Reviewing: Golynski, Munro, Rao: Rank/Select Operations on Large Alphabets: a Tool for Text Indexing

Simon Gog

May 21, 2012

Time complexities of basic operations on sequences

Given sequence T of length n over alphabet Σ of length σ.

	$access(\mathsf{T}, i)$	$rank(\mathsf{T}, i, c)$	$select(\mathsf{T}, i, c)$	space
WT	$\log \sigma$	$\log \sigma$	$\log \sigma$	$n \log \sigma + o(n \log \sigma)$
G-1	$\sigma \cdot \log \log \sigma$	$\log \log \sigma$	1	$nH_0 + \mathcal{O}(n)$*
G-2	$\log \log \sigma$	$\log \log \sigma$	1	$n \log \sigma + o(n \log \sigma)$**
G-2a	1	$\log \log \sigma \cdot \log \log \log \sigma$	$\log \log \sigma$	$n \log \sigma + o(n \log \sigma)$**

We have omitted $\mathcal{O}(\cdot)$ at the specification of the time complexities.

* actually $2n + o(n)$

** $4n + o(n)$ hidden in $o(n \log \sigma)$

Solution overview

- (1) Divide sequence into blocks of length σ.
- (2) Calculate rank and select on the block level.
- (3) Calculate in-block rank and select.
- Step (1) and (2) are used in all solutions

Relation between binary rank/select and general rank/select

- Conceptionally introduce a bitvector for each symbol
- Concatenated in row major order: Array A
- Size of A: $n\sigma$

	e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
_	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
e	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
m	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
y	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Relation between binary rank/select and general rank/select

If we can answer rank/select on A in constant time, we can answer it on T as well.

$$rank(\mathsf{T}, i, c) = rank(A, c \cdot n + i, 1) - rank(A, c \cdot n, 1) \quad (1)$$

$$select(\mathsf{T}, i, c) = select(A, rank(A, c \cdot n, 1) + i + 1, 1) \quad (2)$$

But: A uses to much space!

Compressing A

- Divide A into blocks of length σ
- Count the number of ones in each block A
- Resulting array is C of length n, so $|C| = n \log \sigma bits$.

	e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
$	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
_	0	0	0	0	0	0	0	0	0	1	1	3	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	2	0	0	1
e	1	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
m	0	0	0	3	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	5	1	1	0
y	0	1	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Compressing A

C= 0 1 0 0 3 0 0 1 2 1 1 0 3 1 0 0 0 5 3 0 0

- The sum of all entries in C is n.
- So store it with unary code in array B with $\Rightarrow |B| = 2n$ bits

$B =$ 10111000111010010101100010111100001000111

- Now C is compressed. But how do we answer rank and select with B?
- By adding a select data structure to B we can navigate to blocks in A!
- We jump to block i in A by doing $select(B, i, 1)$
- $rank'(A, \sigma i) = rank(B, select(B, i, 1), 0) = select(B, i, 1) + 1 - i$

Rank and select on blocks A

	e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
$	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
_	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
e	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
m	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
y	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$B = 10111000111010010101100010111100001000111$

- $rank'(A, \sigma i) = select(B, i, 1) + 1 - i$
- $select'(A, i) = rank(B, select(B, i, 0), 1) = select(B, i, 0) + 1 - i$

In-block rank and select (G-1)

- For each block A_j, we store the positions in the range $[0..\sigma-1]$ of the set bits in increasing order in an array E_j.
- Total space: $n \log \sigma$.

Answering select

- Block $x = \mathit{select}'(A, i, 1)$ contains the ith one. There are $y = \mathit{rank}'(A, \sigma x, 0)$ ones before block $x \Rightarrow$ $\mathit{select}(A, i, 1) = x \cdot \sigma + E_x[i - y]$

Answering rank

- i with $j = \lfloor \frac{i}{\sigma} \rfloor$ and $r = i - j \cdot \sigma$ $\mathit{rank}(A, i, 1) = \mathit{rank}'(A, i \cdot \mathit{sigma}) + \mathit{max}\{\{k \mid E_j[k] < r\} \cup \{-1\}\} + 1$ Use y-fast trie for second part to get $\mathcal{O}(\log \log \sigma)$ time

Solution for rank/select and access (G-2)

- Divide T in chunks of size of size σ.
- In each chunk C: For each $c \in \Sigma$ (in lex. order) write its occurrences in C. We get a permutation π.
- Also store a bitvector X which contains the number of occurrences decoded in unary.

	e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
$\pi =$	0	4	5	6	1	2	3	4	1	2	3	6	5	0	0	6	1	2	3	4	5

$X =$ 1111010001100010100010101011111001111000001

$ _ a e m r y $ _ a e m r y $ _ a e m r y

Solution for rank/select and access (G-2)

- *select*(T, i, c): First we determine by *rank* and *select* on A chunk x and the argument j for *select* on C_x.
- *select*$(C_x, j, c) = \pi_X[\mathit{select}(X, c, 1) + j - c]$

	e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
$\pi =$	0	4	5	6	1	2	3	4	1	2	3	6	5	0	0	6	1	2	3	4	5

$X =$ 1111010001100010100010101011111001110000 01
$ _ a e m r y $ _ a e m r y $ _ a e m r y

Solution for rank/select and access (G-2)

- $y = \pi^{-1}(i)$ tells us the corresponding 0 in X.
- Ones before y in X the corresponding character.
- I.e. $select(X, y, 0) - y - 1$

$\pi =$

e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
0	4	5	6	1	2	3	4	1	2	3	6	5	0	0	6	1	2	3	4	5

$X =$ 1111010001100010100010101011111001110000 01

$ _ a e m r y $ _ a e m r y $ _ a e m r y

Solution for rank/select and access (G-2)

- Use X to select the range $[sp..ep]$ of position of c in π.
- Solve predecessor query on $\pi[sp..ep]$

	e	y	y	y	m	m	m	m	_	_	_	$	e	a	a	r	r	r	r	r	a
$\pi =$	0	4	5	6	1	2	3	4	1	2	3	6	5	0	0	6	1	2	3	4	5

```
X = 111101000110001010001010101111100111000001
    $_ae m   ry   $ _   a e m ry$_a  emr     y
```