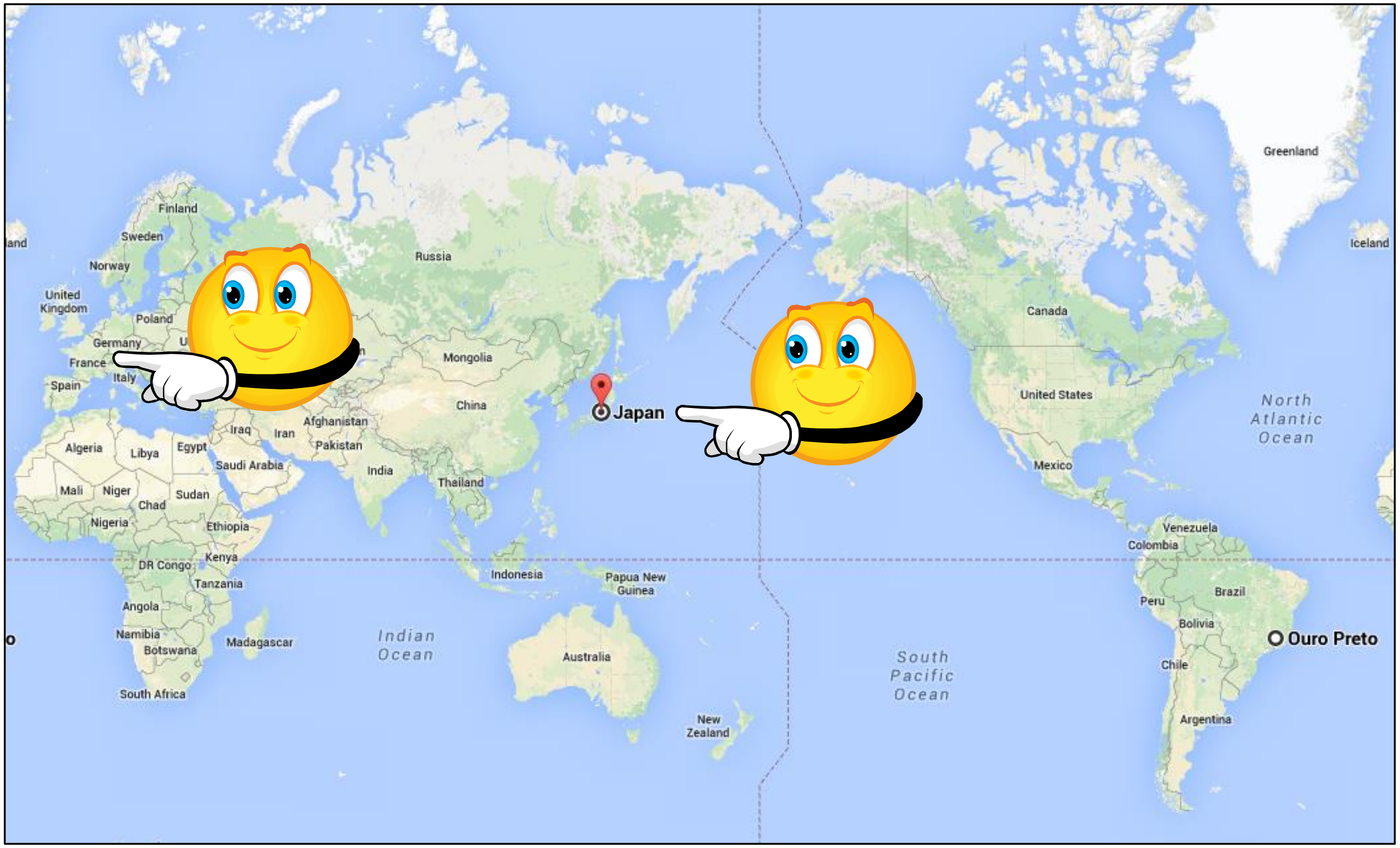
CIAC 2015

An opportunistic text indexing structure based on run length encoding

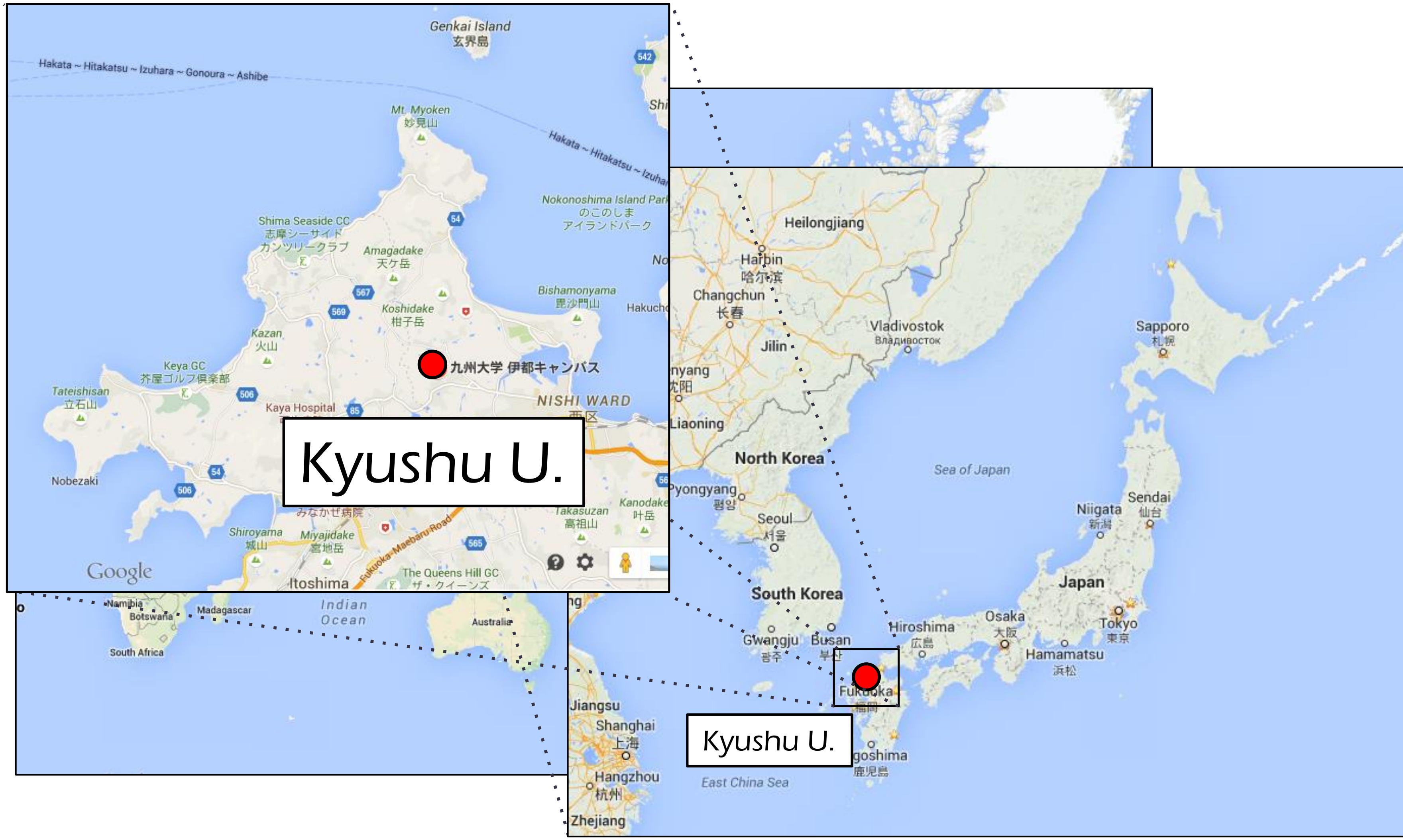
Yuya Tamakoshi, Keisuke Goto, Shunsuke Inenaga,
Hideo Bannai, Masayuki Takeda

Kyushu University, Japan

Kyushu University, Japan



Kyushu University, Japan



Kyushu University, Japan

Itoshima Peninsula

糸島

String Island

String matching

Input: text string T and pattern string P

Output: all occurrences of P in T

String matching

Input: text string T and pattern string P

Output: all occurrences of P in T

text T

We introduce a general framework which is suitable to capture an essence of compressed pattern matching according to various dictionary based compressions. The goal is to find all occurrences of a pattern in a text without decompression, which is one of the most active topics in string matching. Our framework includes such compression methods as Lempel-Ziv family, (LZ77, LZSS, LZ78, LZW), byte-pair encoding, and the static dictionary based method.

pattern P

compress

String matching

Input: text string T and pattern string P

Output: all occurrences of P in T

- ✓ String matching is fundamental to areas such as
  - Information Retrieval
  - Bioinformatics, etc.

Indexed string matching

Preprocess: build index on fixed text T

Query: pattern string P

Answer: all occurrences of P in T

- ✓ Goal is to construct a space-efficient index on T which quickly answers to string matching query.
  - ➢ Text T can be very long (e.g., DNA sequences).
  - ➢ We may receive many different query patterns.

Classical text index: Suffix Array

The suffix array SA of text T is an array which stores the beginning positions of the suffixes of T in lexicographic order [Manber & Myers, 1991].

T = cococacao$

$ is an end-marker which appears only at the end of any string.

Classical text index: Suffix Array

The suffix array *SA* of text *T* is an array which stores the beginning positions of the suffixes of *T* in lexicographic order [Manber & Myers, 1991].

```
 1 cococacao$
 2  ococacao$
 3   cocacao$
 4    ocacao$
 5     cacao$
 6      acao$
 7       cao$
 8        ao$
 9         o$
10          $
```

Classical text index: Suffix Array

The <u>suffix array</u> *SA* of text *T* is an array which stores the beginning positions of the suffixes of *T* in <u>lexicographic order</u> [Manber & Myers, 1991].

1	cococacao$
2	ococacao$
3	cocacao$
4	ocacao$
5	cacao$
6	acao$
7	cao$
8	ao$
9	o$
10	$

Sort

SA	
10	$
6	acao$
8	ao$
5	cacao$
7	cao$
3	cocacao$
1	cococacao$
9	o$
4	ocacao$
2	ococacao$

String matching with suffix array

Binary search a given pattern P on SA

$P =$ `coc`

SA	
10	`$`
6	`acao$`
8	`ao$`
5	`cacao$`
7	`cao$`
3	`cocacao$`
1	`cococacao$`
9	`o$`
4	`ocacao$`
2	`ococacao$`

String matching with suffix array

Binary search a given pattern P on SA

P = coc
>
cao$

SA

10	$
6	acao$
8	ao$
5	cacao$
7	cao$
3	cocacao$
1	cococacao$
9	o$
4	ocacao$
2	ococacao$

String matching with suffix array

Binary search a given pattern *P* on *SA*

P = coc
∧
o$

SA	
10	$
6	acao$
8	ao$
5	cacao$
7	cao$
3	cocacao$
1	cococacao$
9	o$
4	ocacao$
2	ococacao$

String matching with suffix array

Binary search a given pattern P on SA

P = coc
||
cocacao$

SA	
10	$
6	acao$
8	ao$
5	cacao$
7	cao$
✓3	cocacao$
1	cococacao$
9	o$
4	ocacao$
2	ococacao$

String matching with suffix array

Binary search a given pattern P on SA

P = <u>coc</u>

||

<u>coc</u>ocacao$

SA	
10	$
6	acao$
8	ao$
5	cacao$
7	cao$
✓3	<u>coc</u>acao$
✓1	<u>coc</u>ocacao$
9	o$
4	ocacao$
2	ococacao$

String matching with suffix array

Binary search a given pattern P on SA

1 2 3 4 5 6 7 8 9 10

T = cococacao$

P = coc

SA

10	$
6	acao$
8	ao$
5	cacao$
7	cao$
3	cocacao$
1	cococacao$
9	o$
4	ocacao$
2	ococacao$

String matching with suffix array

All occurrences of P in T can be found in $O(m \log u + occ)$ time using SA.

The search time can be improved to $O(m + \log u + occ)$ using the LCP array.

SA	
10	`$`
6	`acao$`
8	`ao$`
5	`cacao$`
7	`cao$`
3	`cocacao$`
1	`cococacao$`
9	`o$`
4	`ocacao$`
2	`ococacao$`

$u = |T|$

$m = |P|$

occ = # occ. of P in T

SA+LCP

Theorem [Manber & Myers, 1991]

There is an index (SA+LCP) which reports all occ occurrences of P in T in $O(m+\log u+occ)$ time, and requires $\underbrace{2u\log u}_{\text{SA \& LCP}} + \underbrace{u\log\sigma}_{\text{Text } T} + \underbrace{O(u)}_{\text{Auxiliary data structure}}$ bits of space.

SA & LCP

Text T

Auxiliary data structure

$u = |T|$
$m = |P|$
$\sigma = |\Sigma|$

✓ This can take too much space for large text T (i.e., for large u).

Compressed index

- ✓ There are a number of compressed indexes which occupy only compressed size of text.
  - FM-index [Ferragina & Mancini, 2000], Compressed Suffix Array [Grossi & Vitter, 2000], Lempel-Ziv index [Gagie et al., 2014], etc.
- ✓ Most of them are **slower** than SA+LCP.

Our proposal

New compressed index based on run length encoding (RLE) of text which is **smaller** & **faster** than SA+LCP.

Run Length Encoding (RLE)

The run length encoding of text T, denoted $RLE(T)$, is a compressed representation of T in which each maximal run $a\ldots a$ of characters is encoded by a^p, where p denotes the length of the maximal run.

T = aaaabbbaacccccccbbbbbaaaaa$

$RLE(T) = \mathtt{a}^4\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$

✓ Applications to RLE include:
- black-white fax messages
- image format (PackBits, TIFF)
- music format (MIDI)

RLE suffixes

Let $n = |RLE(T)|$. For any $1 \le i \le n$, $RLEsuf(i)$ is the suffix of $RLE(T)$ starting with the i-th run.

$RLE(T)$: $\mathtt{a}^4\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$

$n = 7$

$RLEsuf(1)$: $\mathtt{a}^4\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$

$RLEsuf(2)$: $\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$

$RLEsuf(3)$: $\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$

$RLEsuf(4)$: $\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$

$RLEsuf(5)$: $\mathtt{b}^5\mathtt{a}^5\$$

$RLEsuf(6)$: $\mathtt{a}^5\$$

$RLEsuf(7)$: $\$$

Difficulty in indexing RLE suffixes

✓ We want to index only RLE suffixes of the text, but simply sorted RLE suffixes don't work!

sorted RLE suffixes of text

$$\begin{array}{l}
a^5b\cdots \\
a^5b\cdots \\
a^5c\cdots \\
a^4b\cdots \\
a^4c\cdots \\
a^4c\cdots \\
a^4c\cdots \\
a^3b\cdots \\
a^3b\cdots
\end{array}$$

Difficulty in indexing RLE suffixes

- ✓ We want to index only RLE suffixes of the text, but simply sorted RLE suffixes don't work!

sorted RLE suffixes of text

```
aaaaab···
aaaaab···
aaaaac···
aaaab···
aaaac···
aaaac···
aaaac···
aaab···
aaab···
```

Difficulty in indexing RLE suffixes

- ✓ We want to index only RLE suffixes of the text, but simply sorted RLE suffixes don't work!

$RLE(P)$: $\mathtt{a}^2\mathtt{b}^1$

sorted RLE suffixes of text

✓ aaa<u>aab</u>···
✓ aaa<u>aab</u>···
  aaaaac···
✓ aa<u>aab</u>···
  aaaac···
  aaaac···
  aaaac···
✓ a<u>aab</u>···
✓ a<u>aab</u>···

Pattern occurrences are spread out, so we cannot binary search!!

Our ideas to index RLE suffixes

- ✓ When sorting RLE suffixes, we "ignore" the exponents of the <u>first runs</u> of RLE suffixes of text T.
- ✓ To find occurrences of pattern P, we first "ignore" the exponent of the first run of $RLE(P)$, and find its corresponding range.
- ✓ We then pick up only the occurrences of $RLE(P)$ from this range.

Truncated RLE suffixes

$tRLEsuf(i)$ is the suffix of $RLEsuf(i)$ where the first exponent p_i is truncated to 1.

$RLEsuf(1)$:	$\mathtt{a}^4\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$	$tRLEsuf(1)$:	$\mathtt{a}\ \mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$
$RLEsuf(2)$:	$\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$	$tRLEsuf(2)$:	$\mathtt{b}\ \mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$
$RLEsuf(3)$:	$\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$	$tRLEsuf(3)$:	$\mathtt{a}\ \mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$
$RLEsuf(4)$:	$\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$	$tRLEsuf(4)$:	$\mathtt{c}\ \mathtt{b}^5\mathtt{a}^5\$$
$RLEsuf(5)$:	$\mathtt{b}^5\mathtt{a}^5\$$	$tRLEsuf(5)$:	$\mathtt{b}\ \mathtt{a}^5\$$
$RLEsuf(6)$:	$\mathtt{a}^5\$$	$tRLEsuf(6)$:	$\mathtt{a}\ \$$
$RLEsuf(7)$:	$\$$	$tRLEsuf(7)$:	$\$$

Our index: Truncated RLE Suffix Array

The tRLE suffix array *tRLESA* of text *T* is an array which stores the beginning positions of the tRLE suffixes in lexicographical order.

1	$a\ b^3a^2c^7b^5a^5\$$
2	$b\ a^2c^7b^5a^5\$$
3	$a\ c^7b^5a^5\$$
4	$c\ b^5a^5\$$
5	$b\ a^5\$$
6	$a\ \$$
7	$\$$

Sort

tRLESA	
7	$\$$
6	$a\ \$$
1	$a\ b^3a^2c^7b^5a^5\$$
3	$a\ c^7b^5a^5\$$
5	$b\ a^5\$$
2	$b\ a^2c^7b^5a^5\$$
4	$c\ b^5a^5\$$

Monotonicity on Truncated RLE Suffix Array

tRLESA	tRLE suffixes
$\vdots$	$\vdots$
47	$b^{(2)}c^5a^2b^2a^6\cdots$
99	$b^{(9)}c^5a^2b^3a^1\cdots$
11	$b^{(1)}c^5a^2b^5a^4\cdots$
40	$b^{(2)}c^5a^2b^7a^3\cdots$
55	$b^{(3)}c^5a^2b^8c^2\cdots$
72	$b^{(9)}c^5a^2b^6c^3\cdots$
19	$b^{(1)}c^5a^2b^6c^3\cdots$
26	$b^{(5)}c^5a^2b^4c^7\cdots$
4	$b^{(1)}c^5a^2b^1c^8\cdots$
$\vdots$	

Ignored exponents in parentheses

Monotonicity on Truncated RLE Suffix Array

We first look for $\mathtt{b}\mathtt{c}^5\mathtt{a}^2$

$RLE(P)$: $\mathtt{b}^3\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4$

The range $\mathtt{b}\mathtt{c}^5\mathtt{a}^2$ matches

This range can be found by a binary search.

tRLESA	tRLE suffixes
$\vdots$	$\vdots$
47	$\mathtt{b}^{(2)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^2\mathtt{a}^6\cdots$
99	$\mathtt{b}^{(9)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^3\mathtt{a}^1\cdots$
11	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^5\mathtt{a}^4\cdots$
40	$\mathtt{b}^{(2)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^7\mathtt{a}^3\cdots$
55	$\mathtt{b}^{(3)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^8\mathtt{c}^2\cdots$
72	$\mathtt{b}^{(9)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^6\mathtt{c}^3\cdots$
19	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^6\mathtt{c}^3\cdots$
26	$\mathtt{b}^{(5)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4\mathtt{c}^7\cdots$
4	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^1\mathtt{c}^8\cdots$
$\vdots$	$\vdots$

Monotonicity on Truncated RLE Suffix Array

We next look for $\mathrm{b}\mathrm{c}^5\mathrm{a}^2\underline{\mathrm{b}^4}$

$RLE(P)$: $\mathrm{b}^3\mathrm{c}^5\mathrm{a}^2\mathrm{b}^4$

The range $\boxed{\mathrm{b}\mathrm{c}^5\mathrm{a}^2}$ matches

tRLESA	tRLE suffixes
⋮	⋮
47	$\mathrm{b}^{(2)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^2\mathrm{a}^6\cdots$
99	$\mathrm{b}^{(9)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^3\mathrm{a}^1\cdots$
11	$\mathrm{b}^{(1)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^5\mathrm{a}^4\cdots$
40	$\mathrm{b}^{(2)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^7\mathrm{a}^3\cdots$
55	$\mathrm{b}^{(3)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^8\mathrm{c}^2\cdots$
72	$\mathrm{b}^{(9)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^6\mathrm{c}^3\cdots$
19	$\mathrm{b}^{(1)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^6\mathrm{c}^3\cdots$
26	$\mathrm{b}^{(5)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^4\mathrm{c}^7\cdots$
4	$\mathrm{b}^{(1)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^1\mathrm{c}^8\cdots$
⋮	⋮

Monotonicity on Truncated RLE Suffix Array

We next look for $\mathtt{b}\mathtt{c}^5\mathtt{a}^2\underline{\mathtt{b}^4}$

$RLE(P)$: $\mathtt{b}^3\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4$

The range $\mathtt{b}\mathtt{c}^5\mathtt{a}^2$ matches

tRLESA	tRLE suffixes
⋮	⋮
47	$\mathtt{b}^{(2)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{2}}\mathtt{a}^6\cdots$
99	$\mathtt{b}^{(9)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{3}}\mathtt{a}^1\cdots$
11	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{5}}\mathtt{a}^4\cdots$
40	$\mathtt{b}^{(2)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{7}}\mathtt{a}^3\cdots$
55	$\mathtt{b}^{(3)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{8}}\mathtt{c}^2\cdots$
72	$\mathtt{b}^{(9)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{6}}\mathtt{c}^3\cdots$
19	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{6}}\mathtt{c}^3\cdots$
26	$\mathtt{b}^{(5)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{4}}\mathtt{c}^7\cdots$
4	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^{\mathbf{1}}\mathtt{c}^8\cdots$
⋮	⋮

monotonically non-decreasing

monotonically non-increasing

Matching with Truncated RLE Suffix Array

We next look for $\mathrm{b}\mathrm{c}^5\mathrm{a}^2\underline{\mathrm{b}^4}$

RLE(*P*): $\mathrm{b}^3\mathrm{c}^5\mathrm{a}^2\mathrm{b}^4$

tRLESA	tRLE suffixes
⋮	⋮
47	$\mathrm{b}^{(2)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^2\mathrm{a}^6\cdots$
99	$\mathrm{b}^{(9)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^3\mathrm{a}^1\cdots$
11	$\mathrm{b}^{(1)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^5\mathrm{a}^4\cdots$
40	$\mathrm{b}^{(2)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^7\mathrm{a}^3\cdots$
55	$\mathrm{b}^{(3)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^8\mathrm{c}^2\cdots$
72	$\mathrm{b}^{(9)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^6\mathrm{c}^3\cdots$
19	$\mathrm{b}^{(1)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^6\mathrm{c}^3\cdots$
26	$\mathrm{b}^{(5)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^4\mathrm{c}^7\cdots$
4	$\mathrm{b}^{(1)}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^1\mathrm{c}^8\cdots$
⋮	⋮

the range $\boxed{\mathrm{b}\mathrm{c}^5\mathrm{a}^2\mathrm{b}^4}$ matches

Based on the monotonicity, this range can be found by a binary search.

Matching with Truncated RLE Suffix Array

We finally look for $b^{\mathbf{3}}c^5a^2b^4$

$RLE(P)$: $b^3c^5a^2b^4$

the range $\boxed{bc^5a^2b^4}$ matches

tRLESA	tRLE suffixes
⋮	⋮
47	$b^{(2)}c^5a^2b^2a^6\cdots$
99	$b^{(9)}c^5a^2b^3a^1\cdots$
11	$b^{(1)}c^5a^2b^5a^4\cdots$
40	$b^{(2)}c^5a^2b^7a^3\cdots$
55	$b^{(3)}c^5a^2b^8c^2\cdots$
72	$b^{(9)}c^5a^2b^6c^3\cdots$
19	$b^{(1)}c^5a^2b^6c^3\cdots$
26	$b^{(5)}c^5a^2b^4c^7\cdots$
4	$b^{(1)}c^5a^2b^1c^8\cdots$
⋮	⋮

Matching with Truncated RLE Suffix Array

We finally look for $\mathtt{b}^{\mathbf{3}}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4$

$RLE(P)$: $\mathtt{b}^3\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4$

tRLESA	tRLE suffixes
⋮	⋮
47	$\mathtt{b}^{(2)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^2\mathtt{a}^6\cdots$
99	$\mathtt{b}^{(9)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^3\mathtt{a}^1\cdots$
11	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^5\mathtt{a}^4\cdots$
40	$\mathtt{b}^{(2)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^7\mathtt{a}^3\cdots$
55	$\mathtt{b}^{(\mathbf{3})}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^8\mathtt{c}^2\cdots$
72	$\mathtt{b}^{(\mathbf{9})}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^6\mathtt{c}^3\cdots$
19	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^6\mathtt{c}^3\cdots$
26	$\mathtt{b}^{(\mathbf{5})}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4\mathtt{c}^7$
4	$\mathtt{b}^{(1)}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^1\mathtt{c}$
⋮	⋮

the range $\boxed{\mathtt{b}\mathtt{c}^5\mathtt{a}^2\mathtt{b}^4}$ matches

We want only those whose 1st exponents are at least **3**

Matching with Truncated RLE Suffix Array

RLE(*P*): $b^3c^5a^2b^4$

exponents	*tRLESA*	tRLE suffixes
⋮	⋮	⋮
2	47	$b^{(2)}c^5a^2\,b^2a^6\cdots$
9	99	$b^{(9)}c^5a^2\,b^3a^1\cdots$
1	11	$b^{(1)}c^5a^2\,b^5a^4\cdots$
2	40	$b^{(2)}c^5a^2\,b^7a^3\cdots$
3	55	$b^{(3)}c^5a^2\,b^8c^2\cdots$
9	72	$b^{(9)}c^5a^2\,b^6c^3\cdots$
1	19	$b^{(1)}c^5a^2\,b^6c^3\cdots$
5	26	$b^{(5)}c^5a^2\,b^4c^7\cdots$
1	4	$b^{(1)}c^5a^2\,b^1c^8\cdots$
⋮	⋮	⋮

We use an array of ignored exponents of the first runs.

Matching with Truncated RLE Suffix Array

We finally look for $b^{3}c^{5}a^{2}b^{4}$

$RLE(P)$: $b^3c^5a^2b^4$

the range $bc^5a^2b^4$ matches

exponents	*tRLESA*	tRLE suffixes
⋮	⋮	⋮
2	47	$b^{(2)}c^5a^2b^2a^6\ldots$
9	99	$b^{(9)}c^5a^2b^3a^1\ldots$
1	11	$b^{(1)}c^5a^2b^5a^4\ldots$
2	40	$b^{(2)}c^5a^2b^7a^3\ldots$
3	55	$b^{(3)}c^5a^2b^8c^2\ldots$
9	72	$b^{(9)}c^5a^2b^6c^3\ldots$
1	19	$b^{(1)}c^5a^2b^6c^3\ldots$
5	26	$b^{(5)}c^5a^2b^4c^7\ldots$
1	4	$b^{(1)}c^5a^2b^1c^8\ldots$
⋮	⋮	⋮

Matching with Truncated RLE Suffix Array

We finally look for $b^{3}c^{5}a^{2}b^{4}$

$RLE(P)$: $b^{3}c^{5}a^{2}b^{4}$

Range Maximum Query (RMQ)

exponents		*tRLESA*	tRLE suffixes
$\vdots$		$\vdots$	$\vdots$
2		47	$b^{(2)}c^{5}a^{2}b^{2}a^{6}\ldots$
9		99	$b^{(9)}c^{5}a^{2}b^{3}a^{1}\ldots$
1		11	$b^{(1)}c^{5}a^{2}b^{5}a^{4}\ldots$
2		40	$b^{(2)}c^{5}a^{2}b^{7}a^{3}\ldots$
3		55	$b^{(3)}c^{5}a^{2}b^{8}c^{2}\ldots$
9	✓	72	$b^{(9)}c^{5}a^{2}b^{6}c^{3}\ldots$
1		19	$b^{(1)}c^{5}a^{2}b^{6}c^{3}\ldots$
5		26	$b^{(5)}c^{5}a^{2}b^{4}c^{7}\ldots$
1		4	$b^{(1)}c^{5}a^{2}b^{1}c^{8}\ldots$
$\vdots$		$\vdots$	$\vdots$

Matching with Truncated RLE Suffix Array

We finally look for $b^{\mathbf{3}}c^5a^2b^4$

expo

We perform RMQ's recursively, in the 1st & 2nd halves of the range.

$RLE(P)$: $b^3c^5a^2b^4$

	[illegible]		[illegible]	[illegible]
	[illegible]		99	$b^{(9)}c^5a^2b^3a^1\ldots$
RMQ	1		11	$b^{(1)}c^5a^2b^5a^4\ldots$
	2		40	$b^{(2)}c^5a^2b^7a^3\ldots$
	3	✓	55	$b^{(\mathbf{3})}c^5a^2b^8c^2\ldots$
	9	✓	72	$b^{(\mathbf{9})}c^5a^2b^6c^3\ldots$
RMQ	1		19	$b^{(1)}c^5a^2b^6c^3\ldots$
	5	✓	26	$b^{(\mathbf{5})}c^5a^2b^4c^7\ldots$
	1		4	$b^{(1)}c^5a^2b^1c^8\ldots$
	⋮		⋮	⋮

Matching with Truncated RLE Suffix Array

We finally look for $b^{\mathbf{3}}c^5a^2b^4$

exp

Recursion ends when the range maxima is less than **3**.

$RLE(P)$: $b^3c^5a^2b^4$

	2			$b^{(\cdot)}c^5a^2b^{\cdot}a^{\cdot}\ldots$
	9		99	$b^{(9)}c^5a^2b^3a^1\ldots$
RMQ	1		11	$b^{(1)}c^5a^2b^5a^4\ldots$
RMQ	**2**		40	$b^{(2)}c^5a^2b^7a^3\ldots$
	3	✓	55	$b^{(\mathbf{3})}c^5a^2b^8c^2\ldots$
	9	✓	72	$b^{(\mathbf{9})}c^5a^2b^6c^3\ldots$
RMQ	**1**		19	$b^{(1)}c^5a^2b^6c^3\ldots$
	5	✓	26	$b^{(\mathbf{5})}c^5a^2b^4c^7\ldots$
	1		4	$b^{(1)}c^5a^2b^1c^8\ldots$
	⋮		⋮	⋮

Matching with Truncated RLE Suffix Array

RLE(*P*): $b^{3}c^{5}a^{2}b^{4}$

exponents		*tRLESA*	tRLE suffixes
⋮		⋮	⋮
2		47	$b^{(2)}c^{5}a^{2}b^{2}a^{6}\ldots$
9		99	$b^{(9)}c^{5}a^{2}b^{3}a^{1}\ldots$
1		11	$b^{(1)}c^{5}a^{2}b^{5}a^{4}\ldots$
2		40	$b^{(2)}c^{5}a^{2}b^{7}a^{3}\ldots$
3	✓	55	$b^{(3)}c^{5}a^{2}b^{8}c^{2}\ldots$
9	✓	72	$b^{(9)}c^{5}a^{2}b^{6}c^{3}\ldots$
1		19	$b^{(1)}c^{5}a^{2}b^{6}c^{3}\ldots$
5	✓	26	$b^{(5)}c^{5}a^{2}b^{4}c^{7}\ldots$
1		4	$b^{(1)}c^{5}a^{2}b^{1}c^{8}\ldots$
⋮		⋮	⋮

# of RMQ's we perform is *O*(*occ*).

Each RMQ takes *O*(1) time [Fischer & Heum, 2011].

Our results

Theorem 1 (RLE-index)

There is an index which, given $RLE(P)$, reports all occ occurrences of P in T in $O(q+\log n+occ)$ time, and requires $2n\log u + n\log\sigma + n\log n + O(n)$ bits of space.

$u = |T|$
$n = |RLE(T)|$
$(n \leq u)$
$q = |RLE(P)|$
$\sigma = |\Sigma|$

Our results

Theorem 1 (RLE-index)

There is an index which, given $RLE(P)$, reports all occ occurrences of P in T in $\boxed{O(q+\log n+occ)}$ time, and requires $2n\log u + n\log\sigma + n\log n + O(n)$ bits of space.

- ✓ SA+LCP takes $\boxed{O(m+\log u+occ)}$ time for pattern matching ($m = |P|$).
- ✓ Since $q \leq m$ and $n \leq u$ always hold, our index is <u>faster</u> than SA+LCP.

$u = |T|$
$n = |RLE(T)|$
$(n \leq u)$
$q = |RLE(P)|$
$\sigma = |\Sigma|$

Our results

Theorem 1 (RLE-index)

There is an index which, given $RLE(P)$, reports all occ occurrences of P in T in $O(q+\log n+occ)$ time, and requires $\boxed{2n\log u + n\log\sigma + n\log n + O(n)}$ bits of space.

- ✓ SA+LCP requires $\boxed{2u\log u + u\log\sigma + O(u)}$ bits of space.
- ✓ Our RLE-index is <u>smaller</u> when text T is compressible with RLE.

$u = |T|$
$n = |RLE(T)|$
$(n \le u)$
$q = |RLE(P)|$
$\sigma = |\Sigma|$

Our results

Theorem 2 (Construction time & space)

Given $RLE(T)$ of size n, the RLE-index of T can be constructed in $O(n \log n)$ time with $O(n \log u)$ bits of working space.

$u = |T|$
$n = |RLE(T)|$

- ✓ We introduced <u>new combinatorial properties of RLE suffixes</u>.
- ✓ We also use the idea of induced-sorting [Nong et al., 2011] which was originally designed for fast suffix array construction.

Conclusions & Future work

- ✓ Our RLE-index is <u>always faster</u> than SA+LCP.
- ✓ Our RLE-index is <u>smaller</u> than SA+LCP when the text is compressible by RLE (i.e. when the $n \log n$ term is negligible).

- ◆ Comparisons to other compressed index (e.g., FM-index, compressed SA, LZ-index).

FAQ

LCP array and Range Minima

The LCP array of T stores the length of the longest common prefix of neighboring suffixes in SA of T.

SA	LCP	
10	-	`$`
6	0	`acao$`
8	1	`ao$`
5	0	`cacao$`
7	2	`cao$`
3	1	`cocacao$`
1	3	`cococacao$`
9	0	`o$`
4	1	`ocacao$`
2	2	`ococacao$`

LCP array and Range Minima

The LCP array of T stores the length of the longest common prefix of neighboring suffixes in SA of T.

SA	LCP	
10	-	`$`
6	0	`acao$`
8	1	`ao$`
5	0	`cacao$`
7	**2**	`cao$`
3	1	`cocacao$`
1	3	`cococacao$`
9	0	`o$`
4	1	`ocacao$`
2	2	`ococacao$`

LCP array and Range Minima

The LCP array of T stores the length of the longest common prefix of neighboring suffixes in SA of T.

SA	LCP	
10	-	$
6	0	acao$
8	1	ao$
5	0	cacao$
7	2	cao$
3	1	cocacao$
1	**3**	cococacao$
9	0	o$
4	1	ocacao$
2	2	ococacao$

LCP array and Range Minima

The length of the LCP of any suffixes can also be computed by a range minimum query.

LCP		
-		`$`
0		`acao$`
1		`ao$`
0	✓	`cacao$`
2		`cao$`
1		`cocacao$`
3	✓	`cococacao$`
0		`o$`
1		`ocacao$`
2		`ococacao$`

LCP array and Range Minima

The length of the LCP of <u>any</u> suffixes can also be computed by a range <u>minimum</u> query.

Range minimum query

LCP		
-		$
0		acao$
1		ao$
0	✓	cacao$
2		cao$
1		cocacao$
3	✓	cococacao$
0		o$
1		ocacao$
2		ococacao$

LCP array and Range Minima

For any integer array of length k, there is a data structure which supports range minimum query in $O(1)$ time, and requires $2k + o(k)$ bits of extra space [Fischer & Heum, 2011].

S-type and L-type RLE suffixes

$RLEsuf(i)$ is S-type if $RLEsuf(i) < RLEsuf(i+1)$.
$RLEsuf(i)$ is L-type if $RLEsuf(i) > RLEsuf(i+1)$.

- ✓ $a^4b^3a^2c^7b^5a^5\$$ is S-type, because
  $a^4b^3a^2c^7b^5a^5\$ < b^3a^2c^7b^5a^5\$$.
- ✓ $b^3a^2c^7b^5a^5\$$ is L-type, because
  $b^3a^2c^7b^5a^5\$ < a^2c^7b^5a^5\$$.

※ Lex. order < on RLE strings is the same as the lex. order < on decompressed strings.

Properties of lex. order of RLE suffixes

For any $1 \leq i \leq n$, let a_i, p_i be the ith character and exponent of $RLE(T)$, respectively.

Lemma

For any $RLEsuf(i)$ and $RLEsuf(j)$ with $a_i = a_j$,

1. if $RLEsuf(i)$ is L-type and $RLEsuf(j)$ is S-type, then $RLEsuf(i) < RLEsuf(j)$.
2. if $RLEsuf(i)$ and $RLEsuf(j)$ are L-type and $p_i < p_j$, then $RLEsuf(i) < RLEsuf(j)$.
3. if $RLEsuf(i)$ and $RLEsuf(j)$ are S-type and $p_i > p_j$, then $RLEsuf(i) < RLEsuf(j)$.

Properties of lex. order of RLE suffixes

Lemma (Case 1)

For any $RLEsuf(i)$ and $RLEsuf(j)$ with $a_i = a_j$,

1. if $RLEsuf(i)$ is L-type and $RLEsuf(j)$ is S-type, then $RLEsuf(i) < RLEsuf(j)$.

L-type (a > $)

S-type (a < b)

$$\mathtt{a}^5\$ \ < \ \mathtt{a}^4\mathtt{b}^3\mathtt{a}^2\mathtt{c}^7\mathtt{b}^5\mathtt{a}^5\$$$

```
aaaaa$
     ∧
aaaabbbaacccccccbbbbbaaaaa$
```

Properties of lex. order of RLE suffixes

Lemma (Case 2)

For any $RLEsuf(i)$ and $RLEsuf(j)$ with $a_i = a_j$,

2. if $RLEsuf(i)$ and $RLEsuf(j)$ are L-type and $p_i < p_j$, then $RLEsuf(i) < RLEsuf(j)$.

L-type (b > a)

L-type (b > a)

$b^{3}a^{2}c^{7}b^{5}a^{5}\$ < b^{5}a^{5}\$$

bbbaacccccccbbbbbaaaaa$

∧

bbbbbaaaaa$

Properties of lex. order of RLE suffixes

Lemma (Case 3)

For any $RLEsuf(i)$ and $RLEsuf(j)$ with $a_i = a_j$,

3. if $RLEsuf(i)$ and $RLEsuf(j)$ are S-type and $p_i > p_j$, then $RLEsuf(i) < RLEsuf(j)$.

S-type (a < b)

S-type (a < c)

$a^{4}b^{3}a^{2}c^{7}b^{5}a^{5}\$ \quad < \quad a^{2}c^{7}b^{5}a^{5}\$$

aaaabbbaaccccccccbbbbbaaaaa$

$\wedge$

aacccccccbbbbbaaaaa$

Our results

Theorem 3 (accessing *SA*)

There is an index which, given an integer $1 \le j \le u$, answers $SA[j]$ in $O(\log^2 n)$ time, and requires $n(3\log u + \log n + \log\sigma) + 2\sigma\log\frac{u}{\sigma} + O(n\log\log n)$ bits of space.

- ✓ Use a wavelet tree [Grossi et al., 2003] in place of RMQ data structure.
- ✓ Then, we can access arbitrary position of *SA*, using our RLE-index.

$u = |T|$
$n = |RLE(T)|$
$\sigma = |\Sigma|$