

# Advanced Data Structures

## Lecture 08: Compressed Suffix Array

Florian Kurpicz

The slides are licensed under a Creative Commons Attribution-ShareAlike 4.0 International License © ⓘ ⓘ: [www.creativecommons.org/licenses/by-sa/4.0](http://www.creativecommons.org/licenses/by-sa/4.0) | commit c70729e compiled at 2023-06-19-13:15



<https://pingo.scc.kit.edu/915810>

# Recap: Suffix Array

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	b	b	b	b	c
			c	\$	b	b		a	\$	b	a	\$	a
			a		a	c		b		a	a		b
			b		\$	a		c		\$	b		b
			c			b		a			b		a
			a			b		b			a		\$
			b			a		b					
			b			a		b					
			a			\$		a					
			\$					\$					

# Recap: Suffix Array

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	b	c	\$	b	a	a	a	b	b
			b	a	c	a		c	b	b	b	b	c
			c	\$	b	b		a	b	b	c	a	a
			a		a	c		b	c	a	a	\$	b
			b		a	a		c	a	b	b		b
			c		\$	b		a	b	a	a		a
			a			a		b	b	a			\$
			b			a		b	b	a			
			b			\$		a					
			a										
			\$										

- space:  $O(n \log n)$  bits
- space text:  $n \lceil \log \sigma \rceil$  bits

# Recap: Suffix Array

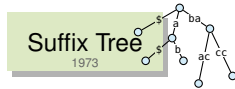
	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
\$	a	a	a	a	a	a	b	b	b	b	b	c	c
	\$	b	b	b	b	b	a	a	b	c	c	a	a
		a	b	a	c	c	\$	b	a	a	a	b	b
		b	a	c	a	a		c	b	b	b	b	c
		c	\$	b	b	b		a	b	c	a	a	a
		a		b	c	c		b	\$	a	\$	b	b
		b		a	a	b		c		b	b	a	a
		c		\$	b	b		a		b	a	\$	\$
		a			b	a		b		a			
		b				a		b					
		b				\$		a					
		a						\$					
		\$											

- space:  $O(n \log n)$  bits
- space text:  $n \lceil \log \sigma \rceil$  bits

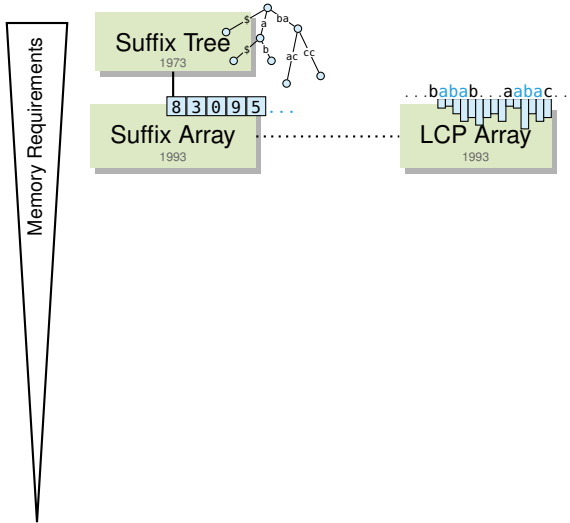
- better: index requiring same space as text
- even better: index requiring same space as compressed text

# (Compressed) Text Indices #Ad

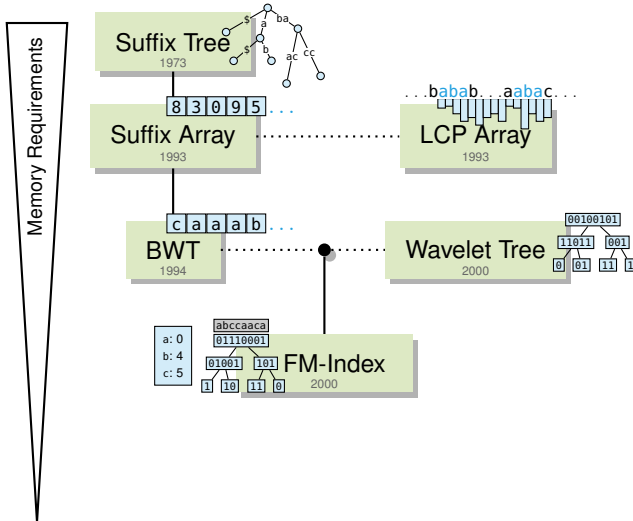
Memory Requirements



# (Compressed) Text Indices #Ad

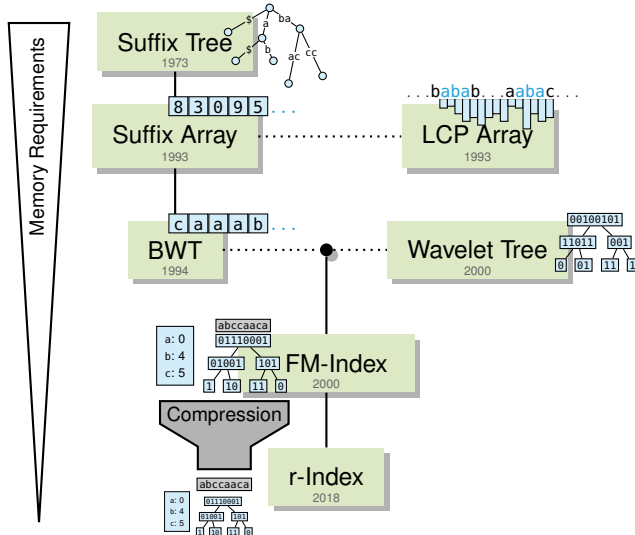


# (Compressed) Text Indices #Ad

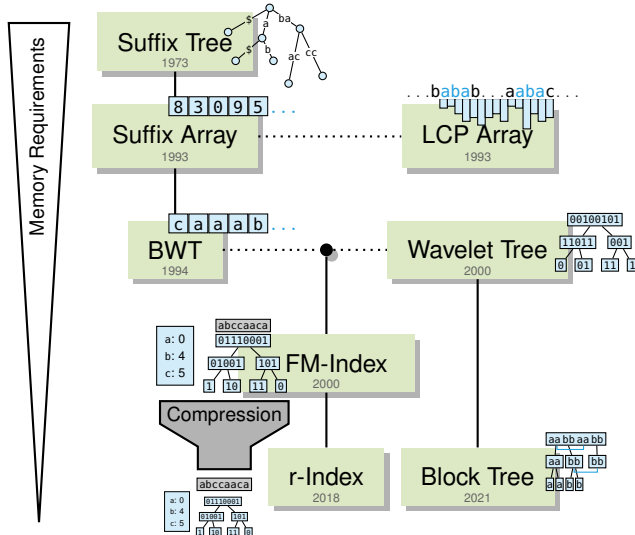




# (Compressed) Text Indices #Ad



# (Compressed) Text Indices #Ad



# Ψ Function

## Definition: Ψ Function

Given a suffix array  $SA$  of length  $n$ ,

$$\Psi(i) = SA^{-1}[SA[i] + 1]$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		a	c	\$	a
			a		a	c		b		b	a		b
			b		\$	a		c		\$			a
			c			b		a			b		b
			a			b		b			a		a
			b			a		a					\$
			a			\$		b					
			\$					\$					

# Ψ Function

## Definition: Ψ Function

Given a suffix array  $SA$  of length  $n$ ,

$$\Psi(i) = SA^{-1}[SA[i] + 1]$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	a	b	b	b	c
			c	\$	b	b		a	b	a	c	a	a
			a		a	c		b	c	a	b	\$	b
			b		\$	a		c		\$	a		a
			c			b		a			b		b
			a			b		b			a		a
			b			a		a			\$		\$
			a			\$		b					b
			\$					\$					a

# Ψ Function

## Definition: Ψ Function

Given a suffix array  $SA$  of length  $n$ ,

$$\Psi(i) = SA^{-1}[SA[i] + 1]$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	a	a	a	a	a
			a	b	c	c	\$	b	b	a	b	b	b
			b	a	a	a		c	a	b	c	a	c
			c	\$	b	b		\$	b	a	b	\$	a
			a		a	c					a		a
			b		\$	a					b		b
			c			b					a		a
			a			a							b
			b			b							a
			a			\$							\$

# Ψ Function

## Definition: Ψ Function

Given a suffix array  $SA$  of length  $n$ ,

$$\Psi(i) = SA^{-1}[SA[i] + 1]$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	a	a	a	a	a
			a	b	c	c	\$	b	b	b	b	b	b
			b	a	a	a		c	a	a	a	a	c
			c	\$	b	b		\$	b	b	b	\$	a
			a		a	c							b
			b		\$	a							a
			c			b							b
			a			a							a
			b			b							\$
			a			a							
			\$			\$							

# Ψ Function

## Definition: Ψ Function

Given a suffix array  $SA$  of length  $n$ ,

$$\Psi(i) = SA^{-1}[SA[i] + 1]$$

- $SA[\Psi(i)] = SA[i] + 1$
- where in  $SA$  is the suffix  $T[SA[i + 1]..n)$
- “successor” function

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	a	b	b	a	c
			c	\$	b	b		a	b	b	c	a	a
			a		a	c		b	c	a	a	\$	b
			b		\$	a				\$	b		b
			c			b		a			b		a
			a			b		b			a		\$
			b			a		a					
			b			\$		b					
			a					\$					

# Ψ Function

## Definition: Ψ Function

Given a suffix array  $SA$  of length  $n$ ,

$$\Psi(i) = SA^{-1}[SA[i] + 1]$$

- $SA[\Psi(i)] = SA[i] + 1$
  - where in  $SA$  is the suffix  $T[SA[i + 1]..n)$
  - “successor” function
- 
- can be used to obtain suffix array
  - can be compressed **i** currently  $O(n \log n)$  bits

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	b	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		b	c	\$	a
			a		a	c		b		a	a		b
			b		\$	a		c		\$	b		b
			c			b		a			b		a
			a			b		b			a		\$
			b			a		a					
			a			\$		b					



# Replacing SA with $\Psi$

- which number does in this example not occur?


	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	a	a	a	a	a
			a	b	c	c	\$	b	b	a	b	b	b
			b	a	a	a		c	a	b	c	a	c
			c	\$	b	b		\$	b	a	b	\$	a
			a		a	c					a		a
			b		\$	b					b		b
			c			a					a		a
			a			b					b		b
			b			a					a		a
			a			\$					\$		\$

# Replacing SA with $\Psi$

- which number does in this example not occur?  
Answer: 3

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	a	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		a	c	\$	a
			a		a	c		b		b	a		b
			b		\$	a		c		\$			a
			c			b		a			b		b
			a			b		b			a		a
			b			a		a					\$
			a			\$		b					


# Replacing SA with $\Psi$

- which number does in this example not occur?  
Answer: 3
- how to obtain  $SA[i]$  using  $\Psi$   PINGO

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	c	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	a	b	b	a	c
			c	\$	b	b		\$	b	a	c	\$	a
			a		a	c					a		b
			b		\$	a					b		a
			c			b					a		b
			a			a							a
			b			b							\$
			a			\$							

# Replacing SA with $\Psi$

- which number does in this example not occur?  
Answer: 3


- how to obtain  $SA[i]$  using  $\Psi$   PINGO

- follow positions until last suffix is found
- last suffix is at position 1
- $n - \#steps$  is SA value
- requires  $O(n)$  time

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	a	b	b	a	c
			c	\$	b	b		a	b	a	c	\$	a
			a		a	c		b	c	a	a		b
			b		\$	a				\$	b		b
			c			b		a			a		a
			a			b		b			a		\$
			b			a		a					b
			a			\$		b					a
								\$					

# Replacing SA with $\Psi$

- which number does in this example not occur?  
Answer: 3

- how to obtain  $SA[i]$  using  $\Psi$   PINGO

- follow positions until last suffix is found
- last suffix is at position 1
- $n - \#steps$  is SA value
- requires  $O(n)$  time

- pattern matching:  $O(mn \log n)$  time
- pattern matching with LCP and RMQ:  
 $O(mn + \log n)$  time

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		b	c	\$	a
			a		a	c		b			a		b
			b		\$	a		c		\$	b		a
			c			b		a			a		b
			a			a		b			\$		a
			b			\$		a					\$
			a					b					
			\$					\$					

# Speeding Up Lookups in $\Psi$ (1/2)

- space SA:  $O(n \log n)$  bits
- space text:  $O(n \log \sigma)$  bits
- space compressed suffix array should not more than text

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		a	a	\$	a
			a		a	c		b		b	b		b
			b		\$	a		c		\$	a		a
			c			b		a			b		b
			a			b		b			a		a
			b			a		a			\$		\$
			a			\$		b					b
			\$					\$					\$

# Speeding Up Lookups in $\Psi$ (1/2)

- space  $SA$ :  $O(n \log n)$  bits
- space text:  $O(n \log \sigma)$  bits
- space compressed suffix array should not more than text


- sample every  $\log n$ -th  $SA$  entry
- $O(n / \log n)$  samples of size  $O(\log n)$  bits
- total space:  $O(n)$  bits

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		b	a	\$	a
			a		a	c		b		a	a		b
			b		\$	a		c		\$	b		b
			c			b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			a			\$		b			\$		\$
			\$					\$					

# Speeding Up Lookups in $\Psi$ (1/2)

- space SA:  $O(n \log n)$  bits
- space text:  $O(n \log \sigma)$  bits
- space compressed suffix array should not more than text

- sample every  $\log n$ -th SA entry
- $O(n / \log n)$  samples of size  $O(\log n)$  bits
- total space:  $O(n)$  bits

- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**


	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		b	c	\$	a
			a		a	c		b		a	a		b
			b		\$	a		c		\$	b		b
			c			b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			a			\$		\$			\$		\$



# Speeding Up Lookups in $\Psi$ (1/2)


- space SA:  $O(n \log n)$  bits
- space text:  $O(n \log \sigma)$  bits
- space compressed suffix array should not more than text

- sample every  $\log n$ -th SA entry
- $O(n / \log n)$  samples of size  $O(\log n)$  bits
- total space:  $O(n)$  bits

- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**


	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		b	c	\$	a
			a		a	a		b		a	a		b
			b		\$	b		c		\$	b		b
			c			b		a			a		a
			a			a		b			b		\$
			b			b		a			a		
			a			a		b			\$		
			\$			\$		\$					

# Speeding Up Lookups in $\Psi$ (2/2)

- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	a	b	b	a	c
			c	\$	b	b		\$	b	a	c	\$	a
			a		a	c		a			a		b
			b		\$	a		b			b		a
			c			b		a			a		b
			a			a		b			a		a
			b			b		a					\$
			a			a		\$					


# Speeding Up Lookups in $\Psi$ (2/2)

- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**


- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	a	b	b	a	c
			c	\$	b	b		a	\$	b	c	\$	a
			a		a	c		b		a	a		b
			b		\$	a		c		\$	b		b
			c			b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			a			\$		b			\$		\$

# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time


- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			a	\$	b	b		a		a	a	a	a
			b		a	a		b		\$	b	\$	b
			c		\$	b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			b			\$		b			\$		\$
			a					a					
			\$					\$					

# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time


- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			c	\$	b	b		a		b	a	a	a
			a		a	c		b		a	a	\$	b
			b		\$	a		c		\$	b		b
			c			b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			a			\$		b			\$		\$
			\$					\$					

# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time


- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			a	\$	b	b		a		a	a	\$	a
			b		a	a		c		\$	b		b
			c		\$	b		a			b		a
			a			b		b			a		b
			b			a		a			\$		a
			a			\$		b					\$
								\$					

# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time


- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			a	\$	b	b		a		a	a	a	a
			b		a	a		b		\$	b	\$	b
			c		\$	b		a			b		b
			a			b		b			a		a
			b			a		a			\$		\$
			b			\$		b					
			a					\$					

# Speeding Up Lookups in $\Psi$ (2/2)

- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**


- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time

- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$


	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			a	\$	b	b		a		a	a	a	a
			b		a	a		b		\$	b	\$	b
			c		\$	b		a			b		a
			a			b		b			a		b
			b			a		a			\$		a
			b			\$		b					\$
			a					\$					



# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time


- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			a	\$	b	b		a		a	a	a	a
			b		a	a		b		\$	b	\$	b
			c		\$	b		a			b		a
			a			b		b			a		b
			b			a		a			\$		a
			b			\$		b					\$
			a					\$					

# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time


- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			a	\$	b	b		a		a	a	a	a
			b		a	a		b		\$	b	\$	b
			c		\$	b		a			b		b
			a			b		b			a		a
			b			a		a			\$		\$
			b			\$		b					
			a					\$					

# Speeding Up Lookups in $\Psi$ (2/2)


- every  $\log n$ -th entry in  $\Psi$
- every  $\log n$ -th step in  $\Psi$
- what is better?  **PINGO**

- every  $\log n$ -th step in  $\Psi$  is better
- sampled **positions** may not be reached in better asymptotic time

- how much time does recovering  $SA$  position from  $\Psi$  require with sampling?  **PINGO**
- answer:  $O(\log n)$


	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	c	b	c
			c	\$	b	b		a		b	a	a	a
			a		a	c		b		\$	a	\$	b
			b		\$	a		c			b		b
			c			b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			a			\$		b			\$		\$
			\$					\$					

# Structure of $\Psi$

- does  $\Psi$  have some structure?  **PINGO**

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5

# Structure of $\Psi$


- does  $\Psi$  have some structure?  **PINGO**

## Lemma: Structure of $\Psi$

$$T[SA[i]] = T[SA[i + 1]] \Rightarrow \Psi(i) < \Psi(i + 1)$$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5

# Structure of $\Psi$

- does  $\Psi$  have some structure?  **PINGO**

## Lemma: Structure of $\Psi$


$$T[SA[i]] = T[SA[i + 1]] \Rightarrow \Psi(i) < \Psi(i + 1)$$

## Proof (Sketch)

- $T[SA[i]] \leq T[SA[i + 1]]$
- if  $T[SA[i]] = T[SA[i + 1]]$  then  
 $T[SA[i] + 1..n] \leq T[SA[i + 1] + 1..n]$
- $T[SA[i] + 1] = T[\Psi(i)]$
- if suffixes share same character, lexicographical order of suffixes without first character holds

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5

# Structure of $\Psi$

- does  $\Psi$  have some structure?  **PINGO**


## Lemma: Structure of $\Psi$

$$T[SA[i]] = T[SA[i + 1]] \Rightarrow \Psi(i) < \Psi(i + 1)$$

## Proof (Sketch)

- $T[SA[i]] \leq T[SA[i + 1]]$
- if  $T[SA[i]] = T[SA[i + 1]]$  then  
 $T[SA[i] + 1..n] \leq T[SA[i + 1] + 1..n]$
- $T[SA[i] + 1] = T[\Psi(i)]$
- if suffixes share same character, lexicographical order of suffixes without first character holds

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
$SA$	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5

- note that not all increasing intervals belong to the same character
- example on the board 

# Compressing Ordered Sequences

## $\Delta$ -Encoding

- store difference between entries
- scanning whole sequence up to value when decoding



# Compressing Ordered Sequences

## Δ-Encoding

- store difference between entries
- scanning whole sequence up to value when decoding

## Elias-Fano (Lecture 05)

- upper and lower halves
- upper half represented in bit vector  $p_i + i$
- lower half plain bit compressed

0	1	2	3	4	5	6	7	8	9
0	1	2	4	7	10	20	21	22	32

- 0: 000000
- 1: 000001
- 2: 000010
- 4: 000100
- 7: 000111
- 10: 001010
- 20: 010100
- 21: 010101
- 22: 010110
- 30: 100000

upper: 11101101000111000100  
 lower: 00011000111000011000


# Compressing Ordered Sequences

## Δ-Encoding

- store difference between entries
- scanning whole sequence up to value when decoding

## Elias-Fano (Lecture 05)

- upper and lower halves
- upper half represented in bit vector  $p_i + i$
- lower half plain bit compressed

- using Elias-Fano is bad for large alphabets
- example on the board 

0	1	2	3	4	5	6	7	8	9
0	1	2	4	7	10	20	21	22	32

- 0: 000000
- 1: 000001
- 2: 000010
- 4: 000100
- 7: 000111
- 10: 001010
- 20: 010100
- 21: 010101
- 22: 010110
- 30: 100000

**upper:** 11101101000111000100

**lower:** 00 01 10 00 11 10 00 01 10 00

# Recap: Elias-Fano Coding Space

## Lemma: Elias-Fano Coding

Given an array containing  $n$  distinct integers from a universe  $\mathcal{U} = [0, n)$ , the array can be represented using

$$n(2 + \log \lceil \frac{u}{n} \rceil) \text{ bits}$$

while allowing  $O(1)$  access time and  $O(\log \frac{u}{n})$  predecessor/successor time

# Compressing Sparse Ordered Sequences

- Elias-Fano coding for each increasing interval
- $\sigma$  many
- only every  $1/\sigma$ -th entry is set (*sparse*)

# Compressing Sparse Ordered Sequences

- Elias-Fano coding for each increasing interval
  - $\sigma$  many
  - only every  $1/\sigma$ -th entry is set (*sparse*)
- 
- if there are  $n$  entries of universe with size  $u$
  - make entries sparse using  $q = u/n$
  - for each value  $x$  store pair  $(x/q, x \% q)$

# Compressing Sparse Ordered Sequences

- Elias-Fano coding for each increasing interval
- $\sigma$  many
- only every  $1/\sigma$ -th entry is set (**sparse**)

- if there are  $n$  entries of universe with size  $u$
- make entries sparse using  $q = u/n$
- for each value  $x$  store pair  $(x/q, x \% q)$

- $u = 512, n = 8, q = 64$
- $(0, 3, 17, 89, 128, 132, 500, 511)$
- $\{0, 0\}, \{0, 3\}, \{0, 7\}, \{1, 25\},$   
 $\{2, 0\}, \{2, 4\}, \{7, 52\}, \{7, 63\}$

# Compressing Sparse Ordered Sequences

- Elias-Fano coding for each increasing interval
- $\sigma$  many
- only every  $1/\sigma$ -th entry is set (sparse)

- store quotient  $(x/q)$  using Elias-Fano
- store remainder  $(x \% q)$  plain using  $\lceil \log q \rceil$  bits

- if there are  $n$  entries of universe with size  $u$
- make entries sparse using  $q = u/n$
- for each value  $x$  store pair  $(x/q, x \% q)$

- $u = 512, n = 8, q = 64$
- $(0, 3, 17, 89, 128, 132, 500, 511)$
- $\{0, 0\}, \{0, 3\}, \{0, 7\}, \{1, 25\},$   
 $\{2, 0\}, \{2, 4\}, \{7, 52\}, \{7, 63\}$

# Compressing Sparse Ordered Sequences

- Elias-Fano coding for each increasing interval
- $\sigma$  many
- only every  $1/\sigma$ -th entry is set (sparse)

- if there are  $n$  entries of universe with size  $u$
- make entries sparse using  $q = u/n$
- for each value  $x$  store pair  $(x/q, x \% q)$

- $u = 512, n = 8, q = 64$
- $(0, 3, 17, 89, 128, 132, 500, 511)$
- $\{0, 0\}, \{0, 3\}, \{0, 7\}, \{1, 25\},$   
 $\{2, 0\}, \{2, 4\}, \{7, 52\}, \{7, 63\}$

- store quotient  $(x/q)$  using Elias-Fano
- store remainder  $(x \% q)$  plain using  $\lceil \log q \rceil$  bits

## Lemma: $\Psi$ with Elias-Fano

Using Elias-Fano with quotienting,  $\Psi$  can be stored using  $O(n\sigma)$  bits



# Compressing Sparse Ordered Sequences

- Elias-Fano coding for each increasing interval
- $\sigma$  many
- only every  $1/\sigma$ -th entry is set (sparse)

- if there are  $n$  entries of universe with size  $u$
- make entries sparse using  $q = u/n$
- for each value  $x$  store pair  $(x/q, x \% q)$

- $u = 512, n = 8, q = 64$
- $(0, 3, 17, 89, 128, 132, 500, 511)$
- $\{0, 0\}, \{0, 3\}, \{0, 7\}, \{1, 25\},$   
 $\{2, 0\}, \{2, 4\}, \{7, 52\}, \{7, 63\}$

- store quotient  $(x/q)$  using Elias-Fano
- store remainder  $(x \% q)$  plain using  $\lceil \log q \rceil$  bits

## Lemma: $\Psi$ with Elias-Fano

Using Elias-Fano with quotienting,  $\Psi$  can be stored using  $O(n\sigma)$  bits

- more precise: two additional bits per character

# Simple Compressed Suffix Array

- compute  $\Psi$  and store samples of SA
- compress  $\Psi$  Elias-Fano with **quotienting**
- binary search on SA ⓘ by decoding  $\Psi$

- space:  $O(n \log \sigma)$  space
- query time:  $O(m \log^2 n)$

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	b	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		a	c	\$	a
			a		a	c		b		a	a		b
			b		\$	a		c		\$	b		b
			c			b		a			a		a
			a			b		b			b		b
			b			a		a			a		a
			a			\$		b			\$		\$

# Improving Compressed Suffix Arrays [GV05] (1/5)


- improve SA lookup to  $O(\log \log n)$  time
- divide-and-conquer approach
- storing  $\Psi$  only for half of the entries
- recurs for the other half

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	b	c	c	a	a
			a	b	c	c	\$	b	a	a	a	b	b
			b	a	a	a		c	\$	b	b	a	c
			c	\$	b	b		a		a	c	\$	a
			a		a	c		b		b	a		b
			b		\$	a		c		\$			a
			c			b		a			b		b
			a			b		b			a		a
			b			a		a					\$
			a			\$		b					
			\$					\$					

# Improving Compressed Suffix Arrays [GV05] (1/5)

- improve SA lookup to  $O(\log \log n)$  time
- divide-and-conquer approach
- storing  $\Psi$  only for half of the entries
- recurs for the other half

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
NEW	13	1	9	3	11	7	5	1	10	6	7	13	4

- for which values do we store  $\Psi$ ?  **PINGO**

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T$	a	b	a	b	c	a	b	c	a	b	b	a	\$
SA	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
	\$	a	a	a	a	a	b	b	b	b	b	c	c
		\$	b	b	b	b	a	a	a	a	a	a	a
			a	b	c	c	\$	b	b	b	b	b	b
			b	a	a	a		c	a	b	c	a	b
			c	\$	b	b		a	b	a	a	\$	a
			a		a	a		c	b	\$	b	b	a
			b		\$	b		a	a		a		b
			c			b		b	b		b		a
			a			a		a	b		a		\$
			b			b		b	a		\$		
			a			a		a	\$				

## Improving Compressed Suffix Arrays (2/5)

- store bit vector marking **odd** SA values
- store only odd SA values
- store  $\Psi$  for even SA values

## Improving Compressed Suffix Arrays (2/5)

- store bit vector marking **odd** SA values
- store only odd SA values
- store  $\Psi$  for even SA values

- store  $\Psi$  as before
- Elias-Fano with quotienting
- **without** sampling

## Improving Compressed Suffix Arrays (2/5)

- store bit vector marking **odd** SA values
- store only odd SA values
- store  $\Psi$  for even SA values

- store  $\Psi$  as before
- Elias-Fano with quotienting
- **without** sampling

- right half (SA) still big
- how to recurs?

# Improving Compressed Suffix Arrays (2/5)

- store bit vector marking **odd** SA values
- store only odd SA values
- store  $\Psi$  for even SA values

- store  $\Psi$  as before
- Elias-Fano with quotienting
- **without** sampling

- right half (SA) still big
- how to recurs?

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1



# Improving Compressed Suffix Arrays (3/5)

- SA half consists only of odd values
- for value  $x$  store  $(x - 1)/2$
- reversible since all values are odd

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (3/5)

- SA half consists only of odd values
- for value  $x$  store  $(x - 1)/2$
- reversible since all values are odd


- 13, 1, 9, 3, 11, 7, 5
- 6, 0, 4, 1, 5, 3, 2

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (3/5)

- SA half consists only of odd values
- for value  $x$  store  $(x - 1)/2$
- reversible since all values are odd

- 13, 1, 9, 3, 11, 7, 5
- 6, 0, 4, 1, 5, 3, 2



- what do we have here?  **PINGO**

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (3/5)

- SA half consists only of odd values
- for value  $x$  store  $(x - 1)/2$
- reversible since all values are odd

- 13, 1, 9, 3, 11, 7, 5
- 6, 0, 4, 1, 5, 3, 2



- what do we have here?  **PINGO**
- permutation  basically a suffix array without text

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (3/5)

- SA half consists only of odd values
- for value  $x$  store  $(x - 1)/2$
- reversible since all values are odd

- 13, 1, 9, 3, 11, 7, 5
- 6, 0, 4, 1, 5, 3, 2

- what do we have here?  **PINGO**
- permutation  basically a suffix array without text

- recurs on the permutation without explicitly storing it

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (4/5)

- recurs  $\log \log n$  times
- guarantees  $O(\log \log n)$  time to obtain *SA* value
- allows to store final *SA* within space bounds

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (4/5)

- recurs  $\log \log n$  times
- guarantees  $O(\log \log n)$  time to obtain *SA* value
- allows to store final *SA* within space bounds

## Lemma: Space Final *SA*

Using the divide-and-conquer approach, the final *SA* requires  $O(n)$  bits of space

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (4/5)

- recurs  $\log \log n$  times
- guarantees  $O(\log \log n)$  time to obtain SA value
- allows to store final SA within space bounds

## Lemma: Space Final SA

Using the divide-and-conquer approach, the final SA requires  $O(n)$  bits of space

## Proof (Sketch)

- after  $\log \log n$  recursions SA has size  $n/2^{\log \log n}$
- each entry requires  $\log n$  bits
- total space:  $O(n)$  bits

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1



# Improving Compressed Suffix Arrays (5/5)

## Lemma: Decoding Time Improved CSA

An SA value can be decoded in  $O(\log \log n)$  time using the improved CSA

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (5/5)

## Lemma: Decoding Time Improved CSA

An *SA* value can be decoded in  $O(\log \log n)$  time using the improved CSA

## Proof (Sketch)

- on each level, odd *SA* values can be decoded using the recursive *SA*
- there are at most  $\log \log n$  levels
- on each level, even *SA* values can be decoded in one step, as the next *SA* value is odd

	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Improving Compressed Suffix Arrays (5/5)

## Lemma: Decoding Time Improved CSA

An SA value can be decoded in  $O(\log \log n)$  time using the improved CSA

## Proof (Sketch)

- on each level, odd SA values can be decoded using the recursive SA
  - there are at most  $\log \log n$  levels
  - on each level, even SA values can be decoded in one step, as the next SA value is odd
- requires rank and select data structures

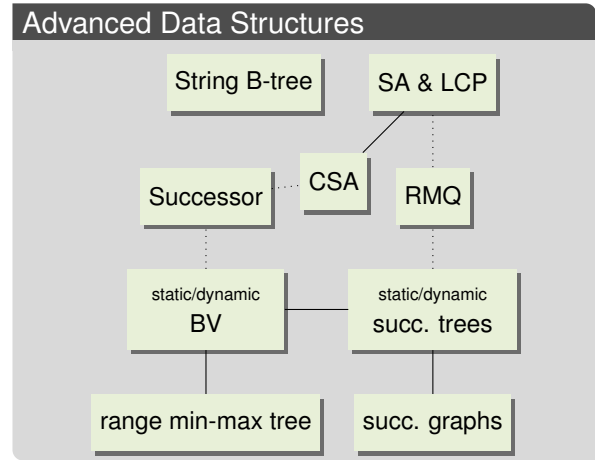
	1	2	3	4	5	6	7	8	9	10	11	12	13
<i>T</i>	a	b	a	b	c	a	b	c	a	b	b	a	\$
<i>SA</i>	13	12	1	9	6	3	11	2	10	7	4	8	5
$\Psi$	-	1	8	9	10	11	2	6	7	12	13	4	5
<i>NEW</i>	13	1	9	3	11	7	5	1	10	6	7	13	4
<i>BV</i>	1	0	1	1	0	1	1	0	0	1	0	0	1

# Conclusion and Outlook

## This Lecture

- compressed suffix array
- note that CSA can be compressed further
- Elias-Fano for sparse sequences

## Advanced Data Structures



# Conclusion and Outlook

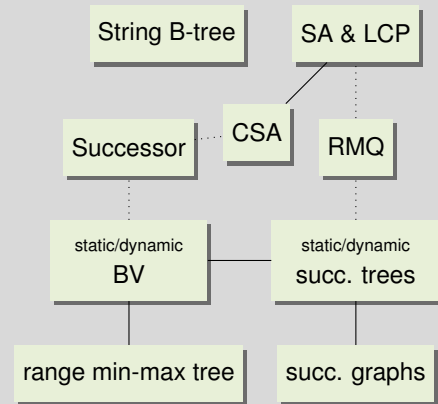
## This Lecture

- compressed suffix array
- note that CSA can be compressed further
- Elias-Fano for sparse sequences

## Next Lecture

- temporal data structures

## Advanced Data Structures



# Bibliography I

- [GV05] Roberto Grossi and Jeffrey Scott Vitter. “Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching”. In: *SIAM J. Comput.* 35.2 (2005), pages 378–407. DOI: [10.1137/S0097539702402354](https://doi.org/10.1137/S0097539702402354).