Rank and Select on Degenerate Strings

Philip Bille
Inge Li Gørtz
Tord Joakim Stordalen

Rank and Select on Degenerate Strings

$$X = \left\{\begin{matrix} A \\ C \\ G \end{matrix}\right\} \left\{\begin{matrix} A \\ T \end{matrix}\right\} \left\{\begin{matrix} C \end{matrix}\right\} \left\{\begin{matrix} T \\ G \end{matrix}\right\}$$

$$X_1 \quad X_2 \quad X_3 \quad X_4$$

Rank and Select on Degenerate Strings

$$X = \begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \begin{Bmatrix} A \\ T \end{Bmatrix} \begin{Bmatrix} C \end{Bmatrix} \begin{Bmatrix} T \\ G \end{Bmatrix}$$

$X_1 \quad X_2 \quad X_3 \quad X_4$

Rank and Select on Degenerate Strings

$$X = \left\{\begin{matrix} A \\ C \\ G \end{matrix}\right\} \quad \left\{\begin{matrix} A \\ T \end{matrix}\right\} \quad \left\{\begin{matrix} C \end{matrix}\right\} \quad \left\{\begin{matrix} T \\ G \end{matrix}\right\}$$

$$X_1 \quad X_2 \quad X_3 \quad X_4$$

- subset-rank(3,*A*)

$$X = \begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \begin{Bmatrix} A \\ T \end{Bmatrix} \begin{Bmatrix} C \end{Bmatrix} \begin{Bmatrix} T \\ G \end{Bmatrix}$$

$$X_1 \quad X_2 \quad X_3 \quad X_4$$

- subset-rank(3,A) = 2

Rank and Select on Degenerate Strings

$$X = \begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \begin{Bmatrix} A \\ T \end{Bmatrix} \begin{Bmatrix} C \end{Bmatrix} \begin{Bmatrix} T \\ G \end{Bmatrix}$$

$$X_1 \quad X_2 \quad X_3 \quad X_4$$

- subset-rank(3,A) = 2
- subset-select(2, C)

Rank and Select on Degenerate Strings

$$X = \begin{Bmatrix} \color{red}{A} \\ \color{blue}{C} \\ G \end{Bmatrix} \begin{Bmatrix} \color{red}{A} \\ T \end{Bmatrix} \begin{Bmatrix} \color{blue}{C} \end{Bmatrix} \begin{Bmatrix} T \\ G \end{Bmatrix}$$

$$\quad X_1 \quad X_2 \quad X_3 \quad X_4$$

- subset-rank(3,A) = 2
- subset-select(2, C) = 3

Rank and Select on Degenerate Strings

$$X = \begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \begin{Bmatrix} A \\ T \end{Bmatrix} \begin{Bmatrix} C \end{Bmatrix} \begin{Bmatrix} T \\ G \end{Bmatrix}$$

$$X_1 \quad X_2 \quad X_3 \quad X_4$$

- $n = \#\text{sets} = 4$
- $n_0 = \#\text{empty sets} = 0$
- $N = \#\text{characters} = 8$
- subset-rank$(i, c) = \#$occurrences of c in $X_1, \ldots, X_i$.
- subset-select$(i, c) =$ position of ith occurrence of c.

Motivation

- Alanko, Biagi, Puglisi, Vuohtoniemi, 2023 - *Subset Wavelet Trees*
  $O(\log \sigma)$ time, $N \log \sigma + 2n_0 + o(N \log \sigma + n_0)$ bits

- Alanko, Puglisi, Vuohtoniemi, 2023 - *Small searchable k-spectra [...]*
  Support k-mer queries using $2k$ subset-rank queries; one-two orders of magnitude faster than previous best solution.

- A number of reductions to regular rank and select, from APV2023, the famous BOSS paper, etc.

Our Contributions

- Introduce N as a parameter and reanalyze existing reductions
- Prove a simple lower bound on space
- A new compact and fast data structure using SIMD

Reductions

Rank-select structure $\mathcal{D}$

- $\mathcal{D}_b(\ell, \sigma)$ bits
- $\mathcal{D}_r(\ell, \sigma)$ rank-time
- $\mathcal{D}_s(\ell, \sigma)$ select-time

Rank-select structure $\mathcal{B}$

- $\mathcal{B}_b(n, n_0)$ bits
- $\mathcal{B}_r(n, n_0)$ rank(·, *1*)-time
- $\mathcal{B}_s(n, n_0)$ select(·, *0*)-time

	Space	subset-rank	subset-select
i†			
ii			
iii			

†: No empty sets in reduction (i)

Reduction (i)

$$\begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \quad \begin{Bmatrix} A \\ T \end{Bmatrix} \quad \begin{Bmatrix} C \end{Bmatrix} \quad \begin{Bmatrix} T \\ G \end{Bmatrix}$$

S = ACG AT C TG

R = 100 10 1 10 1

Reduction (i)

$$\begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \quad \begin{Bmatrix} A \\ T \end{Bmatrix} \quad \begin{Bmatrix} C \end{Bmatrix} \quad \begin{Bmatrix} T \\ G \end{Bmatrix}$$

S = ACG AT C TG

R = 100 10 1 10 1

- $|S| = N$ and $|R| = N + 1$.

Reduction (i)

$$\left\{\begin{matrix} A \\ C \\ G \end{matrix}\right\} \quad \left\{\begin{matrix} A \\ T \end{matrix}\right\} \quad \left\{\begin{matrix} C \end{matrix}\right\} \quad \left\{\begin{matrix} T \\ G \end{matrix}\right\}$$

S = `ACG AT C TG`
R = `100 10 1 10 1`

- $|S| = N$ and $|R| = N + 1$.
- Build $\mathcal{D}$ over S and a constant-time rank-select bitvector over R.

Reduction (i)

$$\begin{Bmatrix} \mathit{A} \\ \mathit{C} \\ \mathit{G} \end{Bmatrix} \quad \begin{Bmatrix} \mathit{A} \\ \mathit{T} \end{Bmatrix} \quad \begin{Bmatrix} \mathit{C} \end{Bmatrix} \quad \begin{Bmatrix} \mathit{T} \\ \mathit{G} \end{Bmatrix}$$

S = ACG AT C TG

R = 100 10 1 10 1

- $|S| = N$ and $|R| = N + 1$.
- Build $\mathcal{D}$ over S and a constant-time rank-select bitvector over R.
- subset-rank(i, c):
  - $k =$ select$(i + 1, \mathit{1})$ in R
  - return rank$(k - 1, c)$ in S

Reduction (i)

$$\begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \quad \begin{Bmatrix} A \\ T \end{Bmatrix} \quad \begin{Bmatrix} C \end{Bmatrix} \quad \begin{Bmatrix} T \\ G \end{Bmatrix}$$

S = ACG AT C TG
R = 100 10 1 10 1

- $|S| = N$ and $|R| = N + 1$.
- Build $\mathcal{D}$ over S and a constant-time rank-select bitvector over R.
- subset-rank(i, c):
    - $k =$ select($i + 1$, 1) in R
    - return rank($k - 1, c$) in S
- subset-select(i, c):
    - $k =$ select(i, c) in S
    - return rank($k, 1$) in R

Reductions

Rank-select structure $\mathcal{D}$

- $\mathcal{D}_b(\ell, \sigma)$ bits
- $\mathcal{D}_r(\ell, \sigma)$ rank-time
- $\mathcal{D}_s(\ell, \sigma)$ select-time

Rank-select structure $\mathcal{B}$

- $\mathcal{B}_b(n, n_0)$ bits
- $\mathcal{B}_r(n, n_0)$ rank(·, `1`)-time
- $\mathcal{B}_s(n, n_0)$ select(·, `0`)-time

	Space	subset-rank	subset-select
i†	$\mathcal{D}_b(N, \sigma) + N + o(N)$	$\mathcal{D}_r(N, \sigma) + O(1)$	$\mathcal{D}_s(N, \sigma) + O(1)$
ii			
iii			

†: No empty sets in reduction (i)

Reductions (ii) and (iii)

$$\begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \quad \begin{Bmatrix} A \\ T \end{Bmatrix} \quad \begin{Bmatrix} \\ \\ \end{Bmatrix} \quad \begin{Bmatrix} T \\ G \end{Bmatrix}$$

S =	ACG	AT	!!	TG	
R =	100	10	1	10	1

- Empty sets $\rightarrow$ we cannot build R in the same way

Reductions (ii) and (iii)

$$\begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \quad \begin{Bmatrix} A \\ T \end{Bmatrix} \quad \begin{Bmatrix} \quad \end{Bmatrix} \quad \begin{Bmatrix} T \\ G \end{Bmatrix}$$

S =	ACG	AT	!!	TG	
R =	100	10	1	10	1

- Empty sets $\rightarrow$ we cannot build R in the same way
- Reduction (ii): Replace $\{\}$ by $\{\epsilon\}$ $\rightarrow N' = N + n_0$ and $\sigma' = \sigma + 1$

Reductions (ii) and (iii)

$$\begin{Bmatrix} A \\ C \\ G \end{Bmatrix} \begin{Bmatrix} A \\ T \end{Bmatrix} \begin{Bmatrix} \\ \end{Bmatrix} \begin{Bmatrix} T \\ G \end{Bmatrix}$$

S =	ACG	AT	!!	TG	
R =	100	10	1	10	1

- Empty sets $\rightarrow$ we cannot build R in the same way
- Reduction (ii): Replace $\{\}$ by $\{\epsilon\} \rightarrow N' = N + n_0$ and $\sigma' = \sigma + 1$
- Reduction (iii):
  - iLet E be the length-n bitstring where $E[i] = 1$ iff $X_i = \emptyset$
  - Let X' be X with the empty sets removed
  - Build (i) over X' and $\mathcal{B}$ over E
  - For queries, filter out empty sets first, then use (i)

Reductions

Rank-select structure $\mathcal{D}$

- $\mathcal{D}_b(\ell, \sigma)$ bits
- $\mathcal{D}_r(\ell, \sigma)$ rank-time
- $\mathcal{D}_s(\ell, \sigma)$ select-time

Rank-select structure $\mathcal{B}$

- $\mathcal{B}_b(n, n_0)$ bits
- $\mathcal{B}_r(n, n_0)$ rank(·, `1`)-time
- $\mathcal{B}_s(n, n_0)$ select(·, `0`)-time

	Space	subset-rank	subset-select
i†	$\mathcal{D}_b(N, \sigma) + N + o(N)$	$\mathcal{D}_r(N, \sigma) + O(1)$	$\mathcal{D}_s(N, \sigma) + O(1)$
ii	same as (i) with $N' = N + n_0$ and $\sigma' = \sigma + 1$		
iii	(i) + $\mathcal{B}_b(n, n_0)$	(i) + $\mathcal{B}_r(n, n_0)$	(i) + $\mathcal{B}_s(n, n_0)$

†: No empty sets in reduction (i)

Plugging in

- Bitvector by Golynski, Munro, Rao, 2006
  - $\ell \log \sigma + o(\ell \log \sigma)$ bits
  - rank in $O(\log \log \sigma)$ time
  - select in constant time

Plugging in

- Bitvector by Golynski, Munro, Rao, 2006
  - $\ell \log \sigma + o(\ell \log \sigma)$ bits
  - rank in $O(\log \log \sigma)$ time
  - select in constant time

(i) $N \log \sigma + N + o(n \log \sigma)$ bits, subset-rank in $O(\log \log \sigma)$ time, subset-select in constant time.

Plugging in

- Bitvector by Golynski, Munro, Rao, 2006
  - $\ell \log \sigma + o(\ell \log \sigma)$ bits
  - rank in $O(\log \log \sigma)$ time
  - select in constant time

(i) $N \log \sigma + N + o(n \log \sigma)$ bits, subset-rank in $O(\log \log \sigma)$ time, subset-select in constant time.

(ii) $(N + n_0) \log(\sigma + 1) + N + n_0 + o((N + n_0) \log \sigma)$ bits, same time bound.

Plugging in

- Bitvector by Golynski, Munro, Rao, 2006
  - $\ell \log \sigma + o(\ell \log \sigma)$ bits
  - rank in $O(\log \log \sigma)$ time
  - select in constant time

(i) $N \log \sigma + N + o(n \log \sigma)$ bits, subset-rank in $O(\log \log \sigma)$ time, subset-select in constant time.

(ii) $(N + n_0) \log(\sigma + 1) + N + n_0 + o((N + n_0) \log \sigma)$ bits, same time bound.

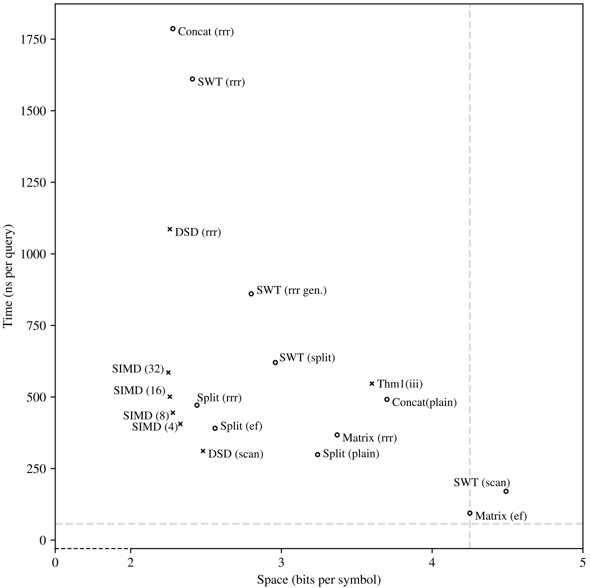
(iii) $N \log \sigma + \mathcal{B}_s(n, n_0)$ bits, and an additional rank or select query in $\mathcal{B}$. In particular, if $n = o(N \log \sigma)$ we can use $n + o(n)$ extra bits and constant time per query to achieve the same results as (i).

Lower Bound and Succinctness

- Let sufficiently large N and $\sigma = \omega(\log N)$ be given
- Assume wlog. that $\log N$ and $N/\log N$ are integers.
- Consider the class $X_1, \ldots, X_n$ where each X_i has size $\log N$ and $n = N/\log N$.
- There are $\binom{\sigma}{\log N}^{N/\log N}$ such strings

$$\begin{aligned}
\log \binom{\sigma}{\log N}^{N/\log N} &= \frac{N}{\log N} \log \binom{\sigma}{\log N} \\
&\geq \frac{N}{\log N} \log \left(\frac{\sigma - \log N}{\log N}\right)^{\log N} \\
&= N \log \left(\frac{\sigma - \log N}{\log N}\right) \\
&= N \log \sigma - o(N \log \sigma)
\end{aligned}$$

Empirical Results - subset-rank Queries



SIMD Implementation

- Standard idea from succinct data structures;
    - Divide string into blocks
    - Precompute the answer for rank queries up to each block ($\sigma = 4$)
    - Compute in-block answers as needed

SIMD Implementation

- Standard idea from succinct data structures;
  - Divide string into blocks
  - Precompute the answer for rank queries up to each block ($\sigma = 4$)
  - Compute in-block answers as needed
- With SIMD: larger blocks $\rightarrow$ smaller data structures

SIMD Implementation

- Standard idea from succinct data structures;
  - Divide string into blocks
  - Precompute the answer for rank queries up to each block ($\sigma = 4$)
  - Compute in-block answers as needed
- With SIMD: larger blocks $\rightarrow$ smaller data structures
- How we use SIMD to answer rank queries in a block?

Rank queries using SIMD

- Split the string into two strings; the high bits and the low bits

  00 *01* *10* *11* ⟶ *0 0 1 1*
  A C G T

  0 1 0 1
  A C G T

- To perform rank(i,*C*), we could scan the two strings looking for a *0* in the hi bit and a *1* in the low bit.

- With SIMD, we can use the operation ***vpternlogq***

vpternlogq

- Given three SIMD vectors and an 8-bit immediate value, computes *any* three-variable boolean function

$A =$	0	0	1	0	0	0	1	0
$B =$	0	1	1	0	1	1	0	0
$C =$	?	?	?	?	?	?	?	?

a	b	c	imm	C	A	A & C
0	0	0	b_0			
0	0	1	b_1			
0	1	0	b_2			
0	1	1	b_3			
1	0	0	b_4			
1	0	1	b_5			
1	1	0	b_6			
1	1	1	b_7			

vpternlogq

- Given three SIMD vectors and an 8-bit immediate value, computes *any* three-variable boolean function

$A =$	0	0	1	0	0	0	1	0
$B =$	0	1	1	0	1	1	0	0
$C =$	?	?	?	?	?	?	?	?

$R =$	0	1	0	0	1	1	0	0

a	b	c	imm	*C*	*A*	*A* & *C*
0	0	0	b_0	0		
0	0	1	b_1	0		
0	1	0	b_2	1		
0	1	1	b_3	1		
1	0	0	b_4	0		
1	0	1	b_5	0		
1	1	0	b_6	0		
1	1	1	b_7	0		

vpternlogq

- Given three SIMD vectors and an 8-bit immediate value, computes *any* three-variable boolean function

$A =$ `0 0 1 0 0 0 1 0`

$B =$ `0 1 1 0 1 1 0 0`

$C =$ `? ? ? ? ? ? ? ?`

$R =$ `0 1 0 0 1 1 0 0`

a	b	c	imm	*C*	*A*	*A* & *C*
0	0	0	b_0	0		
0	0	1	b_1	0		
0	1	0	b_2	1		
0	1	1	b_3	1		
1	0	0	b_4	0		
1	0	1	b_5	0		
1	1	0	b_6	0		
1	1	1	b_7	0		

Use *vpopcntq* (population count) to count the number of matches in the result!

vpternlogq

- Given three SIMD vectors and an 8-bit immediate value, computes *any* three-variable boolean function

$A =$ `0 0 1 0 0 0 1 0`

$B =$ `0 1 1 0 1 1 0 0`

$C =$ `? ? ? ? ? ? ? ?`

$R =$ `0 1 0 0 1 1 0 0`

a	b	c	imm	*C*	*A*	*A* & *C*
0	0	0	b_0	0	1	
0	0	1	b_1	0	1	
0	1	0	b_2	1	0	
0	1	1	b_3	1	0	
1	0	0	b_4	0	0	
1	0	1	b_5	0	0	
1	1	0	b_6	0	0	
1	1	1	b_7	0	0	

Use ***vpopcntq*** (population count) to count the number of matches in the result!

vpternlogq

- Given three SIMD vectors and an 8-bit immediate value, computes *any* three-variable boolean function

$A =$ `0 0 1 0 0 0 1 0`

$B =$ `0 1 1 0 1 1 0 0`

$C =$ `? ? ? ? ? ? ? ?`

$R =$ `0 1 0 0 1 1 0 0`

a	b	c	imm	*C*	*A*	*A* & *C*
0	0	0	b_0	0	1	1
0	0	1	b_1	0	1	1
0	1	0	b_2	1	0	1
0	1	1	b_3	1	0	1
1	0	0	b_4	0	0	0
1	0	1	b_5	0	0	0
1	1	0	b_6	0	0	0
1	1	1	b_7	0	0	0

Use *`vpopcntq`* (population count) to count the number of matches in the result!

Questions?