $$a := \sum_{i=0}^{n-1} a_i \cdot 2^{i \log n} \qquad b := \sum_{j=0}^{n-1} b_j \cdot 2^{j \log n}$$

we get

$$a \cdot b = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i \cdot b_j \cdot 2^{(i+j) \log n} = \sum_{k=0}^{2n-2} 2^{k \log n} \cdot \sum_{i+j=k} a_i \cdot b_j.$$

Spread the bits (leave large enough gaps) to avoid interfering carries!

Results

Theorem

Using Fürer's algorithm, we get Boolean convolution in time $n \log^2 n 2^{O(\log^* n)} \leq O(n \log^3 n)$.
$\rightarrow$ *online Boolean convolution in* $n \log^3 n 2^{O(\log^* n)} \leq O(n \log^4 n)$.

Theorem

Membership for unary Boolean grammars in binary normal form in time $|G| \cdot n \log^3 n \cdot 2^{O(\log^* n)} \leq O(|G| \cdot n \log^4 n)$.

Algorithm is similar to Valiant's: Subdivisions and Boolean matrix multiplication compared to subdivisions and Boolean convolution.