EPR-dictionaries:

A practical and fast data structure for constant time searches in unidirectional and bidirectional FM-indices

Christopher Pockrandt, Marcel Ehrhardt, Knut Reinert

Free University of Berlin, Max-Planck-Institute for Molecular Genetics

FM-Index (by Ferragina and Manzini)

T = mississippi$

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

sort

S = suffix array
F = First column
L = Last column = BWT

i	
1	mississippi$
2	ississippi$m
3	ssissippi$mi
4	sissippi$mis
5	issippi$miss
6	ssippi$missi
7	sippi$missis
8	ippi$mississ
9	ppi$mississi
10	pi$mississip
11	i$mississipp
12	$mississippi

FM-Index (by Ferragina and Manzini)

T = mississippi$

S = suffix array
F = First column
L = Last column = BWT

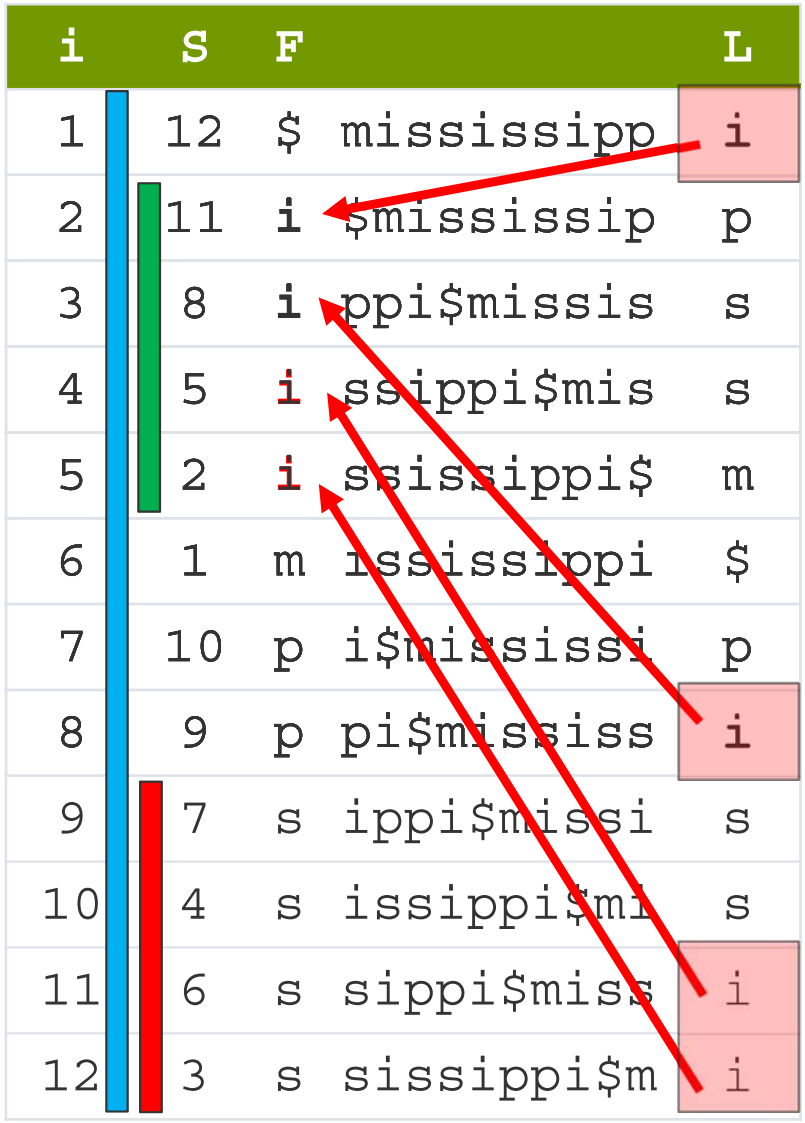
i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

1st p in F corresponds to 1st p in L
2nd i in F corresponds to 2nd i in L
...

We also need a Count table C

c	Count(c)
$	1
i	5
m	6
p	8
s	12

Backward search



i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i sissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

T = mississippi$

Pattern P = “is”

[1,12] corresponds to ε

[9,12] corresponds to “s”

Backward search

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

T = mississippi$

Pattern P = "is"

[1,12] corresponds to ε

[9,12] corresponds to "s"

[4,5] corresponds to "is"

rank_i(8) = 2

Bit vectors

BWT	i	p	s	s	m	$	p	i	s	s	i	i
$	0	0	0	0	0	1	0	0	0	0	0	0
i	1	0	0	0	0	0	0	1	0	0	1	1
m	0	0	0	0	1	0	0	0	0	0	0	0
p	0	1	0	0	0	0	1	0	0	0	0	0
s	0	0	1	1	0	0	0	0	1	1	0	0

Constant time rank support for each bit vector (Jacobson)

	block 1				block 2				block 3			
i	1	0	0	0	0	0	0	1	0	0	1	1

block #	rank
1	1
2	2
3	4

Bit vectors

BWT	i	p	s	s	m	$	p	i	s	s	i	i
$	0	0	0	0	0	1	0	0	0	0	0	0
i	1	0	0	0	0	0	0	1	0	0	1	1
m	0	0	0	0	1	0	0	0	0	0	0	0
p	0	1	0	0	0	0	1	0	0	0	0	0
s	0	0	1	1	0	0	0	0	1	1	0	0

Constant time rank support for each bit vector (Jacobson)

	block 1				block 2				block 3			
i	1	0	0	0	0	0	0	1	0	0	1	1

block #	rank
1	1
2	2
3	4

rank_i(11) = **2** + **1**

table lookup + in-block query

Bit vectors

BWT	i	p	s	s	m	$	p	i	s	s	i	i
$	0	0	0	0	0	1	0	0	0	0	0	0
i	1	0	0	0	0	0	0	1	0	0	1	1
m	0	0	0	0	1	0	0	0	0	0	0	0
p	0	1	0	0	0	0	1	0	0	0	0	0
s	0	0	1	1	0	0	0	0	1	1	0	0

Constant time rank support for each bit vector (Jacobson)

	block 1				block 2				block 3			
i	1	0	0	0	0	0	0	1	0	0	1	1

block #	rank
1	1
2	2
3	4

Only $o(n)$ additional space per char

In total $n \cdot \sigma + o(n \cdot \sigma)$ space

Bidirectional search (Lam et al)

REINERT LAB GENOMICS. PROTEOMICS. ALGORITHMS

Freie Universität Berlin

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F	L
1	12	$	mississipp	i
2	11	i	$mississip	p
3	8	i	ppi$missis	s
4	5	i	ssippi$mis	s
5	2	i	ssissippi$	m
6	1	m	ississippi	$
7	10	p	i$mississi	p
8	9	p	pi$mississ	i
9	7	s	ippi$missi	s
10	4	s	issippi$mi	s
11	6	s	sippi$miss	i
12	3	s	sissippi$m	i

i	S'	F'	L'
1	12	$	ippississi	m
2	10	i	m$ippissis	s
3	1	i	ppississim	$
4	7	i	ssim$ippis	s
5	4	i	ssissim$ip	p
6	11	m	$ippississ	i
7	3	p	ississim$i	p
8	2	p	pississim$	i
9	9	s	im$ippissi	s
10	6	s	issim$ippi	s
11	8	s	sim$ippiss	i
12	5	s	sissim$ipp	i

Bidirectional search (Lam et al)

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

i	S'	F'	L'
1	12	$ ippississi	m
2	10	i m$ippissis	s
3	1	i ppississim	$
4	7	i ssim$ippis	s
5	4	i ssissim$ip	p
6	11	m $ippississ	i
7	3	p ississim$i	p
8	2	p pississim$	i
9	9	s im$ippissi	s
10	6	s issim$ippi	s
11	8	s sim$ippiss	i
12	5	s sissim$ipp	i

Bidirectional search (Lam et al)

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F		L
1	12	$	mississipp	i
2	11	i	$mississip	p
3	8	i	ppi$missis	s
4	5	i	ssippi$mis	s
5	2	i	ssissippi$	m
6	1	m	ississippi	$
7	10	p	i$mississi	p
8	9	p	pi$mississ	i
9	7	s	ippi$missi	s
10	4	s	issippi$mi	s
11	6	s	sippi$miss	i
12	3	s	sissippi$m	i

i	S'	F'		L'
1	12	$	ippississi	m
2	10	i	m$ippissis	s
3	1	i	ppississim	$
4	7	i	ssim$ippis	s
5	4	i	ssissim$ip	p
6	11	m	$ippississ	i
7	3	p	ississim$i	p
8	2	p	pississim$	i
9	9	s	im$ippissi	s
10	6	s	issim$ippi	s
11	8	s	sim$ippiss	i
12	5	s	sissim$ipp	i

Bidirectional search (Lam et al)

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

i	S′	F′	L′
1	12	$ ippississi	m
2	10	i m$ippissis	s
3	1	i ppississim	$
4	7	i ssim$ippis	s
5	4	i ssissim$ip	p
6	11	m $ippississ	i
7	3	p ississim$i	p
8	2	p pississim$	i
9	9	s im$ippissi	s
10	6	s issim$ippi	s
11	8	s sim$ippiss	i
12	5	s sissim$ipp	i

Bidirectional search (Lam et al)

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

i	S'	F'	L'
1	12	$ ippississi	m
2	10	i m$ippissis	s
3	1	i ppississim	$
4	7	i ssim$ippis	s
5	4	i ssissim$ip	p
6	11	m $ippississ	i
7	3	p ississim$i	p
8	2	p pississim$	i
9	9	s im$ippissi	s
10	6	s issim$ippi	s
11	8	s sim$ippiss	i
12	5	s sissim$ipp	i

Bidirectional search (Lam et al)

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8			
9			
10			
11	6	s sippi$miss	i
12	3	s sissippi$m	i

i	S′	F′	L′
1	12	$ ippississi	m
2	10	i m$ippissis	s
3	1	i ppississim	$
4	7	i ssim$ippis	s
5	4	i ssissim$ip	p
6	11	m $ippississ	i
7	3	p ississim$i	p
8			
9			
10			
11	8	s sim$ippiss	i
12	5	s sissim$ipp	i

A forward search takes

O(Σ) backward searches / rank queries!

Bidirectional search (Lam et al)

T = mississippi$ P = is T^{rev} = ippissisim$

i	S	F	L
1	12	$ mississipp	i
2	11	i $mississip	p
3	8	i ppi$missis	s
4	5	i ssippi$mis	s
5	2	i ssissippi$	m
6	1	m ississippi	$
7	10	p i$mississi	p
8	9	p pi$mississ	i
9	7	s ippi$missi	s
10	4	s issippi$mi	s
11	6	s sippi$miss	i
12	3	s sissippi$m	i

i	S′	F′	L′
1	12	$ ippississi	m
2	10	i m$ippissis	s
3	1	i ppississim	$
4	7	i ssim$ippis	s
5	4	i ssissim$ip	p
6	11	m $ippississ	i
7	3	p ississim$i	p
8	2	p pississim$	i
9	9	s im$ippissi	s
10	6	s issim$ippi	s
11	8	s sim$ippiss	i
12	5	s sissim$ipp	i

Prefixsum Rank queries

Observation: A forward search can be done with O(1) backward-searches using less space (compared to Lam et al., 2009).

→ **Use prefix sums !**

BWT	m	s	$	s	p	i	p	i	s	s	i	i
$	0	0	1	0	0	0	0	0	0	0	0	0
i	0	0	1	0	0	1	0	1	0	0	1	1
m	1	0	1	0	0	1	0	1	0	0	1	1
p	**1**	**0**	**1**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**
~~s~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	~~1~~

Disadvantage : $n \cdot (\sigma - 1) + o(n \cdot \sigma)$ space

Use bit representation of BWT

	block 1				block 2				block 3			
BWT	A	C	A	T	G	T	A	C	A	G	C	A
	00	01	00	11	10	11	00	01	00	10	01	00

→Needs only $n \cdot \log \sigma + o(n \cdot \log \sigma \cdot \sigma)$

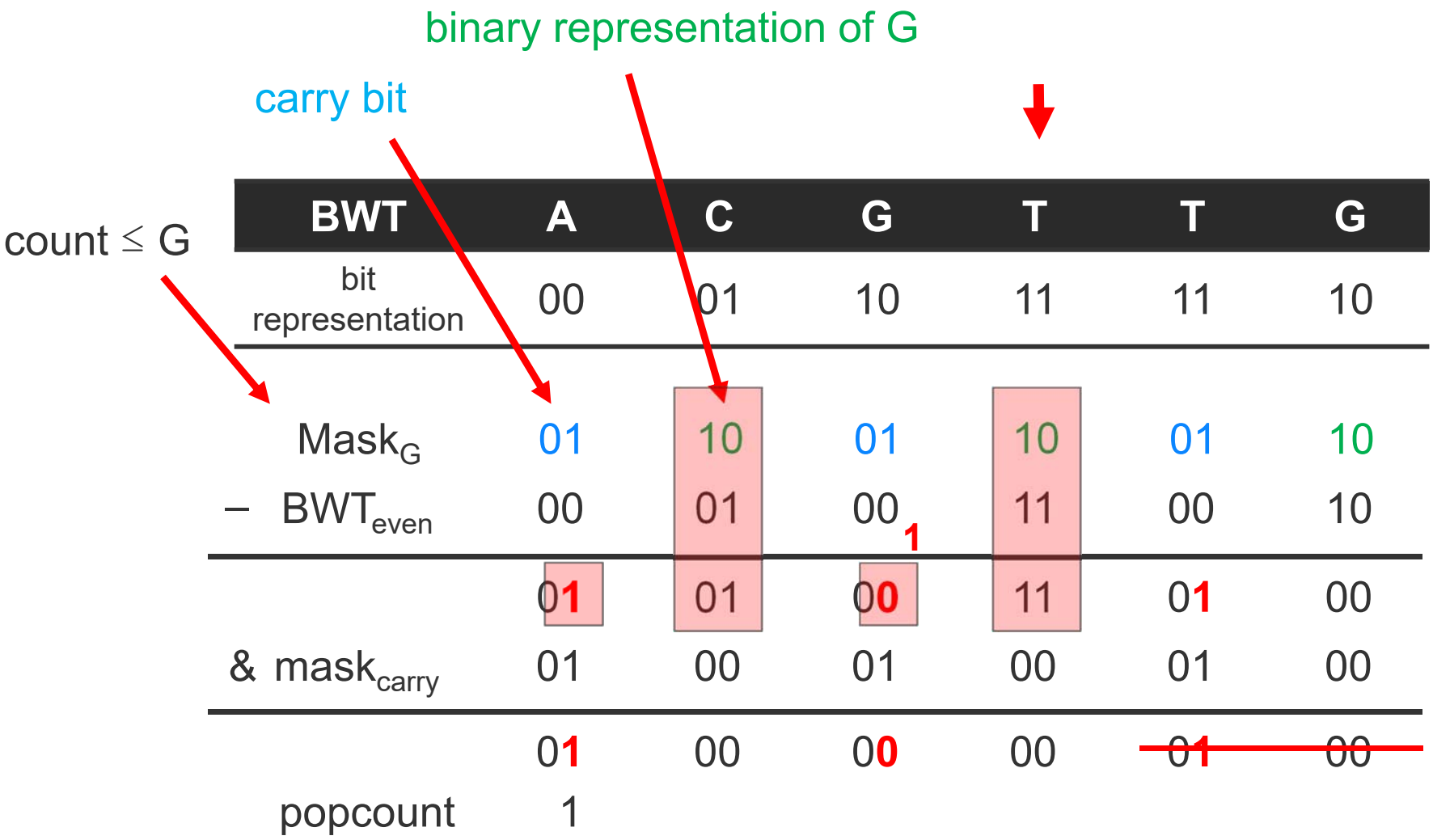
count ≤ G

id	rank ≤ A	rank ≤ C	rank ≤ G	rank ≤ T
1	2	3	3	4
2	3	5	6	8
3	5	8	9	12

How to count in-block query in constant time?

Rank queries (in-block query)

Example: Count **even** occurrences of characters ≤ G



Comparison to wavelet trees

REINERT LAB GENOMICS. PROTEOMICS. ALGORITHMS

Freie Universität Berlin

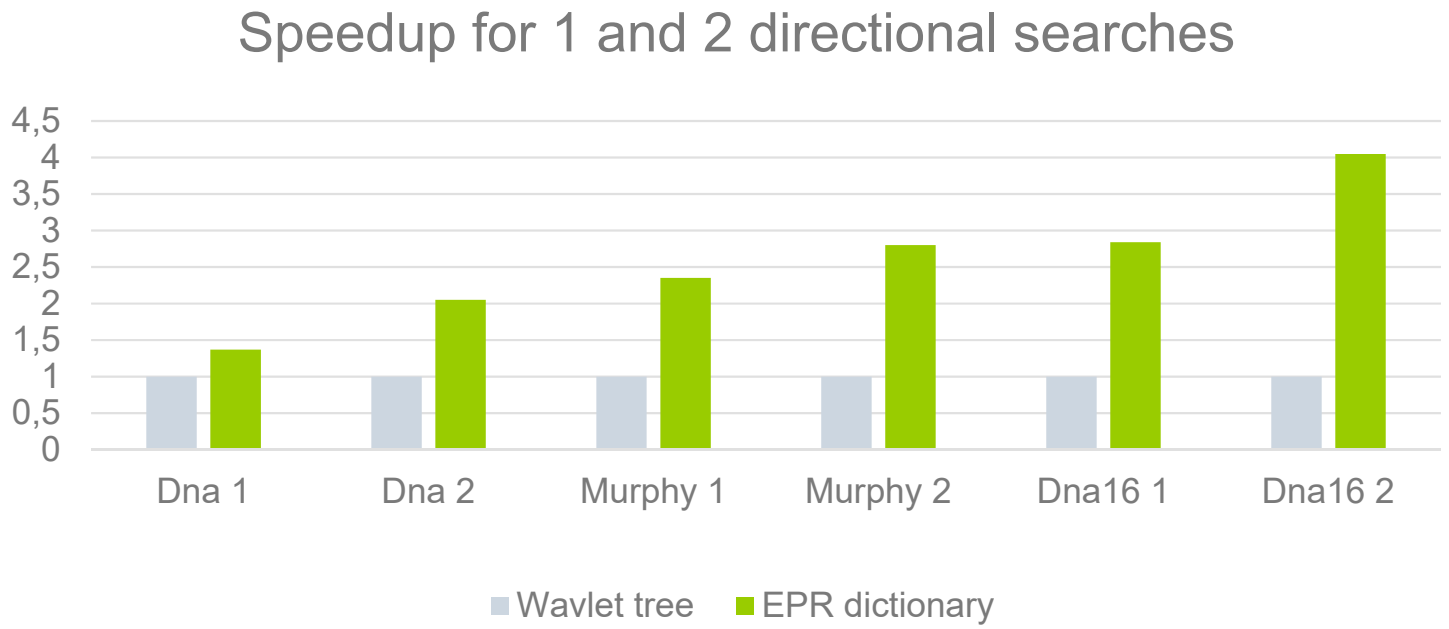
Enhanced Prefixsum Rank dictionaries

- backward and forward searches: $O(1)$
- $O(n \cdot \log \sigma) + o(n \cdot \sigma \cdot \log \sigma)$ space consumption

Wavelet trees

- backward and forward searches: $O(\log \sigma)$
- $O(n \cdot \log \sigma)$ space consumption

Runtime comparisons



Index	Dna		Murphy10		Dna16	
	time	factor	time	factor	time	factor
WaveletTree	20.73s	1.00	52.35s	1.00	66.45s	1.00
EPR	**15.09s**	**1.37**	**22.27s**	**2.35**	**23.39s**	**2.84**
2WaveletTree	41.21s	1.00	66.59s	1.00	98.74s	1.00
2EPR	**20.09s**	**2.05**	**23.80s**	**2.80**	**24.39s**	**4.05**

1 million queries of length 200, sampled from a text of length 10^8

Space consumption

Index	Dna	Murphy10	Dna16
WaveletTree	30	51	60
EPR	**42**	**156**	**227**

rank data structure in Megabytes

Conclusion

We presented a practical rank dictionary that

- has the same big O space consumption as wavelet trees
- allows for constant time backward and forward searches
- has a speed-up factor between 2 and 4
- ~17% more space for DNA (CSA with sampling rate of 10)

- Available in the SeqAn library (www.seqan.de)
- Funding from the MPI for molecular genetics

Thank you for your attention

Runtime comparisons

Index	Dna		Murphy10		Dna16	
	time	factor	time	factor	time	factor
WaveletTree	20.73s	1.00	52.35s	1.00	66.45s	1.00
EPR	**15.09s**	**1.37**	**22.27s**	**2.35**	**23.39s**	**2.84**
2WaveletTree	41.21s	1.00	66.59s	1.00	98.74s	1.00
2SDSL	43.54s	0.95	74.69s	0.89	89.07s	1.11
2Schnattinger	59.59s	0.69	91.30s	0.73	107.04s	0.92
2EPR	**20.09s**	**2.05**	**23.80s**	**2.80**	**24.39s**	**4.05**

1 million queries of length 200, sampled from a text of length 10^8

Space consumption

Index	Dna	Murphy10	Dna16	Protein
EPR	**42**	**156**	**227**	**478**
WaveletTree	30	51	60	72
2EPR	**84**	**311**	**454**	**955**
2WaveletTree	60	102	120	144
2SDSL	68	105	122	145
Schnattinger	75	108	123	146

rank data structure in Megabytes