Backward Search
FM-Index

(Full-text index in Minute space)

Motivation

How does it work?

- *Burrows-Wheeler Transform*

- ***compressed text*** ***full-text indexing information.***

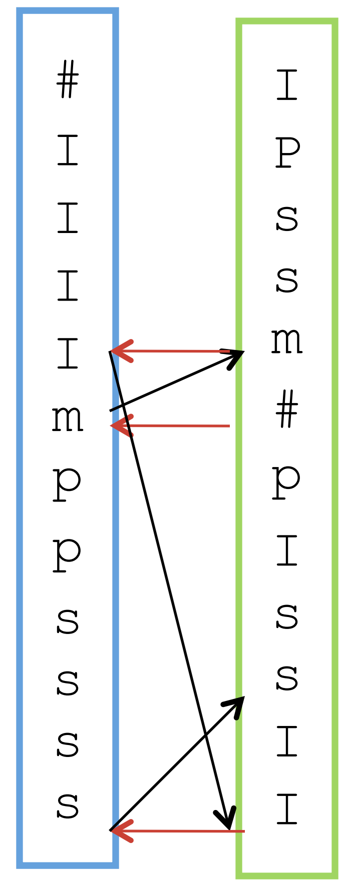
-
  - **Count**
  - **Locate**

Burrows-Wheeler Transform

```
mississippi#
ississippi#m
 ssissippi#mi
sissippi#mis          i   ssippi#mis   s
issippi#miss
ssippi#missi          m   issi
sippi#missis          p   i#mississi   p
ippi#mississ          p   pi#mississ   i
ppi#mississi          s   ippi#missi   s
pi#mississip          s   issippi#mi   s
i#mississipp          s   sippi#miss   i
#mississippi          s   sissippi#m   i
```

Burrows–Wheeler Transform

:



Next: Count P in T

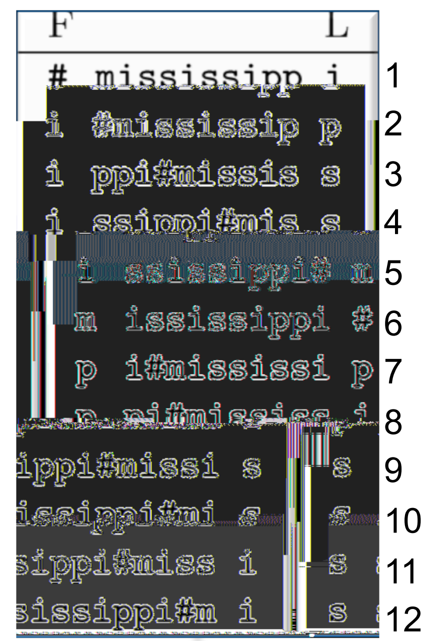
- **Backward-search algorithm**
-
-
    - Σ
    -

= #

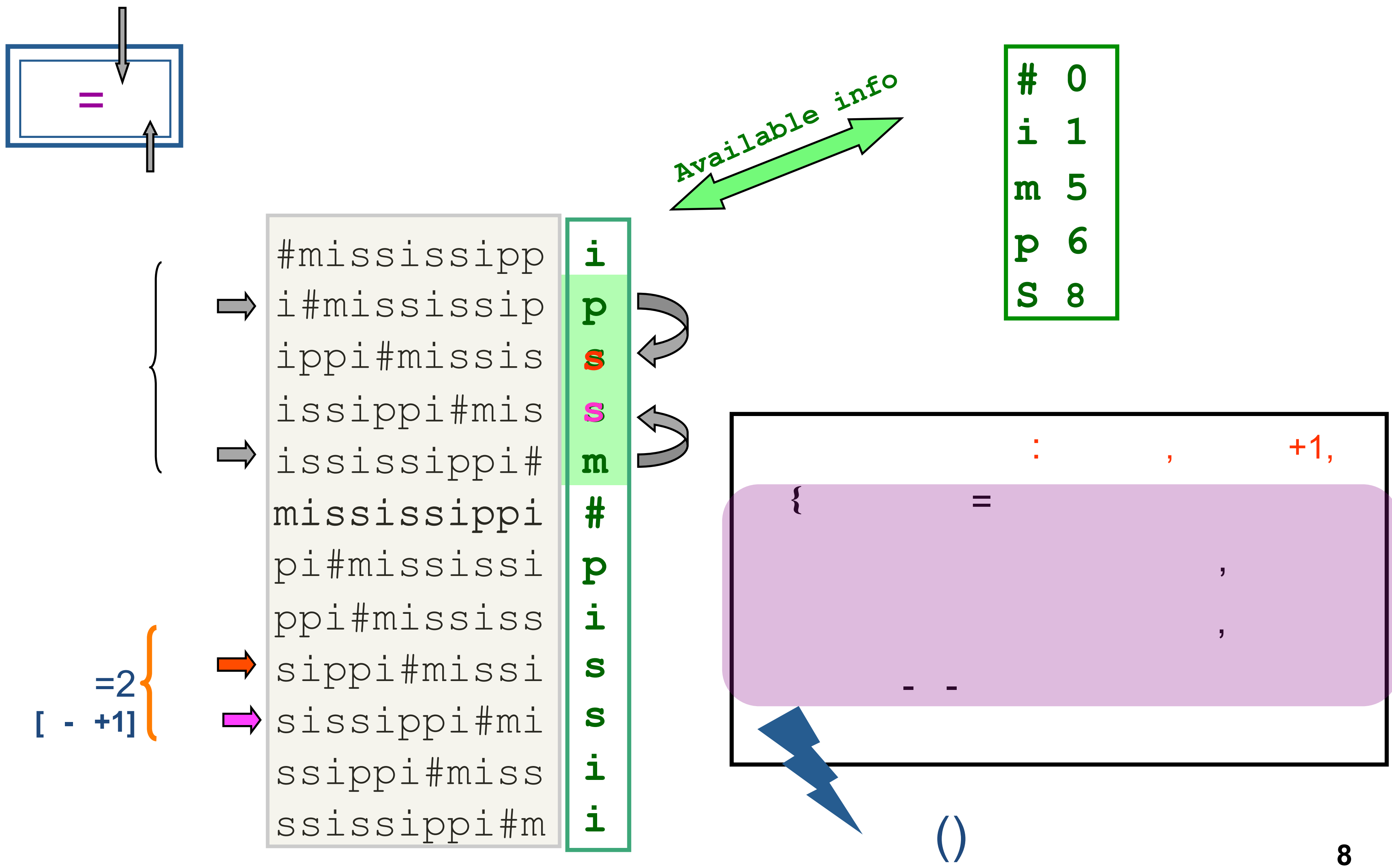
1	5	6	8

(, 5) = 2
(,12) = 4

≡



SUBSTRING SEARCH IN T (COUNT THE PATTERN OCCURRENCES)



Backward search example

- P = pssi
  - 3
  -
  - + (,1) +1 = 8+0+1 = 9
  - + (,5) = 8+2 = 10
  - 2

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

= | 1 | 5 | 6 | 8 |

Algorithm backward_search($P[1,p]$)

(1) $i \leftarrow p$, $c \leftarrow P[p]$, First $\leftarrow C[c]+1$, Last $\leftarrow C[c+1]$;
(2) while ((First $\leq$ Last) and ($i \geq 2$)) do
(3) $c \leftarrow P[i-1]$;
(4) First $\leftarrow C[c] + \text{Occ}(c, \text{First}-1)+1$;
(5) Last $\leftarrow C[c] + \text{Occ}(c, \text{Last})$;
(6) $i \leftarrow i-1$;
(7) if (Last < First) then return "no rows prefixed by $P[1,p]$" else return (First, Last).

Backward-search example

- P = pssi
  - 2
  -
  - + (,8) +1 = 8+2+1 = 11
  - + (,10) = 8+4 = 12
  - 2

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

= | 1 | 5 | 6 | 8 |

Algorithm backward_search($P[1,p]$)

(1) $i \leftarrow p$, $c \leftarrow P[p]$, First $\leftarrow C[c]+1$, Last $\leftarrow C[c+1]$;

(2) while ((First $\leq$ Last) and ($i \geq 2$)) do

(3) $c \leftarrow P[i-1]$;

(4) First $\leftarrow C[c] + \text{Occ}(c, \text{First}-1)+1$;

(5) Last $\leftarrow C[c] + \text{Occ}(c, \text{Last})$;

(6) $i \leftarrow i-1$;

(7) if (Last < First) then return "no rows prefixed by $P[1,p]$" else return (First, Last).

Backward search example

- P = pssi
  - 1
  -
  - + (,10) +1 = 6+2+1 = 9
  - + (,12) = 6+2 = 8
  -

F		L	
#	mississipp	i	1
i	#mississip	p	2
i	ppi#missis	s	3
i	ssippi#mis	s	4
i	ssissippi#	m	5
m	ississippi	#	6
[illegible]	[illegible]	[illegible]	7
[illegible]	[illegible]	[illegible]	8
s	ippi#missi	s	9
s	issippi#mi	s	10
s	sippi#miss	i	11
s	sissippi#m	i	12

= | 1 | 5 | 6 | 8 |

Algorithm backward_search($P[1,p]$)

(1) $i \leftarrow p$, $c \leftarrow P[p]$, First $\leftarrow C[c]+1$, Last $\leftarrow C[c+1]$;

(2) while ((First $\leq$ Last) and ($i \geq 2$)) do

(3) $c \leftarrow P[i-1]$;

(4) First $\leftarrow C[c] + \text{Occ}(c, \text{First} - 1) + 1$;

(5) Last $\leftarrow C[c] + \text{Occ}(c, \text{Last})$;

(6) $i \leftarrow i - 1$;

(7) if (Last < First) then return "no rows prefixed by $P[1,p]$" else return (First, Last).

Assignment 2

ASSIGNMENT 2

- bwtsearch
  - Bwtsearch -e fileToBeEncoded outputFile
  - Bwtsearch -d fileToBeDecoded
    - standard output
  - Bwtsearch -s fileEncoded "queryString"
    - Output all the lines contain "queryString"
    - Highlight "queryString" if capable
    - The search results need to be sorted according to their line numbers.

ASSIGNMENT 2

Assignment 2

ASSIGNMENT 2

- one
-

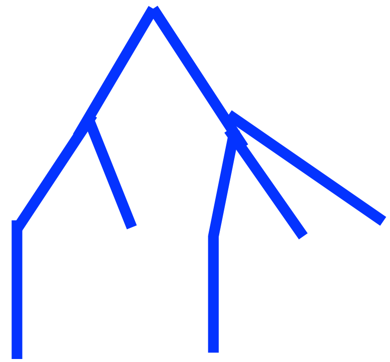
Lecture 5

SUCCINCT SUFFIX ARRAYS BASED ON RUN-LENGTH ENCODING *

VELI MÄKINEN†
Dept. of Computer Science, University of Helsinki
Gustaf Hällströmin katu 2b, 00014 University of Helsinki, Finland
[redacted]
Dept. of Computer Science, University of Chil
Blanco Encalada 2120, Santiago, Chile
gnavarro@dcc.uchile.cl

A B

Arbitrary Ordered Trees



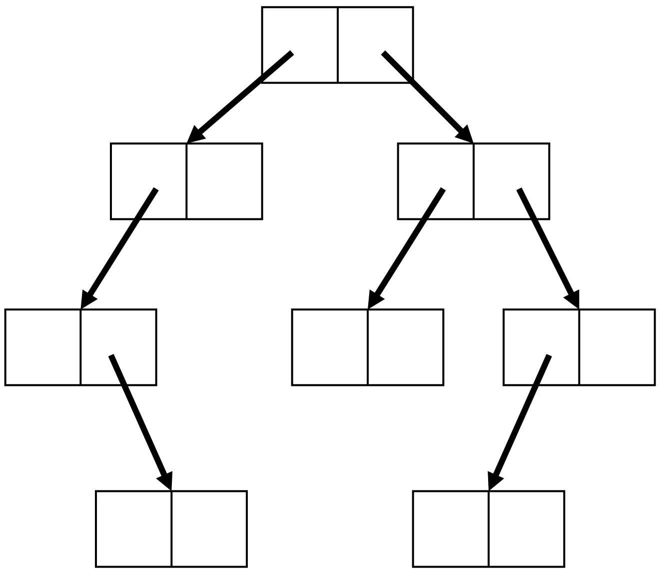
“ “ “ ”

SPACE FOR TREES

STANDARD REPRESENTATION

B :

A -
2 2



(
).

, (. ,)
, , .

CAN WE IMPROVE THE SPACE BOUND?

- 2^2

  .

- 2

  .

- 2

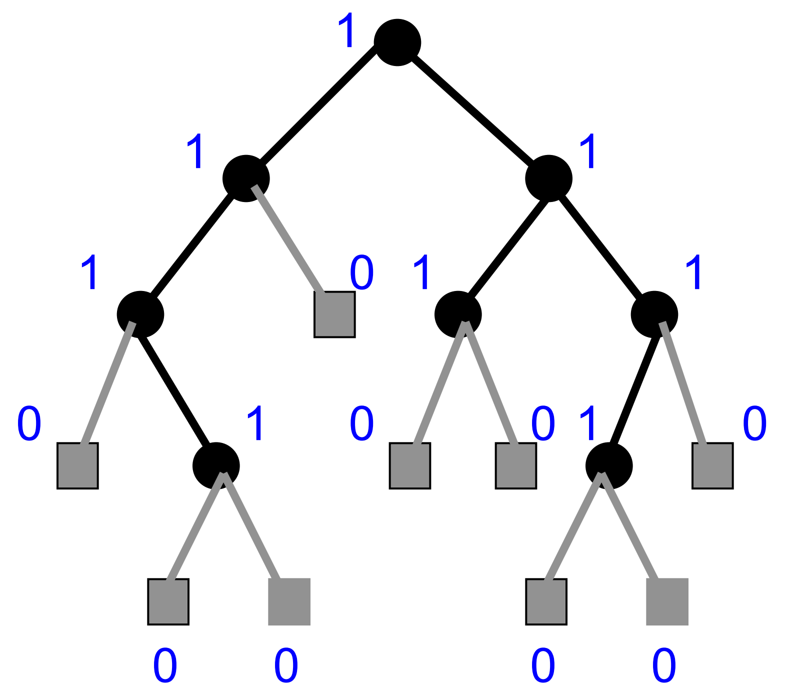
  ?

HEAP-LIKE NOTATION FOR A BINARY TREE

A

1
0

1 1 1 1 0 1 1 0 1 0 0 1 0 0 0 0 0



A

2 +1 .

?

RANK/SELECT ON A BIT VECTOR

B

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
B:	0	1	1	0	1	0	0	0	1	1	0	1	1	1	1

$_1(\ ) = \#\ 1$ B

$_1(\ ) =$ - 1 B

($_0$ $_0$)

,

( )- ,

.

$_1(5) = 3$

$_1(4) = 9$

$_0(5) = 2$

$_0(4) = 7$

A .

.

Binary tree representation

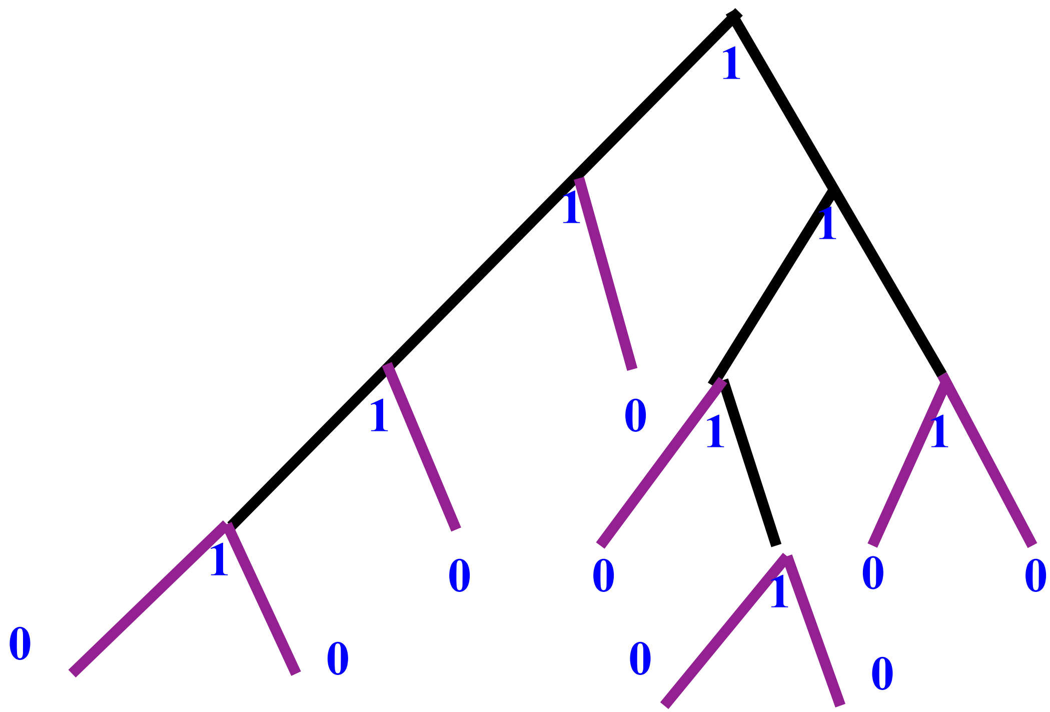
- A
  2 + ( ) :
  -
  -
  -

.

- 1 1 1 1 0 1 1 1 0

Heap-like Notation for a Binary Tree

A



1 2 3 4 5 6

1 1 1 1 0 1 1 1 0 0 1 0 00000 2 +1

1 2 3 4 5 6 7 8 9 0 1 2 34567

ORDERED TREES

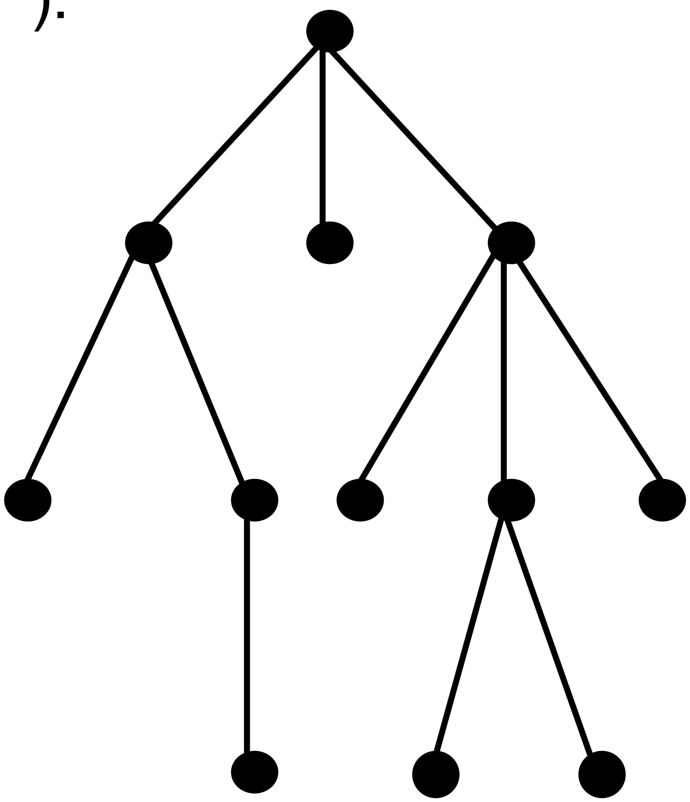
A ( ):

:

- ( ) =
- ( ) =
- ( ) =

:

- ( ) = 2
- ( ) = 4



ORDERED TREES

- A 2 + ( )
  ,
  .
- \- -
  ( )
  (+1).
- 2 + ( ) ,
  ( ) .
- .

SUPPORTING OPERATIONS

A

1 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0

1 2 3 4 5 6 7 8 9 10 11 12

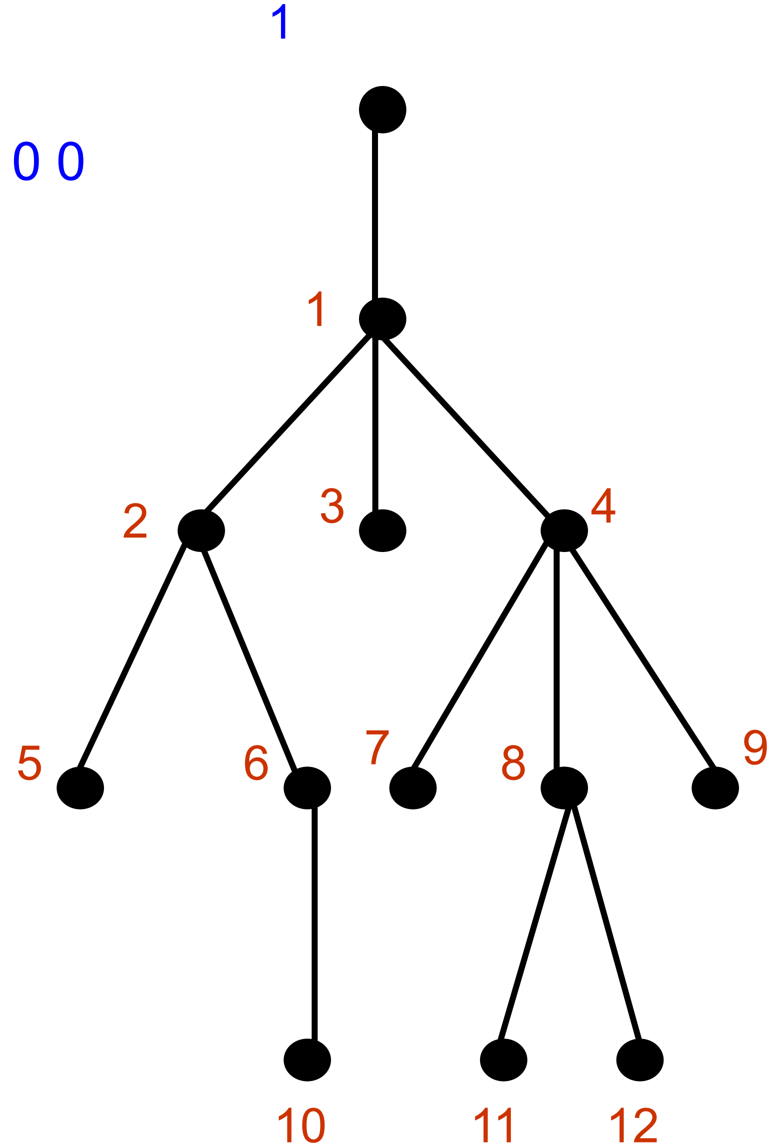
- 1

( ) = # 0 - 1

- 0

: , - ,

( )



Simple FM-index

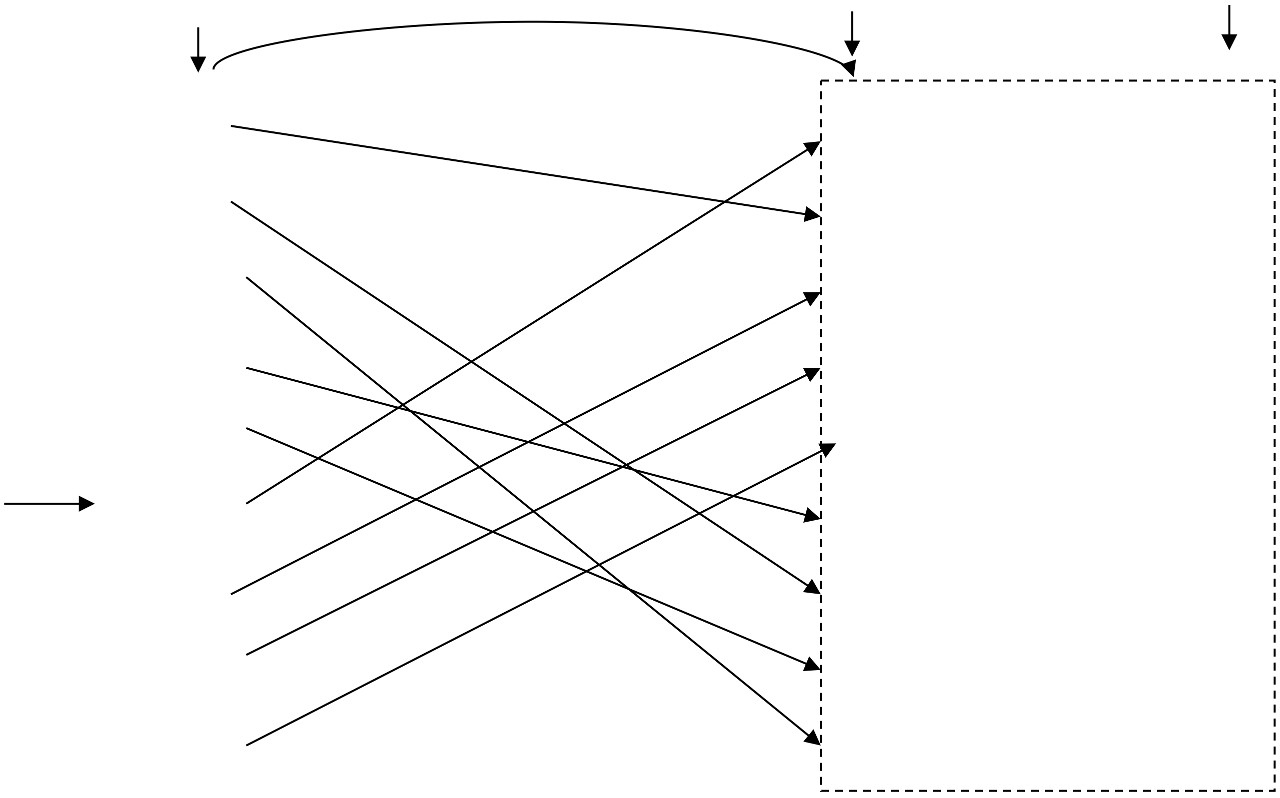
- *Burrows-Wheeler-transformed*
-
-

Burrows-Wheeler transformation

Example

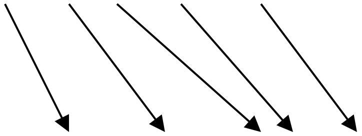
12345

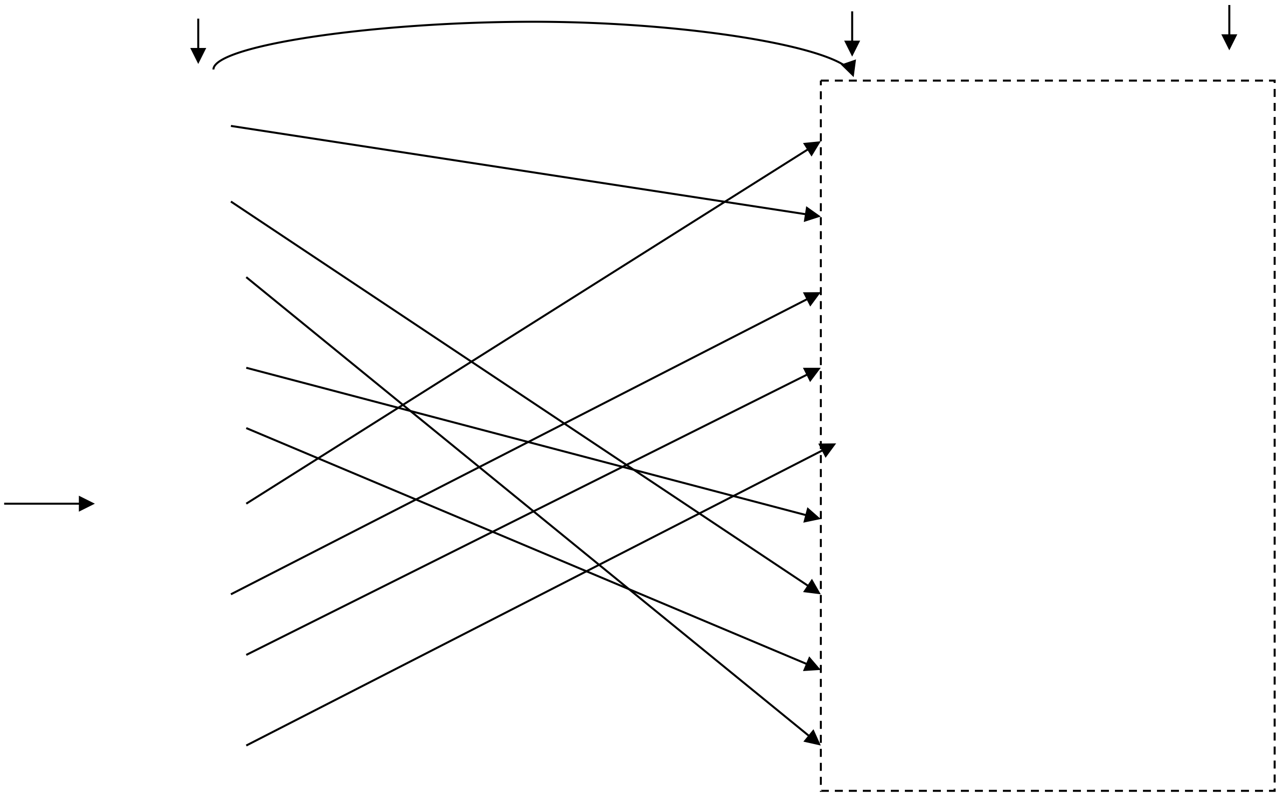
#



Implicit LF[i]

Rank/Select





Recall: Backward search on bwt(T)

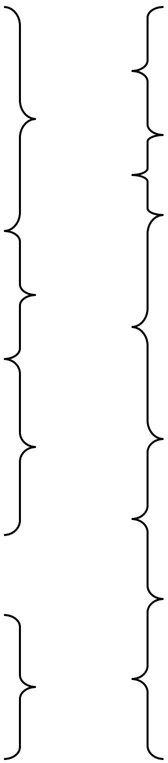
- Observation

Backward search on BWT(T)...

- σ σ
- σ
-

Run-Length FM-index

RUN-LENGTH FM-INDEX...



CHANGES TO FORMULAS

Theorem:

$\neq$

Example, L[i]=c

Exercise

WHAT IS B'

i **B** **S**

USUALLY B’ IS GIVEN TO SAVE COMPUTATIONS

<u>*i*</u>	<u>**B**</u>	<u>**S**</u>	<u>**B’**</u>

Reverse BWT from row 6

i	**B**	**S**	**B'**

Reverse BWT

i	**B**	**S**	**B'**

Reverse BWT

i	**B**	**S**	**B'**

Reverse BWT

i	**B**	**S**	**B'**

Reverse BWT

i **B** **B'**

BACKWARD SEARCH

i **B** **S** **B'**

Backward search

i	**B**	**S**	**B'**