Reconstructing words
Using queries on subwords or factors

Gwenaël Richomme and Matthieu Rosenfeld

December 5, 2023

Definitions

An **alphabet** is a finite set of **letters**
A **word** $w \in \mathcal{A}^*$ is a finite sequence of letters

Definitions

An **alphabet** is a finite set of **letters**
A **word** $w \in \mathcal{A}^*$ is a finite sequence of letters

For all letter a and integer j, $a^j = \underbrace{a \dots a}_{j \text{ times}}$

Definitions

An **alphabet** is a finite set of **letters**
A **word** $w \in \mathcal{A}^*$ is a finite sequence of letters

For all letter a and integer j, $a^j = \underbrace{a \dots a}_{j \text{ times}}$

Example: $a^5 = aaaaa$

Definitions

An **alphabet** is a finite set of **letters**
A **word** $w \in \mathcal{A}^*$ is a finite sequence of letters

For all letter a and integer j, $a^j = \underbrace{a \dots a}_{j \text{ times}}$

Example: $a^5 = aaaaa$

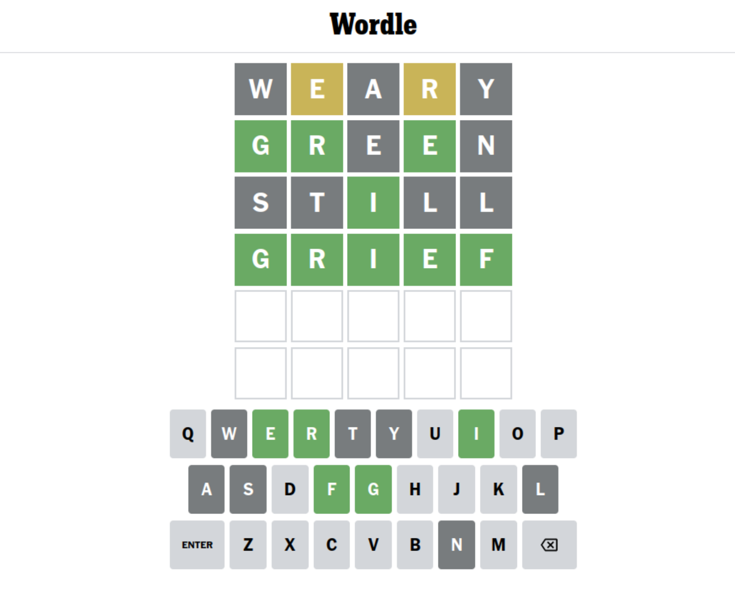
Reconstructing words over $\mathcal{A}$ with the set of queries $\mathcal{Q}$

$\mathbf{W} \in \mathcal{A}^*$ is only known by an oracle

Our task:

- Reconstruct $\mathbf{W}$ by asking queries from $\mathcal{Q}$ about $\mathbf{W}$ to the oracle
- Minimize the number of queries in terms of $n = |\mathbf{W}|$ and $k = |\mathcal{A}|$

A first example



The queries

u is a **subword** of $w \iff u$ is a subsequence of w

u is a **factor** of $w \iff u$ is a **contiguous** subsequence of w

The queries

u is a **subword** of w $\iff$ u is a subsequence of w

u is a **factor** of w $\iff$ u is a **contiguous** subsequence of w

Example: $\quad aba \quad \begin{array}{c}\text{is a subword of}\\ \text{is not a factor of}\end{array} \quad babbaa$

The queries

u is a **subword** of $w \iff u$ is a subsequence of w

u is a **factor** of $w \iff u$ is a **contiguous** subsequence of w

Example: aba $\begin{matrix}\text{is a subword of}\\ \text{is not a factor of}\end{matrix}$ $babbaa$

Four families of queries about W

$\Big\{$ "How many time does u occurs as a subword in **W** ?" $: u \in \mathcal{A}^*\Big\}$

The queries

u is a **subword** of $w \iff u$ is a subsequence of w

u is a **factor** of $w \iff u$ is a **contiguous** subsequence of w

Example: *aba* is a subword of / is not a factor of *babbaa*

Four families of queries about W

$\{$ "How many time does u occurs as a subword in **W** ?" $: u \in \mathcal{A}^*\}$

$\{$ "Is u a subword of **W** ?" $: u \in \mathcal{A}^*\}$

The queries

u is a **subword** of $w \iff u$ is a subsequence of w

u is a **factor** of $w \iff u$ is a **contiguous** subsequence of w

Example: $\quad aba \quad \begin{matrix}\text{is a subword of} \\ \text{is not a factor of}\end{matrix} \quad babbaa$

Four families of queries about W

$\{$ "How many time does u occurs as a subword in **W** ?" $: u \in \mathcal{A}^*\}$

$\{$ "Is u a subword of **W** ?" $: u \in \mathcal{A}^*\}$

$\{$ "Is u a factor of **W** ?" $: u \in \mathcal{A}^*\}$

The queries

u is a **subword** of $w \iff u$ is a subsequence of w

u is a **factor** of $w \iff u$ is a **contiguous** subsequence of w

Example: aba $\begin{array}{c}\text{is a subword of}\\ \text{is not a factor of}\end{array}$ $babbaa$

Four families of queries about W

$$\begin{array}{ll}
\{\ \text{“How many time does } u \text{ occurs as a subword in } \mathbf{W}\text{ ?”} & : u \in \mathcal{A}^*\} \\
\{\ \text{“Is } u \text{ a subword of } \mathbf{W}\text{ ?”} & : u \in \mathcal{A}^*\} \\
\{\ \text{“Is } u \text{ a factor of } \mathbf{W}\text{ ?”} & : u \in \mathcal{A}^*\} \\
\{\ \text{“How many time does } u \text{ occurs as a factor in } \mathbf{W}\text{ ?”} & : u \in \mathcal{A}^*\}
\end{array}$$

Factor queries

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W :

- Is 00 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 00

- Is 00 a factor of W ? Yes

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 00

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W :   00

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 00

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W :   000

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes

A first example

The unknown word $W \in \{0, 1\}^*$ has length 9.

We try to build a factor of W : 000

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W :   0001

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 0001

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : $0001\underline{0}$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : $\quad 0001\underline{0}$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W :   000101

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 000101

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 0001010̲

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : $\quad 000101\underline{0}$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 00010<u>100</u>

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No

A first example

The unknown word $W \in \{0, 1\}^*$ has length 9.

We try to build a factor of W : 00010100

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 000101$\underline{000}$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 000101$\underline{000}$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : $000101\underline{000}|$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : $000101\underline{000}|$

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ?

A first example

The unknown word $W \in \{0, 1\}^*$ has length 9.

We try to build a factor of W : 0001010$\underline{00}$|

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ? Yes

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 0001010$\underline{00}$|

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ? Yes
- Is 00010100 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 10001010 |

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ? Yes
- Is 00010100 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 10001010 |

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ? Yes
- Is 00010100 a factor of W ? No
- Is 110001010 a factor of W ?

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 010001010 |

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ? Yes
- Is 00010100 a factor of W ? No
- Is 110001010 a factor of W ? No

A first example

The unknown word $W \in \{0,1\}^*$ has length 9.

We try to build a factor of W : 010001010 |

- Is 00 a factor of W ? Yes
- Is 0000 a factor of W ? No
- Is 000 a factor of W ? Yes
- Is 0001 a factor of W ? Yes
- Is 00011 a factor of W ? No
- Is 000101 a factor of W ? Yes
- Is 0001011 a factor of W ? No
- Is 00010101 a factor of W ? No
- Is 000101001 a factor of W ? No
- Is 0001010001 a factor of W ? No
- Is 0001010 a factor of W ? Yes
- Is 00010100 a factor of W ? No
- Is 110001010 a factor of W ? No

The corresponding strategy

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Binary search on i

The corresponding strategy

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Binary search on i
2. Try to extend it on the right:
   Try adding 1, if it fails add 0
   If it fails $k+1$ consecutive times, find the position of failure

The corresponding strategy

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Binary search on i
2. Try to extend it on the right:
   Try adding 1, if it fails add 0
   If it fails $k+1$ consecutive times, find the position of failure
3. Extend on the left until reaching the desired size:
   Try adding 1, if it fails add 0

The corresponding strategy

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Binary search on i
2. Try to extend it on the right:
   Try adding 1, if it fails add 0
   If it fails $k+1$ consecutive times, find the position of failure
3. Extend on the left until reaching the desired size:
   Try adding 1, if it fails add 0

1. in $\lceil \log_2 n \rceil + O(1)$ queries

The corresponding strategy

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Binary search on i
2. Try to extend it on the right:
   Try adding 1, if it fails add 0
   If it fails $k+1$ consecutive times, find the position of failure
3. Extend on the left until reaching the desired size:
   Try adding 1, if it fails add 0

1. in $\lceil \log_2 n \rceil + O(1)$ queries
2.+3. in $(n-k)+k+O(1)$ queries

The corresponding strategy

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Binary search on i
2. Try to extend it on the right:
   Try adding 1, if it fails add 0
   If it fails $k+1$ consecutive times, find the position of failure
3. Extend on the left until reaching the desired size:
   Try adding 1, if it fails add 0

$$\left.\begin{array}{l}\text{1. in } \lceil \log_2 n \rceil + O(1) \text{ queries} \\ \text{2.+3. in } (n-k) + k + O(1) \text{ queries}\end{array}\right\} \leq n + \lceil \log_2 n \rceil + O(1)$$

The corresponding strategy

If $\mathbf{W}$ taken uniformly at random (Iwama,Teruyama, and Tsuyama, 2018):

The algorithm (Skiena and Sundaram, 1995)

1. Find k the largest i such that 0^i is factor of $\mathbf{W}$:
   Greedy search on i starting at $i = \log_2 n$
2. Try to extend it on the right:
   Try adding 1, if it fails add 0
   If it fails $k+1$ consecutive times, find the position of failure
3. Extend on the left until reaching the desired size:
   Try adding 1, if it fails add 0

$$\left.\begin{array}{l} \text{1. in expected } O(1) \text{ queries} \\ \text{2.+3. in } (n-k)+k+O(1) \text{ queries} \end{array}\right\} \leq n + O(1)$$

Lower bound on the number of $\exists$-factor queries

Lemma

For any $n \geq 1$, there is no algorithm that reconstructs any $\mathbf{W} \in \{0,1\}^n$ in less than n $\exists$-factor queries.

Lower bound on the number of $\exists$-factor queries

Lemma

For any $n \geq 1$, there is no algorithm that reconstructs any $\mathbf{W} \in \{0,1\}^n$ in less than n $\exists$-factor queries.

Every query has 2 possible outputs

Lower bound on the number of $\exists$-factor queries

Lemma

For any $n \geq 1$, there is no algorithm that reconstructs any $\mathbf{W} \in \{0,1\}^n$ in less than n $\exists$-factor queries.

Every query has 2 possible outputs

If the number of queries is bounded by $n-1$, then the number of possible outputs of the algorithm is bounded by 2^{n-1}

Lower bound on the number of $\exists$-factor queries

Lemma

For any $n \geq 1$, there is no algorithm that reconstructs any $\mathbf{W} \in \{0,1\}^n$ in less than n $\exists$-factor queries.

Every query has 2 possible outputs

If the number of queries is bounded by $n-1$, then the number of possible outputs of the algorithm is bounded by 2^{n-1}

This is not possible since $|\{0,1\}^n| = 2^n$ $\square$

The results

Number of ∃-factor queries needed to guess an unknown binary word $\mathbf{W} \in \{0,1\}^n$ when n is known:

$\geq n$	simple argument

The results

Number of $\exists$-factor queries needed to guess an unknown binary word $\mathbf{W} \in \{0,1\}^n$ when n is known:

$\geq n$	simple argument
$\leq n + \lceil \log n \rceil + O(1)$	Skiena and Sundaram, 1995

The results

Number of $\exists$-factor queries needed to guess an unknown binary word $\mathbf{W} \in \{0,1\}^n$ when n is known:

$\geq n$	simple argument
$\leq n + \lceil \log n \rceil + O(1)$	Skiena and Sundaram, 1995
In average $\leq n + O(1)$	Iwama,Teruyama, and Tsuyama, 2018

The results

Number of $\exists$-factor queries needed to guess an unknown binary word $\mathbf{W} \in \{0,1\}^n$ when n is known:

$\geq n$	simple argument
$\leq n + \lceil \log n \rceil + O(1)$	Skiena and Sundaram, 1995
In average $\leq n + O(1)$	Iwama,Teruyama, and Tsuyama, 2018
$\leq n + \frac{\lceil \log n \rceil}{2} + O(1)$	Richomme and Rosenfeld, 2022

$\exists$-subword queries

$\exists$-subword: reconstructing a word from its subwords

u is a **subword** of w $\iff$ u is a subsequence of w $\iff$ $u \sqsubseteq w$

$\exists$-subword: reconstructing a word from its subwords

u is a **subword** of $w \iff u$ is a subsequence of $w \iff u \sqsubseteq w$

Example: $010 \sqsubseteq 001101$

$\exists$-subword: reconstructing a word from its subwords

u is a **subword** of $w \iff u$ is a subsequence of $w \iff u \sqsubseteq w$
Example: $010 \sqsubseteq 001101$

Giwen an unknown word $\mathbf{W}$ over $\mathcal{A}$, you can ask queries of the form

$$\boxed{u \sqsubseteq \mathbf{W}?}$$

How many queries do you need to reconstruct $\mathbf{W}$?

$\exists$-subword: reconstructing a word from its subwords

u is a **subword** of $w \iff u$ is a subsequence of $w \iff u \sqsubseteq w$
Example: $010 \sqsubseteq 001101$

Giwen an unknown word $\mathbf{W}$ over $\mathcal{A}$, you can ask queries of the form

$$\boxed{u \sqsubseteq \mathbf{W}?}$$

How many queries do you need to reconstruct $\mathbf{W}$?

Lemma

For any $n \geq 1$, there is no algorithm that reconstruct any $\mathbf{W} \in \{0,1\}^n$ in less than n queries.

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)

Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)

Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

Useful tool

For all i and j,
$$0^i1^j0^{t-i} \sqsubseteq \mathbf{W} \iff j \leq s_i$$

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)

Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

Useful tool

For all i and j,
$$0^i1^j0^{t-i} \sqsubseteq \mathbf{W} \iff j \leq s_i$$

For each i, we find s_i ...

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)
Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

Useful tool

For all i and j, $$0^i1^j0^{t-i} \sqsubseteq \mathbf{W} \iff j \leq s_i$$

For each i, we find s_i **greedily**:

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)
Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

Useful tool

For all i and j, $$0^i1^j0^{t-i} \sqsubseteq \mathbf{W} \iff j \leq s_i$$

For each i, we find s_i **greedily**:

$\boxed{0^i1^j0^{t-i} \sqsubseteq \mathbf{W}}$ for $j = 1, 2\ldots$ until it says no ($s_i + 1$ queries for each s_i)

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)
Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

Useful tool

For all i and j, $$0^i1^j0^{t-i} \sqsubseteq \mathbf{W} \iff j \leq s_i$$

For each i, we find s_i **greedily**:

$\boxed{0^i1^j0^{t-i} \sqsubseteq \mathbf{W}}$ for $j = 1, 2\ldots$ until it says no ($s_i + 1$ queries for each s_i)

Total number of queries:

$$O(\log_2 n) + \sum_i (s_i + 1) = O(\log_2 n) + n + 1$$

$\exists$-subword on binary alphabet - upper bound

Find t the number of 0 in $\mathbf{W}$ (binary search in $O(\log_2 n)$)
Then $\mathbf{W} = 1^{s_0}01^{s_1}0\ldots01^{s_t}$ for some integers $s_0, \ldots, s_t$.

Useful tool

For all i and j, $$0^i1^j0^{t-i} \sqsubseteq \mathbf{W} \iff j \leq s_i$$

For each i, we find s_i **greedily**:

$\boxed{0^i1^j0^{t-i} \sqsubseteq \mathbf{W}}$ for $j = 1, 2\ldots$ until it says no ($s_i + 1$ queries for each s_i)

Total number of queries:

$$O(\log_2 n) + \sum_i (s_i + 1) = O(\log_2 n) + n + 1$$

Theorem (Skiena and Sundaram, 93)

There is an algorithm that reconstructs any unknown non-empty binary word $\mathbf{W}$ *in at most* $n + 2\lceil \log_2 n \rceil$ *queries.*

The gap over the binary alphabet

Theorem (Skiena and Sundaram, 93)

There is an algorithm that reconstructs any unknown non-empty binary word $\mathbf{W}$ *in at most* $n + 2\lceil \log_2 n \rceil$ *queries.*

We need at least n queries

The gap over the binary alphabet

Theorem (Skiena and Sundaram, 93)

There is an algorithm that reconstructs any unknown non-empty binary word $\mathbf{W}$ *in at most* $n + 2\lceil \log_2 n \rceil$ *queries.*

We need at least n queries

Can we do better than $n + O(\log_2 n)$?

Or even $n + O(1)$?

Over an alphabet of size $k \geq 3$

The number of queries needed to reconstruct an unkown word w over $\mathcal{A}$

$\geq \log_2(k^n) = n \log_2 k$	simple argument

Over an alphabet of size $k \geq 3$

The number of queries needed to reconstruct an unkown word w over $\mathcal{A}$

$\geq \log_2(k^n) = n \log_2 k$	simple argument
$\leq\ 1.59\, n \log_2(k) + 2k \log_2(n) + 5k$	Skiena and Sundaram, 1995

Over an alphabet of size $k \geq 3$

The number of queries needed to reconstruct an unkown word w over $\mathcal{A}$

$\geq \log_2(k^n) = n \log_2 k$	simple argument
$\leq \underbrace{1.59}_{\text{Not optimal ?}} n \log_2(k) + 2k \log_2(n) + 5k$	Skiena and Sundaram, 1995

Over an alphabet of size $k \geq 3$

The number of queries needed to reconstruct an unkown word w over $\mathcal{A}$

$\geq \log_2(k^n) = n \log_2 k$	simple argument
$\leq 1.59\, n \log_2(k) + 2k \log_2(n) + 5k$	Skiena and Sundaram, 1995
$\leq n\lceil \log_2(k) \rceil + k\lfloor \log_2(n) \rfloor + 2k$	Richomme and Rosenfeld, 2022

Over an alphabet of size $k \geq 3$

The number of queries needed to reconstruct an unkown word w over $\mathcal{A}$

$\geq \log_2(k^n) = n \log_2 k$	simple argument
$\leq 1.59\, n \log_2(k) + 2k \log_2(n) + 5k$	Skiena and Sundaram, 1995
$\leq n\lceil \log_2(k) \rceil + \underbrace{k\lfloor \log_2(n) \rfloor}_{\text{Not optimal ?}} + 2k$	Richomme and Rosenfeld, 2022

#-subword queries

$\boxed{\binom{\mathsf{w}}{\cdot}?}$-queries

$\binom{u}{v}$ denotes the number of occurrences of v as a subword of u

Example

$$\binom{0010}{01} = ?$$

$\boxed{\binom{w}{\cdot}?}$-queries

$\binom{u}{v}$ denotes the number of occurrences of v as a subword of u

Example

$$\binom{0010}{01} = 1$$

$\boxed{\binom{w}{\cdot}?}$-queries

$\binom{u}{v}$ denotes the number of occurrences of v as a subword of u

Example

$$\binom{0010}{01} = 2$$

$\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries

$\binom{u}{v}$ denotes the number of occurrences of v as a subword of u

Example

$$\binom{0010}{01} = 2$$

$\boxed{\binom{\mathbf{W}}{\cdot}?}$**-queries**

$\mathbf{W} \in \mathcal{A}^*$ only known by the oracle.

For any $u \in \mathcal{A}^*$, you can ask the query $\boxed{\binom{\mathbf{W}}{u}?}$

Minimize the number of $\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries needed to reconstruct $\mathbf{W}$

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?}$

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \implies$ There are 3 0 in $\mathbf{W}$

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \implies$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?}$

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \Longrightarrow$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?} \quad = \quad 4 \quad \Longrightarrow$ There are 2 1 in $\mathbf{W}$

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \Longrightarrow$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?} \quad = \quad 4 \quad \Longrightarrow$ There are 2 1 in $\mathbf{W}$

W

00011

00101

00110

01001

01010

01100

10001

10010

10100

11000

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1:	$\boxed{\binom{\mathbf{W}}{0}?}$	$=$	3	$\Longrightarrow$	There are 3 0 in $\mathbf{W}$
Q2:	$\boxed{\binom{\mathbf{W}}{1}?}$	$=$	4	$\Longrightarrow$	There are 2 1 in $\mathbf{W}$
Q3:	$\boxed{\binom{\mathbf{W}}{01}?}$	$=$	4		

$\mathbf{W}$
00011
00101
00110
01001
01010
01100
10001
10010
10100
11000

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \Longrightarrow$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?} \quad = \quad 4 \quad \Longrightarrow$ There are 2 1 in $\mathbf{W}$

Q3: $\boxed{\binom{\mathbf{W}}{01}?} \quad = \quad 4$

$\mathbf{W}$	$\binom{\mathbf{W}}{01}$
00011	6
00101	5
00110	4
01001	4
01010	3
01100	2
10001	3
10010	2
10100	1
11000	0

An example

An unknown word $\mathbf{W} \in \{0, 1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \implies$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?} \quad = \quad 4 \quad \implies$ There are 2 1 in $\mathbf{W}$

Q3: $\boxed{\binom{\mathbf{W}}{01}?} \quad = \quad 4$

$\mathbf{W}$	$\binom{\mathbf{W}}{01}$
00011	6
00101	5
00110	4
01001	4
01010	3
01100	2
10001	3
10010	2
10100	1
11000	0

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?}$ $=$ 3 $\Longrightarrow$ There are 3 0 in **W**

Q2: $\boxed{\binom{\mathbf{W}}{1}?}$ $=$ 4 $\Longrightarrow$ There are 2 1 in **W**

Q3: $\boxed{\binom{\mathbf{W}}{01}?}$ $=$ 4

Q4: $\boxed{\binom{\mathbf{W}}{00110}?}$

W

00011
00101
00110
01001
01010
01100
10001
10010
10100
11000

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} = 3 \implies$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?} = 4 \implies$ There are 2 1 in $\mathbf{W}$

Q3: $\boxed{\binom{\mathbf{W}}{01}?} = 4$

Q4: $\boxed{\binom{\mathbf{W}}{00110}?} = 0$

$\mathbf{W}$

00011

00101

00110

01001

01010

01100

10001

10010

10100

11000

An example

An unknown word $\mathbf{W} \in \{0,1\}^*$

Q1: $\boxed{\binom{\mathbf{W}}{0}?} \quad = \quad 3 \quad \Longrightarrow$ There are 3 0 in $\mathbf{W}$

Q2: $\boxed{\binom{\mathbf{W}}{1}?} \quad = \quad 4 \quad \Longrightarrow$ There are 2 1 in $\mathbf{W}$

Q3: $\boxed{\binom{\mathbf{W}}{01}?} \quad = \quad 4$

Q4: $\boxed{\binom{\mathbf{W}}{00110}?} \quad = \quad 0$

$\mathbf{W} = 01001$

$\mathbf{W}$

00011

00101

00110

01001

01010

01100

10001

10010

10100

11000

Linear number of $\boxed{\binom{\mathsf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathsf{W} \iff \binom{\mathsf{W}}{u} \geq 1$$

Linear number of $\boxed{\binom{\mathsf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathsf{W} \iff \binom{\mathsf{W}}{u} \geq 1$$

$\implies$ we can solve this in $n + O(\log n)$

Linear number of $\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathbf{W} \iff \binom{\mathbf{W}}{u} \geq 1$$

$\Longrightarrow$ we can solve this in $n + O(\log n)$

Theorem (Fleischmann, Lejeune, Manea, Nowotka and Rigo, 2020)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ in at most $\left\lfloor \frac{n}{2} \right\rfloor + 1$ queries.

Linear number of $\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathbf{W} \iff \binom{\mathbf{W}}{u} \geq 1$$

$\implies$ we can solve this in $n + O(\log n)$

Theorem (Fleischmann, Lejeune, Manea, Nowotka and Rigo, 2020)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ in at most $\left\lfloor \frac{n}{2} \right\rfloor + 1$ queries.

Write $\mathbf{W}$ as $$\mathbf{W} = 0^{s_0}10^{s_1}1\dots10^{s_t}$$

Obtain the s_i one by one $$s_i = \binom{\mathbf{W}}{1^i10^{t-i}}.$$

Linear number of $\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathbf{W} \iff \binom{\mathbf{W}}{u} \geq 1$$

$\implies$ we can solve this in $n + O(\log n)$

Theorem (Fleischmann, Lejeune, Manea, Nowotka and Rigo, 2020)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ in at most $\left\lfloor \frac{n}{2} \right\rfloor + 1$ queries.

Write $\mathbf{W}$ as
$$\mathbf{W} = 0^{s_0}10^{s_1}1\ldots10^{s_t}$$

Obtain the s_i one by one
$$s_i = \binom{\mathbf{W}}{1^i10^{t-i}}.$$

Theorem (Richomme and Rosenfeld)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ in at most $O(\sqrt{n \log_2 n})$ queries.

Linear number of $\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathbf{W} \iff \binom{\mathbf{W}}{u} \geq 1$$

$\implies$ we can solve this in $n + O(\log n)$

Theorem (Fleischmann, Lejeune, Manea, Nowotka and Rigo, 2020)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ in at most $\left\lfloor \frac{n}{2} \right\rfloor + 1$ queries.

Write $\mathbf{W}$ as

$$\mathbf{W} = 0^{s_0}10^{s_1}1\ldots10^{s_t}$$

Obtain the s_i one by one $s_i = \binom{\mathbf{W}}{1^i10^{t-i}}$.

Theorem (Richomme and Rosenfeld)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ in at most $O(\sqrt{n\log_2 n})$ queries.

Obtain multiple s_i at once

Linear number of $\boxed{\binom{\mathbf{W}}{\cdot}?}$-queries

$$u \sqsubseteq \mathbf{W} \iff \binom{\mathbf{W}}{u} \geq 1$$

$\implies$ we can solve this in $n + O(\log n)$

Theorem (Fleischmann, Lejeune, Manea, Nowotka and Rigo, 2020)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ *in at most* $\left\lfloor \frac{n}{2} \right\rfloor + 1$ *queries.*

Write $\mathbf{W}$ as $$\mathbf{W} = 0^{s_0}10^{s_1}1\ldots10^{s_t}$$

Obtain the s_i one by one $$s_i = \binom{\mathbf{W}}{1^i10^{t-i}}.$$

Theorem (Richomme and Rosenfeld)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ *in at most* $O(\sqrt{n\log_2 n})$ *queries.*

Obtain multiple s_i at once $\left(\sqrt{\frac{n}{\log_2 n}}\right)$

Toy problem

Suppose that we know $\mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

Toy problem

Suppose that we know $\mathbf{W} = 0^{s_0} 1 0^{s_1} 1 0^{s_2} 1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r}$$

Toy problem

Suppose that we know $\quad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct **W** in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + \binom{r+1}{r} s_1 + \binom{r+2}{r} s_0$$

Toy problem

Suppose that we know $\mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

Toy problem

Suppose that we know $\qquad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}$$

Toy problem

Suppose that we know $\quad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \underbrace{\frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}}_{<1}$$

Toy problem

Suppose that we know $\quad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \underbrace{\frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}}_{<1} = \left\lfloor \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} \right\rfloor$$

Toy problem

Suppose that we know $\quad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \underbrace{\frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}}_{<1} = \left\lfloor \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} \right\rfloor$$

$$s_1 =$$

Toy problem

Suppose that we know $\mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \underbrace{\frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}}_{<1} = \left\lfloor \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} \right\rfloor$$

$$s_1 = \frac{\binom{\mathbf{W}}{ab^r} - \frac{(r+2)(r+1)}{2}s_0}{r+1} - \frac{s_2}{r+1} =$$

Toy problem

Suppose that we know $\quad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ?

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \underbrace{\frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}}_{<1} = \left\lfloor \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} \right\rfloor$$

$$s_1 = \frac{\binom{\mathbf{W}}{ab^r} - \frac{(r+2)(r+1)}{2}s_0}{r+1} - \frac{s_2}{r+1} = \left\lfloor \frac{\binom{\mathbf{W}}{ab^r} - \frac{(r+2)(r+1)}{2}s_0}{r+1} \right\rfloor$$

Toy problem

Suppose that we know $\qquad \mathbf{W} = 0^{s_0}10^{s_1}10^{s_2}1^r$

with $r \gg s_i$ and we know r but not the s_i.

Can we reconstruct $\mathbf{W}$ in one query ? YES

$$\binom{\mathbf{W}}{01^r} = s_2 + (r+1)s_1 + \frac{(r+2)(r+1)}{2}s_0$$

In particular,

$$s_0 = \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} - \underbrace{\frac{2s_2 + 2(r+1)s_1}{(r+2)(r+1)}}_{<1} = \left\lfloor \frac{2\binom{\mathbf{W}}{01^r}}{(r+2)(r+1)} \right\rfloor$$

$$s_1 = \frac{\binom{\mathbf{W}}{ab^r} - \frac{(r+2)(r+1)}{2}s_0}{r+1} - \frac{s_2}{r+1} = \left\lfloor \frac{\binom{\mathbf{W}}{ab^r} - \frac{(r+2)(r+1)}{2}s_0}{r+1} \right\rfloor$$

The proof idea

Lemma

Let $\mathbf{W} = 0^{s_0}10^{s_1}1\dots10^{s_t}$ *and let* $m \in \mathbb{N}$. *Suppose that*

- t *is known,*
- $\forall i$, *either* $s_i < m$ *or* s_i *is known.*

Then, at most $4m$ *queries are needed to reconstruct* $\mathbf{W}$.

The proof idea

Lemma

Let $\mathbf{W} = 0^{s_0}10^{s_1}1\ldots10^{s_t}$ *and let* $m \in \mathbb{N}$. *Suppose that*

- t *is known,*
- $\forall i$, *either* $s_i < m$ *or* s_i *is known.*

Then, at most $4m$ *queries are needed to reconstruct* $\mathbf{W}$.

Let $I = \{i : s_i \geq m\}$, we need to be able to efficiently find I.

The proof idea

Lemma

Let $\mathbf{W} = 0^{s_0}10^{s_1}1\dots10^{s_t}$ *and let* $m \in \mathbb{N}$. *Suppose that*

- t *is known,*
- $\forall i$, *either* $s_i < m$ *or* s_i *is known.*

Then, at most $4m$ *queries are needed to reconstruct* $\mathbf{W}$.

Let $I = \{i : s_i \geq m\}$, we need to be able to efficiently find I.

Lemma

For an unknown word $\mathbf{W} = 0^{s_0}10^{s_1}1\dots10^{s_t}$, *the set* I *can be computed in at most* $\frac{2n\lceil \log_2 n \rceil}{m}$ *queries.*

The “algorithm”

- Compute the numbers of 0 and 1 — 2 queries

The "algorithm"

- Compute the numbers of 0 and 1 — 2 queries
- Finds the large blocks of 0 — $\frac{2n\lceil \log_2 n \rceil}{m}$ queries

The "algorithm"

- Compute the numbers of 0 and 1 — 2 queries
- Finds the large blocks of 0 — $\frac{2n\lceil \log_2 n \rceil}{m}$ queries
- Compute the remaining s_i — $4m$ queries

The "algorithm"

- Compute the numbers of 0 and 1 — 2 queries
- Finds the large blocks of 0 — $\frac{2n\lceil \log_2 n \rceil}{m}$ queries
- Compute the remaining s_i — $4m$ queries

Take $m = \sqrt{n \log_2 n} \implies$

The "algorithm"

- Compute the numbers of 0 and 1 — 2 queries
- Finds the large blocks of 0 — $\frac{2n\lceil \log_2 n \rceil}{m}$ queries
- Compute the remaining s_i — $4m$ queries

Take $m = \sqrt{n \log_2 n} \implies$

Theorem (Richomme and Rosenfeld)

There is an algorithm that reconstructs any unknown word $\mathbf{W} \in \{0,1\}^n$ *in at most* $O(\sqrt{n \log_2 n})$ *queries.*

A better strategy for uniform random word ?

Random word typically do not contain large blocks of 0.

A better strategy for uniform random word ?

Random word typically do not contain large blocks of 0.

Lemma

Let n be an integer and $\mathbf{W}$ be a binary word taken uniformly at random in $\{0,1\}^n$, then

$$\mathbb{P}(0^{\lceil 2 \log n \rceil} \textit{ is a factor of } \mathbf{W}) \leq \frac{1}{n}\,.$$

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2 \log n \rceil$ — 0 queries

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2 \log n \rceil$ — 0 queries
- Compute the remaining s_i — $4\lceil 2 \log n \rceil$ queries

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2 \log n \rceil$ — 0 queries
- Compute the remaining s_i — $4\lceil 2 \log n \rceil$ queries
- Did we found the correct $\mathbf{W}$? — 1 query

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2 \log n \rceil$ — 0 queries
- Compute the remaining s_i — $4\lceil 2 \log n \rceil$ queries
- Did we found the correct $\mathbf{W}$? — 1 query
  - Yes $\implies$ nothing to do — 0 query

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2 \log n \rceil$ — 0 queries
- Compute the remaining s_i — $4\lceil 2 \log n \rceil$ queries
- Did we found the correct $\mathbf{W}$? — 1 query
  - Yes $\implies$ nothing to do — 0 query
  - No $\implies$ Apply the previous algorithm — $O(\sqrt{n \log n})$ queries

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2\log n\rceil$ — 0 queries
- Compute the remaining s_i — $4\lceil 2\log n\rceil$ queries
- Did we found the correct $\mathbf{W}$? — 1 query
    - Yes $\implies$ nothing to do — 0 query
    - No $\implies$ Apply the previous algorithm — $O(\sqrt{n\log n})$ queries

The expected number of queries is at most

$$2 + 4\lceil 2\log n\rceil + 1 + \frac{O(\sqrt{n\log n})}{n} = 8\log n + O(1)$$

A better strategy for uniform random word !

- Compute the numbers of 0 and 1 — 2 queries
- Pretend that there is no s_i larger than $\lceil 2\log n \rceil$ — 0 queries
- Compute the remaining s_i — $4\lceil 2\log n \rceil$ queries
- Did we found the correct **W**? — 1 query
    - Yes $\implies$ nothing to do — 0 query
    - No $\implies$ Apply the previous algorithm — $O(\sqrt{n\log n})$ queries

The expected number of queries is at most

$$2 + 4\lceil 2\log n \rceil + 1 + \frac{O(\sqrt{n\log n})}{n} = 8\log n + O(1)$$

Theorem (Richomme and Rosenfeld)

There is a deterministic algorithm that given a random uniform word **W** *from* $\{0,1\}^n$ *reconstructs* **W** *in an expected number of queries*

$$O(\log n)\,.$$

The result

	Worst case complexity	Average case complexity
Previous result	$\leq n/2$	$\leq n/2$
Our result	$O(\sqrt{n \log n})$	$O(\log n)$
Lower bounds	??	??

The result

	Worst case complexity	Average case complexity
Previous result	$\leq n/2$	$\leq n/2$
Our result	$O(\sqrt{n \log n})$	$O(\log n)$
Lower bounds	??	??

Questions:

- A non-trivial lower bound on the number of queries needed?

The result

	Worst case complexity	Average case complexity
Previous result	$\leq n/2$	$\leq n/2$
Our result	$O(\sqrt{n \log n})$	$O(\log n)$
Lower bounds	??	??

Questions:

- A non-trivial lower bound on the number of queries needed?
- Improve the worst case complexity. Can we go down to $O(\log(n))$?

Conclusion

Many interesting open questions that did not receive a lot of attention ?

Conclusion

Many interesting open questions that did not receive a lot of attention ?

Theorem (Fici, Prezza and Venturini, 2021)

Let C be a compressor, and let $S \in \{0,1\}^n$ be an unknown binary string of known length n. Then, there is an algorithm that reconstructs S using $O(|C(S)|)$ substring queries.

Conclusion

Many interesting open questions that did not receive a lot of attention ?

Theorem (Fici, Prezza and Venturini, 2021)

Let C be a compressor, and let $S \in \{0, 1\}^n$ be an unknown binary string of known length n. Then, there is an algorithm that reconstructs S using $O(|C(S)|)$ substring queries.

A good lesson: binary search is sometime worst than greedy search.

Thanks !