Burrows Wheeler Aligner (BWA)

SBS BIOHACKATHON || PYTHON



Scope

- 1. Introduction to Indexing
- 2. Suffix Array
- 3. Burrows Wheeler Aligner
- 4. Auxiliary Data Structures for BWA FM-Index
- 5. Combining Workshop 1, 2 & 3!
- 6. Additional Resources









Background

NTU SBS Graduate 2020

Bioinformatics Specialist

- A*STAR Genome Institute of Singapore (GIS)
- Nalagenetics

Part Time Masters in BioMedical Data Science

Indexing



Grouping

nest site hunting, 482-87 Macrotermes (termites), 59-60 honeypot ants, see Myrmecocystus male recognition, 298 hormones, 106-9 mass communication, 62-63, 214-18 see also exocrine glands mating, multiple, 155 house (nest site) hunting, 482-92 maze following, 119 Hymenoptera (general), xvi Megalomyrmex (ants), 457 haplodiploid sex determination, 20-22 Megaponera (ants), see Pachycondyla Hypoponera (ants), 194, 262, 324, 388 Melipona (stingless bees), 129 Melophorus (ants), repletes, 257 inclusive fitness, 20-23, 29-42 information measurement, 251-52 memory, 117-19, 213 Messor (harvester ants), 212, 232 intercastes, 388-89 see also ergatogynes; ergatoid queens; mind, 117-19 Monomorium, 127, 212, 214, 216-17, gamergates Iridomyrmex (ants), 266, 280, 288, 321 292 Isoptera, see termites motor displays, 235-47 mound-building ants, 2 juvenile hormone, caste, 106-9, 372 multilevel selection, 7, 7-13, 24-29 mutilation, ritual, 366-73 kin recognition, 293-98 mutualism, see symbioses, ants kin selection, 18-19, 23-24, 28-42, 299, Myanmyrma (fossil ants), 318 Myopias (ants), 326

Ordering

Suffix

What is a suffix?

A suffix is a **substring** at the end of a string of characters. For our purpose, suffixes are non-empty.

Suffixes of 'Horse'

E

SE

RSE

ORSE

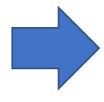
HORSE

Suffix Array

What is a Suffix Array?

A suffix array is an array which contains all the **sorted** suffixes of a string (eg. lexicographically)

Index	Suffix
0	camel
1	amel
2	mel
3	el
4	1



1	Index	Suffix
1	L	amel
()	camel
•	3	el
2	<u>'</u>	1
6	2	mel

Suffix Array

What is a Suffix Array?

The actual 'suffix array' is the array of sorted indices.

This provides a compressed representation of the sorted suffixes without actually needing to store the suffixes

Index	Suffix
1	amel
0	camel
3	el
4	1
2	mel

Basic Code Implementation

Suffix Array

```
Key Concepts:
```

Loop

Slicing

Data Structure: Dictionary

Sort The Dictionary (Lexicographically)

Retrieve the Keys of the Dictionary (Indices)

Burrows-Wheeler Transform (or, BWT) is a **block compression algorithm** and is used in programs like bzip. The compression is **lossless** and allows for **fast querying**.

```
Let's say our word of the
day is:
MISSISSIPPI
Step 1:
 Add a terminator symbol ($)
 MISSISSIPPI$
```

Step 2: Create a suffix array

```
<u>SA</u>
  mississippi$
                       12
  ississippi$
                       11
  ssissippi$
                           ippi$
  sissippi$
                           issippi$
   issippi$
                           ississippi$
  ssippi$
                           mississippi$
   sippi$
                           pi$
                       10
  ippi$
                           ppi$
                        9
   ppi$
                           sippi$
10 pi$
                           sissippi$
                           ssippi$
11 i$
12 $
                           ssissippi$
```

Step 3:

Burrows Wheeler Transform

<u>SA</u>

```
12
                         $mississippi
                         i$mississipp
11
   ippi$
                         ippi$
   issippi$
                         issippi$
   ississippi$
                         ississippi$
   mississippi$
                         mississippi$
   pi$
                         pi$
10
   ppi$
                         ppi$
9
   sippi$
                         sippi$
   sissippi$
                         sissippi$
   ssippi$
                         ssippi$
   ssissippi$
                         ssissippi$
```

Step 3:

Burrows Wheeler Transform

<u>SA</u>

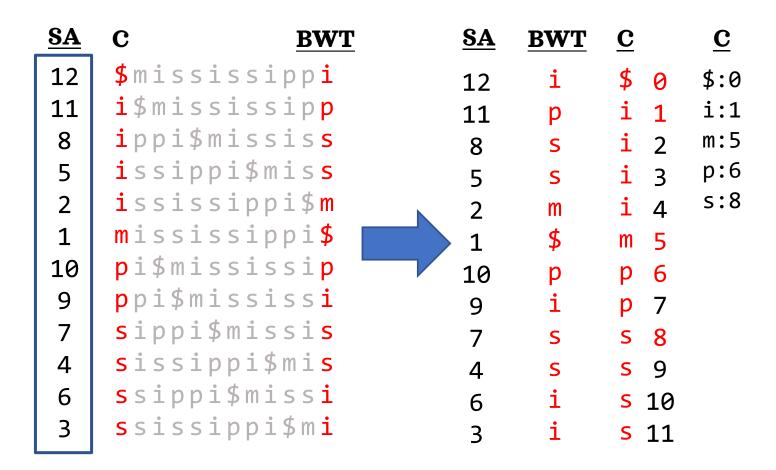
```
12
                        $mississippi
                        i$mississipp
11
   ippi$
                        ippi$mississ
   issippi$
                        issippi$miss
   ississippi$
                        ississippi$m
   mississippi$
                        mississippi$
   pi$
                        pi$mississip
10
   ppi$
                        ppi$mississi
9
   sippi$
                        sippi$missis
   sissippi$
                        sissippi$mis
   ssippi$
                        ssippi$missi
   ssissippi$
                        ssissippi$mi
```

Step 3:

Burrows Wheeler Transform

```
<u>SA</u>
                           {f BWT}
    $mississippi
12
    i$mississipp
11
                            p
    ippi$mississ
                            S
    issippi$miss
                            S
    ississippi$m
                            m
    mississippi$
    pi$mississip
10
                            p
    ppi$mississi
9
    sippi$missis
                            S
    sissippi$mis
                            S
    ssippi$missi
                            i
    ssissippi$mi
```

Step 3: Burrows Wheeler Transform



Essentially What We Need

<u>SA</u>	\mathbf{c}	$\mathbf{\underline{BWT}}$	<u>SA</u>	$\overline{\mathbf{BWT}}$	\mathbf{c}
12	\$mississip	рр і	12	i	\$:0
11	i \$mississi	i p p	11	р	i:1
8	ippi\$missi	İSS	8	S	m:5
5	issippi\$mi	İSS	5	S	p:6
2	i ssissippi	i\$m	2	m	s:8
1	mississipp	pi\$	1	\$	
10	pi\$mississ	sip	10	р	
9	<pre>ppi\$missis</pre>	ss i	9	i	
7	sippi\$miss	si s	7	S	
4	sissippi\$r	nis	4	S	
6	ssippi\$mis	ss i	6	i	
3	s sissippi	≸m i	3	i	

When we index the BWT!!

<u>SA</u>	\mathbf{c}	\mathbf{BWT}	<u>SA</u>	$\underline{\mathbf{BWT}}$	<u>c</u>
12	<pre>\$ mississi</pre>	ppi ₀	12	i	\$:0
11	i ₀ \$mississ	ipp ₀	11	р	i:1
8	i ₁ ppi\$miss		8	S	m:5
5	i ₂ ssippi\$m	iss ₁	5	S	p:6
2	i ₃ ssissipp	i\$m ₀	2	m	s:8
1	m ₀ ississip	pi\$	1	\$	
10	p ₀ i\$missis	· ·	10	р	
9	p ₁ pi\$missi	<u>-</u>	9	i	
7	s ₀ ippi\$mis	SiS ₂	7	S	
4	s ₁ issippi\$	mis ₃	4	S	
6	s ₂ sippi\$mi	_	6	i	
3	s ₃ sissippi	\$ m i 3	3	i	

Step 3:

Burrows Wheeler Transform

```
BWT
<u>SA</u>
                                             BWT
12
                   $ mississippia
                   i<sub>a</sub>$mississipp<sub>a</sub>
11
                   i<sub>1</sub>ppi$mississ<sub>0</sub>
                   i,ssippi$miss<sub>1</sub>
                   i<sub>3</sub>ssissippi$m<sub>a</sub>
                   m<sub>o</sub>ississippi$
                   p<sub>0</sub>i$mississip<sub>1</sub>
10
                   p<sub>1</sub>pi$mississi<sub>1</sub>
                   s<sub>a</sub>ippi$missis<sub>2</sub>
                   S<sub>1</sub>issippi$mis<sub>3</sub>
                   s<sub>2</sub>sippi$missi<sub>2</sub>
                   s<sub>3</sub>sissippi$mi<sub>3</sub>
                        Right Context
```

BWA Querying

Let's say Pattern = iss

```
\mathbf{F}
                                           <u>SA</u>
       $ mississippia
       i<sub>a</sub>$mississipp<sub>a</sub>
       i<sub>1</sub>ppi$mississ<sub>a</sub>
                                                  ippi$
       i<sub>2</sub>ssippi$miss<sub>1</sub>
                                                  issippi$
[3,5)
                                    [3,5)
                                                  ississippi$
                                                  mississippi$
       m<sub>a</sub>ississippi$
       p<sub>0</sub>i$mississip<sub>1</sub>
                                                  pi$
       p<sub>1</sub>pi$mississi<sub>1</sub>
                                                  ppi$
       s<sub>0</sub>ippi$missis<sub>2</sub>
                                                sippi$
       S<sub>1</sub>issippi$mis<sub>3</sub>
                                                  sissippi$
       s,sippi$missi,
                                                ssippi$
       s<sub>3</sub>sissippi$mi<sub>3</sub>
                                                  ssissippi$
```

Unlike Suffix Array, we don't immediately know where the matches are in Text

BWA Querying

This is the information we have now How do we query?

<u>SA</u>	\mathbf{BWT}	<u>c</u>
12	i	\$:0
11	р	i:1
8	S	m:5
5	S	p:6
2	m	s:8
1	\$	
10	р	
9	i	
7	S	
4	S	
6	i	
3	i	

Pattern = iss

ISS ISS \mathbf{F} \mathbf{L} \$ mississippia \$ mississippia \$ mississippi_a i_a\$mississipp_a i_a\$mississipp_a i_a\$mississipp_a i₁ppi\$mississ_a i₁ppi\$mississ_a i₁ppi\$mississ₀ i,ssippi\$miss₁ i₂ssippi\$miss₁ i₂ssippi\$miss₁ i₃ssissippi\$m₀ i₃s issippi\$m_a i₃ssissippi\$m₀ m_aississippi\$ m_aississippi\$ m_oissippi\$ p_ai\$mississip₁ p_ai\$mississip₁ p₀i\$hississip₁ p₁pi\$mississi₁ p₁pi\$mississi₁ p₁pi\$mississi₁ s₀ippi\$\lssis, Saippi\$missi S₁issippi\$mis₃ s, sippi\$miss s₃sissippi\$

ISS

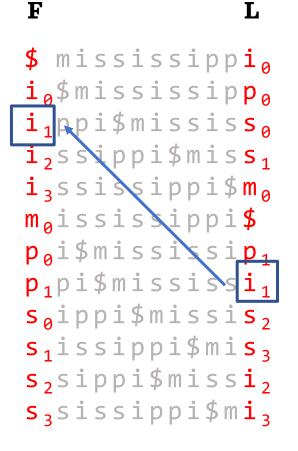
 \mathbf{L}

SIP

\mathbf{F} \$ mississippia i_a\$mississipp_a i₁ppi\$mississ₀ i,ssippi\$miss₁ i₃ssissippi\$m₀ m_aississippi\$ S₀ippi\$missis₂ S₁issippi\$mis₃ s,sippi\$missi,

s₃sissippi\$mi₃

SIP



SIP

```
\mathbf{L}
12 $ mississippia
11 i<sub>0</sub>$mississipp<sub>0</sub>
     i<sub>2</sub>ssippi$m/is<mark>s</mark>1
    i<sub>3</sub>ssissippi$m<sub>0</sub>
    m<sub>o</sub>issis/sippi$
\underline{10} p_0 i \$ m i ssissip_1
    p<sub>1</sub>p<sub>1</sub>/$mississi<sub>1</sub>
7 S<sub>0</sub> fppi$missis<sub>2</sub>
\underline{4} S_1 is sippi $ m i S_3
6 S<sub>2</sub>sippi$missi<sub>2</sub>
3 S<sub>3</sub>sissippi$mi<sub>3</sub>
```

Now Its your Turn!

Let's say we have this information; can you reform the word? Hint, first step is to order the letters in BWT Lexicographically

<u>SA</u>	$\mathbf{\underline{BWT}}$
15	N
7	Н
10	K
3	S
1	S
8	Α
6	0
12	Т
4	В
9	C
14	0
5	I
13	Н
2	В
0	\$
11	Α

Now Its your Turn!

Let's say we have this information; can you reform the word? Hint, first step is to order the letters in BWT Lexicographically

SA	$\mathbf{B}\mathbf{W}\mathbf{T}$		
15	N	\$	N
15 7		Α	Н
7	Н	Α	Κ
10	K	В	5
3	S	В	\ \ \ \ \
1	S		ر ۸
8	Α	C	A
6	0	Н	O
12	T	H I	S S A O T B
12 4		I	В
4	В	K	
9	C	N	C 0
14	0	N	I
5	I	0	
13	Н	O	Н
	В	S	В
2 0		0 0 S S	\$
Ø	\$	Т	Α
11	Α	-	

Now Its your Turn!

Let's say we have this information; can you help query the string "GAT"?

$\underline{\mathbf{S}}\underline{\mathbf{A}}$	\mathbf{BWT}
10	Т
2	G
6	G
4	Т
1	G
5	C
0	\$
8	T
9	G
3	Α
7	Α

Auxiliary Data Structure

Fm-Index!

FM-Index

OCC Data Structure - Store BWA details

	BWT	I	М	Р	S
\$mississippi	i	1	0	0	0
i \$mississip p	р	1	0	1	0
ippi\$missis s	S				
issippi\$mis s	S				
ississippi\$m	m				
mississippi\$	\$				
pi\$mississip	р				
ppi\$mississ i	i				
sippi\$missis	S				
sissippi\$mis	S				
ssippi\$missi	i				
ssissippi\$mi	i				

FM-Index

OCC Data Structure

	BWT	I	М	Р	S
\$mississipp i	i	1	0	0	0
i \$mississip p	р	1	0	1	0
ippi\$missis s	S	1	0	1	1
issippi\$mis s	S	1	0	1	2
ississippi\$m	m	1	1	1	2
mississippi\$	\$	1	1	1	2
pi\$mississip	р	1	1	2	2
ppi\$mississ i	i	2	1	2	2
sippi\$missis	S	2	1	2	3
sissippi\$mis	S	2	1	2	4
ssippi\$missi	i	3	1	2	4
ssissippi\$mi	i	4	1	2	4

OCC Data Structure: Pattern = iss

F	BWT	I	М	P	S
\$	i	1	0	0	0
i	р	1	0	1	0
i	S	1	0	1	1
i	S	1	0	1	2
i	m	1	1	1	2
m	\$	1	1	1	2
р	р	1	1	2	2
р	i	2	1	2	2
S	S	2	1	2	3
S	S	2	1	2	4
S	i	3	1	2	4
S	i	4	1	2	4

F	BWT	I	М	Р	S
\$	i	1	0	0	0
i	р	1	0	1	0
i	S	1	0	1	1
i	S	1	0	1	2
i	m	1	1	1	2
m	\$	1	1	1	2
р	р	1	1	2	2
р	i	2	1	2	2
S	S	2	1	2	3
S	S	2	1	2	4
S	i	3	1	2	4
S	i	4	1	2	4

F	BWT	I	М	Р	S
\$	i	1	0	0	0
i	р	1	0	1	0
i	S	1	0	1	1
i	S	1	0	1	2
i	m	1	1	1	2
m 🐧	\$	1	1	1	2
p	р	1	1	2	2
р	i	2	1	2	2
S	M	2	1	2	3
S	S	2	1	2	4
S	i	3	1	2	4
S	i	4	1	2	4

BWA FM-Index Code Implementation!

Exact Matching

Pipeline example

Burrows Wheeler Aligner

Bit By Bit

BackTracking [CIGAR score]

Edit Distance

4 situations:

- 1. Match (Move diagonally)
- 2. Substitution (Move diagonally + 1)
- 3. Insertion (Move right + 1)
- 4. Deletion (Move down + 1)

Find the lowest score

Query: TGATA Hit : TGGACT

		T	G	Α	Т	Α
	0	1	2	3	4	5
Т	1	0				
G	2					
G	3					
Α	4					
С	5					
Т	6					

		Т	G	Α	Т	Α
	0	1	2	3	4	5
Т	1	0	1	2	3	4
G	2	1	0	1	2	3
G	3	2	1	1	2	3
Α	4	3	2	1	2	2
С	5	4	3	2	2	3
Т	6	4	3	3	2	3

References

```
Burrows-Wheeler Indexing By Ben Langmead:
```

https://www.youtube.com/watch?v=5G2Db41pSHE&list=PL
2mpR0RYFQsADmYpW2YWBrXJZ 6EL 3nu

BWA Paper:

https://academic.oup.com/bioinformatics/article/25/
14/1754/225615

GitHub Repository:

https://github.com/micro-irfan/SBSBiohackathon

Where to Next?

Rosalind Bioinformatics Challenge

Coursera Bioinformatics (UC San Diego / John Hopkins)

William Fiset Data Structures Course

RNA-seq Youtube Code Along

Find a lab attachment

Kaggle (For machine learning; data analyst / scientist)

Build a github repository!

Code code code!