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

[MED-145] Genomics: Genome Indexing II

Grade: Third Year (Medical Informatics Program)

Sara El-Metwally, Ph.D.

Faculty of Computers and Information,

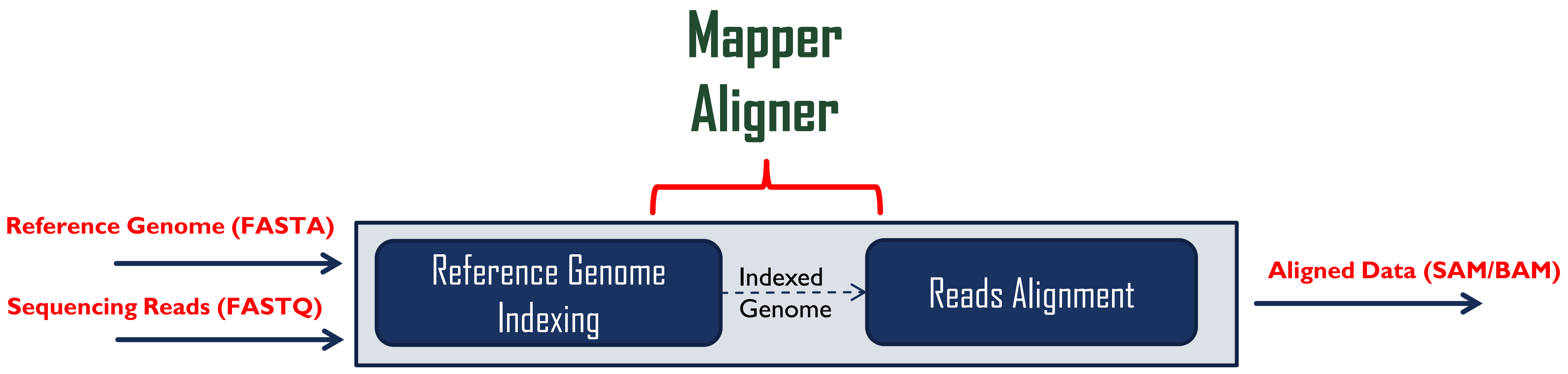
Mansoura University,

Egypt.

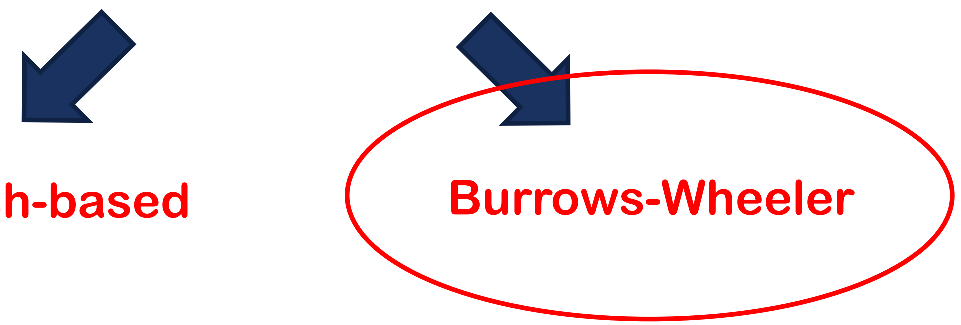
AGENDA

- Suffix Array
- How to build a Suffix array for a reference genome.
- How to use Suffix array to search for a pattern in a reference genome.
- Advantages/Disadvantages of Suffix array based approach.
- What is Burrows Wheeler Transform, BWT
- How to build a BWT for a reference genome.

TYPICAL MAPPING/ALIGNMENT WORKFLOW



Genome Indexing and Mapping Approaches



SUFFIX ARRAY

- An important step towards modern algorithms was the invention of a data structure called the suffix array of a text.
- A suffix is the end of a text from a given position.

Example:
Construct the suffix array of GATGCGAGAGATG?

Step 1: Add a terminator character $, which has lower lexical order than all other characters.

GATGCGAGAGATG$

SUFFIX ARRAY

Step 2: Generate all suffixes of the string: GATGCGAGAGATG$

```
$
G$
TG$
ATG$
GATG$
AGATG$
GAGATG$
AGAGATG$
GAGAGATG$
CGAGAGATG$
GCGAGAGATG$
TGCGAGAGATG$
ATGCGAGAGATG$
GATGCGAGAGATG$
```

SUFFIX ARRAY

Suffix	Start pos.
$	
G$	
TG$	
ATG$	
GATG$	
AGATG$	
GAGATG$	
AGAGATG$	
GAGAGATG$	
CGAGAGATG$	
GCGAGAGATG$	
TGCGAGAGATG$	
ATGCGAGAGATG$	
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	$

SUFFIX ARRAY

Suffix	Start pos.
$	13
G$	
TG$	
ATG$	
GATG$	
AGATG$	
GAGATG$	
AGAGATG$	
GAGAGATG$	
CGAGAGATG$	
GCGAGAGATG$	
TGCGAGAGATG$	
ATGCGAGAGATG$	
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	$

SUFFIX ARRAY

Suffix	Start pos.
$	13
G$	
TG$	
ATG$	
GATG$	
AGATG$	
GAGATG$	
AGAGATG$	
GAGAGATG$	
CGAGAGATG$	
GCGAGAGATG$	
TGCGAGAGATG$	
ATGCGAGAGATG$	
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	$

SUFFIX ARRAY

Suffix	Start pos.
$	13
G$	12
TG$	
ATG$	
GATG$	
AGATG$	
GAGATG$	
AGAGATG$	
GAGAGATG$	
CGAGAGATG$	
GCGAGAGATG$	
TGCGAGAGATG$	
ATGCGAGAGATG$	
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	$

SUFFIX ARRAY

Suffix	Start pos.
$	13
G$	12
TG$	11
ATG$	10
GATG$	9
AGATG$	8
GAGATG$	7
AGAGATG$	6
GAGAGATG$	5
CGAGAGATG$	4
GCGAGAGATG$	3
TGCGAGAGATG$	2
ATGCGAGAGATG$	1
GATGCGAGAGATG$	0

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

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
$
G$
TG$
ATG$
GATG$
AGATG$
GAGATG$
AGAGATG$
GAGAGATG$
CGAGAGATG$
GCGAGAGATG$
TGCGAGAGATG$
ATGCGAGAGATG$
GATGCGAGAGATG$

Step 3: Sort all suffixes in the lexicographical order (i.e. dictionary order).

Note: $ has a lower lexical order than all other characters.

SUFFIX ARRAY

$	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			
										$			

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	$

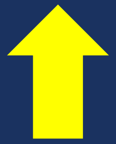
SUFFIX ARRAY

Suffix
$
G$
TG$
ATG$
GATG$
AGATG$
GAGATG$
AGAGATG$
GAGAGATG$
CGAGAGATG$
GCGAGAGATG$
TGCGAGAGATG$
ATGCGAGAGATG$
GATGCGAGAGATG$

Sorted Suffix es
$
AGAGATG$
AGATG$
ATG$
ATGCGAGAGATG$
CGAGAGATG$
G$
GAGAGATG$
GAGATG$
GATG$
GATGCGAGAGATG$
GCGAGAGATG$
TG$
TGCGAGAGATG$

SUFFIX ARRAY

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	$



Sorted Suffix es	Started Pos.
$	13
AGAGATG$	
AGATG$	
ATG$	
ATGCGAGAGATG$	
CGAGAGATG$	
G$	
GAGAGATG$	
GAGATG$	
GATG$	
GATGCGAGAGATG$	
GCGAGAGATG$	
TG$	
TGCGAGAGATG$	

SUFFIX ARRAY

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	$

Sorted Suffix es	Started Pos.
$	13
AGAGATG$	6
AGATG$	
ATG$	
ATGCGAGAGATG$	
CGAGAGATG$	
G$	
GAGAGATG$	
GAGATG$	
GATG$	
GATGCGAGAGATG$	
GCGAGAGATG$	
TG$	
TGCGAGAGATG$	

SUFFIX ARRAY

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	$

Sorted Suffix es	Started Pos.
$	13
AGAGATG$	6
AGATG$	8
ATG$	10
ATGCGAGAGATG$	1
CGAGAGATG$	4
G$	12
GAGAGATG$	5
GAGATG$	7
GATG$	9
GATGCGAGAGATG$	0
GCGAGAGATG$	3
TG$	11
TGCGAGAGATG$	2

SUFFIX ARRAY

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	$

Sorted Suffix es	Started Pos.
$	13
AGAGATG$	6
AGATG$	8
ATG$	10
ATGCGAGAGATG$	1
CGAGAGATG$	4
G$	12
GAGAGATG$	5
GAGATG$	7
GATG$	9
GATGCGAGAGATG$	0
GCGAGAGATG$	3
TG$	11
TGCGAGAGATG$	2

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

Suffix Array

SUFFIX ARRAY

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			
										$			

SUFFIX ARRAY

- How can we use the suffix array of the human genome to solve the query problem?

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

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

Suffix Array

❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

- How can we use the suffix array of the human genome to solve the query problem?

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

`Extract the suffix that starts at the position 12 from FASTA file`

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

Suffix Array

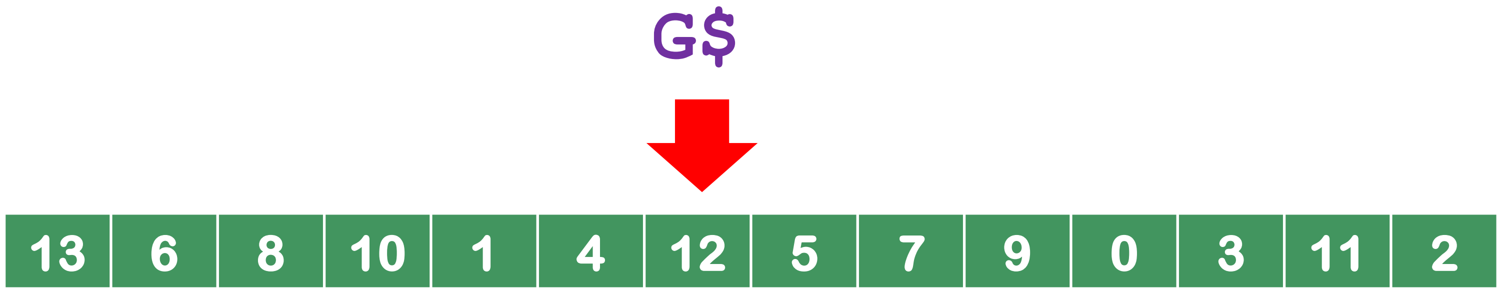
❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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



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

Suffix Array

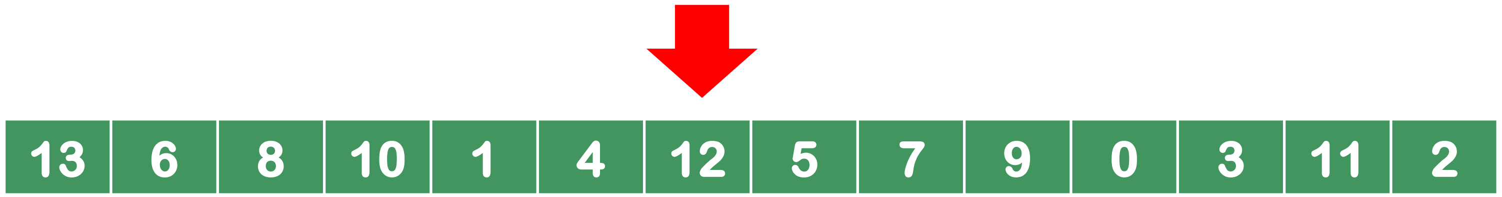
❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

- **How can we use the suffix array of the human genome to solve the query problem?**

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

`Compare the extracted suffix with the query sequence`



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

Suffix Array

❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

- How can we use the suffix array of the human genome to solve the query problem?

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

$$\texttt{G\$} \quad \& \quad \texttt{GAGA} \longrightarrow \texttt{G\$} < \texttt{GAGA}$$

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

Suffix Array

❖ Since the suffixes are sorted, we can proceed by bisection.

❖ We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

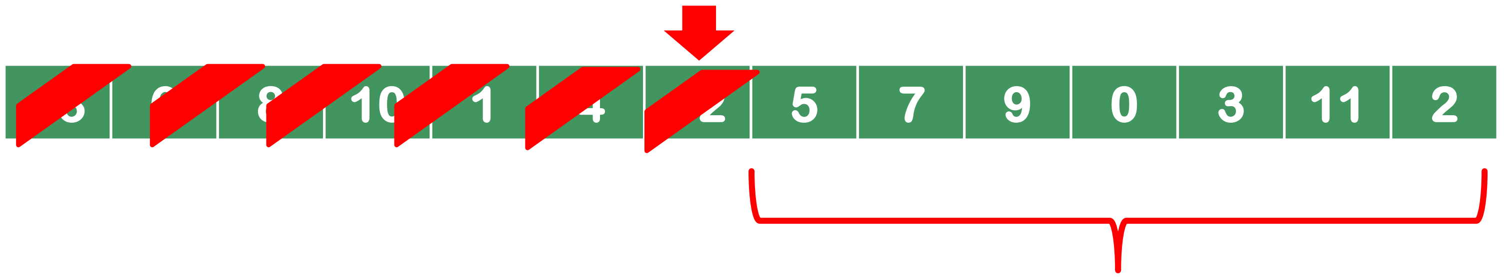
SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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

G$ < GAGA



Suffix Array

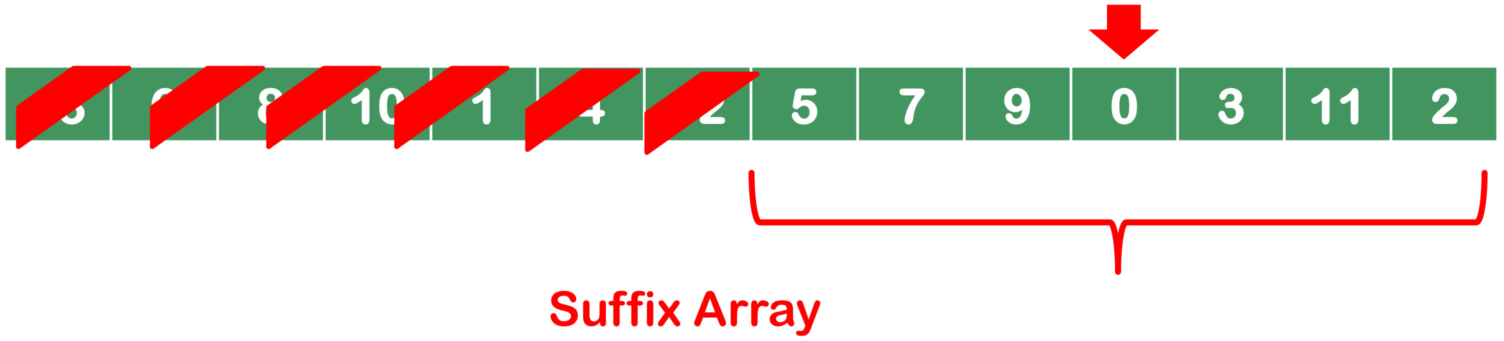
❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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



❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

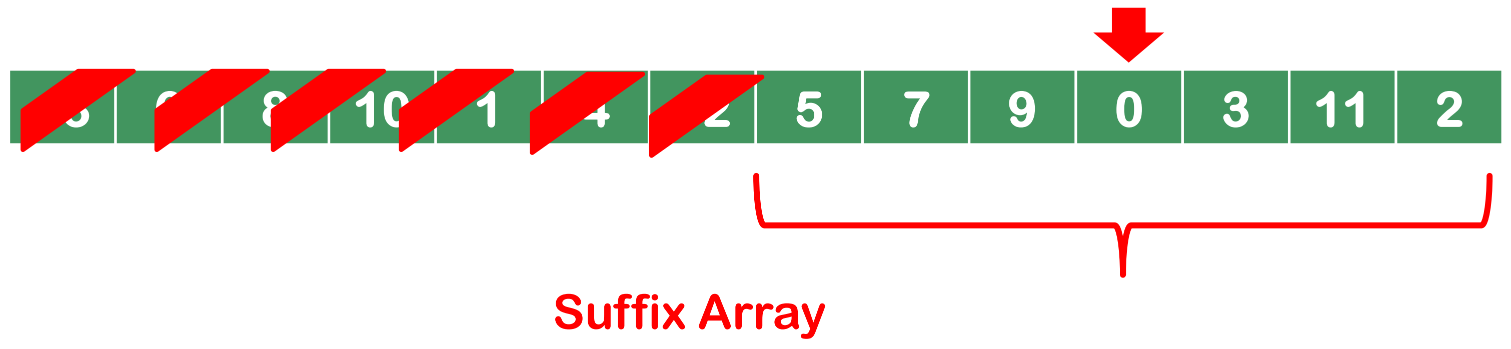
SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?**

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

GATGCGAGAGATG$ > GAGA



❖ Since the suffixes are sorted, we can proceed by bisection.

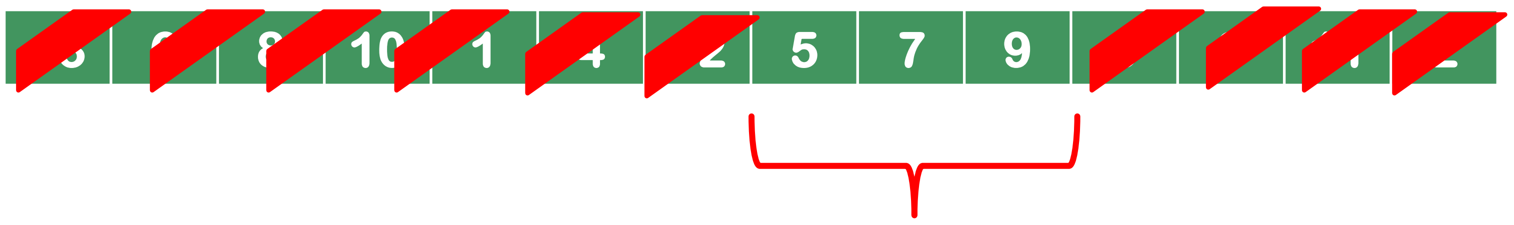
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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



Suffix Array

❖ Since the suffixes are sorted, we can proceed by bisection.

❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

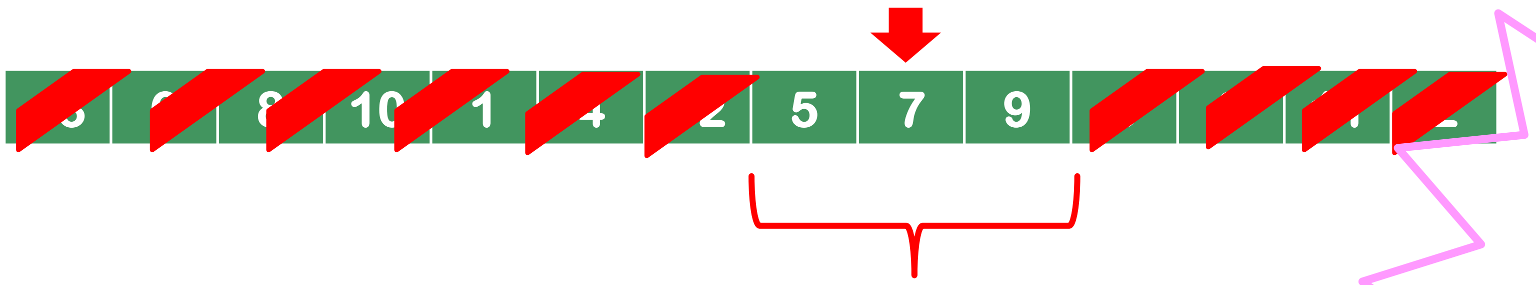
SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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

GAGATG$ > GAGA



Suffix Array

Query located at position 7

❖ Since the suffixes are sorted, we can proceed by bisection.

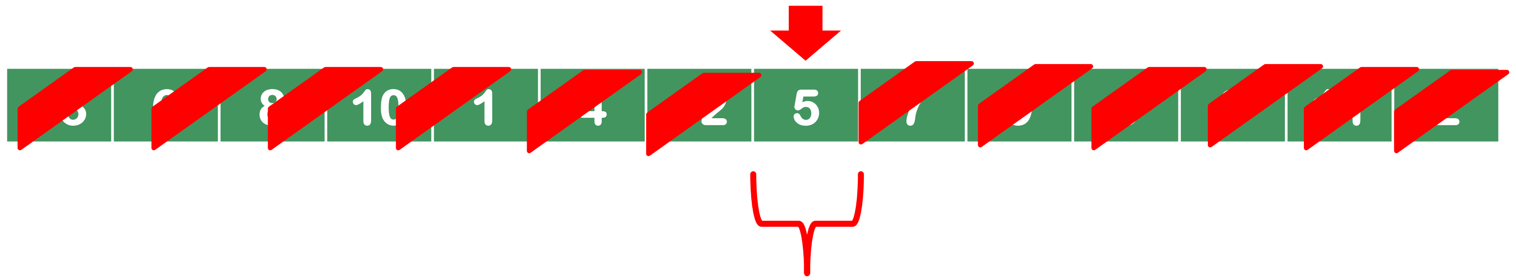
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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



Suffix Array

❖ Since the suffixes are sorted, we can proceed by bisection.

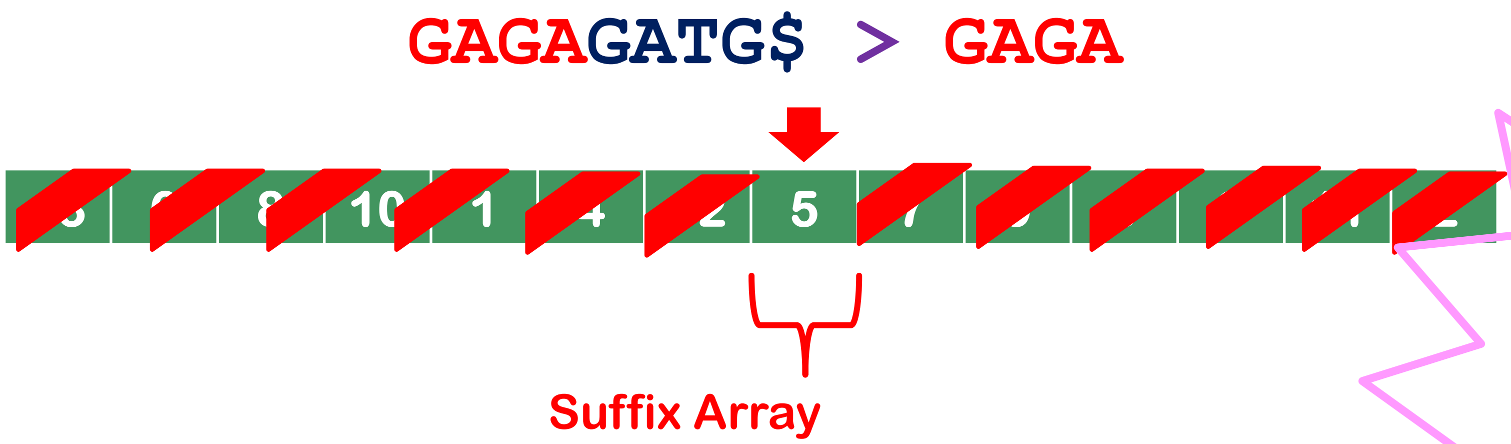
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

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 the suffix array of the human genome to solve the query problem?

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

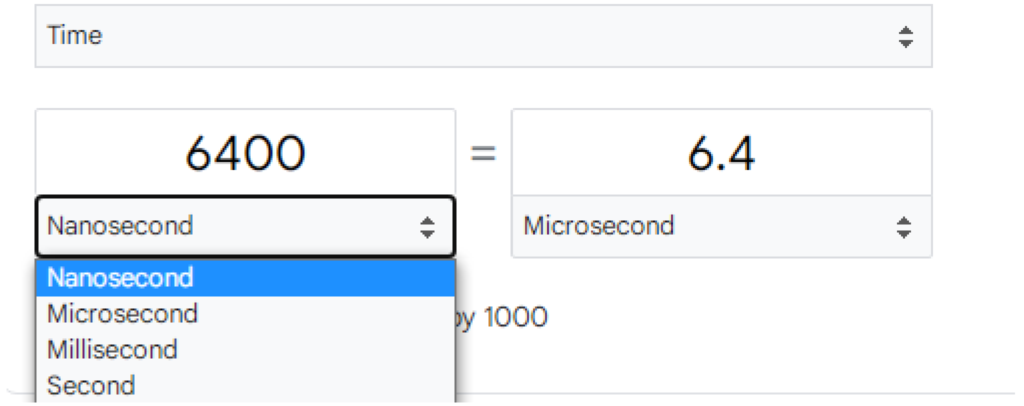


Query located at position 5

❖ Since the suffixes are sorted, we can proceed by bisection.
❖We lookup the middle entry of the suffix array, which points to a particular position of the human genome.

SUFFIX ARRAY

- The suffix array of the human genome has **N = 3.2** billion entries, so we need at most **$\lfloor \log_2(N) \rfloor + 1 = 32$** steps to find out whether any query is present or not.
- We would need a few extra steps to find out the number of occurrences in case it is present.
- Each step consists of **2 memory accesses**, one in the suffix array, one in the human genome to read the suffix.
- Counting approximately **100 ns** per memory access and ignoring the time for string comparison, this brings us around 6-7 ms per query.



Time

6400 Nanosecond = 6.4 Microsecond

Source: http://blog.thegrandlocus.com/2016/07/a-tutorial-on-burrows-wheeler-indexing-methods

SUFFIX ARRAY

- We can encode every position of the genome with a 4 byte integer, and we need to store 3.2 billion entries, so we need 11.92 GB (Vs. 200 GB in Hash based Mapping).
- Still a lot, but notice that this approach solves all the practical difficulties associated with dictionaries. We can easily look for sequences of any length in the suffix array.

BURROWS WHEELER TRANSFORM (BWT)

- The significance of BWT for most of the world is as a data compression technique (Basis for the bzip2 compression algorithm).
- BWT leads to a block-sorted data structure that is well suited to searching short strings in a larger text.
- The FM index uses the BWT to enable search with time linear in the length of the search string.
- The BWT applies a reversible transformation to a block of input text. The transformation does not itself compress the data, but reorders it to make it easy to compress with simple algorithms such as move-to-front coding.

BURROWS WHEELER TRANSFORM (BWT)

Example:
Construct BWT of GATGCGAGAGATG?

Step 1: Add a terminator character $, which has lower lexical order than all other characters.

GATGCGAGAGATG$

BURROWS WHEELER TRANSFORM (BWT)

Step 2: form all rotations of the input text T="GATGCGAGAGATG$"

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

BURROWS WHEELER TRANSFORM (BWT)

Step 3: Sort the rotated strings lexicographically.

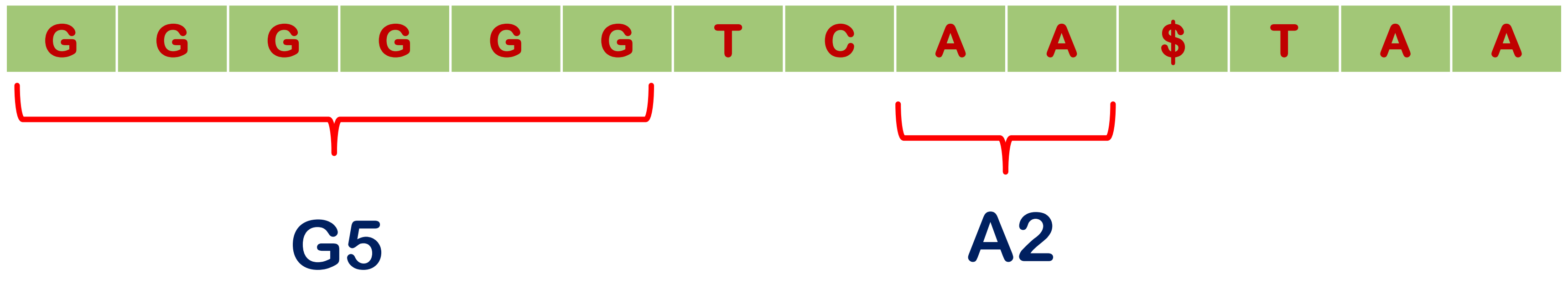
Burrows Wheeler Matrix.

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

BWT

BURROWS WHEELER TRANSFORM (BWT)

G5TCA2$TA2



The BWT tends to contain lots of "runs " of identical characters, which is a good feature to have for compression algorithms such as run-length encoding.

Thank you!