Burrows-Wheeler Transform

Anna Budkina

MIPT

16th October 2023

Telegram Chat

Glossary

- Alphabet $\{a_1, a_2, \ldots, a_A\}$
- String S: $s_1 s_2 \ldots s_k$
- Substring $S[i : j]$: $s_i s_{i+1} \ldots s_{j-1}$
- Prefix $S[: j]$: $s_1 s_2 \ldots s_j$
- Suffix $S[j :]$ $s_j s_{j+1} \ldots s_k$

Pattern Matching Task

Input:

- String S : $s_1 s_2 \dots s_n$
- String P (pattern): $p_1 p_2 \dots p_m$, $m \leq n$

Output:

- Substring $S[i : j]$: $S[i, j] = P$.

Suffix Tree

index	suffix
6	$
5	A
2	$AABA$
3	ABA
0	$ABAABA$
4	BA
1	$BAABA$

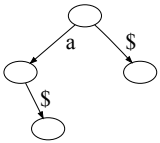
Suffix Tree

index	suffix
6	$
5	A
2	$AABA$
3	ABA
0	$ABAABA$
4	BA
1	$BAABA$

Suffix Tree

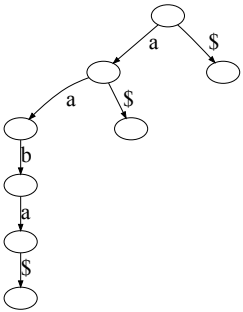
index	suffix
6	$
5	*A*$
2	*AABA*$
3	*ABA*$
0	*ABAABA*$
4	*BA*$
1	*BAABA*$

Suffix Tree



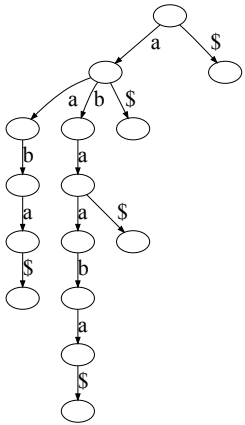
index	suffix
6	$
5	*A*$
2	*AABA*$
3	*ABA*$
0	*ABAABA*$
4	*BA*$
1	*BAABA*$

Suffix Tree



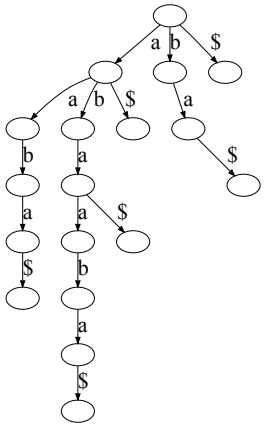
index	suffix
6	$
5	A
2	$AABA$
3	ABA
0	$ABAABA$
4	BA
1	$BAABA$

Suffix Tree



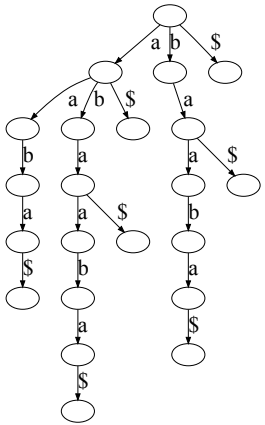
index	suffix
6	$
5	*A*$
2	*AABA*$
3	*ABA*$
0	*ABAABA*$
4	*BA*$
1	*BAABA*$

Suffix Tree



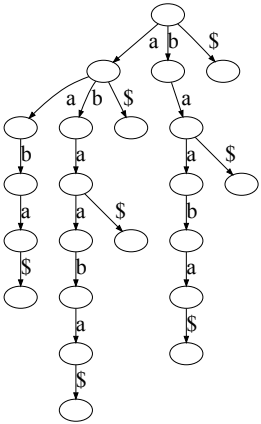
index	suffix
6	$
5	*A*$
2	*AABA*$
3	*ABA*$
0	*ABAABA*$
4	*BA*$
1	*BAABA*$

Suffix Tree



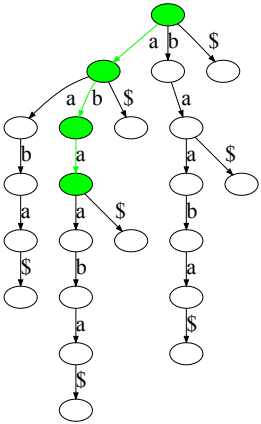
index	suffix
6	$
5	*A*$
2	*AABA*$
3	*ABA*$
0	*ABAABA*$
4	*BA*$
1	*BAABA*$

Suffix Tree: finding pattern



$P = ABA$

Suffix Tree: finding pattern



$P = ABA$

Burrows-Wheeler Transform

S : $ABAABA\$$ ($ - end of string)

Burrows-Wheeler Transform

$ A B A A B A

$S : ABAABA\$$ ($ - end of string) Rotations

Burrows-Wheeler Transform

$	A	B	A	A	B	A
A	$	A	B	A	A	B

S : $ABAABA$\$ (\$ - end of string) → Rotations

Burrows-Wheeler Transform

S : $ABAABA\$$ ($ - end of string) $\xrightarrow{\text{Rotations}}$

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A

Burrows-Wheeler Transform

$S : ABAABA\$$ ($ - end of string) $\xrightarrow{\text{Rotations}}$

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A

Burrows-Wheeler Transform

$S : ABAABA\$$ ($ - end of string) $\xrightarrow{\text{Rotations}}$

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B

Burrows-Wheeler Transform

$S : ABAABA\$$ ($ - end of string) $\xrightarrow{\text{Rotations}}$

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B
B	A	A	B	A	$	A

Burrows-Wheeler Transform

S : $ABAABA\$$ ($ - end of string)

Rotations →

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B
B	A	A	B	A	$	A
A	B	A	A	B	A	$

Burrows-Wheeler Transform: how to build

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B
B	A	A	B	A	$	A
A	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A

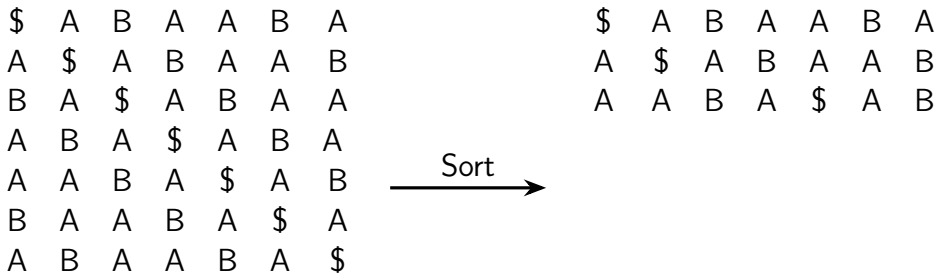
Burrows-Wheeler Transform: how to build

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B
B	A	A	B	A	$	A
A	B	A	A	B	A	$

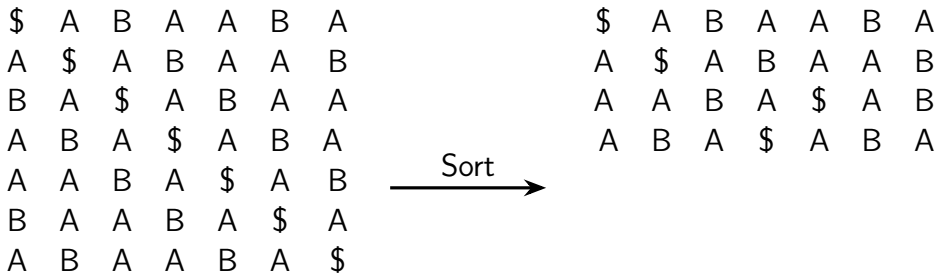
Sort

$	A	B	A	A	B	A
A	$	A	B	A	A	B

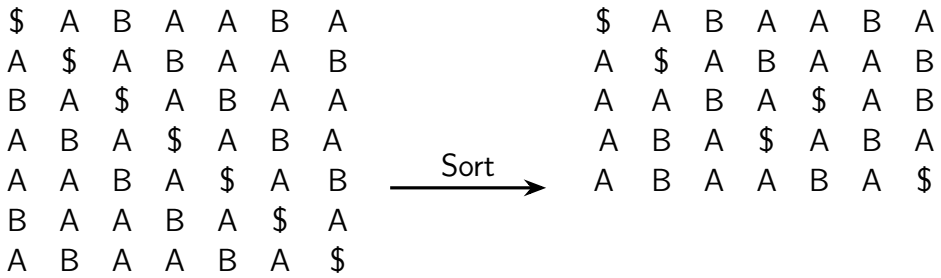
Burrows-Wheeler Transform: how to build



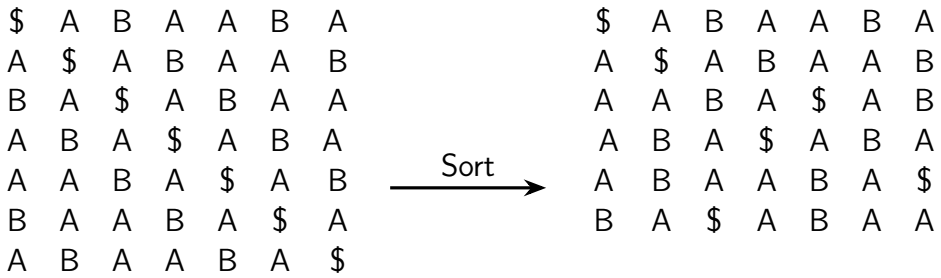
Burrows-Wheeler Transform: how to build



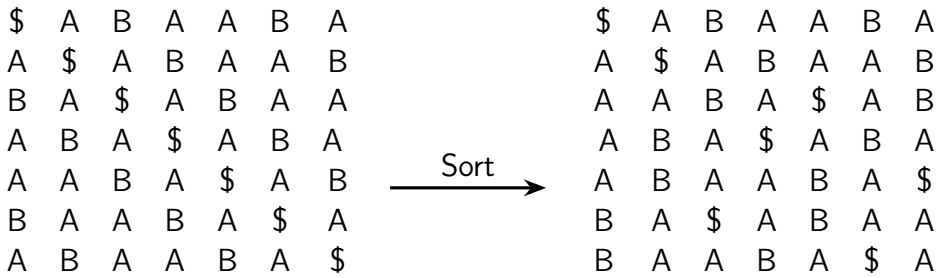
Burrows-Wheeler Transform: how to build



Burrows-Wheeler Transform: how to build



Burrows-Wheeler Transform: how to build



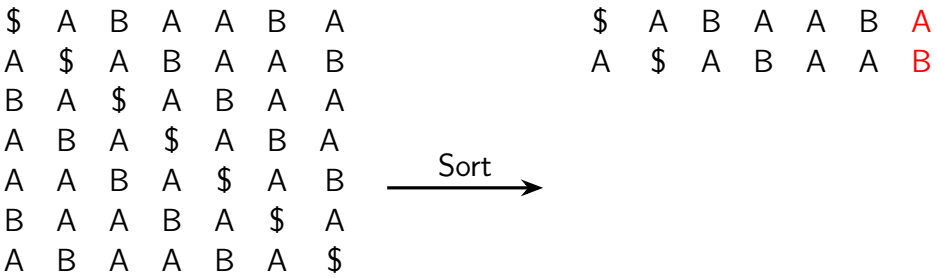
Burrows-Wheeler Transform: how to build

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B
B	A	A	B	A	$	A
A	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A

Burrows-Wheeler Transform: how to build



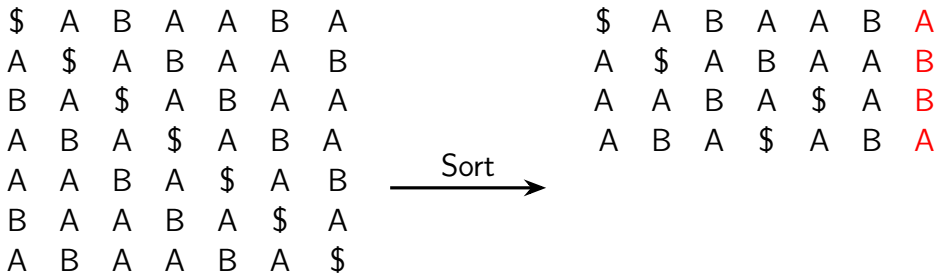
Burrows-Wheeler Transform: how to build

$	A	B	A	A	B	A
A	$	A	B	A	A	B
B	A	$	A	B	A	A
A	B	A	$	A	B	A
A	A	B	A	$	A	B
B	A	A	B	A	$	A
A	B	A	A	B	A	$

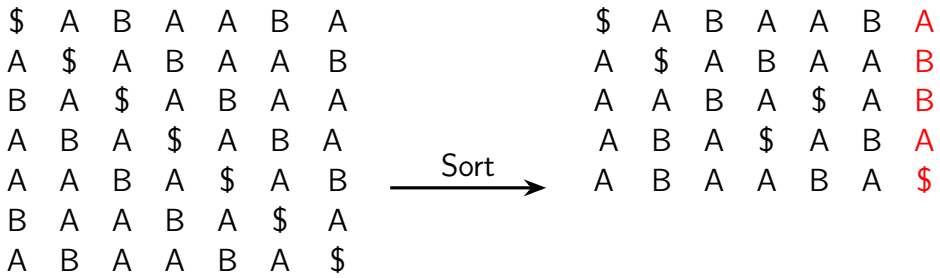
Sort

$	A	B	A	A	B	A
A	$	A	B	A	A	B
A	A	B	A	$	A	B

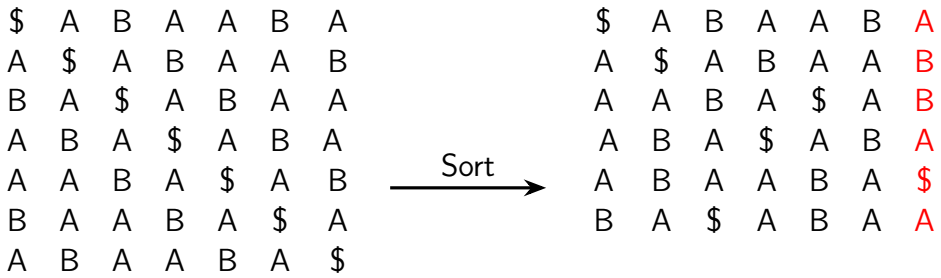
Burrows-Wheeler Transform: how to build



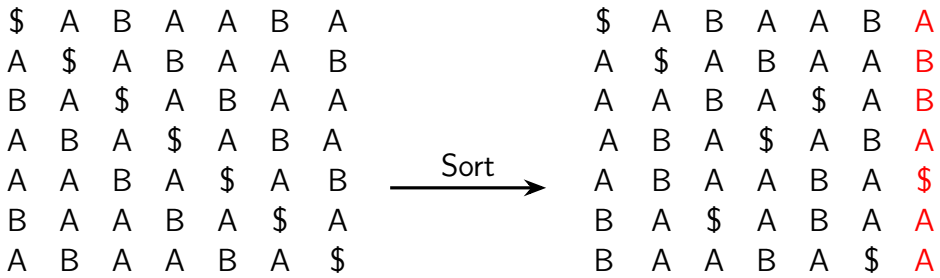
Burrows-Wheeler Transform: how to build



Burrows-Wheeler Transform: how to build



Burrows-Wheeler Transform: how to build



BWT compressing

$S = ABAABA\$$
$BWT(S) = ABBA\$AA$

BWT compressing

$S = ABAABA\$$
$BWT(S) = ABBA\$AA$
Let us code with RLE:

- 1 A
- 2 B
- 1 $
- 2 A

BWT compressing

$S = ABAABA\$$
$BWT(S) = ABBA\$AA$
Let us code with RLE:

- 1 A
- 2 B
- 1 $
- 2 A

We will get string:
$1A2B1\$2A$

BWT: reversing: T ranking

Enumerate letters in the word S:
$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

\$	A	B	A	A	B	A_3
A_3	\$	A	B	A	A	B_1
B_1	A	\$	A	B	A	A_2
A_2	B	A	\$	A	B	A_1
A_1	A	B	A	\$	A	B_0
B_0	A	A	B	A	\$	A_0
A_0	B	A	A	B	A	\$

Sort →

\$	A	B	A	A	B	A_3

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
B_1	A	$	A	B	A	A_2
A_2	B	A	$	A	B	A_1
A_1	A	B	A	$	A	B_0
B_0	A	A	B	A	$	A_0
A_0	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

\$	A	B	A	A	B	A_3
A_3	\$	A	B	A	A	B_1
B_1	A	\$	A	B	A	A_2
A_2	B	A	\$	A	B	A_1
A_1	A	B	A	\$	A	B_0
B_0	A	A	B	A	\$	A_0
A_0	B	A	A	B	A	\$

Sort →

\$	A	B	A	A	B	A_3
A_3	\$	A	B	A	A	B_1
A_1	A	B	A	\$	A	B_0

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
B_1	A	$	A	B	A	A_2
A_2	B	A	$	A	B	A_1
A_1	A	B	A	$	A	B_0
B_0	A	A	B	A	$	A_0
A_0	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
A_1	A	B	A	$	A	B_0
A_2	B	A	$	A	B	A_1

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
B_1	A	$	A	B	A	A_2
A_2	B	A	$	A	B	A_1
A_1	A	B	A	$	A	B_0
B_0	A	A	B	A	$	A_0
A_0	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
A_1	A	B	A	$	A	B_0
A_2	B	A	$	A	B	A_1
A_0	B	A	A	B	A	$

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
B_1	A	$	A	B	A	A_2
A_2	B	A	$	A	B	A_1
A_1	A	B	A	$	A	B_0
B_0	A	A	B	A	$	A_0
A_0	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
A_1	A	B	A	$	A	B_0
A_2	B	A	$	A	B	A_1
A_0	B	A	A	B	A	$
B_1	A	$	A	B	A	A_2

BWT with T ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
B_1	A	$	A	B	A	A_2
A_2	B	A	$	A	B	A_1
A_1	A	B	A	$	A	B_0
B_0	A	A	B	A	$	A_0
A_0	B	A	A	B	A	$

Sort →

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
A_1	A	B	A	$	A	B_0
A_2	B	A	$	A	B	A_1
A_0	B	A	A	B	A	$
B_1	A	$	A	B	A	A_2
B_0	A	A	B	A	$	A_0

BWT: LF mapping

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$\$$	A	B	A	A	B	A_3
A_3	$\$$	A	B	A	A	B_1
A_1	A	B	A	$\$$	A	B_0
A_2	B	A	$\$$	A	B	A_1
A_0	B	A	A	B	A	$\$$
B_1	A	$\$$	A	B	A	A_2
B_0	A	A	B	A	$\$$	A_0

BWT: LF mapping

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
A_1	A	B	A	$	A	B_0
A_2	B	A	$	A	B	A_1
A_0	B	A	A	B	A	$
B_1	A	$	A	B	A	A_2
B_0	A	A	B	A	$	A_0

Last-First mapping

BWT: LF mapping: ordering

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

\$	A	B	A	A	B	A_3
A_3	\$	A	B	A	A	B_1
A_1	A	B	A	\$	A	B_0
A_2	B	A	\$	A	B	A_1
A_0	B	A	A	B	A	\$
B_1	A	\$	A	B	A	A_2
B_0	A	A	B	A	\$	A_0

BWT: LF mapping: ordering

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

\$	A	B	A	A	B	A_3
A_3	\$	A	B	A	A	B_1
A_1	A	B	A	\$	A	B_0
A_2	B	A	\$	A	B	A_1
A_0	B	A	A	B	A	\$
B_1	A	\$	A	B	A	A_2
B_0	A	A	B	A	\$	A_0

Same ordering!

BWT: LF mapping: ordering

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_3
A_3	$	A	B	A	A	B_1
A_1	A	B	A	$	A	B_0
A_2	B	A	$	A	B	A_1
A_0	B	A	A	B	A	$
B_1	A	$	A	B	A	A_2
B_0	A	A	B	A	$	A_0

Same ordering!

BWT: LF mapping: B-ranking

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

$	A	B	A	A	B	A_0
A_0	$	A	B	A	A	B_0
A_1	A	B	A	$	A	B_1
A_2	B	A	$	A	B	A_1
A_3	B	A	A	B	A	$
B_0	A	$	A	B	A	A_2
B_1	A	A	B	A	$	A_3

Sort letters by ascending order in First column.

Important remark!

We have built the FM index!

BWT: reversing from Fm index

$S = A_0 B_0 A_1 A_2 B_1 A_3 \$$

Idea

For each letter in F column we can get the previous letter in corresponding L column.

Question?

What is the last letter in S?

BWT: reversing

Starting from \$.

\$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	\$
B_0	A_2
B_1	A_3

Question

How to get string row with A_0?

BWT: reversing

Starting from \$.

\$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	\$
B_0	A_2
B_1	A_3

Question

How to get string row with A_0?

Answer

Rows in F are sorted in lexicographical order.

$Index(A_0) = 1 + 0$

- 1 - for \$
- 0 - index of A_0

BWT: reversing - Step 2

$\$A_0$.

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with B_0?

BWT: reversing - Step 2

$\$A_0$.

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with B_0?

Answer

$Index(B_0) = 1 + 4 + 0 = 5$

- 1 - for $
- 4 - number of A-s
- 0 - index of B_0

BWT: reversing - Step 2

$\$A_0B_0.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with A_2?

BWT: reversing - Step 2

$\$A_0B_0.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with A_2?

Answer

$Index(A_2) = 1 + 2 = 3$

- 1 - for $
- 2 - index of A_2

BWT: reversing - Step 3

$\$A_0B_0A_2.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with A_1?

BWT: reversing - Step 3

$\$A_0B_0A_2.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with A_1?

Answer

$Index(A_1) = 1 + 1 = 2$

- 1 - for $
- 1 - index of A_1

BWT: reversing - Step 4

$\$A_0B_0A_2A_1.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with B_1?

BWT: reversing - Step 4

$\$A_0B_0A_2A_1.$

\$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	\$
B_0	A_2
B_1	A_3

Question

How to get string row with B_1?

Answer

$Index(A_1) = 1 + 4 + 1 = 6$

- 1 - for \$
- 4 - number of A-s
- 1 - index of B_1

BWT: reversing - Step 5

$\$A_0B_0A_2A_1B_1.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with A_3?

BWT: reversing - Step 5

$\$A_0B_0A_2A_1B_1.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to get string row with A_3?

Answer

$Index(A_1) = 1 + 3 = 4$

- 1 - for $
- 3 - index of A_3

BWT: reversing - Step 5

$\$A_0B_0A_2A_1B_1A_3.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

Have we finished?

BWT: reversing - Step 5

$\$A_0B_0A_2A_1B_1A_3.$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

Have we finished?

Answer

Not yet!

We should reverse string:

$A_3B_1A_1A_2B_0A_0\$$

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = AB\mathbf{A}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to go next?

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = AB\mathbf{A}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to go next?

Answer

Watch for the previous letter! B.

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = A\mathbf{BA}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

Why B are stored sequentially?

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = A\mathbf{BA}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

Why B are stored sequentially?

Answer

Because in row we find strings, started with *BA*!

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = A\mathbf{BA}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to go next?

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = A\mathbf{BA}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

Question

How to go next?

Answer

Watch for the previous letter! A.

BWT: pattern search with FM-indexing

$S = ABAABA\$$

$P = \mathbf{ABA}$

$	A_0
A_0	B_0
A_1	B_1
A_2	A_1
A_3	$
B_0	A_2
B_1	A_3

BWT: any problems?

- Scanning in last column L is slow

BWT: any problems?

- Scanning in last column L is slow
  - Store prefix sums of As and Bs in L

F	L	N(A)	N(B)
\$	A_0	1	0
A_0	B_0	1	1
A_1	B_1	1	2
A_2	A_1	2	2
A_3	\$	2	2
B_0	A_2	3	2
B_1	A_3	4	2

BWT: any problems?

We need space to store ranks

- Store "checkpoints"

F	L	N(A)	N(B)
\$	A_0	-	-
A_0	B_0	1	1
A_1	B_1	-	-
A_2	A_1	-	-
A_3	\$	-	-
B_0	A_2	3	2
B_1	A_3	-	-

BWT: any problems?

We need space to store ranks

- Store "checkpoints"

F	L	N(A)	N(B)
\$	A_0	-	1-1
A_0	B_0	1	1
A_1	B_1	-	-
A_2	A_1	-	-
A_3	\$	-	2-0
B_0	A_2	3	2
B_1	A_3	-	-

BWT: any problems?

We want to get positions of P in T string

- Naive solution: Add suffix array

SA	F	L
6	\$	A_0
5	A_0	B_0
2	A_1	B_1
3	A_2	A_1
0	A_3	\$
4	B_0	A_2
1	B_1	A_3

BWT: any problems?

We want to get positions of P in T string

- Naive solution: Add suffix array
- Build "partial suffix array": randomly select elements of suffix array to save

SA	F	L
6	\$	A_0
-	A_0	B_0
2	A_1	B_1
-	A_2	A_1
0	A_3	\$
4	B_0	A_2
-	B_1	A_3

BWT: vs opponents

- Suffix Tree: >40 GB
- Suffix Array: >16 GB
- FM-index: <3GB

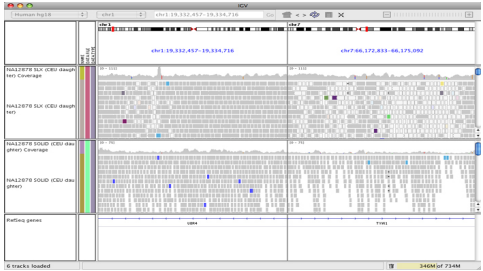
Burrows-Wheeler Transform

Aligning short-sequencing reads to genome.

- BWA-mem
- Bowtie, Bowtie2 (DNA-seq)
- Tophat, Tophat2, Hisat2 (RNA-seq)

Nucleotide Sequence Alignment

RNA-seq



DNA-seq

Nucleotide Sequence Alignment

```
@HD     VN:1.0  SO:coordinate
@SQ     SN:chr20        LN:64444167
@PG     ID:TopHat       VN:2.0.14       CL:/srv/dna_tools/tophat/tophat -N 3 --read-edit-dist 5 --read-rea
lign-edit-dist 2 -i 50 -I 5000 --max-coverage-intron 5000 -M -o out /data/user446/mapping_tophat/index/chr
20 /data/user446/mapping_tophat/L6_18_GTGAAA_L007_R1_001.fastq
HWI-ST1145:74:C101DACXX:7:1102:4284:73714        16      chr20   190930  3       100M    *       0       0
        CCGTGTTTAAAGGTGGATGCGGTCACCTTCCCAGCTAGGCTTAGGGATTCTTAGTTGGCCTAGGAAATCCAGCTAGTCCTGTCTCTCAGTCCCCCCTCT
C       BBDCCDDCCDDDDCDDDDDDCDCCCDBC?DDDDDDDDDDDDDDDDCDCDDDDDDDDDDCCCCEDDDC?DDDDDDDDDDDDDDDDDDDDDBDHFFFFDC@@
        AS:i:-15        XM:i:3  XO:i:0  XG:i:0  MD:Z:55C20C13A9 NM:i:3  NH:i:2  CC:Z:=  CP:i:55352714   HI:i:0
HWI-ST1145:74:C101DACXX:7:1114:2759:41961        16      chr20   193953  50      100M    *       0       0
        TGCTGGATCATCTGGTTAGTGGCTTCTGACTCAGAGGACCTTCGTCCCCTGGGGCAGTGGACCTTCCAGTGATTCCCCTGACATAAGGGGCATGGACGA
G       DCDDDDEDDDDDDDCDDDDDDDCCCDDDCDDDDDEEC>DFFFEJJJJJIGJJJIHGBHHGJIJJJJJJGJJJIJJJJJIHJJJJJJHHHHHFFFFFCCC
        AS:i:-16        XM:i:3  XO:i:0  XG:i:0  MD:Z:60G16T18T3 NM:i:3  NH:i:1
HWI-ST1145:74:C101DACXX:7:1204:14760:4030        16      chr20   270877  50      100M    *       0       0
        GGCTTTATTGGTAAAAAGGAATAGCAGATTTAATCAGAAATTCCCACCTGGCCCAGCAGCACCAACCAGAAAGAAGGGAAGAAGACAGGAAAAAACCA
C       DDDDDDDDDCCDDDDDDDDDDEEEEEEEFFFEFFEGHHHHFGDJJIHJJIJIJJJJIIIIGGFJJIHIIIIJJJJJJIGHHFAHGFHJHFGGHFFFDD@BB
        AS:i:-11        XM:i:2  XO:i:0  XG:i:0  MD:Z:0A85G13    NM:i:2  NH:i:1
HWI-ST1145:74:C101DACXX:7:1210:11167:8699        0       chr20   271218  50      50M4700N50M     *       0
        0       GTGGCTCTTCCACAGGAATGTTGAGGATGACATCCATGTCTGGGGTGCACTTGGGTCTCCGAAGCAGAACATCCTCAAATATGACCTCTCG
accepted_hits.sam
```

SAM Format