Mansoura University
Faculty of Computers and Information
Department of Computer Science
Second Semester: 2020-2021

[MED-145] Genomics: Genome Indexing IIII

Burrows Wheeler Transform & FM-index

Grade: Third Year (Medical Informatics Program)

Sara El-Metwally, Ph.D.

Faculty of Computers and Information,

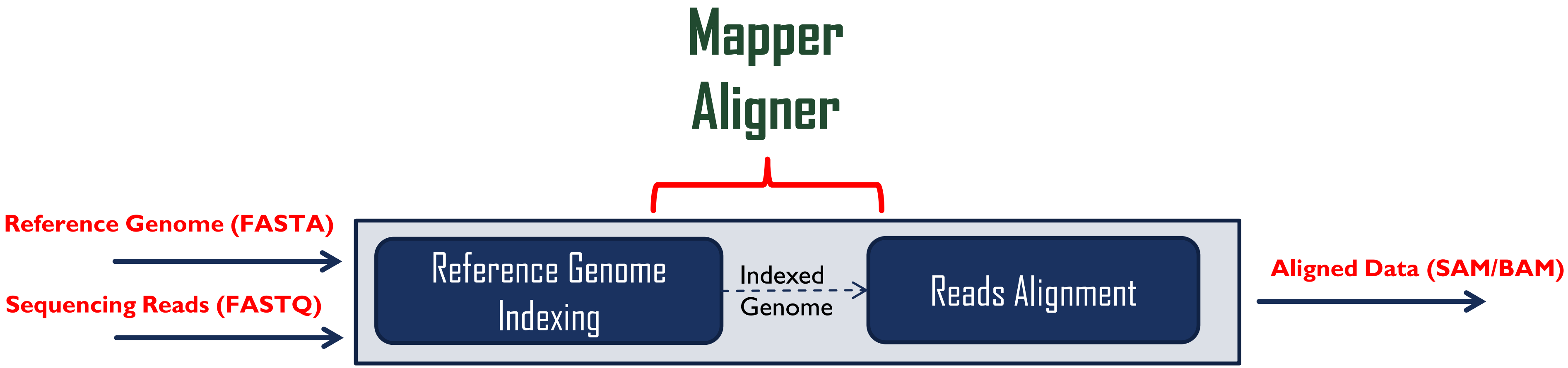
Mansoura University,

Egypt.

AGENDA

- Rev. of Suffix Array & BWT
- Relation of Suffix Array to BWT
- How to use BWT to find the pattern in Genome
- What is FM-index
- How to use FM-index to resolve the query problem over BWT.

TYPICAL MAPPING/ALIGNMENT WORKFLOW



Mapper

Aligner

Reference Genome (FASTA)

Sequencing Reads (FASTQ)

Reference Genome Indexing

Indexed Genome

Reads Alignment

Aligned Data (SAM/BAM)

Genome Indexing and Mapping Approaches

hash-based

Burrows-Wheeler

REV. (SUFFIX ARRAY)

Example:

Construct Suffix Array of GATGCGAGAGATG?

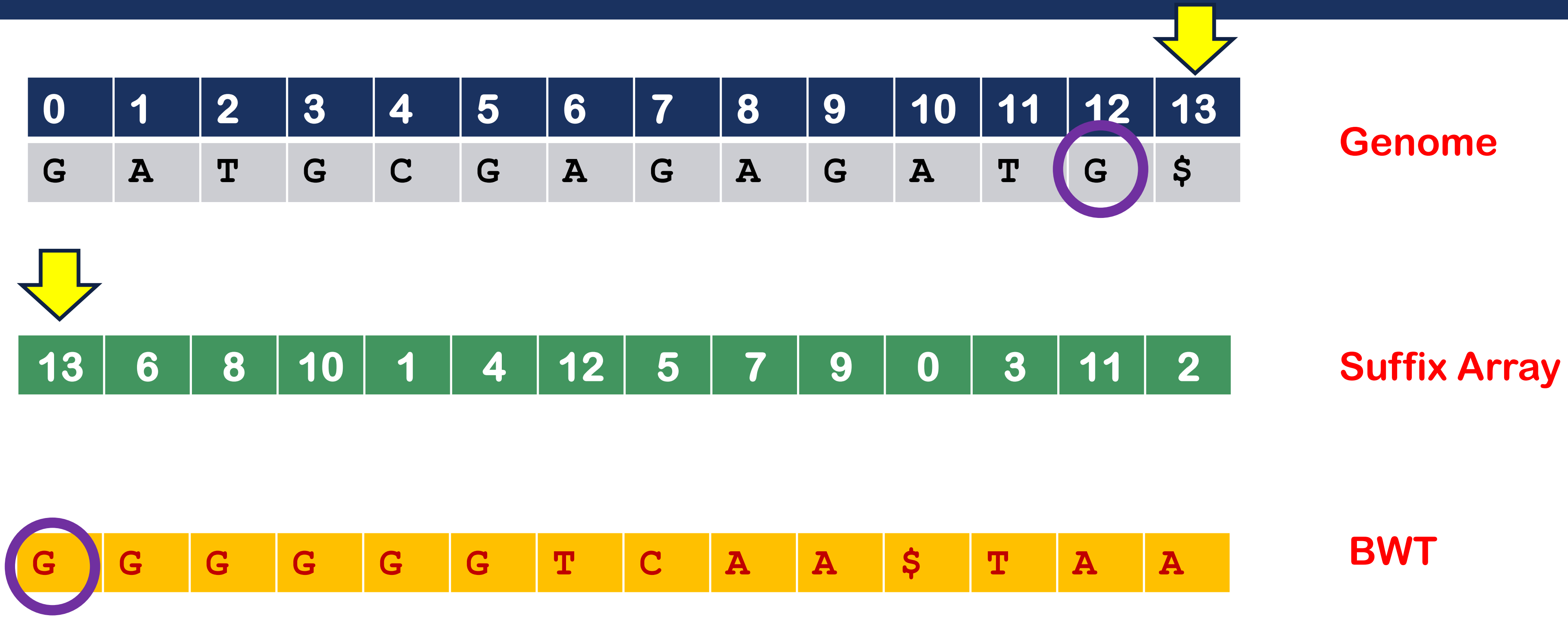
13	6	8	10	1	4	12	5	7	9	0	3	11	2
$	A	A	A	A	C	G	G	G	G	G	G	T	T
	G	G	T	T	G	$	A	A	A	A	C	G	G
	A	A	G	G	A		G	G	T	T	G	$	C
	G	T	$	C	G		A	A	G	G	A		G
	A	G		G	A		G	T	$	C	G		A
	T	$		A	G		A	G		G	A		G
	G			G	A		T	$		A	G		A
	$			A	T		G			G	A		G
				G	G		$			A	T		A
				A	$					G	G		T
				T						A	$		G
				G						T			$
				$						G			
										$			

REV. (BWT)

Example:
Construct BWT of GATGCGAGAGATG?

BWT(GATGCGAGAGATG$) = GGGGGGTCAA$TAA

NOTES Genome (FASTA): GATGCGAGAGATG



Genome

0	1	2	3	4	5	6	7	8	9	10	11	12	13
G	A	T	G	C	G	A	G	A	G	A	T	G	$

Suffix Array

13	6	8	10	1	4	12	5	7	9	0	3	11	2

BWT

G	G	G	G	G	G	T	C	A	A	$	T	A	A

$$\mathrm{BWT}(T) = \begin{cases} T[SA[i]-1] & \text{if} \quad SA[i] > 0 \\ \$ & \text{if} \quad SA[i] = 0 \end{cases}$$

i	SA[i]	SA[i] -1	T[SA[i]-1]
0	13	12	T[12]=G

NOTES Genome (FASTA): GATGCGAGAGATG

Genome

0	1	2	3	4	5	6	7	8	9	10	11	12	13
G	A	T	G	C	G	A	G	A	G	A	T	G	$

Suffix Array

13	6	8	10	1	4	12	5	7	9	0	3	11	2

BWT

G	G	G	G	G	G	T	C	A	A	$	T	A	A

$$\mathrm{BWT}(T) = \begin{cases} T[SA[i]-1] & \text{if} \quad SA[i] > 0 \\ \$ & \text{if} \quad SA[i] = 0 \end{cases}$$

i	SA[i]	SA[i] -1	T[SA[i]-1]
1	6	5	T[5]=G

NOTES Genome (FASTA): GATGCGAGAGATG

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	G	A	T	G	C	G	A	G	A	G	A	T	G	$

Genome

13	6	8	10	1	4	12	5	7	9	0	3	11	2

Suffix Array

G	G	G	G	G	G	T	C	A	A	$	T	A	A

BWT

$$\mathrm{BWT}(T) = \begin{cases} T[SA[i]-1] & \text{if} \quad SA[i] > 0 \\ \$ & \text{if} \quad SA[i] = 0 \end{cases}$$

i	SA[i]	SA[i] -1	T[SA[i]-1]
10	0	----	$

before

NOTES

Burrows Wheeler Matrix.

0	1	2	3	4	5	6	7	8	9	10	11	12	13
G	A	T	G	C	G	A	G	A	G	A	T	G	$

$GATGCGAGAGATG
AGAGATG$GATGCG
AGATG$GATGCGAG
ATG$GATGCGAGAG
ATGCGAGAGATG$G
CGAGAGATG$GATG
G$GATGCGAGAGAT
GAGAGATG$GATGC
GAGATG$GATGCGA
GATG$GATGCGAGA
GATGCGAGAGATG$
GCGAGAGATG$GAT
TG$GATGCGAGAGA
TGCGAGAGATG$GA

Sorted Suffixes

13	6	8	10	1	4	12	5	7	9	0	3	11	2
$	A	A	A	A	C	G	G	G	G	G	G	T	T
	G	G	T	T	G	$	A	A	A	A	C	G	G
	A	A	G	G	A		G	G	T	T	G	$	C
	G	T	$	C	G		A	A	G	G	A		G
	A	G		G	A		G	T	$	C	G		A
	T	$		A	G		A	G		G	A		G
	G			G	A		T	$		A	G		A
	$			A	T		G			G	A		G
				G	G		$			A	T		A
				A	$					G	G		T
				T						A	$		G
				G						T			$
				$						G			
										$			

NOTES

Burrows Wheeler Matrix.

$\texttt{\$GATGCGAGAGATG}_0$

$\texttt{A}_0\texttt{GAGATG\$GATGCG}_1$

$\texttt{A}_1\texttt{GATG\$GATGCGAG}_2$

$\texttt{A}_2\texttt{TG\$GATGCGAGAG}_3$

$\texttt{A}_3\texttt{TGCGAGAGATG\$G}_4$

$\texttt{C}_0\texttt{GAGAGATG\$GATG}_5$

$\texttt{G}_0\texttt{\$GATGCGAGAGAT}_0$

$\texttt{G}_1\texttt{AGAGATG\$GATGC}_0$

$\texttt{G}_2\texttt{AGATG\$GATGCGA}_0$

$\texttt{G}_3\texttt{ATG\$GATGCGAGA}_1$

$\texttt{G}_4\texttt{ATGCGAGAGATG\$}$

$\texttt{G}_5\texttt{CGAGAGATG\$GAT}_1$

$\texttt{T}_0\texttt{G\$GATGCGAGAGA}_2$

$\texttt{T}_1\texttt{GCGAGAGATG\$GA}_3$

NOTES

Burrows Wheeler Matrix.

$\mathtt{\$GATGCGAGAGATG_0}$
$\mathtt{A_0GAGATG\$GATGCG_1}$
$\mathtt{A_1GATG\$GATGCGAG_2}$
$\mathtt{A_2TG\$GATGCGAGAG_3}$
$\mathtt{A_3TGCGAGAGATG\$G_4}$
$\mathtt{C_0GAGAGATG\$GATG_5}$
$\mathtt{G_0\$GATGCGAGAGAT_0}$
$\mathtt{G_1AGAGATG\$GATGC_0}$
$\mathtt{G_2AGATG\$GATGCGA_0}$
$\mathtt{G_3ATG\$GATGCGAGA_1}$
$\mathtt{G_4ATGCGAGAGATG\$}$
$\mathtt{G_5CGAGAGATG\$GAT_1}$
$\mathtt{T_0G\$GATGCGAGAGA_2}$
$\mathtt{T_1GCGAGAGATG\$GA_3}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

GAGA

NOTES

Burrows Wheeler Matrix.

$\text{\$GATGCGAGAGAT}G_0$

$A_0\text{GAGATG\$GATGC}G_1$

$A_1\text{GATG\$GATGCGA}G_2$

$A_2\text{TG\$GATGCGAGA}G_3$

$A_3\text{TGCGAGAGATG\$}G_4$

$C_0\text{GAGAGATG\$GAT}G_5$

$G_0\text{\$GATGCGAGAGA}T_0$

$G_1\text{AGAGATG\$GATG}C_0$

$G_2\text{AGATG\$GATGCG}A_0$

$G_3\text{ATG\$GATGCGAG}A_1$

$G_4\text{ATGCGAGAGATG\$}$

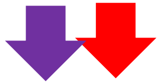
$G_5\text{CGAGAGATG\$GA}T_1$

$T_0\text{G\$GATGCGAGAG}A_2$

$T_1\text{GCGAGAGATG\$G}A_3$

How can we use BWT of the human genome to solve the query problem?

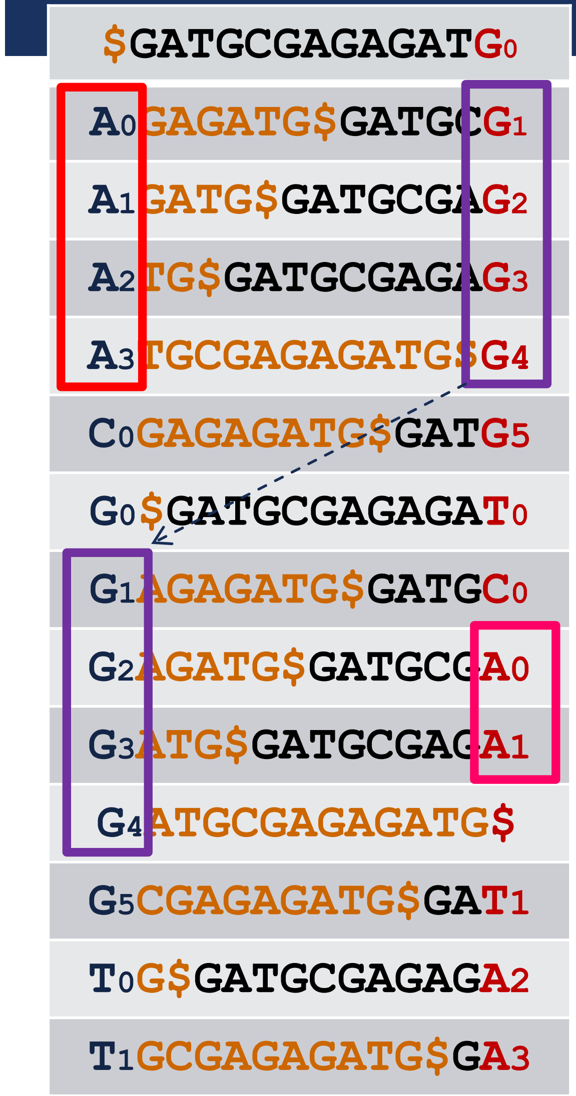
Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .



GAGA

NOTES

Burrows Wheeler Matrix.



$\$\mathrm{GATGCGAGAGAT}G_0$

$A_0\mathrm{GAGATG\$GATGC}G_1$

$A_1\mathrm{GATG\$GATGCGA}G_2$

$A_2\mathrm{TG\$GATGCGAGA}G_3$

$A_3\mathrm{TGCGAGAGATG\$}G_4$

$C_0\mathrm{GAGAGATG\$GAT}G_5$

$G_0\mathrm{\$GATGCGAGAGA}T_0$

$G_1\mathrm{AGAGATG\$GATG}C_0$

$G_2\mathrm{AGATG\$GATGCG}A_0$

$G_3\mathrm{ATG\$GATGCGAG}A_1$

$G_4\mathrm{ATGCGAGAGATG}\$$

$G_5\mathrm{CGAGAGATG\$GA}T_1$

$T_0\mathrm{G\$GATGCGAGAG}A_2$

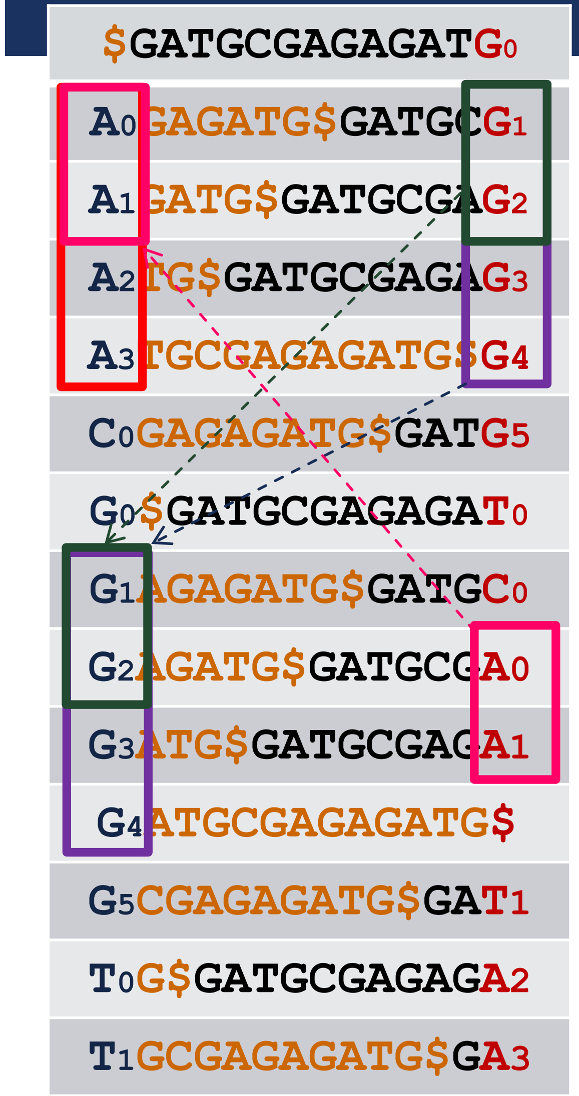
$T_1\mathrm{GCGAGAGATG\$G}A_3$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

NOTES

Burrows Wheeler Matrix.



$\mathrm{\$GATGCGAGAGATG_0}$

$\mathrm{A_0GAGATG\$GATGCG_1}$

$\mathrm{A_1GATG\$GATGCGAG_2}$

$\mathrm{A_2TG\$GATGCGAGAG_3}$

$\mathrm{A_3TGCGAGAGATG\$G_4}$

$\mathrm{C_0GAGAGATG\$GATG_5}$

$\mathrm{G_0\$GATGCGAGAGAT_0}$

$\mathrm{G_1AGAGATG\$GATGC_0}$

$\mathrm{G_2AGATG\$GATGCGA_0}$

$\mathrm{G_3ATG\$GATGCGAGA_1}$

$\mathrm{G_4ATGCGAGAGATG\$}$

$\mathrm{G_5CGAGAGATG\$GAT_1}$

$\mathrm{T_0G\$GATGCGAGAGA_2}$

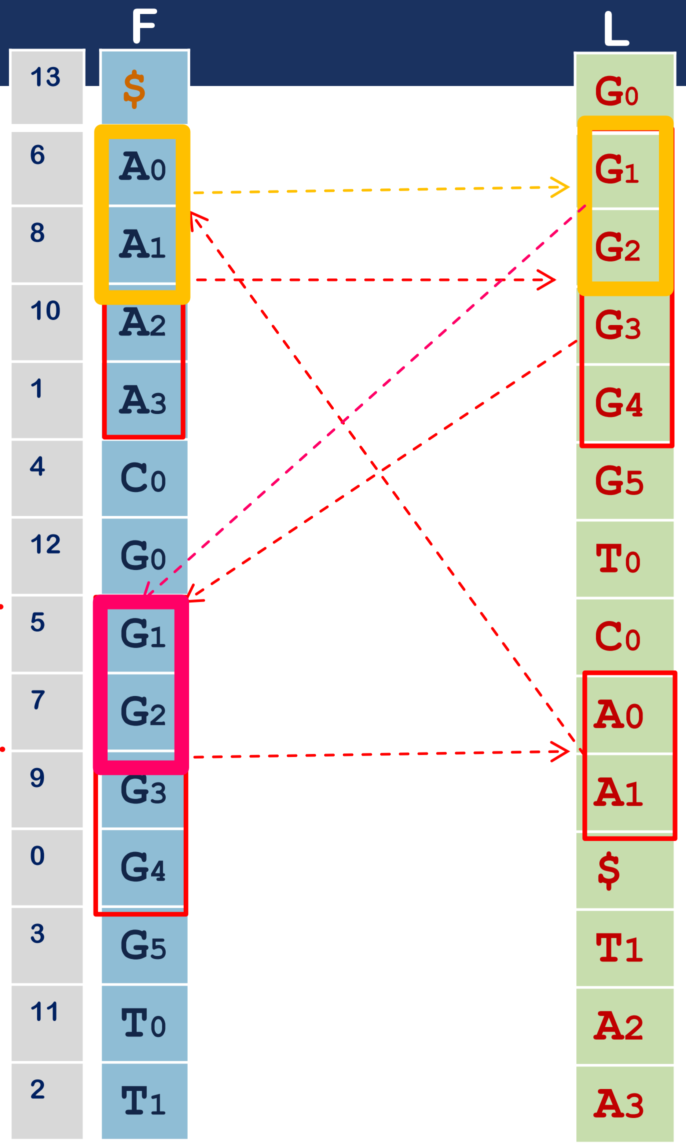
$\mathrm{T_1GCGAGAGATG\$GA_3}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .



	F	L
13	\$	G_0
6	A_0	G_1
8	A_1	G_2
10	A_2	G_3
1	A_3	G_4
4	C_0	G_5
12	G_0	T_0
5	G_1	C_0
7	G_2	A_0
9	G_3	A_1
0	G_4	\$
3	G_5	T_1
11	T_0	A_2
2	T_1	A_3

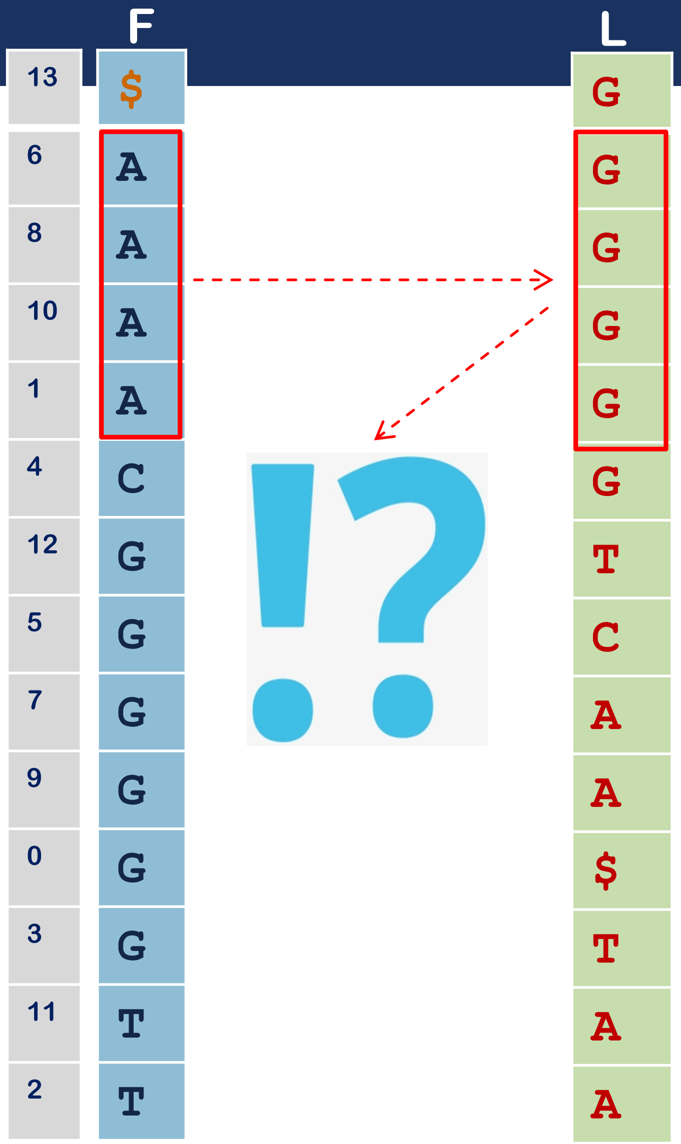
Pattern found at positions 5, 7

GAGA

0	1	2	3	4	5	6	7	8	9	10	11	12	13
G	A	T	G	C	G	A	G	A	G	A	T	G	$

How can we use BWT of the human genome to solve the query problem?

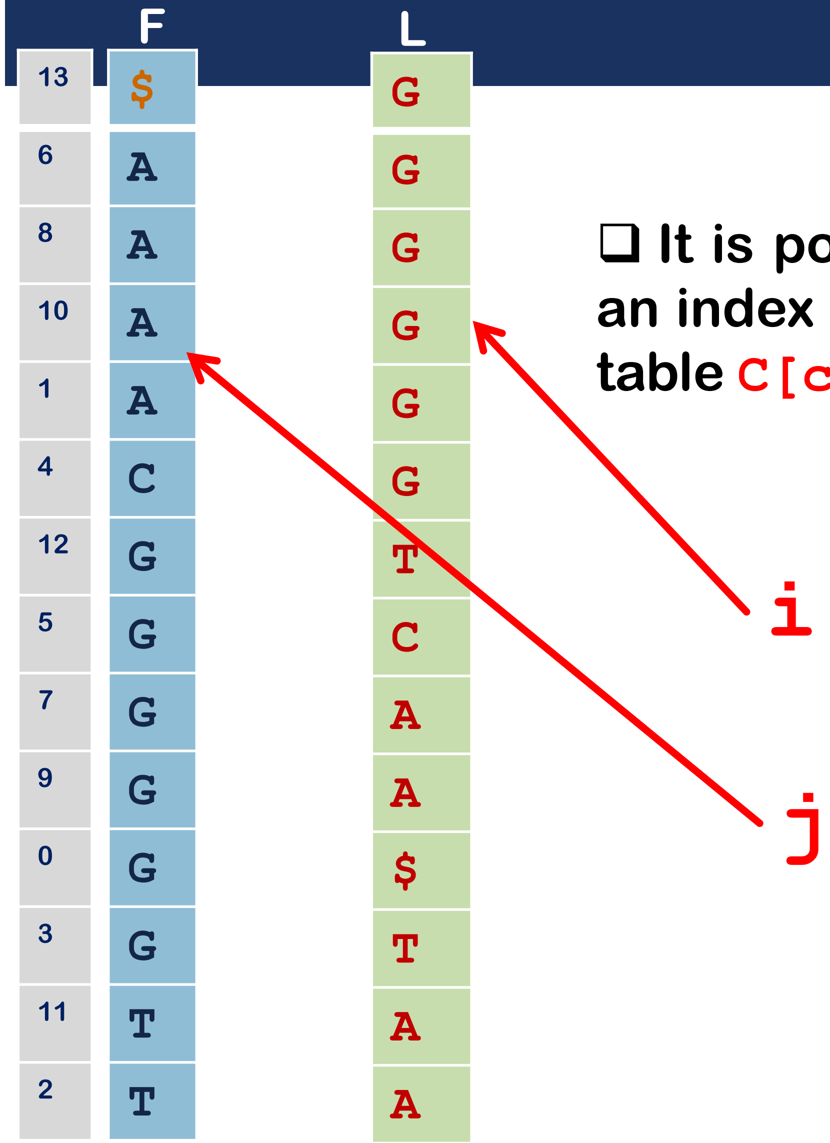
Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .



	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .



	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ It is possible to make a last-to-first column mapping `LF(i)` from an index `i` to an index `j`, such that `F[j] = L[i]`, with the help of a table `C[c]` and a function `Occ(c, k)`.

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table `C[c]` that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

✓Step 1: Determine the text alphabet Σ.

$$\Sigma=\{\$,A,C,G,T\}$$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table `C[c]` that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]					

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table `C[c]` that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]	0				

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table C[c] that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]	0				

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table `C[c]` that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]	0	1			

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table C[c] that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]	0	1			

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ Create a table `C[c]` that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]	0	1	5		

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

□ Create a table `C[c]` that, for each character c in the alphabet, contains the number of occurrences of lexically smaller characters in the text.

$$\Sigma=\{\$,A,C,G,T\}$$

✓Step 2: Create a table that has a column for each letter in Σ.

c	$	A	C	G	T
C[c]	0	1	5	6	12

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function Occ(c, k) is the number of occurrences of character c in the prefix L[0..k] .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13

$\Sigma = \{\$, A, C, G, T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$														
A														
C														
G														
T														

$$\Sigma=\{\$,A,C,G,T\}$$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function Occ(c, k) is the number of occurrences of character c in the prefix L[0..k] .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0													
A														
C														
G														
T														

$\Sigma=\{\$,A,C,G,T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A														
C														
G														
T														

$\Sigma=\{\$,A,C,G,T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function Occ(c, k) is the number of occurrences of character c in the prefix L[0..k] .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0													
C														
G														
T														

$\Sigma=\{\$,A,C,G,T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C														
G														
T														

$\Sigma = \{\$, A, C, G, T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0													
G														
T														

$\Sigma=\{\$,A,C,G,T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G														
T														

$$\Sigma=\{\$,A,C,G,T\}$$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1													
T														

$\Sigma=\{\$,A,C,G,T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T														

$\Sigma=\{\$,A,C,G,T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0													

$\Sigma = \{\$, A, C, G, T\}$

How can we use BWT of the human genome to solve the query problem?

Suppose a FASTA file has a sequence `GATGCGAGAGATG` and the query sequence `GAGA` .

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The function `Occ(c, k)` is the number of occurrences of character c in the prefix `L[0..k]` .

✓Step 1: Create a table that has the following attributes.

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

$$\Sigma=\{\$,A,C,G,T\}$$

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

- ❑ The last-to-first mapping can now be defined as:

LF(i) = (C[L[i]] + Occ(L[i], i))-1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

- The last-to-first mapping can now be defined as:

  `LF(i) = C[L[i]] + Occ(L[i], i) -1`

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0				

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

- ❑ The last-to-first mapping can now be defined as:
  LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G			

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ The last-to-first mapping can now be defined as:

LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6		

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ **The last-to-first mapping can now be defined as:**

LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6	1	

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	1	1	1	1	1	1	2	2	2

Occ(c, k)

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

- ❑ The last-to-first mapping can now be defined as:

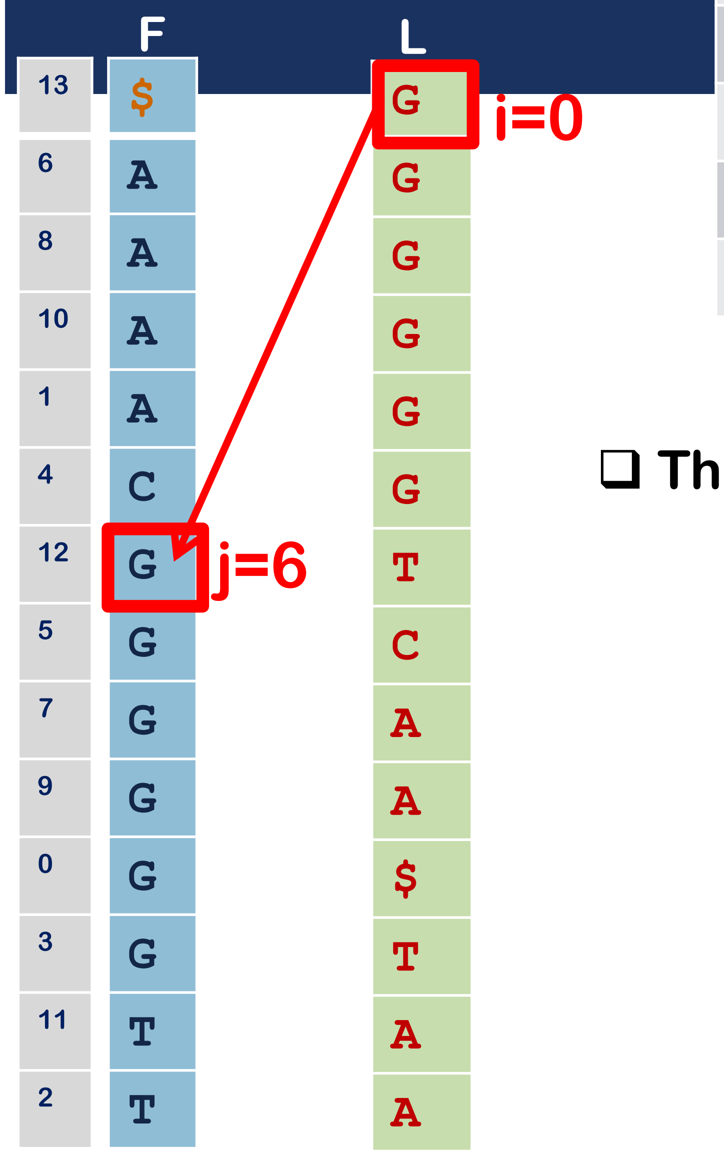
LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6	1	7

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)



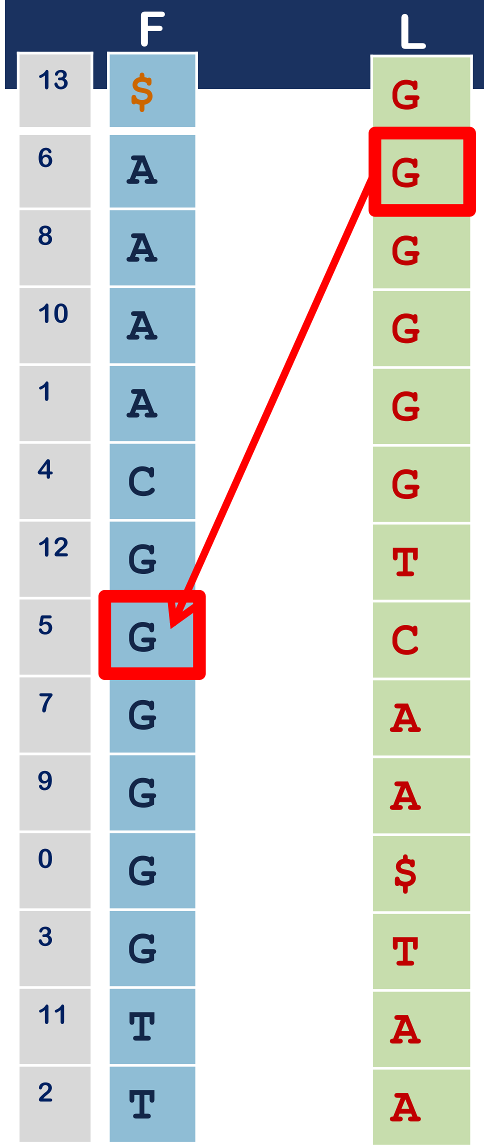
	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

- ❑ The last-to-first mapping can now be defined as:

LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6	1	7-1=6

c	$	A	C	G	T
C[c]	0	1	5	6	12



	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ The last-to-first mapping can now be defined as:

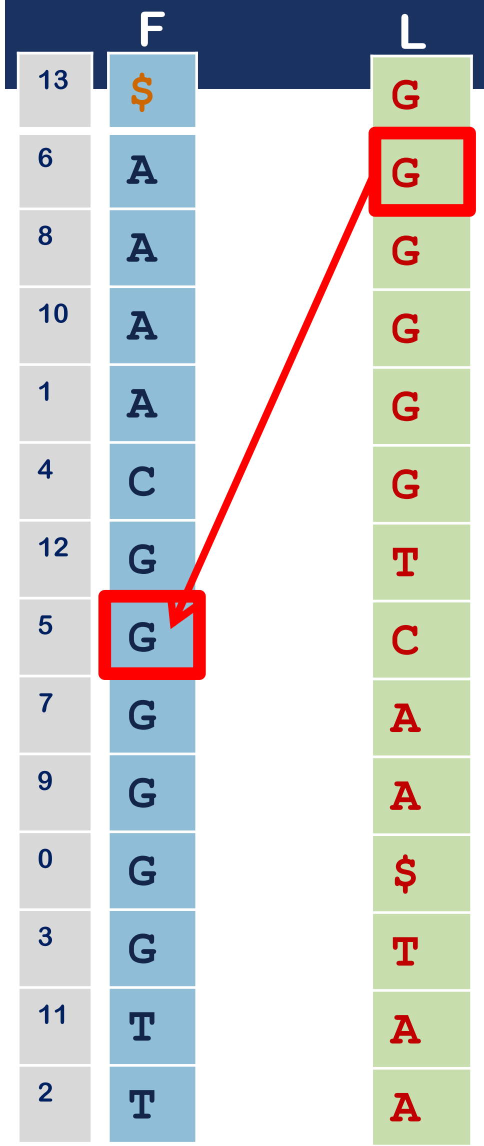
LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6	1	7-1=6
1	G	6	2	8-1=7

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)



	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ The last-to-first mapping can now be defined as:

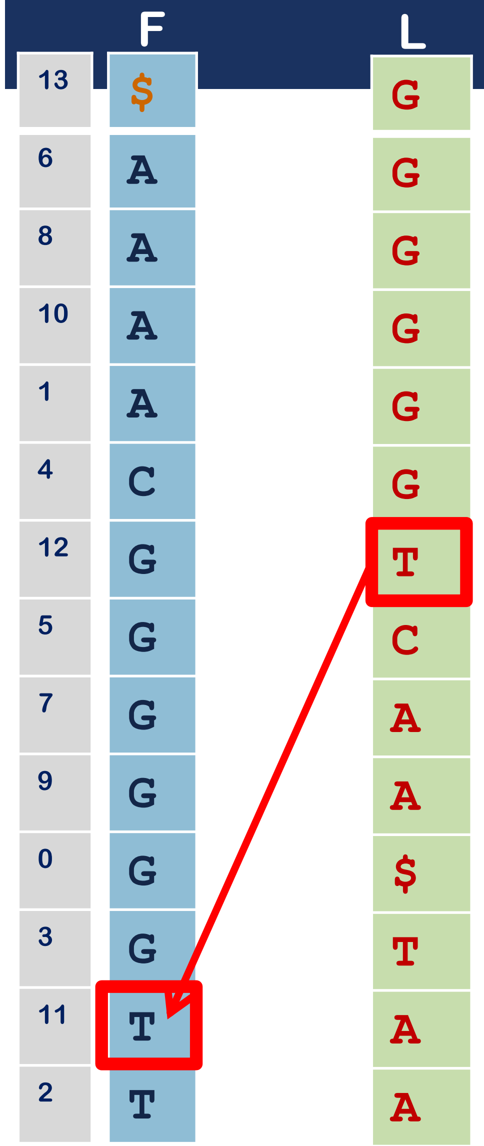
LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6	1	7-1=6
1	G	6	2	8-1=7

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)



	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

- ❑ The last-to-first mapping can now be defined as:

LF(i) = C[L[i]] + Occ(L[i], i) -1

i	L[i]	C[L[i]]	Occ(L[i], i)	LF(i)
0	G	6	1	7-1=6
1	G	6	2	8-1=7
6	T	12	1	13-1=12

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

FM-index

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ FM-index is a compressed full-text substring index based on the Burrows-Wheeler transform, with some similarities to the suffix array.

❑ It was created by Paolo Ferragina and Giovanni Manzini, who describe it as an opportunistic data structure as it allows compression of the input text while still permitting fast substring queries.

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

FM-index

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

- ❑ It can be used to efficiently find the number of occurrences of a pattern within the compressed text, as well as locate the position of each occurrence.

- ❑ The query time, as well as the required storage space, has a sublinear complexity with respect to the size of the input data.

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ **Count**: The operation count takes a pattern P[1..p] and returns the number of occurrences of that pattern in the original text T.

GAGA

The initial range is set to [C[A]..C[A+1]-1]

C[A]	C[A+1]-1	[..]

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ **Count**: The operation count takes a pattern P[1..p] and returns the number of occurrences of that pattern in the original text T.

GAGA

The initial range is set to [C[A]..C[A+1]-1]

C[A]	C[A+1]-1	[..]
1	5-1=4	[1..4]

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

```
$GATGCGAGAGATG
AGAGATG$GATGCG
AGATG$GATGCGAG
ATG$GATGCGAGAG
ATGCGAGAGATG$G
CGAGAGATG$GATG
G$GATGCGAGAGAT
GAGAGATG$GATGC
GAGATG$GATGCGA
GATG$GATGCGAGA
GATGCGAGAGATG$
GCGAGAGATG$GAT
TG$GATGCGAGAGA
TGCGAGAGATG$GA
```

GAGA

The initial range is set to [C[A]..C[A+1]-1]

C[A]	C[A+1]-1	[..]
1	5-1=4	[1..4]

Range of sorted suffixes that started with letter A in BWT Matrix (i.e. F column)

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ Count: The operation count takes a pattern P[1..p] and returns the number of occurrences of that pattern in the original text T.

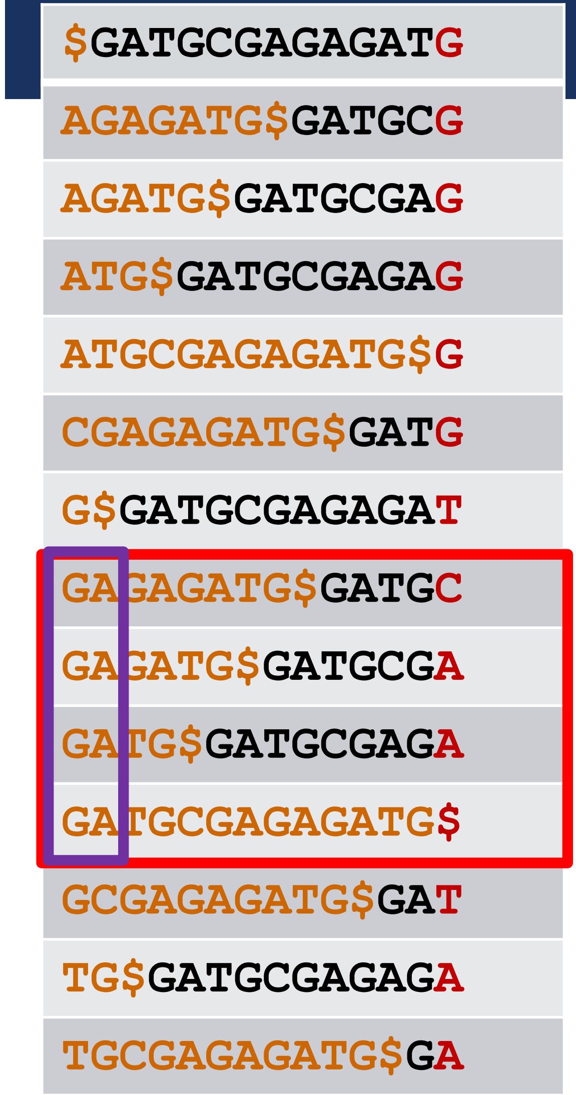
GAGA

The new range is [C[G] + Occ(G, start-1)..C[G] + Occ(G, end)-1]

[1..4]

C[G]	Occ(G, start -1)	Occ(G, end) -1	[..]
6	Occ(G,0)=1	Occ(G,4)=5-1=4	[7..10]

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .



$GATGCGAGAGATG
AGAGATG$GATGCG
AGATG$GATGCGAG
ATG$GATGCGAGAG
ATGCGAGAGATG$G
CGAGAGATG$GATG
G$GATGCGAGAGAT
GAGAGATG$GATGC
GAGATG$GATGCGA
GATG$GATGCGAGA
GATGCGAGAGATG$
GCGAGAGATG$GAT
TG$GATGCGAGAGA
TGCGAGAGATG$GA

GAGA

The new range is [C[G] + Occ(G, start-1)..C[G] + Occ(G, end)-1]

[1..4]

C[G]	Occ(G, start -1)	Occ(G, end) -1	[..]
6	Occ(G,0)=1	Occ(G,4)=5-1=4	[7..10]

Range of sorted suffixes that started with letters GA in BWT Matrix (i.e. F column)

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

- ❑ **Count**: The operation count takes a pattern P[1..p] and returns the number of occurrences of that pattern in the original text T.

GAGA

The new range is [C[A] + Occ(A, start-1)..C[A] + Occ(A, end)-1]

[7..10]

C[A]	Occ(A, start -1)	Occ(A, end) -1	[..]
1	Occ(A,6)=0	Occ(A,10)=2-1=1	[1..2]

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

$GATGCGAGAGATG
AGAGATG$GATGCG
AGATG$GATGCGAG
ATG$GATGCGAGAG
ATGCGAGAGATG$G
CGAGAGATG$GATG
G$GATGCGAGAGAT
GAGAGATG$GATGC
GAGATG$GATGCGA
GATG$GATGCGAGA
GATGCGAGAGATG$
GCGAGAGATG$GAT
TG$GATGCGAGAGA
TGCGAGAGATG$GA

GAGA

The new range is [C[A] + Occ(A, start-1)..C[A] + Occ(A, end)-1]

[7..10]

C[A]	Occ(A, start -1)	Occ(A, end) -1	[..]
1	Occ(A,6)=0	Occ(A,10)=2-1=1	[1..2]

Range of sorted suffixes that started with letters AGA in BWT Matrix (i.e. F column)

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ **Count**: The operation count takes a pattern P[1..p] and returns the number of occurrences of that pattern in the original text T.

GAGA

The new range is [C[G] + Occ(G, start-1)..C[G] + Occ(G, end)-1]

[1..2]

C[G]	Occ(G, start -1)	Occ(G, end) -1	[..]
6	Occ(G,0)=1	Occ(G,2)=3-1=2	[7..8]

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

$GATGCGAGAGATG
AGAGATG$GATGCG
AGATG$GATGCGAG
ATG$GATGCGAGAG
ATGCGAGAGATG$G
CGAGAGATG$GATG
G$GATGCGAGAGAT
GAGAGATG$GATGC
GAGATG$GATGCGA
GATG$GATGCGAGA
GATGCGAGAGATG$
GCGAGAGATG$GAT
TG$GATGCGAGAGA
TGCGAGAGATG$GA

GAGA

The new range is [C[G] + Occ(G, start-1)..C[G] + Occ(G, end)-1]

[1..2]

C[G]	Occ(G, start -1)	Occ(G, end) -1	[..]
6	Occ(G,0)=1	Occ(G,2)=3-1=2	[7..8]

Range of sorted suffixes that started with letters GAGA in BWT Matrix (i.e. F column)

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

❑ **Count**: The operation count takes a pattern P[1..p] and returns the number of occurrences of that pattern in the original text T.

GAGA → [7..8]

The count is the same as the size of the range: 8 - 7 + 1 = 2.

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA .

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ Locate: Find the pattern locations through the text.

GAGA → [7..8]

Locate(7)= SA[7]= 5

Locate(8)= SA[8]= 7

Suppose a FASTA file has a sequence GATGCGAGAGATG and the query sequence GAGA.

c	$	A	C	G	T
C[c]	0	1	5	6	12

	F	L
13	$	G
6	A	G
8	A	G
10	A	G
1	A	G
4	C	G
12	G	T
5	G	C
7	G	A
9	G	A
0	G	$
3	G	T
11	T	A
2	T	A

L	G	G	G	G	G	G	T	C	A	A	$	T	A	A
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$	0	0	0	0	0	0	0	0	0	0	1	1	1	1
A	0	0	0	0	0	0	0	0	1	2	2	2	3	4
C	0	0	0	0	0	0	0	1	1	1	1	1	1	1
G	1	2	3	4	5	6	6	6	6	6	6	6	6	6
T	0	0	0	0	0	0	1	1	1	1	1	2	2	2

Occ(c, k)

❑ Locate: Find the pattern locations through the text.

GAGA Pattern Found at Positions 5 and 7 in T

0	1	2	3	4	5	6	7	8	9	10	11	12	13
G	A	T	G	C	G	A	G	A	G	A	T	G	$

Thank you!