Read Alignment

Practical workshop on Large-Scale Genomic Data Analyses:
GWAS in structured populations

November 26th, 2018

Selene L. Fernández-Valverde

regRNAlab.github.io

@SelFdz

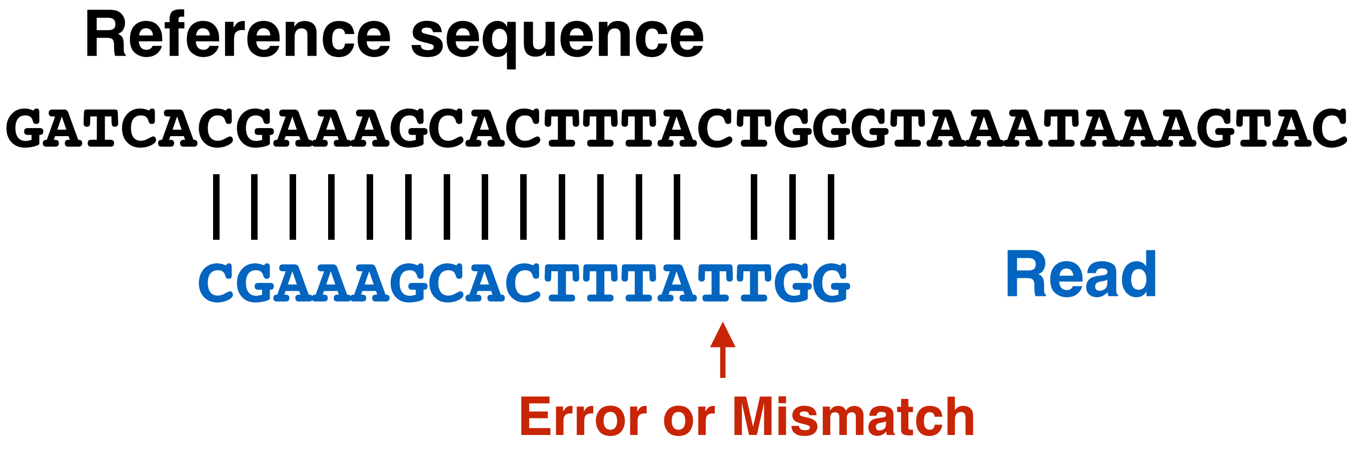
Learning objectives

In this lesson we'll learn:

- To align raw NGS reads to a genomic reference
- To understand the SAM and BAM formats

What does it mean to map a sequence?

- It is to identify the position of origin (high similarity) of ***reads*** or transcripts sequenced in a **reference sequence** (genomes or transcripts)



Reference sequence

```
GATCACGAAAGCACTTTACTGGGTAAATAAAGTAC
     ||||||||||||| |||
     CGAAAGCACTTTATTGG      Read
```

Error or Mismatch

We cannot use BLAST

- BLAST does a local alignment, which makes it very useful to look for partial and/or divergent alignments in large databases.

- BLAST is very slow to align sequences, which makes it impractical to align millions of sequences.

- Since we generally expect a high level of similarity to the reference in a massive sequencing experiment we need a semi-global and very fast alignment algorithm.

Burrows-Wheeler transform (BWT)

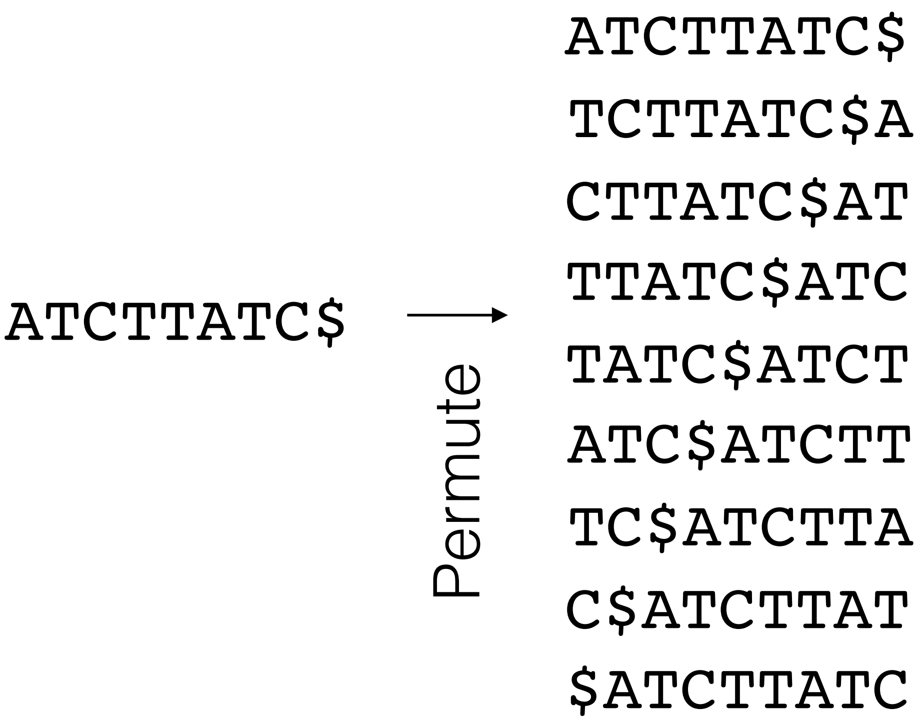
- Discovered by David Wheeler in 1983.
- Reversible permutation of the characters in a string - originally used to compress data.
- In 2005 it was found to be extremely useful in finding substrings.
- In 2009 it began to be used to align readings resulting from massive sequencing experiments.
- Together with compressed indexes (e.g. FM index) it allows the alignment time to grow linearly with the number of sequences.
- Allows to align ~ 100 million reads per hour (Bowtie - 1 thread only)

Generating a BWT

```
ATCTTATC$
```

$ - Character that indicates the end of a string

Generating a BWT



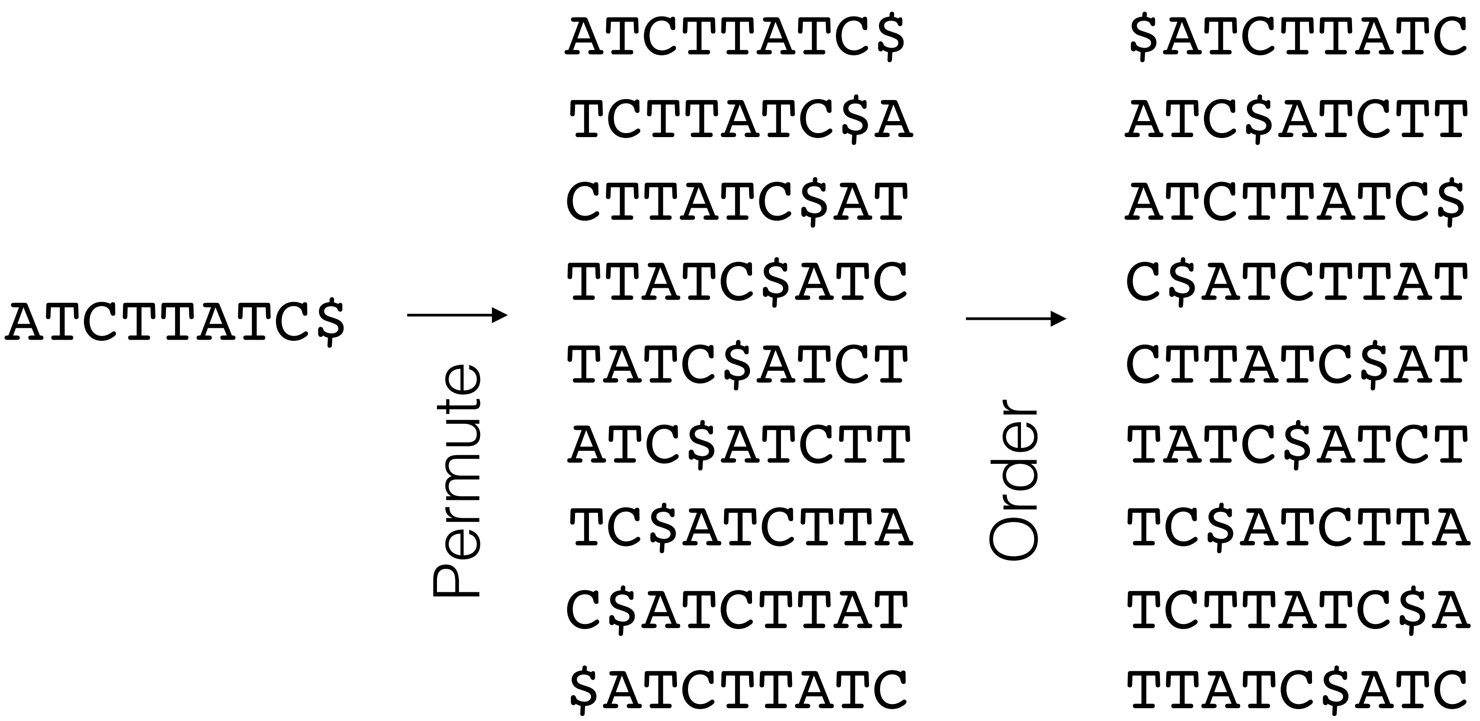
ATCTTATC$

Permute

ATCTTATC$
TCTTATC$A
CTTATC$AT
TTATC$ATC
TATC$ATCT
ATC$ATCTT
TC$ATCTTA
C$ATCTTAT
$ATCTTATC

$ - Character that indicates the end of a string

Generating a BWT



ATCTTATC$

Permute

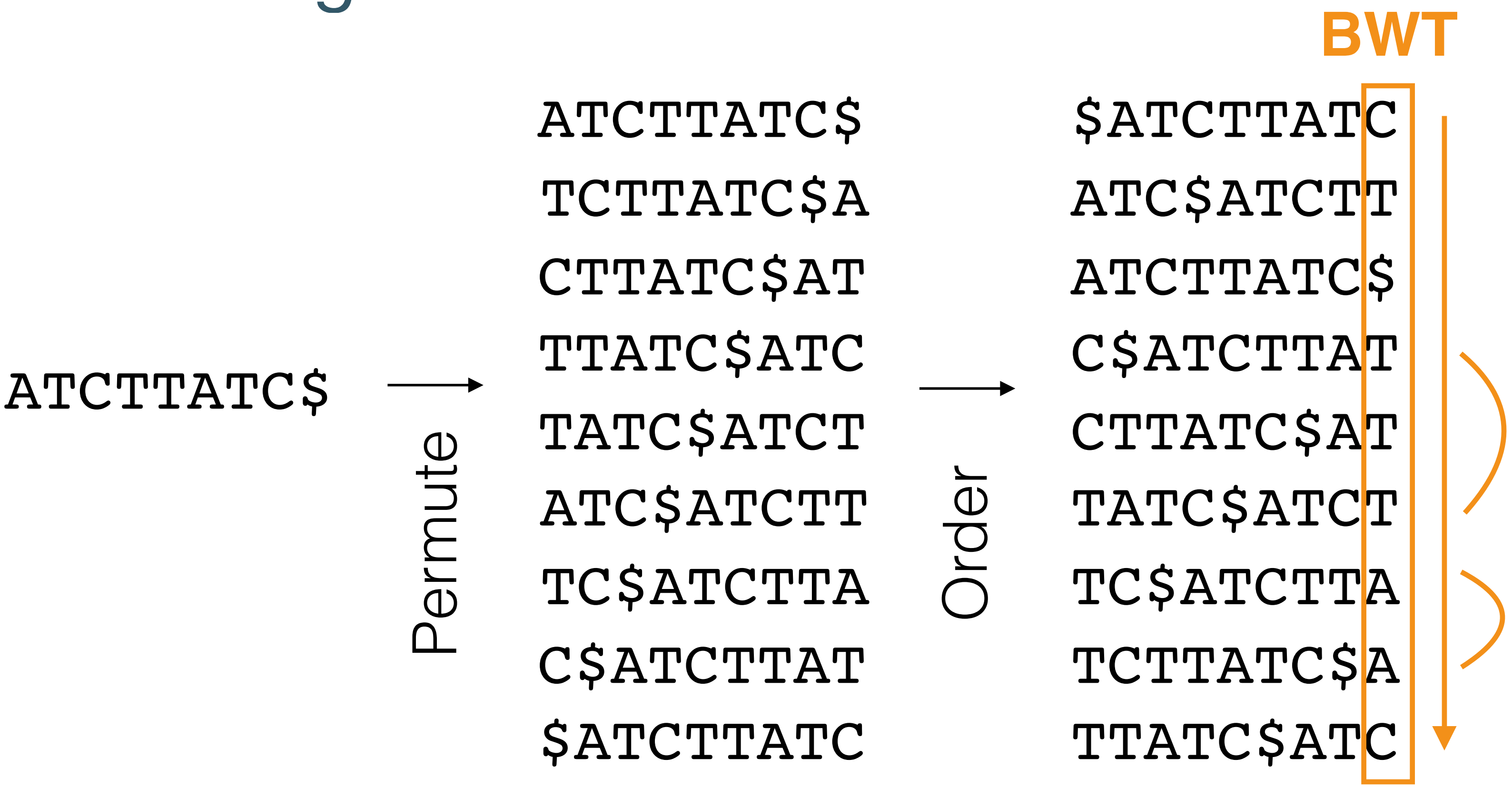
ATCTTATC$
TCTTATC$A
CTTATC$AT
TTATC$ATC
TATC$ATCT
ATC$ATCTT
TC$ATCTTA
C$ATCTTAT
$ATCTTATC

Order

$ATCTTATC
ATC$ATCTT
ATCTTATC$
C$ATCTTAT
CTTATC$AT
TATC$ATCT
TC$ATCTTA
TCTTATC$A
TTATC$ATC

$ - Character that indicates the end of a string

Generating a BWT



ATCTTATC$

Permute

ATCTTATC$
TCTTATC$A
CTTATC$AT
TTATC$ATC
TATC$ATCT
ATC$ATCTT
TC$ATCTTA
C$ATCTTAT
$ATCTTATC

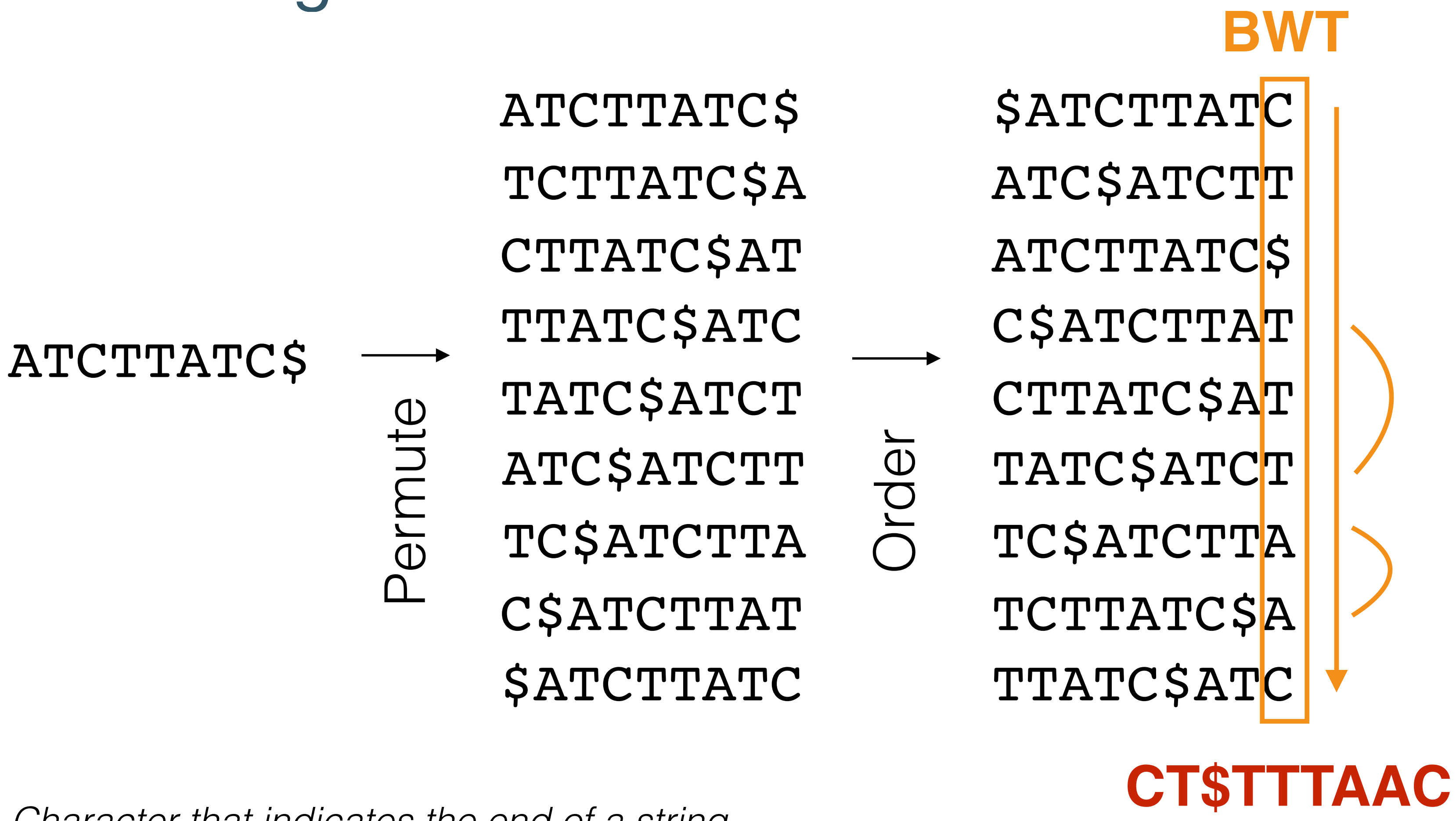
Order

BWT

$ATCTTATC
ATC$ATCTT
ATCTTATC$
C$ATCTTAT
CTTATC$AT
TATC$ATCT
TC$ATCTTA
TCTTATC$A
TTATC$ATC

$ - Character that indicates the end of a string

Generating a BWT



ATCTTATC$

Permute

ATCTTATC$
TCTTATC$A
CTTATC$AT
TTATC$ATC
TATC$ATCT
ATC$ATCTT
TC$ATCTTA
C$ATCTTAT
$ATCTTATC

Order

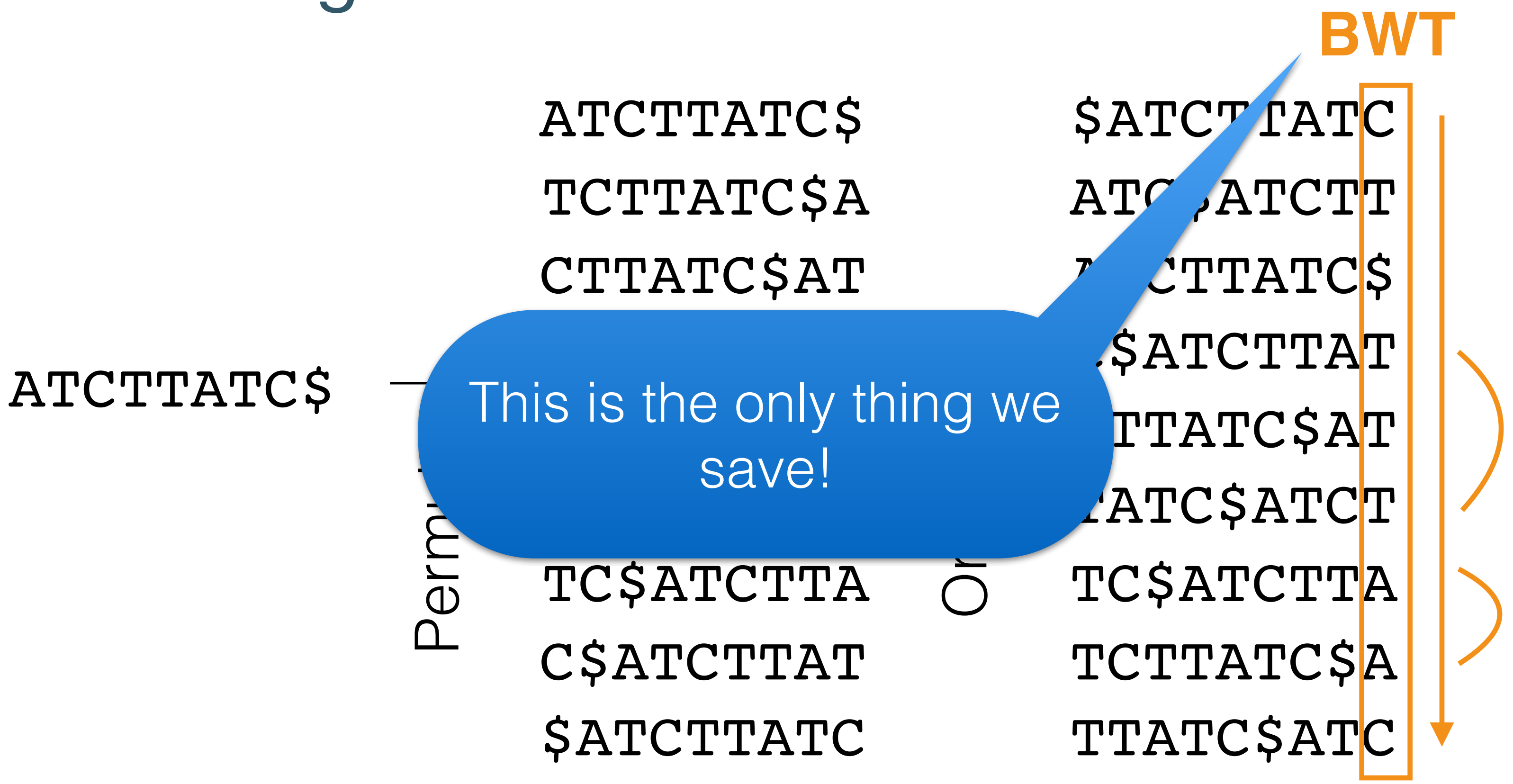
$ATCTTATC
ATC$ATCTT
ATCTTATC$
C$ATCTTAT
CTTATC$AT
TATC$ATCT
TC$ATCTTA
TCTTATC$A
TTATC$ATC

BWT

CT$TTTAAC

$ - Character that indicates the end of a string

Generating a BWT



This is the only thing we save!

CT$TTTAAC

$ - Character that indicates the end of a string

FT Property

Row			
0	$\$_0$	ATCTTAT	C_0
1	A_0	TC$ATCT	T_0
2	A_1	TCTTATC	$\$_0$
3	C_0	$ATCTTA	T_1
4	C_1	TTATC$A	T_2
5	T_0	ATC$ATC	T_3
6	T_1	C$ATCTT	A_0
7	T_2	CTTATC$	A_1
8	T_3	TATC$AT	C_1

FT Property

Row			BWT
0	$\$_0$	ATCTTAT	C_0
1	A_0	TC$ATCT	T_0
2	A_1	TCTTATC	$\$_0$
3	C_0	$ATCTTA	T_1
4	C_1	TTATC$A	T_2
5	T_0	ATC$ATC	T_3
6	T_1	C$ATCTT	A_0
7	T_2	CTTATC$	A_1
8	T_3	TATC$AT	C_1

F- First *L- Last*

FT Property

Row			BWT
0	$\$_0$	ATCTTAT	C_0
1	A_0	TC$ATCT	T_0
2	A_1	TCTTATC	$\$_0$
3	C_0	$ATCTTA	T_1
4	C_1	TTATC$A	T_2
5	T_0	ATC$ATC	T_3
6	T_1	C$ATCTT	A_0
7	T_2	CTTATC$	A_1
8	T_3	TATC$AT	C_1

F- First

L- Last

The range of characters are kept in the first (F) and last (L) column.

The first column can be rebuilt ordering the last

Reverting the BWT transform

Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Original sequence

Reverting the BWT transform

Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Original sequence

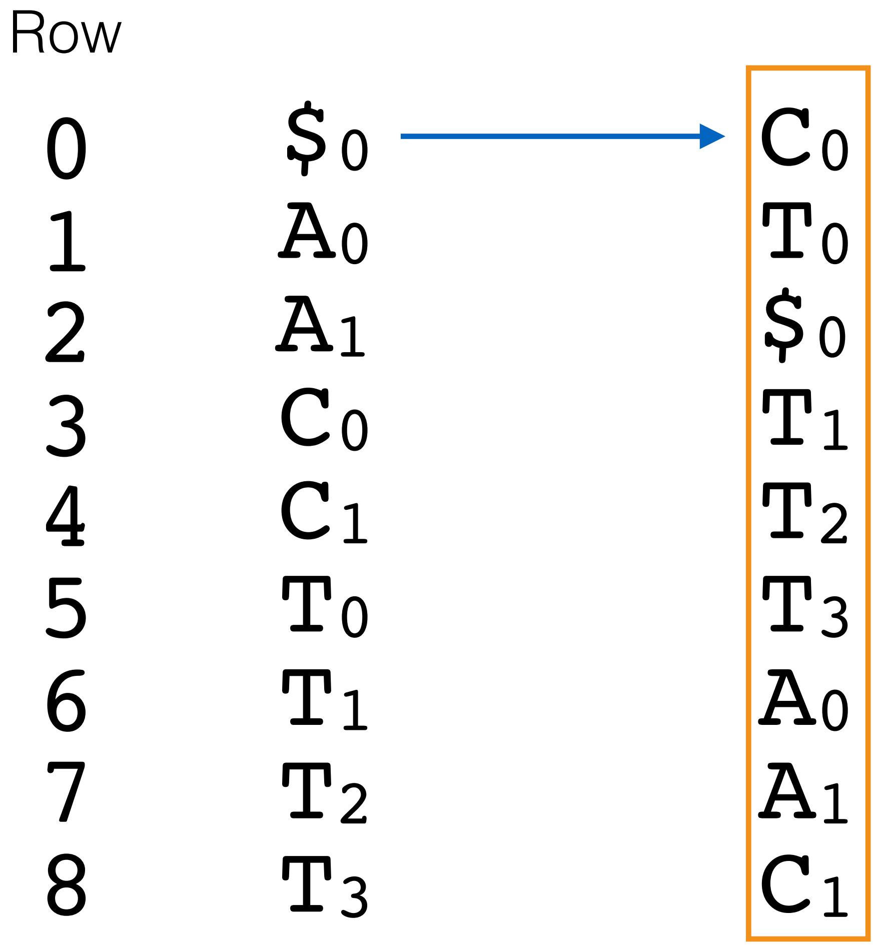
Reverting the BWT transform

Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$\$_0$

Original sequence

Reverting the BWT transform

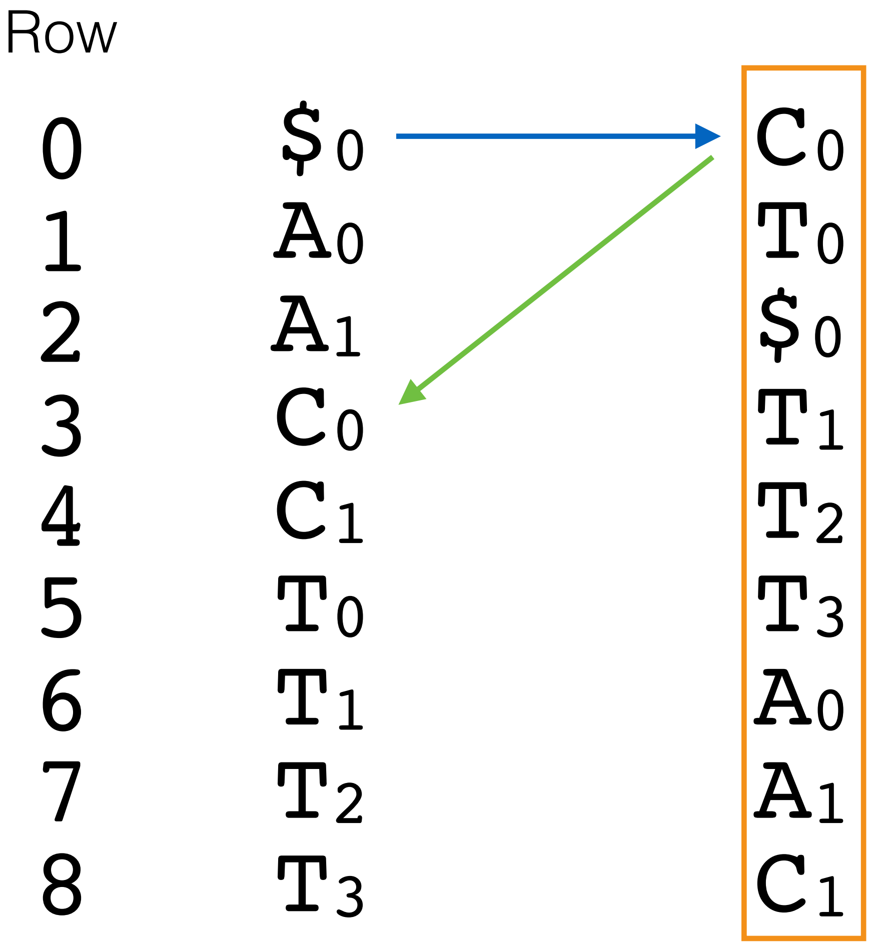


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$C_0 \$_0$

Original sequence

Reverting the BWT transform

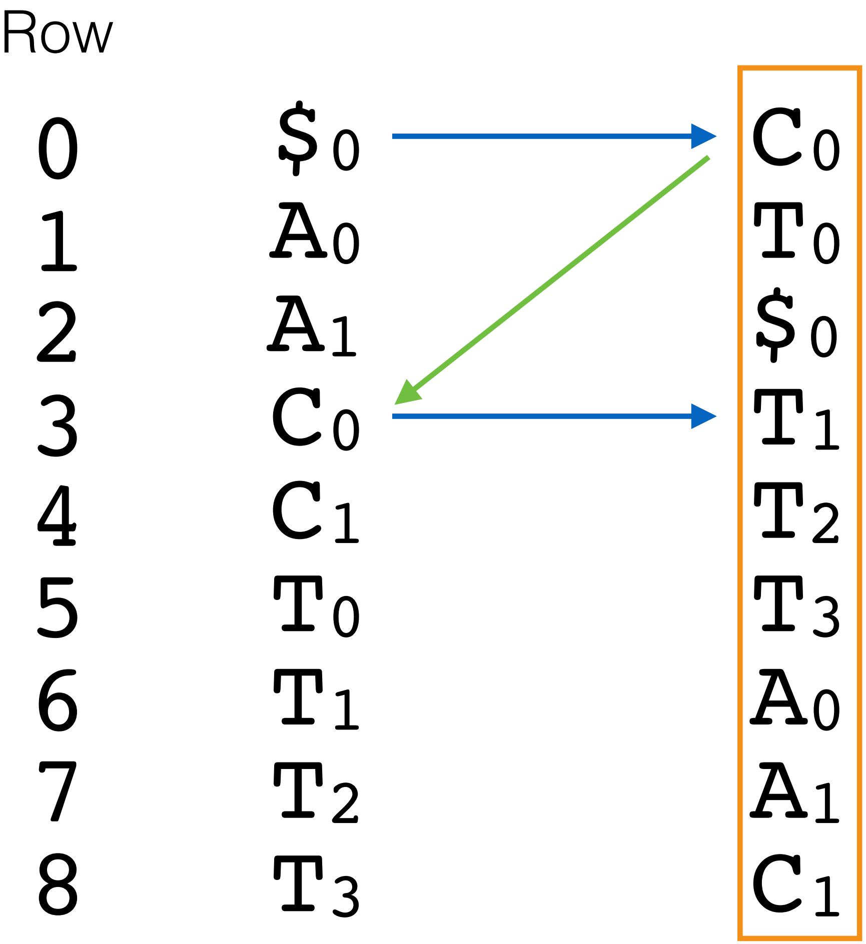


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$C_0 \ \$_0$

Original sequence

Reverting the BWT transform

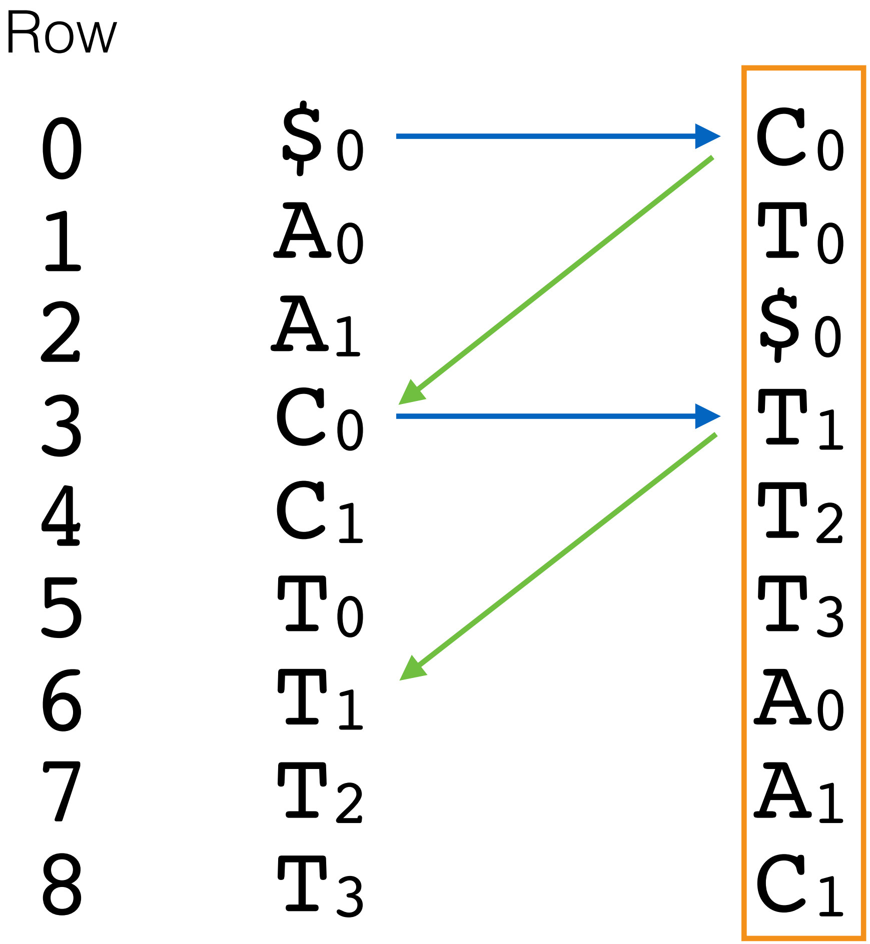


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

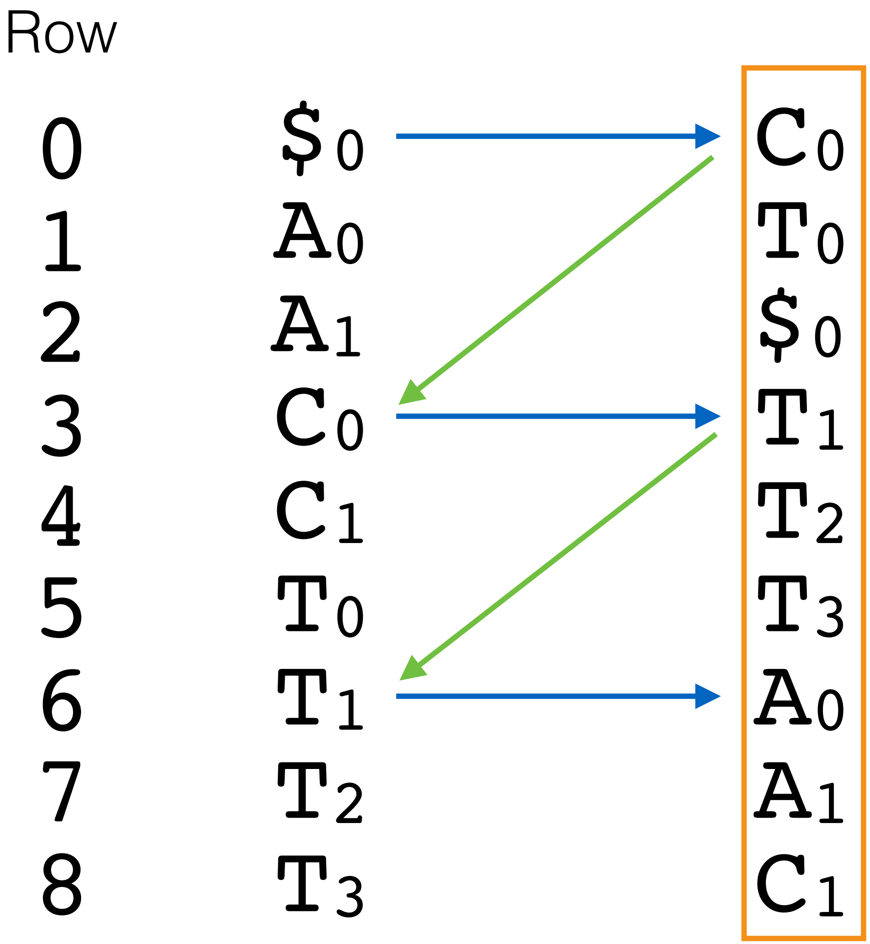


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

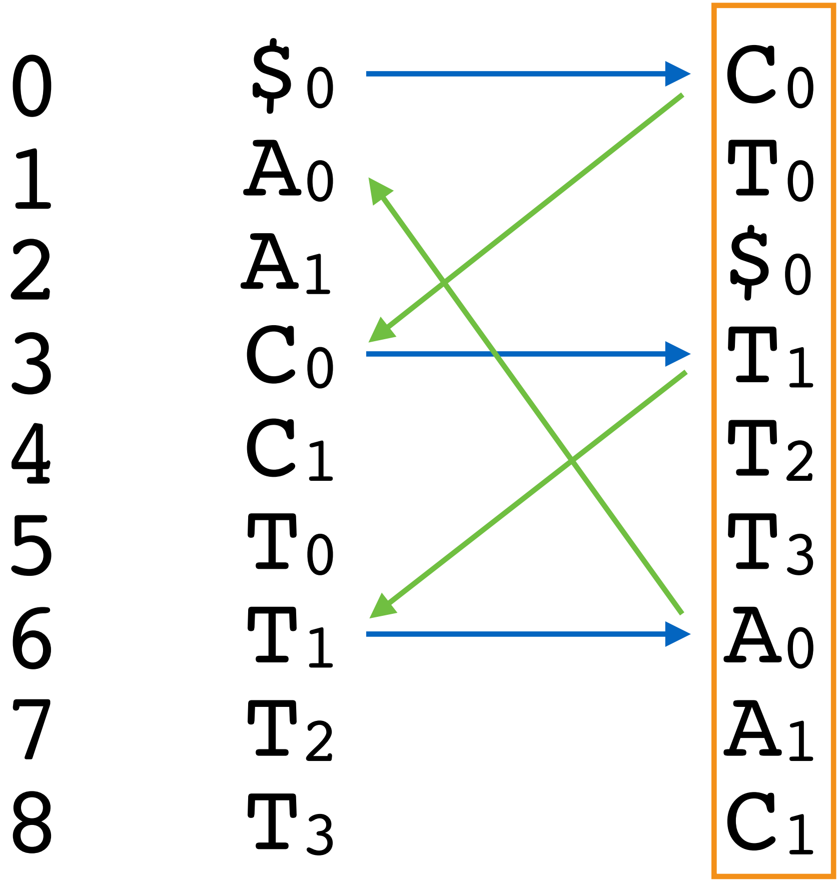


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

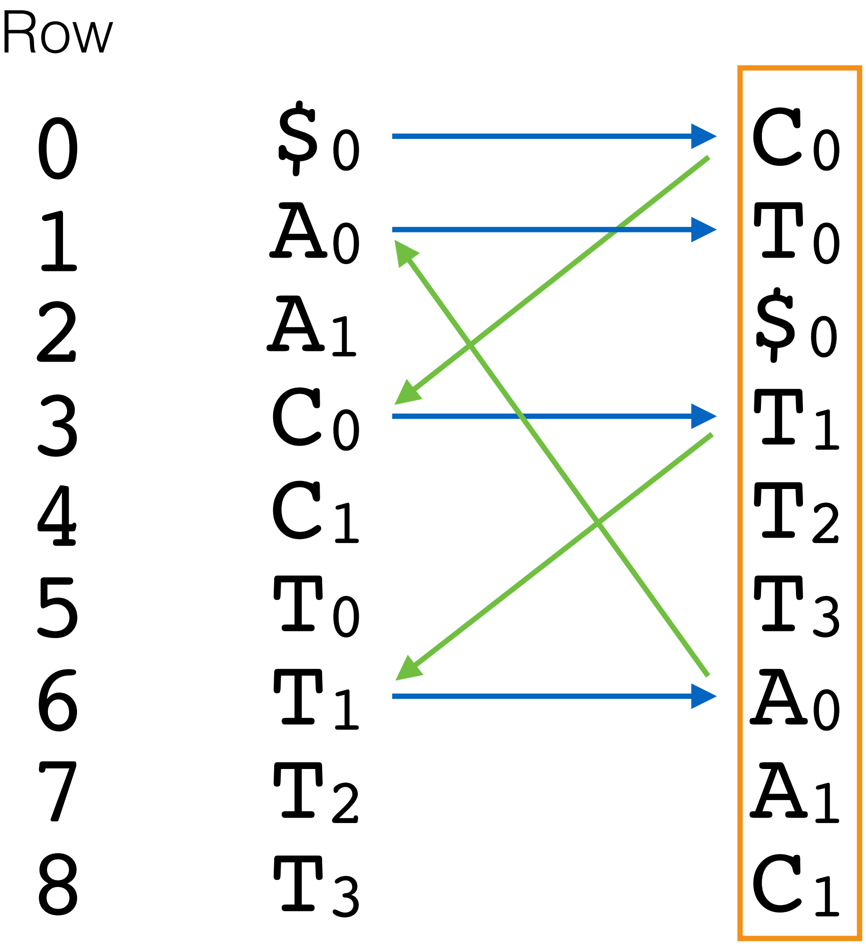


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

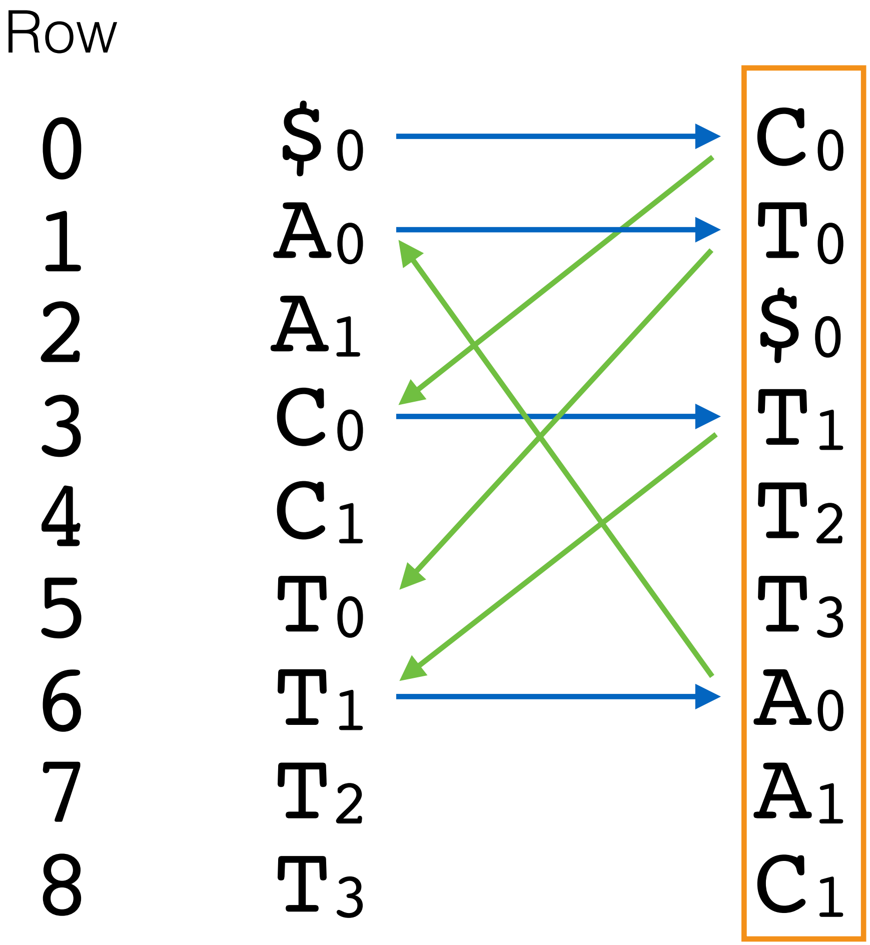


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

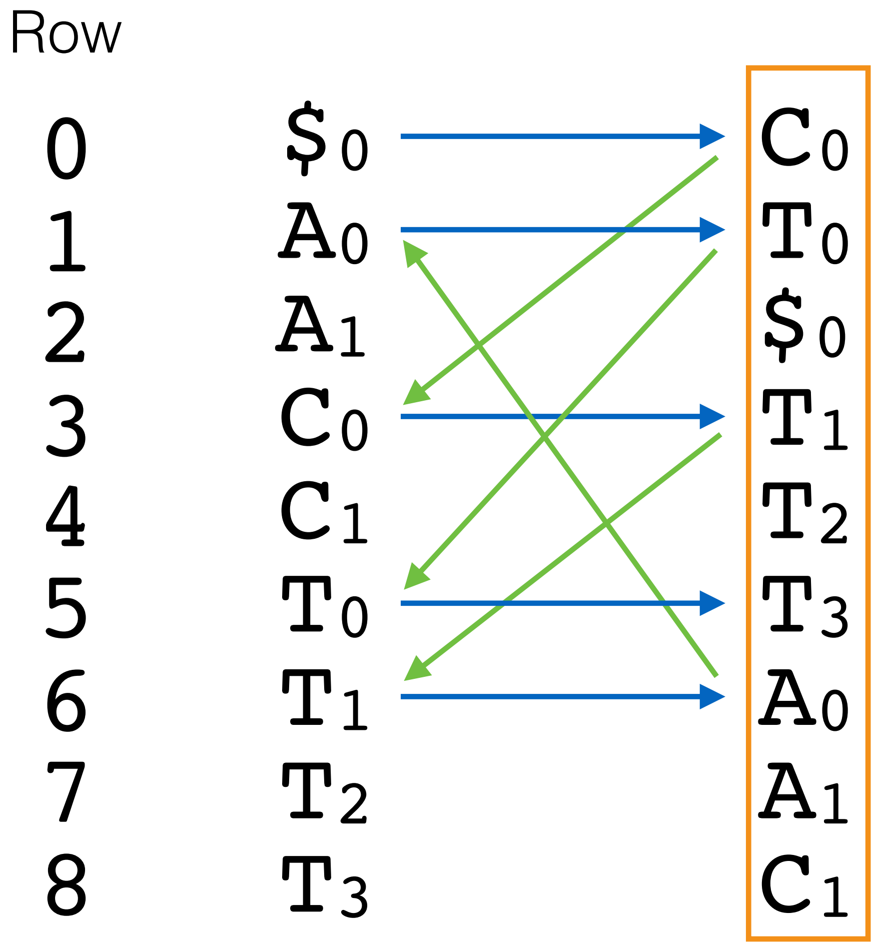


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

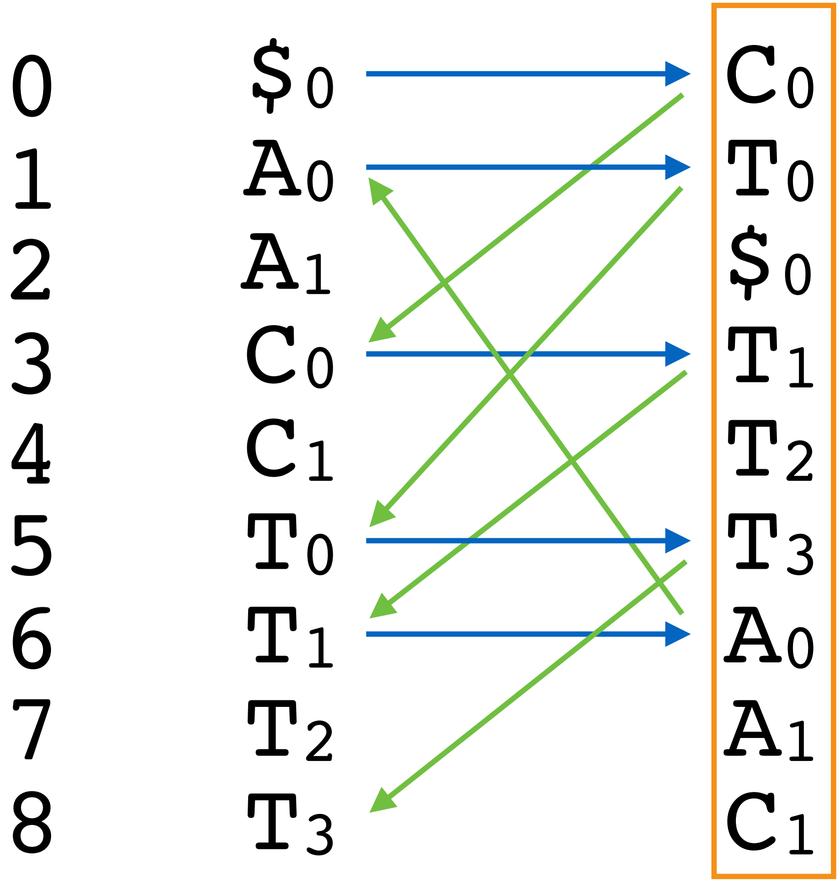


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

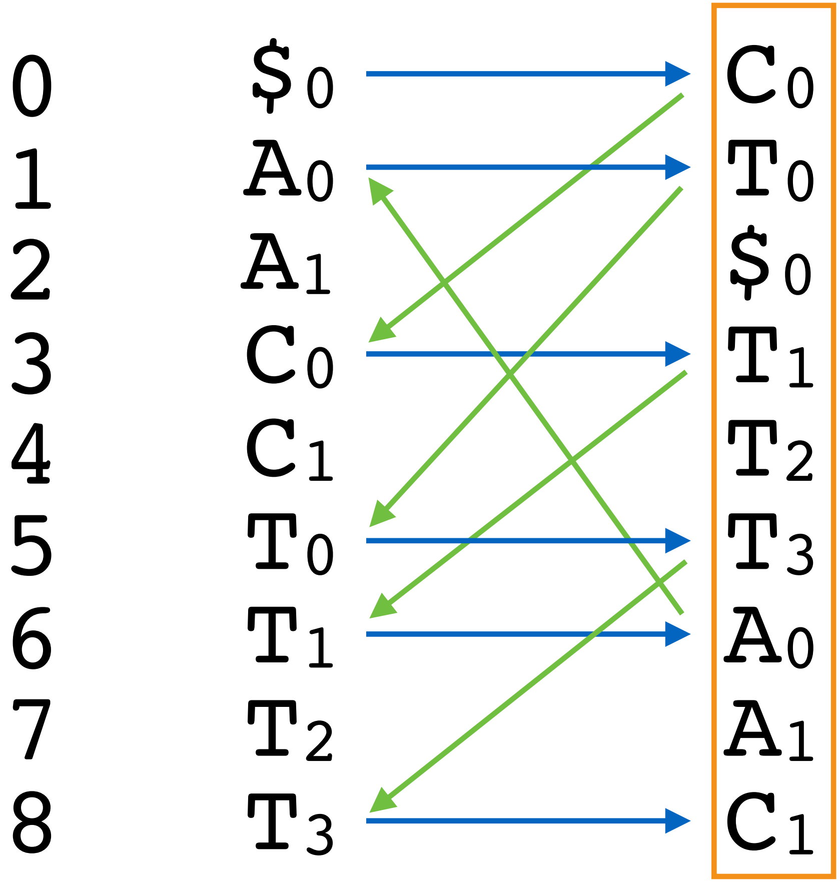


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

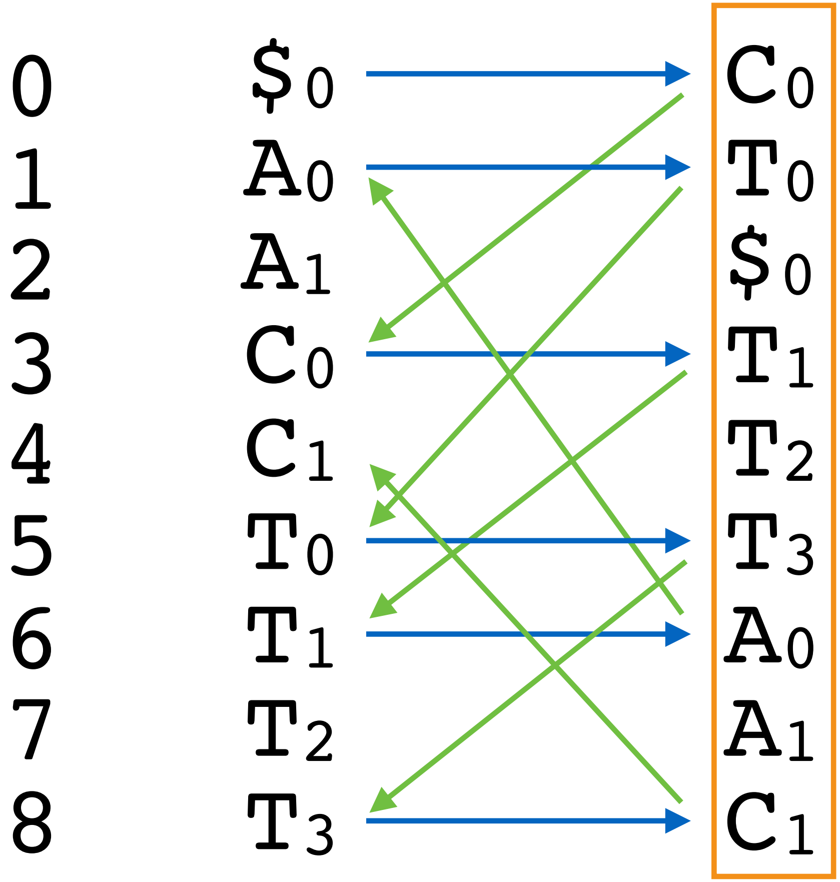


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

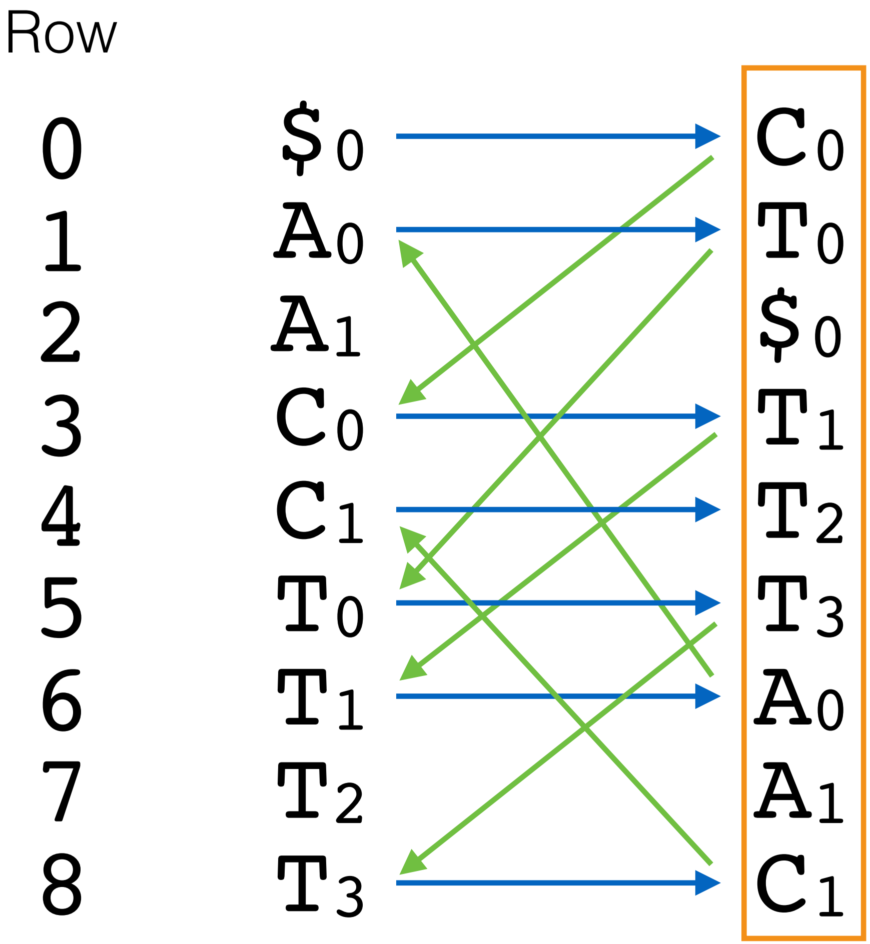


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

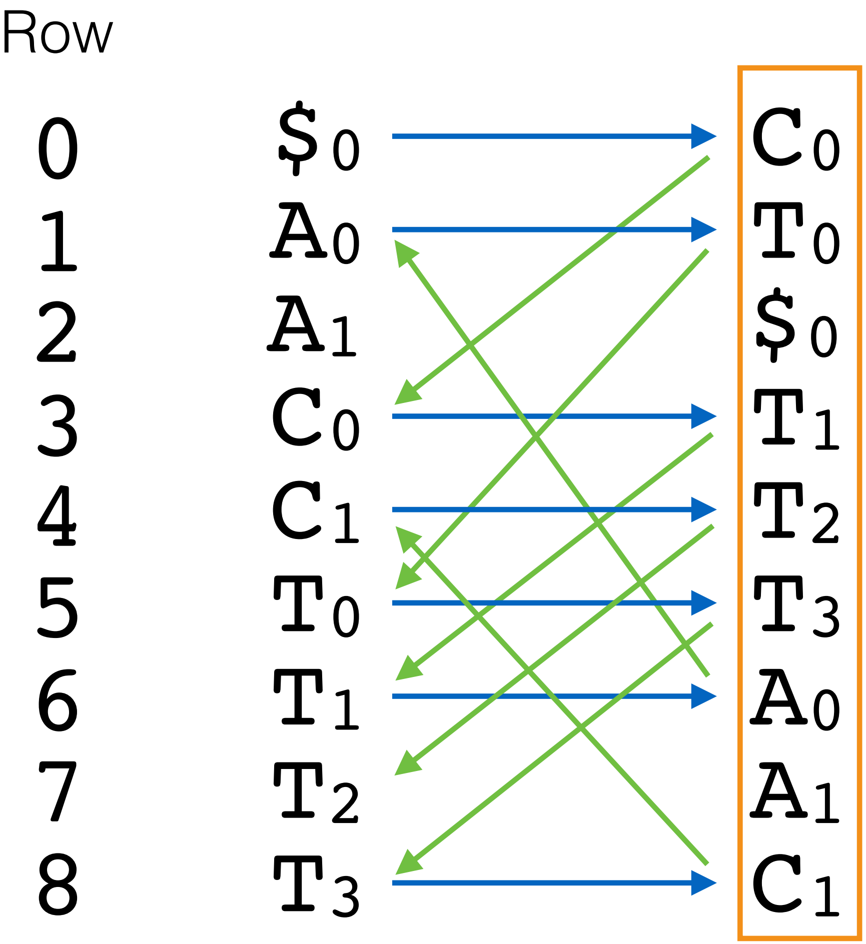


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_2 C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

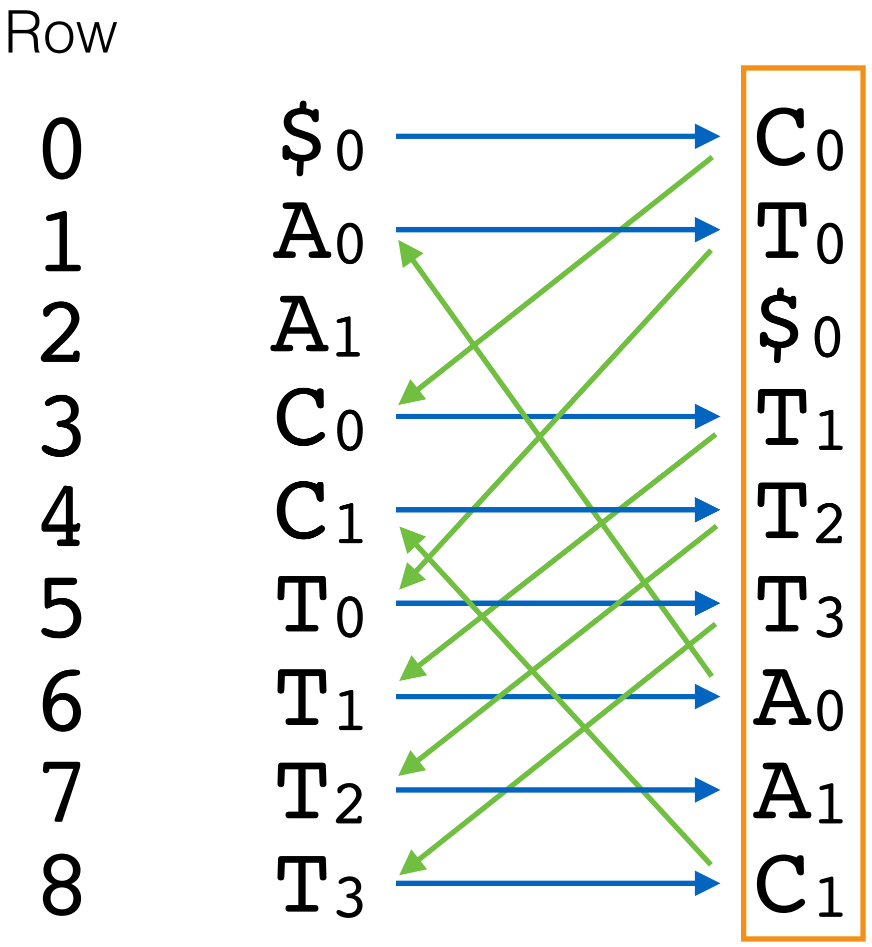


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$T_2\,C_1\,T_3\,T_0\,A_0\,T_1\,C_0\,\$_0$

Original sequence

Reverting the BWT transform

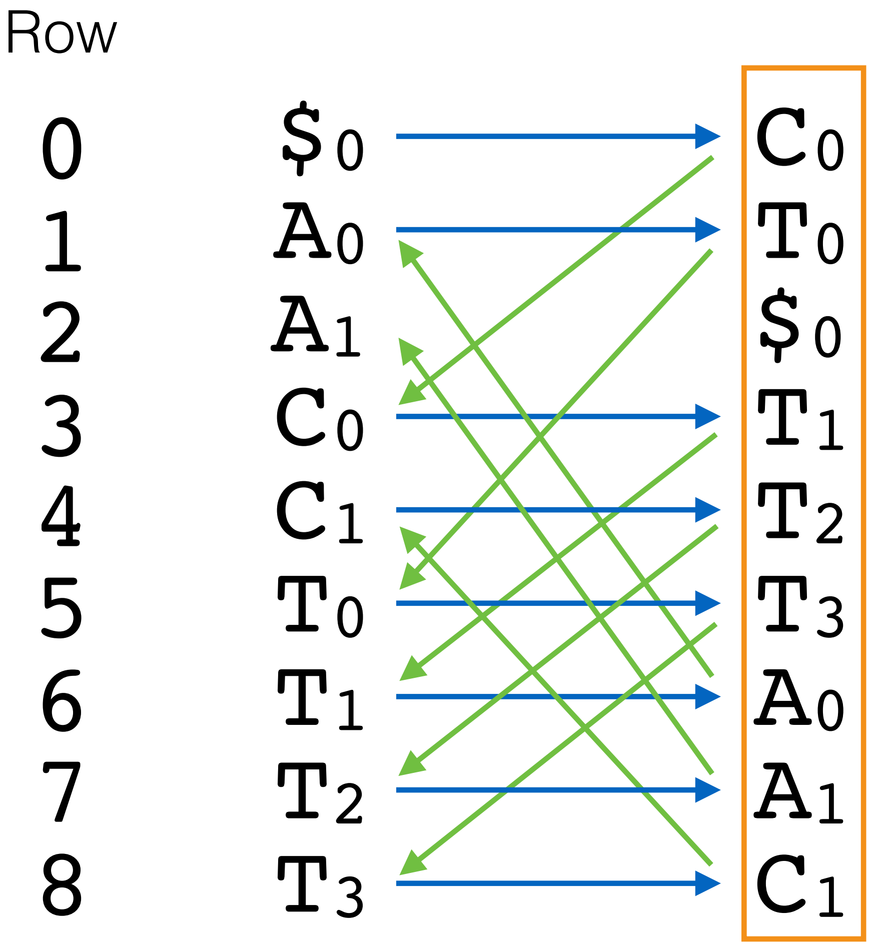


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$A_1 T_2 C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform

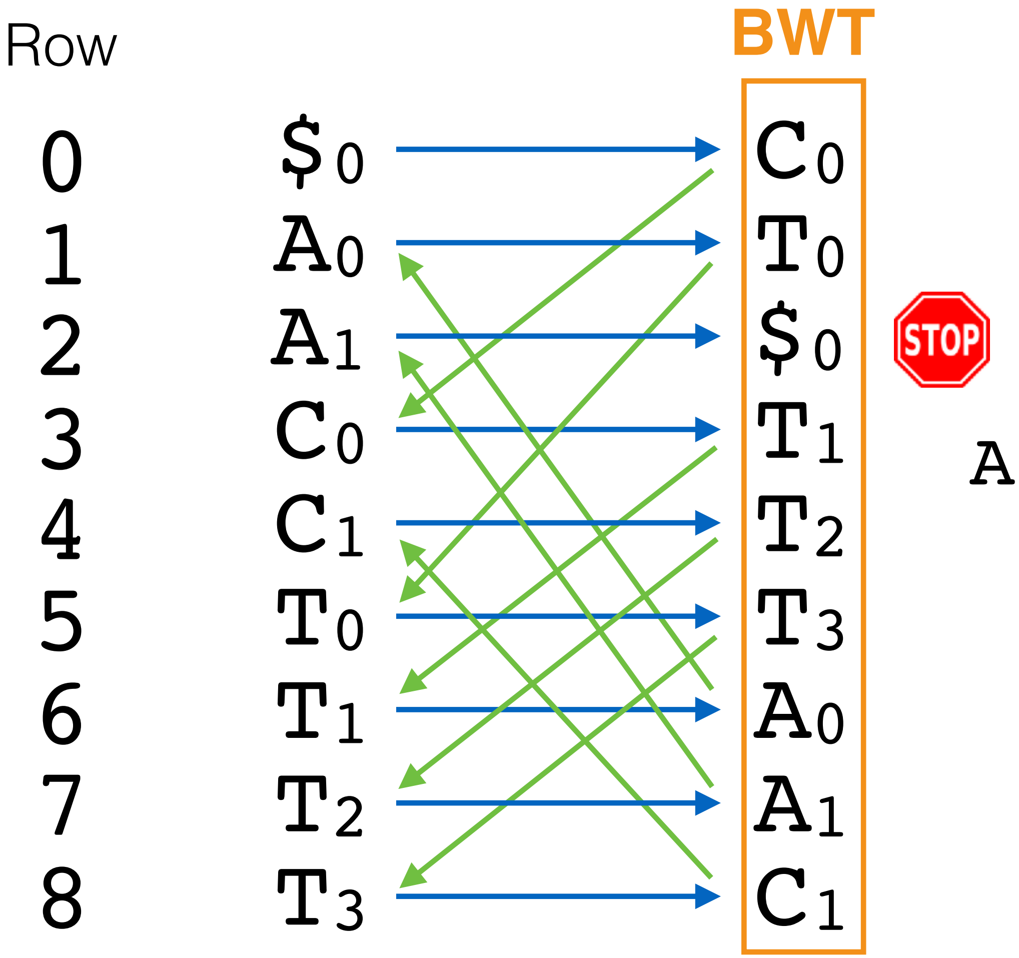


Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$A_1 T_2 C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Reverting the BWT transform



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

$A_1 T_2 C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

Original sequence

Using BWT to map

Row		
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Using BWT to map

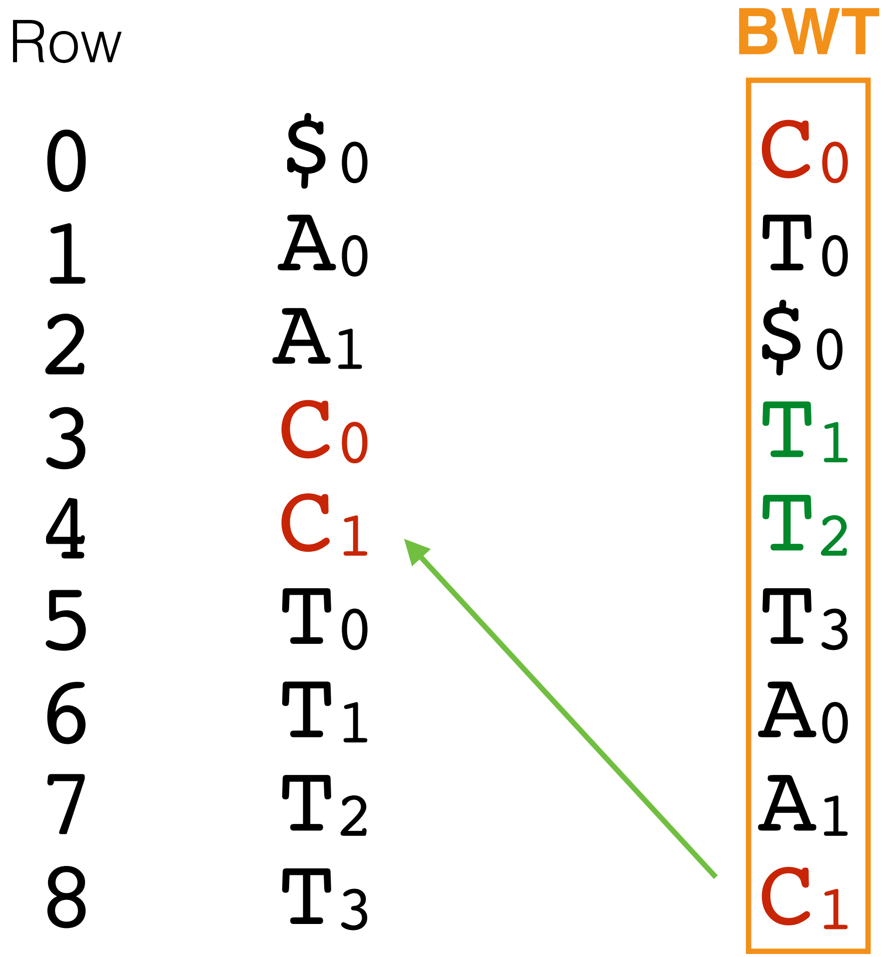
Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map

Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

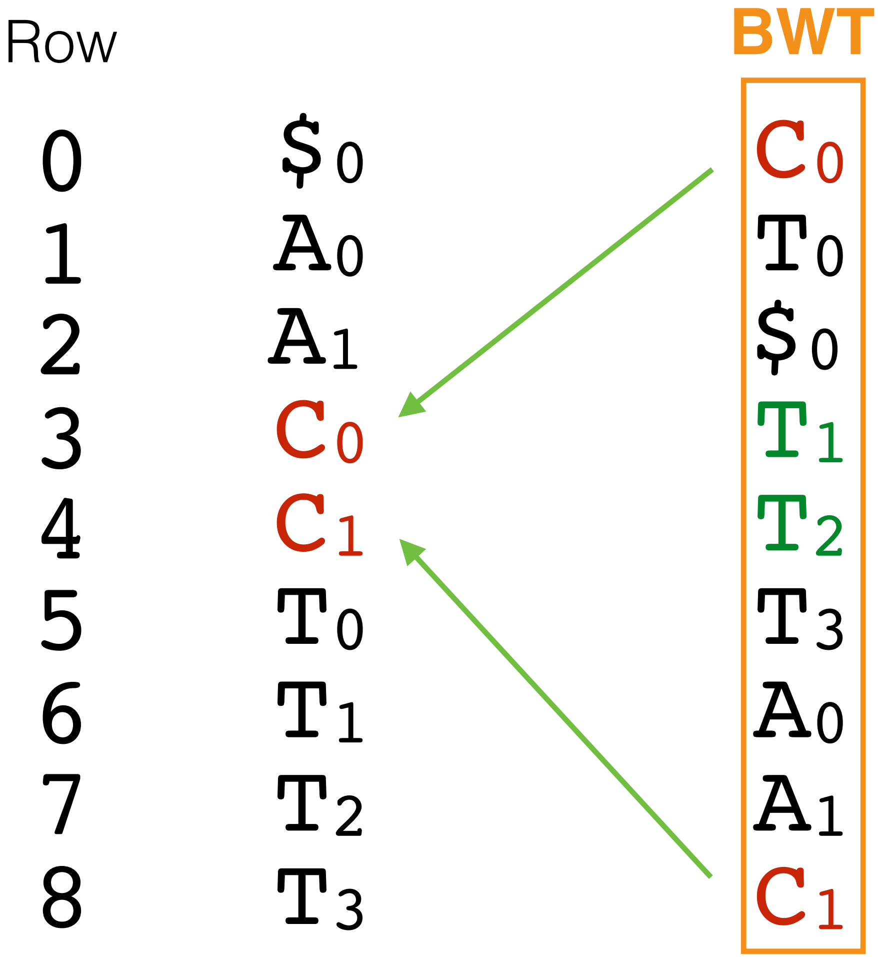
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

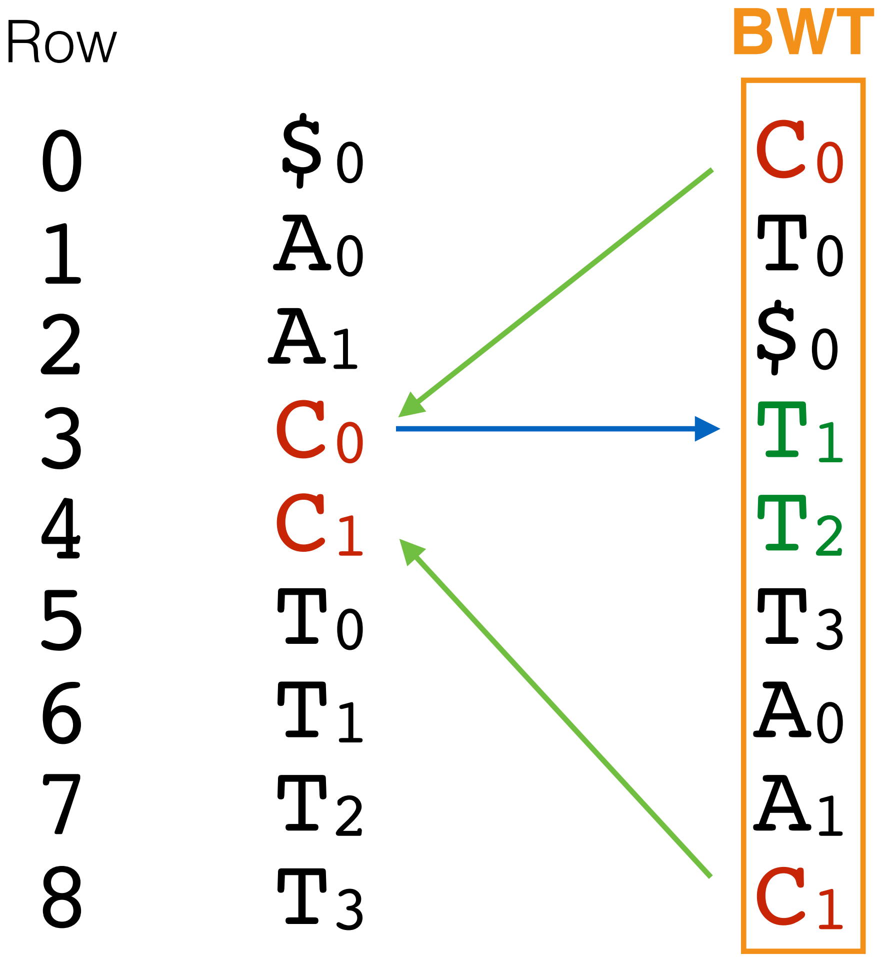
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

`Read: TTATC`

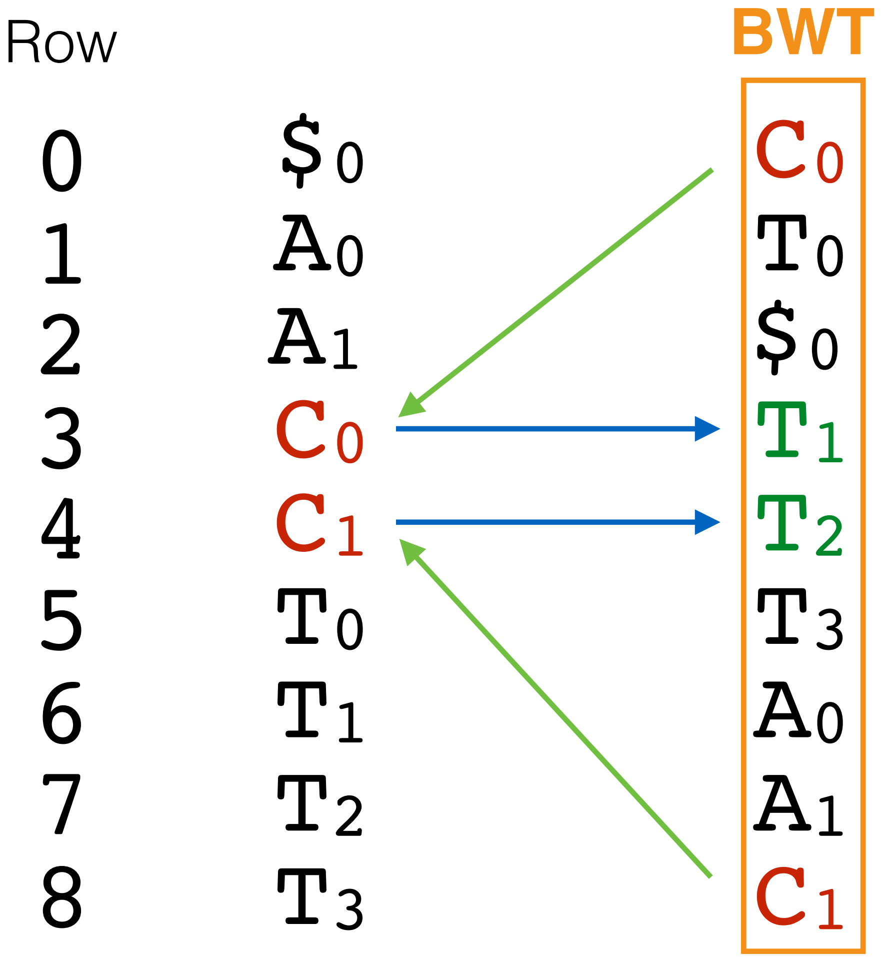
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map

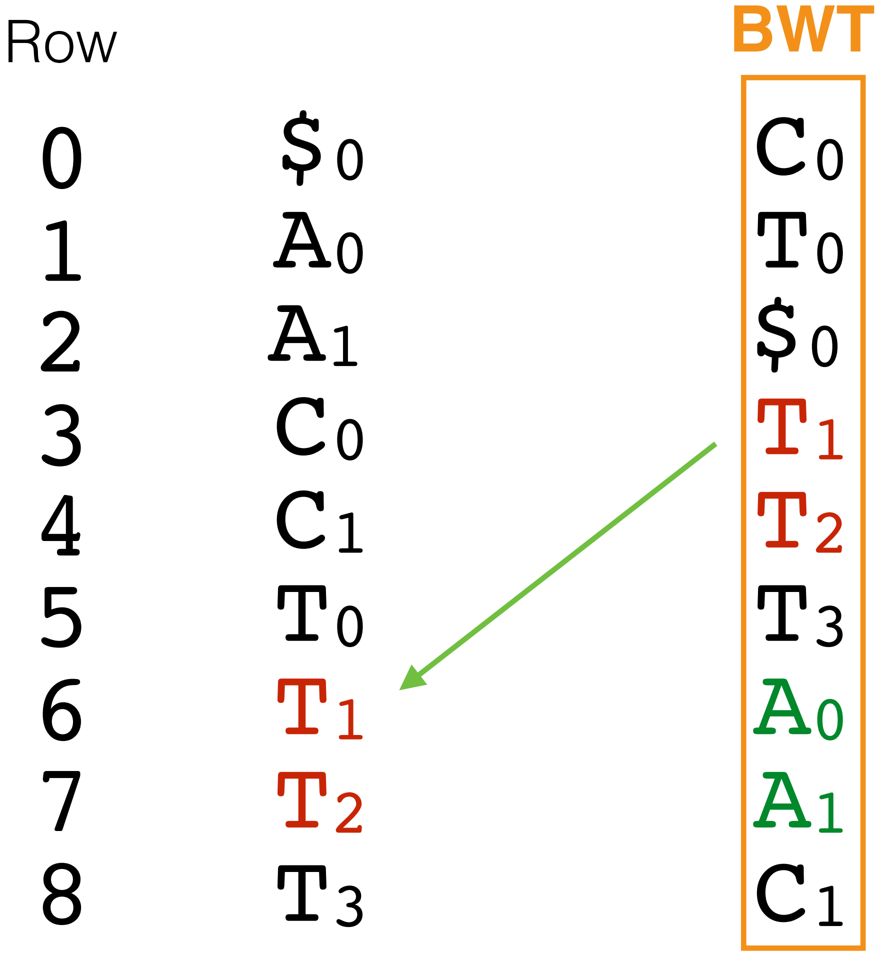
Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Using BWT to map

Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

`Read: TTATC`

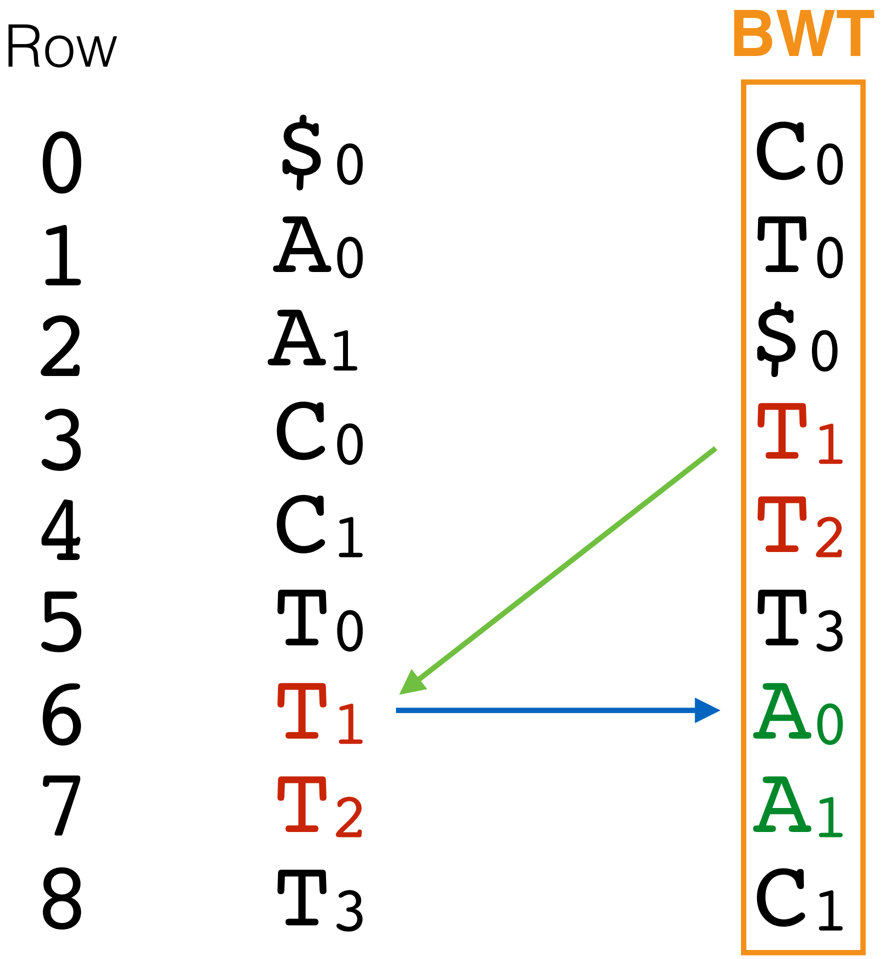
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

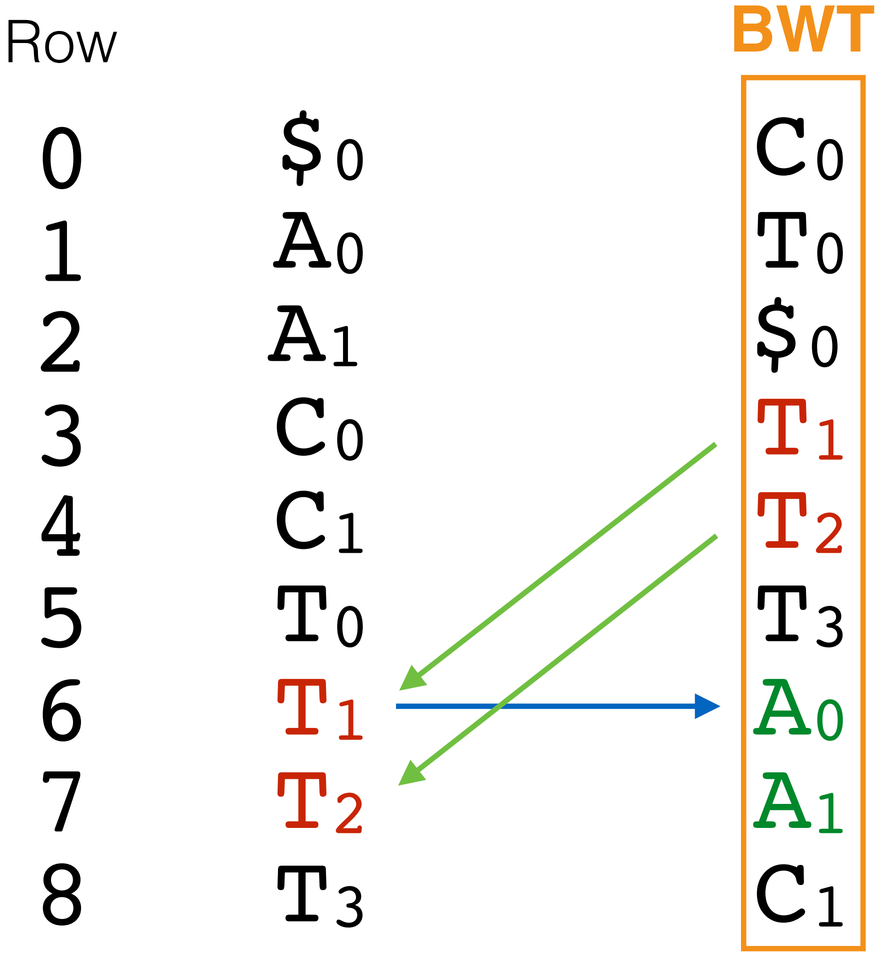
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

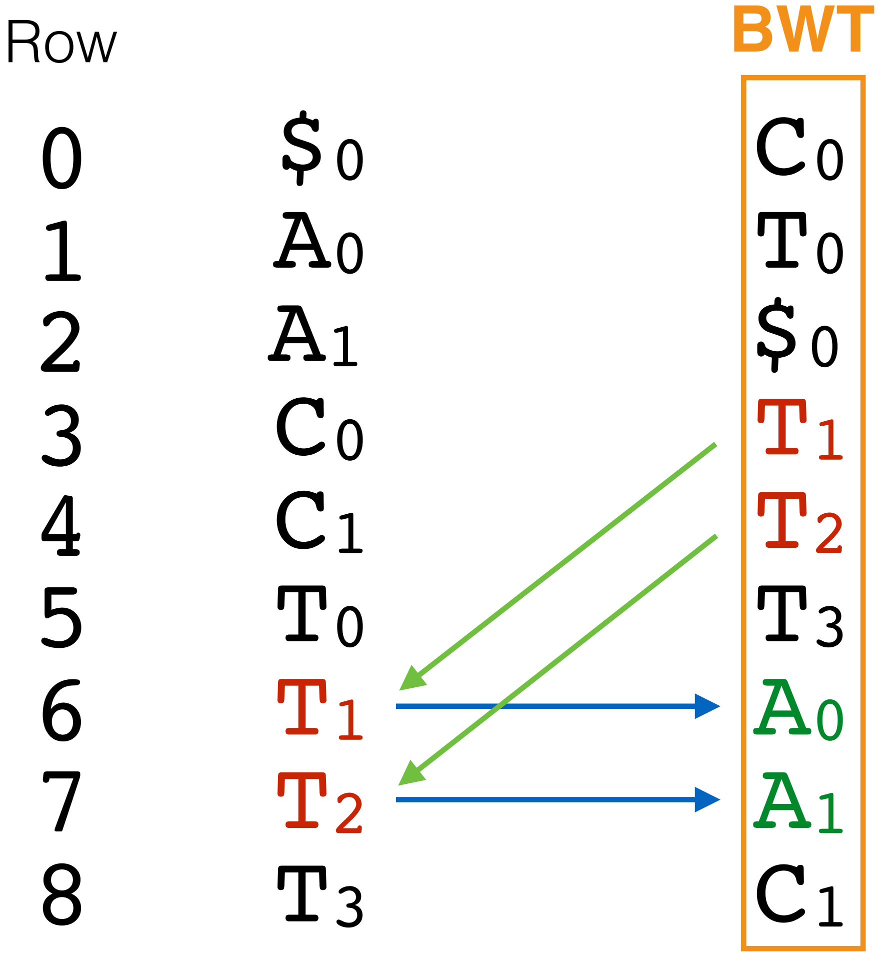
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

`Read: TTATC`

Using BWT to map

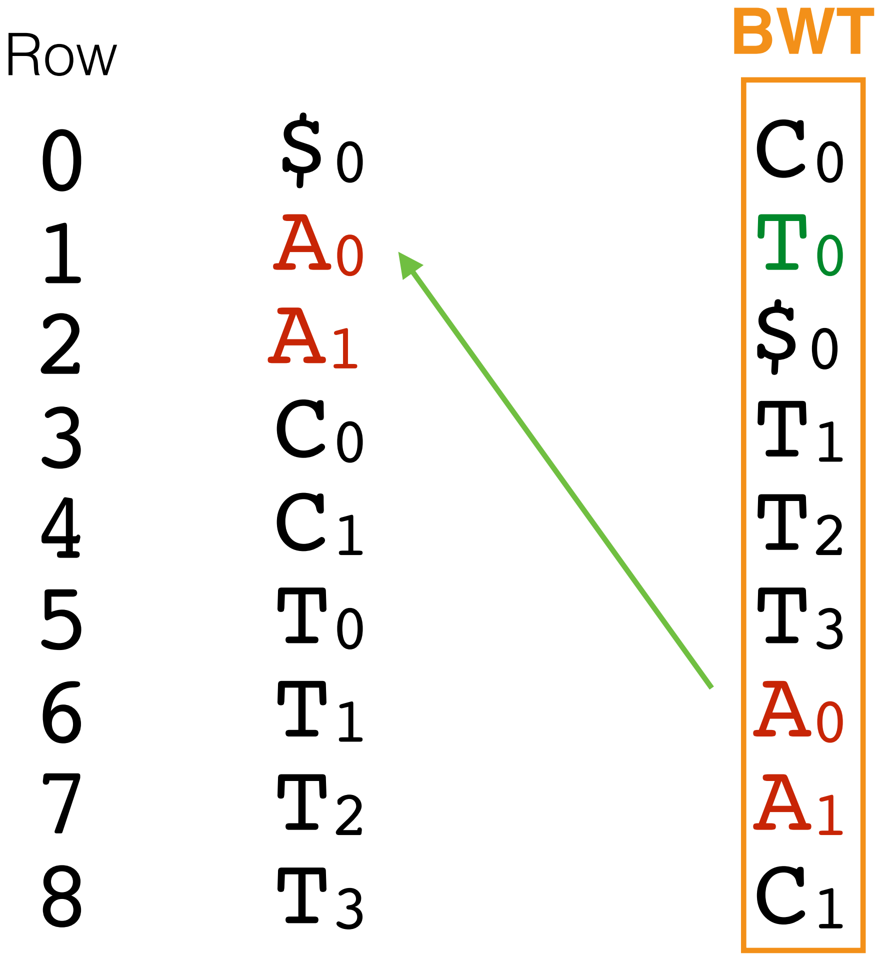
Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Using BWT to map

Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

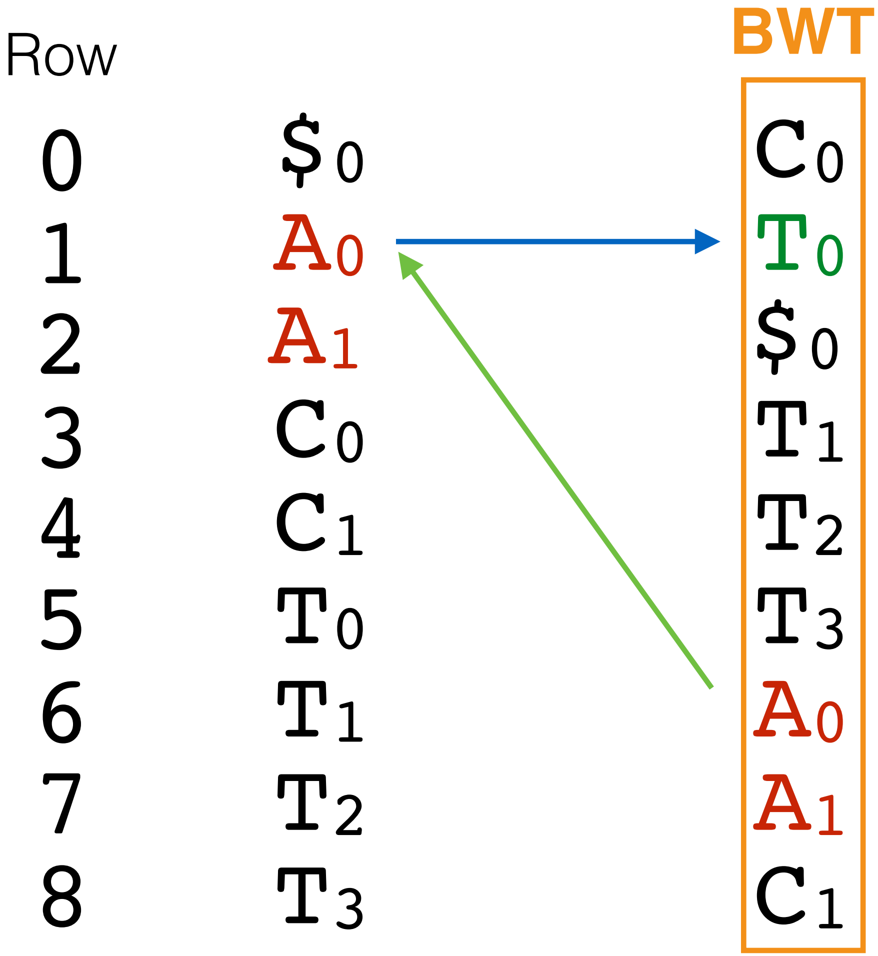
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

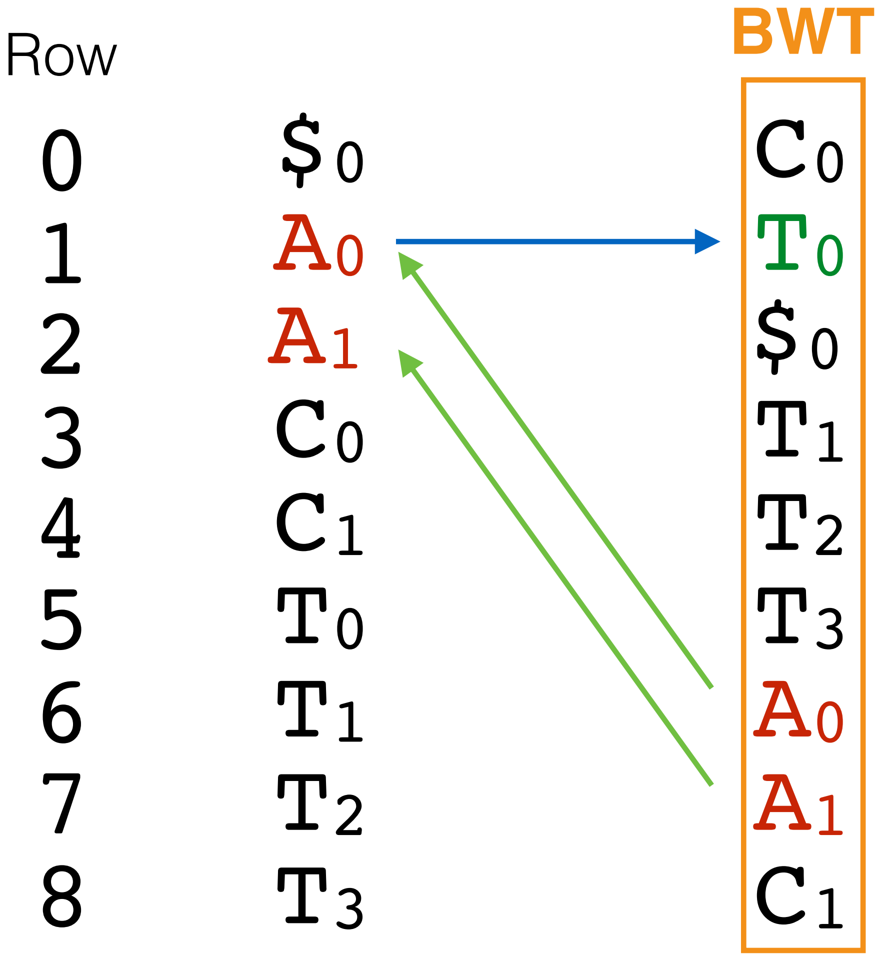
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

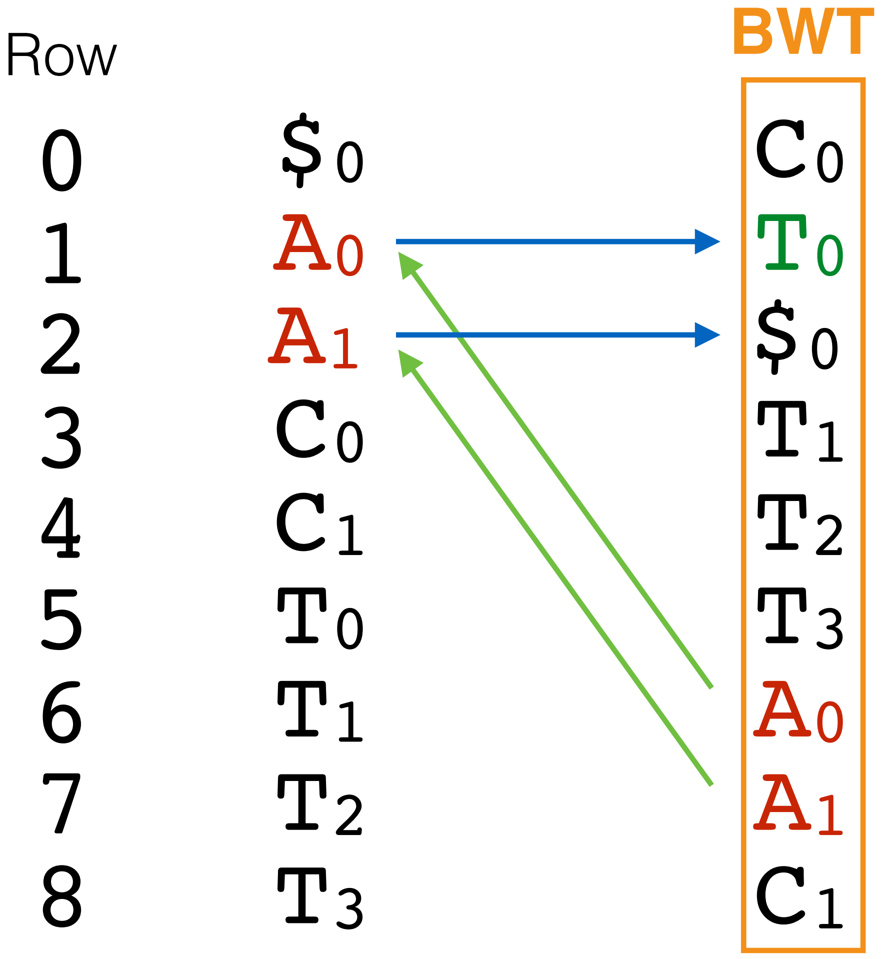
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map

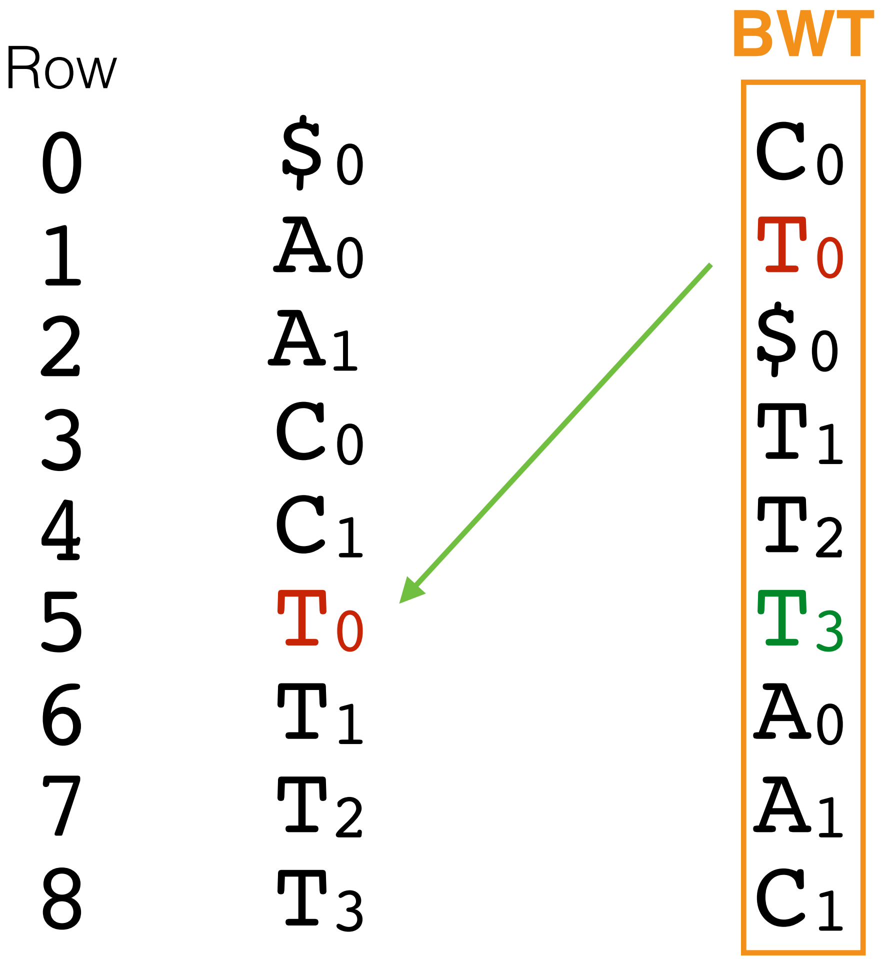
Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Using BWT to map

Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

`Read: TTATC`

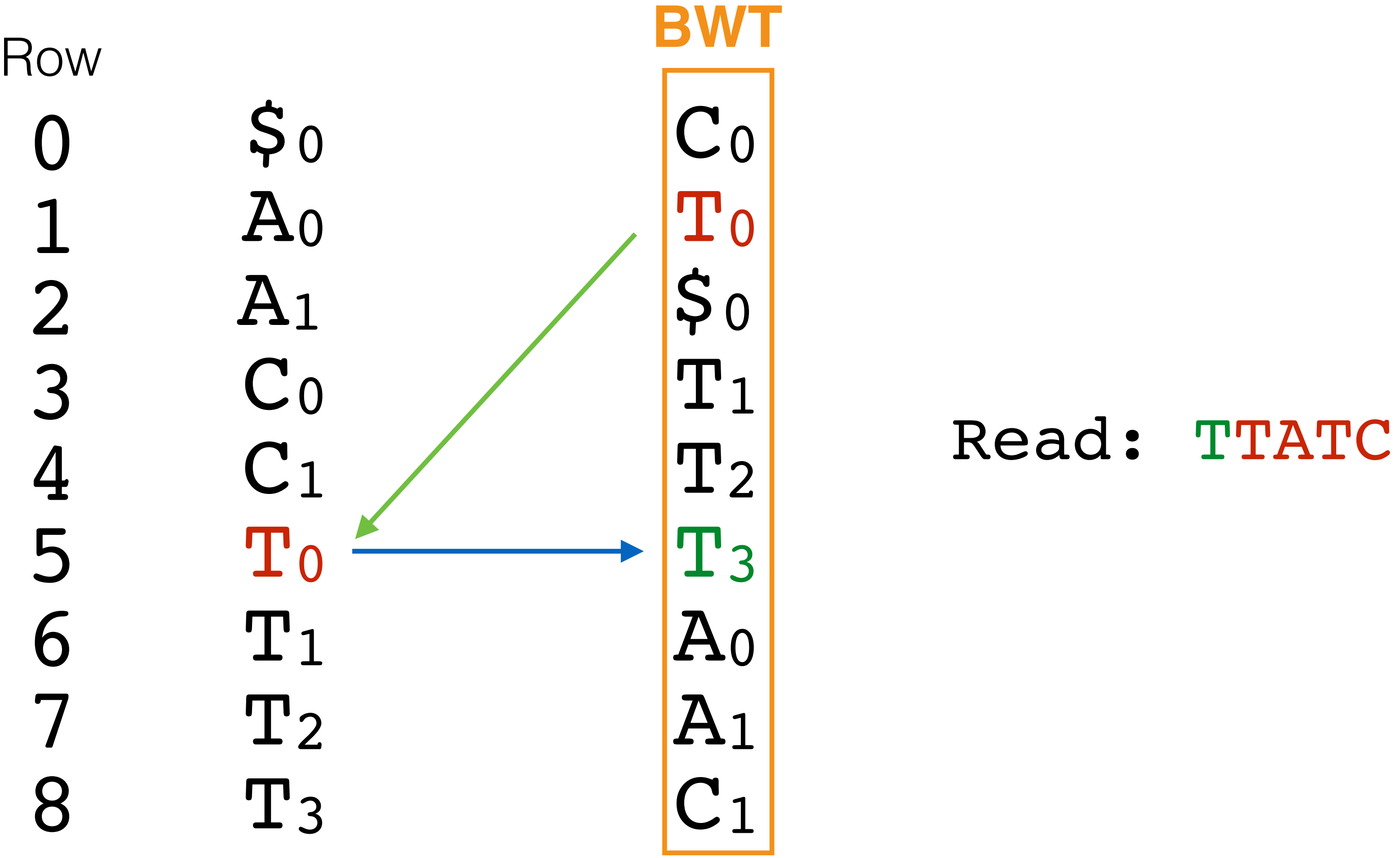
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

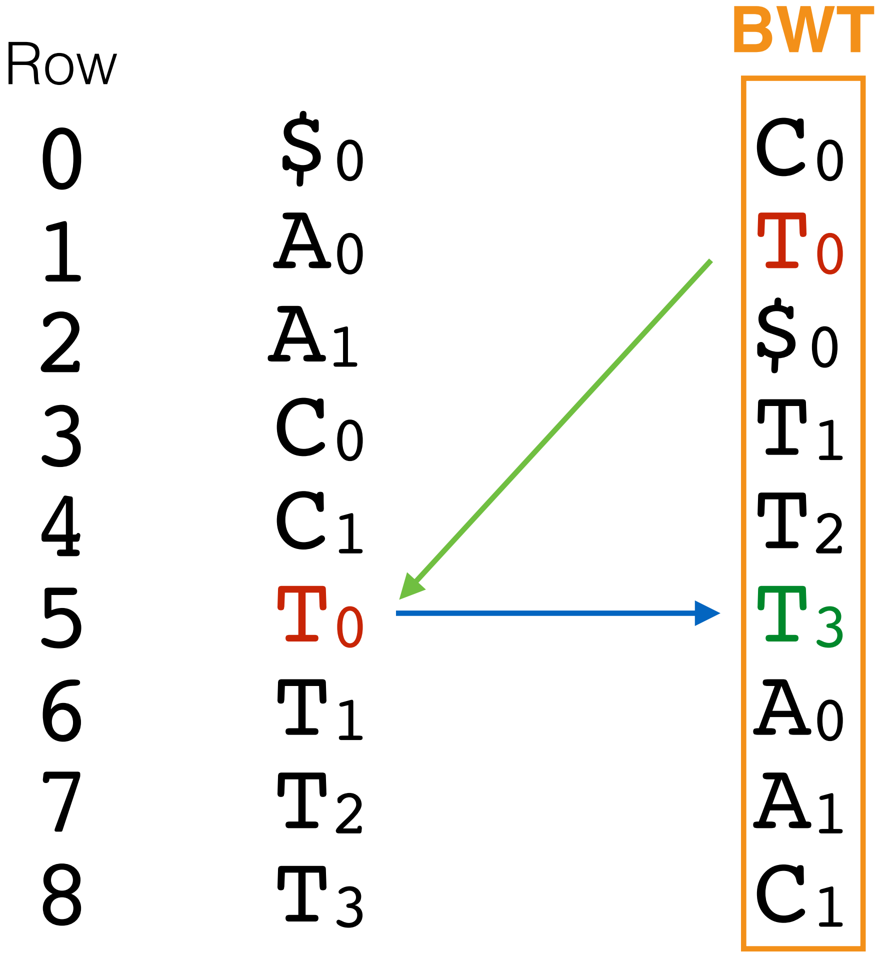
Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

Read: TTATC

Using BWT to map



Row		BWT
0	$\$_0$	C_0
1	A_0	T_0
2	A_1	$\$_0$
3	C_0	T_1
4	C_1	T_2
5	T_0	T_3
6	T_1	A_0
7	T_2	A_1
8	T_3	C_1

`Read: TTATC`

The read maps to our sequence but … Where is it in the genome?

Using BWT to map

Row		BWT	Suffix array
0	$\$_0$	C_0	8
1	A_0	T_0	5
2	A_1	$\$_0$	0
3	C_0	T_1	7
4	C_1	T_2	2
5	T_0	T_3	4
6	T_1	A_0	6
7	T_2	A_1	1
8	T_3	C_1	3

A suffix could indicate us where is it in the original sequence. Uses a lot of space if we have millions of positions

Using BWT to map

Row		BWT	Suffix array
0	$\$_0$	C_0	8
1	A_0	T_0	5
2	A_1	$\$_0$	0
3	C_0	T_1	7
4	C_1	T_2	2
5	T_0	T_3	4
6	T_1	A_0	6
7	T_2	A_1	1
8	T_3	C_1	3

`Read: TTATC`

A suffix could indicate us where is it in the original sequence. Uses a lot of space if we have millions of positions

Using BWT to map

Row		BWT	Suffix array
0	$\$_0$	C_0	8
1	A_0	T_0	5
2	A_1	$\$_0$	0
3	C_0	T_1	7
4	C_1	T_2	2
5	T_0	T_3	4
6	T_1	A_0	6
7	T_2	A_1	1
8	T_3	C_1	3

Read: TTATC

$A_1 T_2 C_1 T_3 T_0 A_0 T_1 C_0 \$_0$

A suffix could indicate us where is it in the original sequence. Uses a lot of space if we have millions of positions

Full-text Minute-size (FM) index

Row		BWT	Checkpoints
0	$\$_0$	C_0	
1	A_0	T_0	
2	A_1	$\$_0$	
3	C_0	T_1	[A:0,T:1,C:1,G:0]
4	C_1	T_2	
5	T_0	T_3	
6	T_1	A_0	
7	T_2	A_1	
8	T_3	C_1	[A:2,T:4,C:1,G:0]

What we do is use "checkpoints" length of the BWT to indicate us the position. When we found a match, we look for the “checkpoint" closest to identify Your position in the reference (genome or transcriptome).

This is known as
FM index and it is very small.

Full-text Minute-size (FM) index

Row		BWT	Checkpoints
0	$\$_0$	C_0	
1	A_0	T_0	
2	A_1	$\$_0$	
3	C_0	T_1	[A:0,T:1,C:1,G:0]
4	C_1	T_2	
5	T_0	T_3	
6	T_1	A_0	
7	T_2	A_1	
8	T_3	C_1	[A:2,T:4,C:1,G:0]

Read: TTATC

What we do is use "checkpoints" length of the BWT to indicate us the position. When we found a match, we look for the “checkpoint" closest to identify Your position in the reference (genome or transcriptome).

This is known as FM index and it is very small.

Errors or Mismatches

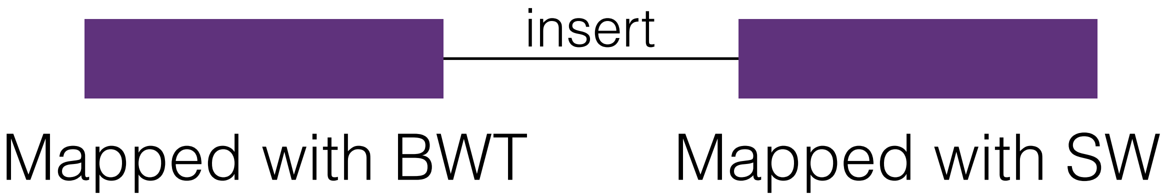
- If no perfect alignment of the reading to the reference sequence is identified, the partial alignments are taken and the candidate nucleus is changed to mismatch (A, T, C, G) and the aim is to continue extending the site with similarity to the reading of interest.

- This is known as "backtracking" and is generally limited to an arbitrary number of cycles to avoid increasing the alignment time too much.

- More backtracking is done in nucleotides with low quality.

- Since the calculation time is linear, it is not so slow to try to do this to find the place of origin of readings with errors.

Paired-end reads



- Many times a single read is found using alignment via BWT. Since we know the approximate size of the insert some algorithms use Smith-Waterman (SW) alignments to find their pair in the neighbouring region.

Programs to align **reads** to a reference

- bwa - (http://bio-bwa.sourceforge.net/)
- bowtie (http://bowtie-bio.sourceforge.net/index.shtml)
- STAR (https://github.com/alexdobin/STAR) - Recommended for RNA-Seq data

Practical - Aligning reads using BWA

https://liz-fernandez.github.io/MxBiobank_NGS/02-mapping.html