Computing All Squares in Compressed Texts

Lesha Khvorost[†]

[†]Institute of Mathematics and Computer Science
Ural Federal University

26th September RuFiDiM 2012

String algorithms are deeply explored.

Two research directions are popular now:

- To improve classic string algorithms to process huge inputs (**I/O efficient algorithms**);

ATLAS project at CERN produces 70Tb/s of valuable information.

- To change data representation from the native to a more compact one (**algorithms on compressed representations**).

СКБ Контур

String algorithms are deeply explored.

Two research directions are popular now:

- To improve classic string algorithms to process huge inputs (**I/O efficient algorithms**);

ATLAS project at CERN produces 70Tb/s of valuable information.

- To change data representation from the native to a more compact one (**algorithms on compressed representations**).

String algorithms are deeply explored.

Two research directions are popular now:

- To improve classic string algorithms to process huge inputs (**I/O efficient algorithms**);

ATLAS project at CERN produces 70Tb/s of valuable information.

- To change data representation from the native to a more compact one (**algorithms on compressed representations**).

String algorithms are deeply explored.

Two research directions are popular now:

- To improve classic string algorithms to process huge inputs (**I/O efficient algorithms**);

ATLAS project at CERN produces 70Tb/s of valuable information.

- To change data representation from the native to a more compact one (**algorithms on compressed representations**).

Features of algorithms on compressed representations

- We save storage space and measure the complexity of the algorithm in terms of compressed input;
- By default we work with preprocessed data.

Features of algorithms on compressed representations

- We save storage space and measure the complexity of the algorithm in terms of compressed input;
- By default we work with preprocessed data.

Definition

A **straight-line program** (SLP) $\mathbb{S}$ is a sequence of assignments of the form:

$$\mathbb{S}_1 = expr_1,\ \mathbb{S}_2 = expr_2, \ldots, \mathbb{S}_n = expr_n,$$

where $\mathbb{S}_i$ are **rules** and $expr_i$ are expressions of the form:

- $expr_i \in \Sigma$ (**terminal** rules), or
- $expr_i = \mathbb{S}_\ell \cdot \mathbb{S}_r$ $(\ell, r < i)$ (**nonterminal** rules).

Example (the SLP that derives the 6th Fibonacci word)

$$\mathbb{F}_0 \to b,\ \mathbb{F}_1 \to a,\ \mathbb{F}_2 \to \mathbb{F}_1 \cdot \mathbb{F}_0,\ \mathbb{F}_3 \to \mathbb{F}_2 \cdot \mathbb{F}_1,$$
$$\mathbb{F}_4 \to \mathbb{F}_3 \cdot \mathbb{F}_2,\ \ \mathbb{F}_5 \to \mathbb{F}_4 \cdot \mathbb{F}_3,\ \ \mathbb{F}_6 \to \mathbb{F}_5 \cdot \mathbb{F}_4.$$

Definition

A **straight-line program** (SLP) $\mathbb{S}$ is a sequence of assignments of the form:

$$\mathbb{S}_1 = expr_1,\ \mathbb{S}_2 = expr_2, \ldots, \mathbb{S}_n = expr_n,$$

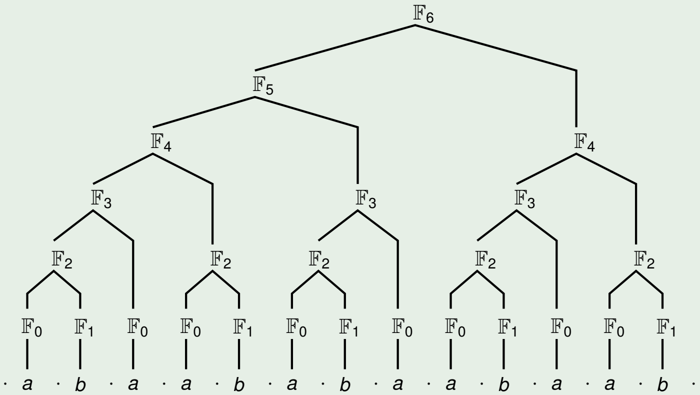
where $\mathbb{S}_i$ are **rules** and $expr_i$ are expressions of the form:

- $expr_i \in \Sigma$ (**terminal** rules), or
- $expr_i = \mathbb{S}_\ell \cdot \mathbb{S}_r$ $(\ell, r < i)$ (**nonterminal** rules).

Example (the SLP that derives the 6th Fibonacci word)

$$\mathbb{F}_0 \rightarrow b,\ \mathbb{F}_1 \rightarrow a,\ \mathbb{F}_2 \rightarrow \mathbb{F}_1 \cdot \mathbb{F}_0,\ \mathbb{F}_3 \rightarrow \mathbb{F}_2 \cdot \mathbb{F}_1,$$
$$\mathbb{F}_4 \rightarrow \mathbb{F}_3 \cdot \mathbb{F}_2,\ \ \mathbb{F}_5 \rightarrow \mathbb{F}_4 \cdot \mathbb{F}_3,\ \ \mathbb{F}_6 \rightarrow \mathbb{F}_5 \cdot \mathbb{F}_4.$$

Example (the parse of the SLP that derives the 6th Fibonacci word)



Tools

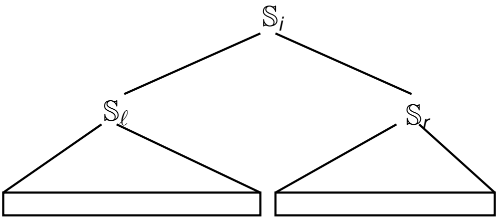
- Collecting some information from children (**Pattern Matching**, **Longest Common Substring**);

Sometimes count operations are extremely expensive (**Hamming Distance**, **Levenstein Distance**).

Are there other tools for SLPs processing?

Tools

- Collecting some information from children (**Pattern Matching**, **Longest Common Substring**);

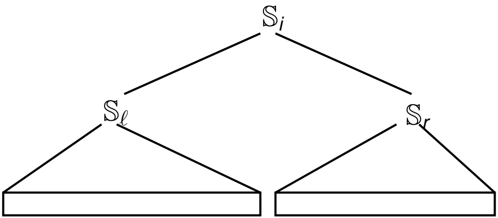


Sometimes count operations are extremely expensive (**Hamming Distance**, **Levenstein Distance**).

Are there other tools for SLPs processing?

Tools

- Collecting some information from children (**Pattern Matching**, **Longest Common Substring**);

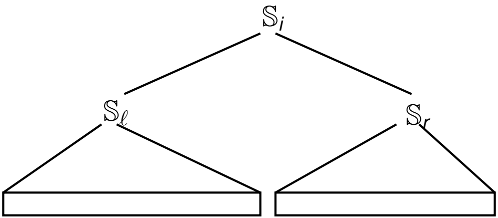


Sometimes count operations are extremely expensive (**Hamming Distance**, **Levenstein Distance**).

Are there other tools for SLPs processing?

Tools

- Collecting some information from children (**Pattern Matching**, **Longest Common Substring**);



Sometimes count operations are extremely expensive (**Hamming Distance**, **Levenstein Distance**).

Are there other tools for SLPs processing?

Definition

A string is called a square if it can be obtained by concatenating two copies of some string.
For example, $abc \cdot abc$. We say that abc is the root of the square.

CAS Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: the S-table that holds information about all squares in S.

Main difficulties:

- The text may contains exponentially many squares (in length of a compressed representation) (a^n contains $O(n^2)$ squares and has a $\log n$ representation);
- It is hard to collect information from children.

СКБ Контур

Definition

A string is called a square if it can be obtained by concatenating two copies of some string.
For example, $abc \cdot abc$. We say that abc is the root of the square.

CAS Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: the S-table that holds information about all squares in S.

Main difficulties:

- The text may contains exponentially many squares (in length of a compressed representation)
  (a^n contains $O(n^2)$ squares and has a $\log n$ representation);
- It is hard to collect information from children.

СКБ Контур

Definition

A string is called a square if it can be obtained by concatenating two copies of some string.
For example, $abc \cdot abc$. We say that abc is the root of the square.

CAS Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: the S-table that holds information about all squares in S.

Main difficulties:

- The text may contains exponentially many squares (in length of a compressed representation)
  (a^n contains $O(n^2)$ squares and has a $\log n$ representation);
- It is hard to collect information from children.

Definition

A string is called a square if it can be obtained by concatenating two copies of some string.
For example, $abc \cdot abc$. We say that abc is the root of the square.

CAS Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: the S-table that holds information about all squares in S.

Main difficulties:

- The text may contains exponentially many squares (in length of a compressed representation)
  (a^n contains $O(n^2)$ squares and has a $\log n$ representation);
- It is hard to collect information from children.

Definition

A string S is called a palindrome if $S = S^R$ (for example *ababa*).

Computing All Palindromes Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: a data structure that holds information about all palindromes in S.

Theorem [W. Matsubara, S. Inenaga et al] (2008)

Computing all palindromes problem can be solved in $O(|\mathbb{S}|^4)$ time with $O(|\mathbb{S}|^2)$ space.

СКБ Контур

Definition

A string S is called a palindrome if $S = S^R$ (for example *ababa*).

Computing All Palindromes Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: a data structure that holds information about all palindromes in S.

Theorem [W. Matsubara, S. Inenaga et al] (2008)

Computing all palindromes problem can be solved in $O(|\mathbb{S}|^4)$ time with $O(|\mathbb{S}|^2)$ space.

Definition

A string S is called a palindrome if $S = S^R$ (for example *ababa*).

Computing All Palindromes Problem

INPUT: an SLP $\mathbb{S}$ that derives the text S;
OUTPUT: a data structure that holds information about all palindromes in S.

Theorem [W. Matsubara, S. Inenaga et al] (2008)

Computing all palindromes problem can be solved in $O(|\mathbb{S}|^4)$ time with $O(|\mathbb{S}|^2)$ space.

Main result

Theorem

There is an algorithm that solves CAS using $O(|\mathbb{S}|^4 \cdot \log^2 |S|)$ time and $O(|\mathbb{S}| \cdot \max(|\mathbb{S}|, \log |S|))$ space.

S-table

The *S-table* is a rectangular table $S(\mathbb{S})$ of size $(\lfloor \log |S| \rfloor + 1) \times (|\mathbb{S}| + 1)$ that stores families of squares.
There are three types of families:

- Simple family, parameterized by $\{|x|, c_\ell, c_r\}$;
- Two types of complex families.

Example (Simple family of squares)

$S = bbabbabb$, family: $\{3, 3, 5\}$, squares: $\{(bba)^2, (bab)^2, (abb)^2\}$.

СКБ Контур

S-table

The *S-table* is a rectangular table $S(\mathbb{S})$ of size $(\lfloor \log |S| \rfloor + 1) \times (|\mathbb{S}| + 1)$ that stores families of squares.
There are three types of families:

- Simple family, parameterized by $\{|x|, c_\ell, c_r\}$;
- Two types of complex families.

Example (Simple family of squares)

$S = bbabbabb$, family: $\{3, 3, 5\}$, squares: $\{(bba)^2, (bab)^2, (abb)^2\}$.

СКБ Контур

S-table

The *S-table* is a rectangular table $S(\mathbb{S})$ of size $(\lfloor\log|S|\rfloor+1)\times(|\mathbb{S}|+1)$ that stores families of squares.
There are three types of families:

- Simple family, parameterized by $\{|x|, c_\ell, c_r\}$;
- Two types of complex families.

Example (Simple family of squares)

$S = bbabbabb$, family: $\{3,3,5\}$, squares: $\{(bba)^2,(bab)^2,(abb)^2\}$.

СКБ Контур

S-table

The *S-table* is a rectangular table $S(\mathbb{S})$ of size $(\lfloor \log |S| \rfloor + 1) \times (|\mathbb{S}| + 1)$ that stores families of squares.
There are three types of families:

- Simple family, parameterized by $\{|x|, c_\ell, c_r\}$;
- Two types of complex families.

Example (Simple family of squares)

$S = bbabbabb$, family: $\{3, 3, 5\}$, squares: $\{(bba)^2, (bab)^2, (abb)^2\}$.

СКБ Контур

Problems solvable by usage of S-table

For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not ($O(1)$);
- to find information about all squares of fixed length ($O(|\mathbb{S}|)$);
- to check whether on not a text S is square-free ($O(|\mathbb{S}| \log |S|)$);
- to compute total number of squares that are contained in S ($O(|\mathbb{S}| \log |S|)$);
- to find a maximal by length square that occurs in S ($O(|\mathbb{S}| \log |S|)$);
- to find information about all squares that starts from a fixed position i ($O(|\mathbb{S}| \log |S|)$).

Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

Контур

Problems solvable by usage of S-table

For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not ($O(1)$);
- to find information about all squares of fixed length ($O(|\mathbb{S}|)$);
- to check whether on not a text S is square-free ($O(|\mathbb{S}|\log|S|)$);
- to compute total number of squares that are contained in S ($O(|\mathbb{S}|\log|S|)$);
- to find a maximal by length square that occurs in S ($O(|\mathbb{S}|\log|S|)$);
- to find information about all squares that starts from a fixed position i ($O(|\mathbb{S}|\log|S|)$).

Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

Problems solvable by usage of S-table

For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not ($O(1)$);
- to find information about all squares of fixed length ($O(|\mathbb{S}|)$);
- to check whether on not a text S is square-free ($O(|\mathbb{S}| \log |S|)$);
- to compute total number of squares that are contained in S ($O(|\mathbb{S}| \log |S|)$);
- to find a maximal by length square that occurs in S ($O(|\mathbb{S}| \log |S|)$);
- to find information about all squares that starts from a fixed position i ($O(|\mathbb{S}| \log |S|)$).

Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

Problems solvable by usage of S-table

For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not ($O(1)$);
- to find information about all squares of fixed length ($O(|\mathbb{S}|)$);
- to check whether on not a text S is square-free ($O(|\mathbb{S}| \log |S|)$);
- to compute total number of squares that are contained in S ($O(|\mathbb{S}| \log |S|)$);
- to find a maximal by length square that occurs in S ($O(|\mathbb{S}| \log |S|)$);
- to find information about all squares that starts from a fixed position i ($O(|\mathbb{S}| \log |S|)$).

Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

Контур

Problems solvable by usage of S-table

For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not $(O(1))$;
- to find information about all squares of fixed length $(O(|\mathbb{S}|))$;
- to check whether on not a text S is square-free $(O(|\mathbb{S}|\log|S|))$;
- to compute total number of squares that are contained in S $(O(|\mathbb{S}|\log|S|))$;
- to find a maximal by length square that occurs in S $(O(|\mathbb{S}|\log|S|))$;
- to find information about all squares that starts from a fixed position i $(O(|\mathbb{S}|\log|S|))$.

Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

Контур

Problems solvable by usage of S-table

For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not ($O(1)$);
- to find information about all squares of fixed length ($O(|\mathbb{S}|)$);
- to check whether on not a text S is square-free ($O(|\mathbb{S}| \log |S|)$);
- to compute total number of squares that are contained in S ($O(|\mathbb{S}| \log |S|)$);
- to find a maximal by length square that occurs in S ($O(|\mathbb{S}| \log |S|)$);
- to find information about all squares that starts from a fixed position i ($O(|\mathbb{S}| \log |S|)$).

Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

Контур

Problems solvable by usage of S-table

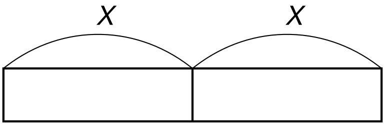
For a given $\mathbb{S}$ that derives a text S

- to check whether S is a square or not ($O(1)$);
- to find information about all squares of fixed length ($O(|\mathbb{S}|)$);
- to check whether on not a text S is square-free ($O(|\mathbb{S}|\log|S|)$);
- to compute total number of squares that are contained in S ($O(|\mathbb{S}|\log|S|)$);
- to find a maximal by length square that occurs in S ($O(|\mathbb{S}|\log|S|)$);
- to find information about all squares that starts from a fixed position i ($O(|\mathbb{S}|\log|S|)$).

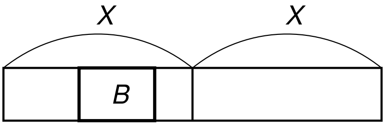
Open problem

For a given $\mathbb{S}$ that derives a text S to construct an SLP that derives concatenation of all squares that contain in S.

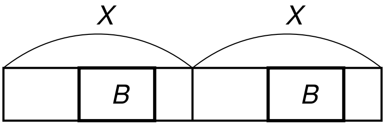
Detecting squares



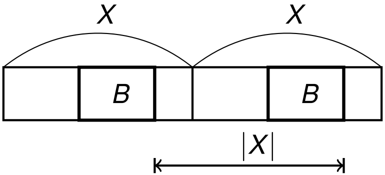
Detecting squares



Detecting squares



Detecting squares



Conclusion

- ~~We present the algorithm of 6th power that solves CAS Problem~~
- We present the new way of SLPs processing

Open problem

Is it possible to improve the algorithm's complexity?

Conclusion

- ~~We present the algorithm of 6th power that solves CAS Problem~~
- We present the new way of SLPs processing

Open problem

Is it possible to improve the algorithm's complexity?

Conclusion

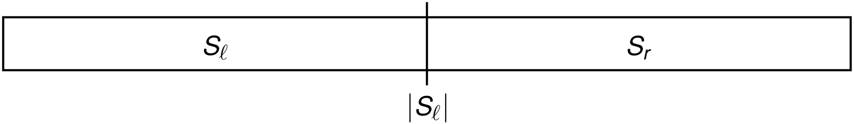
- ~~We present the algorithm of 6th power that solves CAS Problem~~
- We present the new way of SLPs processing

Open problem

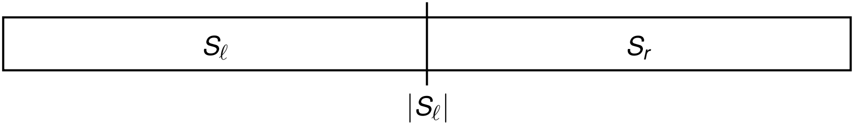
Is it possible to improve the algorithm's complexity?

СКБ Контур

Serching squares



Serching squares



Let $|S_i| = 2^n. Lengths\ of\ roots: \{1\}, \{2\}, \{3, 4\}, \ldots, \{2^{n-2}+1, \ldots, 2^{n-1}\}$.

Serching squares

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

$|S_\ell|$

Let $|S_i| = 2^n$.*Lengths of roots* :$\{1\}, \{2\}, \{3, 4\}, \ldots, \{2^{n-2}+1, \ldots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

СКБ Контур

Serching squares

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

$|S_\ell|$

Let $|S_i| = 2^n$.*Lengths of roots* :$\{1\}$, $\{2\}$, $\{3, 4\}$, $\ldots$, $\{2^{n-2}+1, \ldots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

СКБ Контур

Serching squares

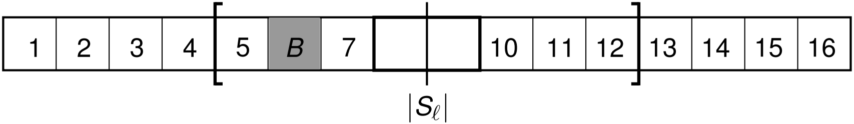
1	2	3	4	[5	B	7	8	9	10	11	12]	13	14	15	16

$|S_\ell|$

Let $|S_i| = 2^n$.*Lengths of roots* :{1}, {2}, {3, 4}, . . . , $\{2^{n-2}+1, \ldots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

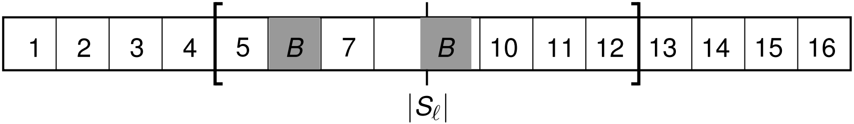
Serching squares



Let $|S_i| = 2^n$.*Lengths of roots* :{1}, {2}, {3, 4}, . . . , $\{2^{n-2}+1, \dots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

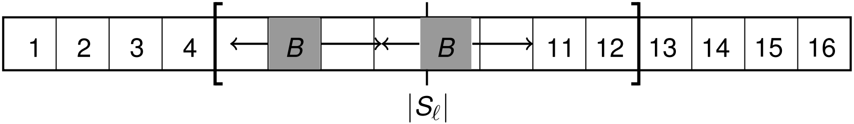
Serching squares



Let $|S_i| = 2^n$.*Lengths of roots* :{1}, {2}, {3, 4}, $\ldots$, $\{2^{n-2}+1, \ldots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

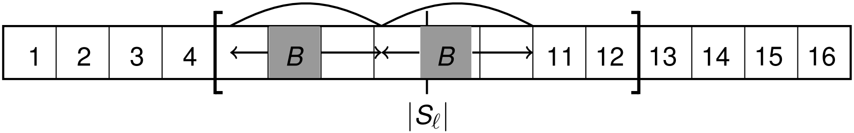
Serching squares



Let $|S_i| = 2^n$. *Lengths of roots* :$\{1\}, \{2\}, \{3, 4\}, \ldots, \{2^{n-2}+1, \ldots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

Serching squares

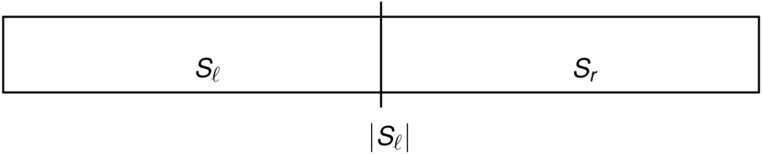


Let $|S_i| = 2^n$. *Lengths of roots* :{1}, {2}, {3, 4}, . . . , $\{2^{n-2}+1, \dots, 2^{n-1}\}$.

Let length of a block B be equal to 2^{n-3}.

Palindromes location

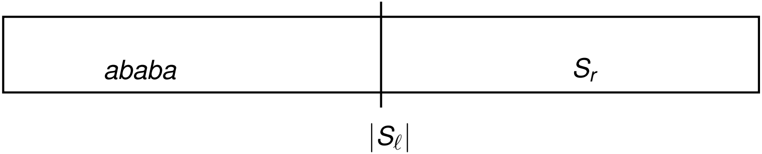
Let $\mathbb{S}$ be equal to $\mathbb{S}_\ell \cdot \mathbb{S}_r$



How we can use infromation about squares from children?

Palindromes location

Let $\mathbb{S}$ be equal to $\mathbb{S}_\ell \cdot \mathbb{S}_r$



How we can use infromation about squares from children?

Palindromes location

Let $\mathbb{S}$ be equal to $\mathbb{S}_\ell \cdot \mathbb{S}_r$

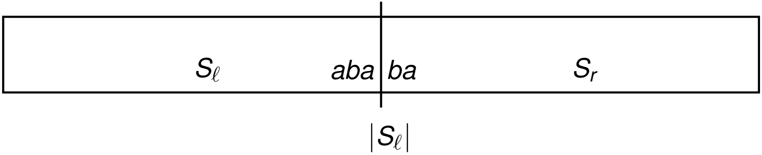
S_ℓ	*ababa*

$|S_\ell|$

How we can use infromation about squares from children?

Palindromes location

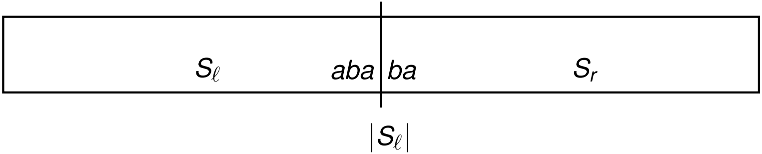
Let $\mathbb{S}$ be equal to $\mathbb{S}_\ell \cdot \mathbb{S}_r$



How we can use infromation about squares from children?

Palindromes location

Let $\mathbb{S}$ be equal to $\mathbb{S}_\ell \cdot \mathbb{S}_r$



How we can use infromation about squares from children?