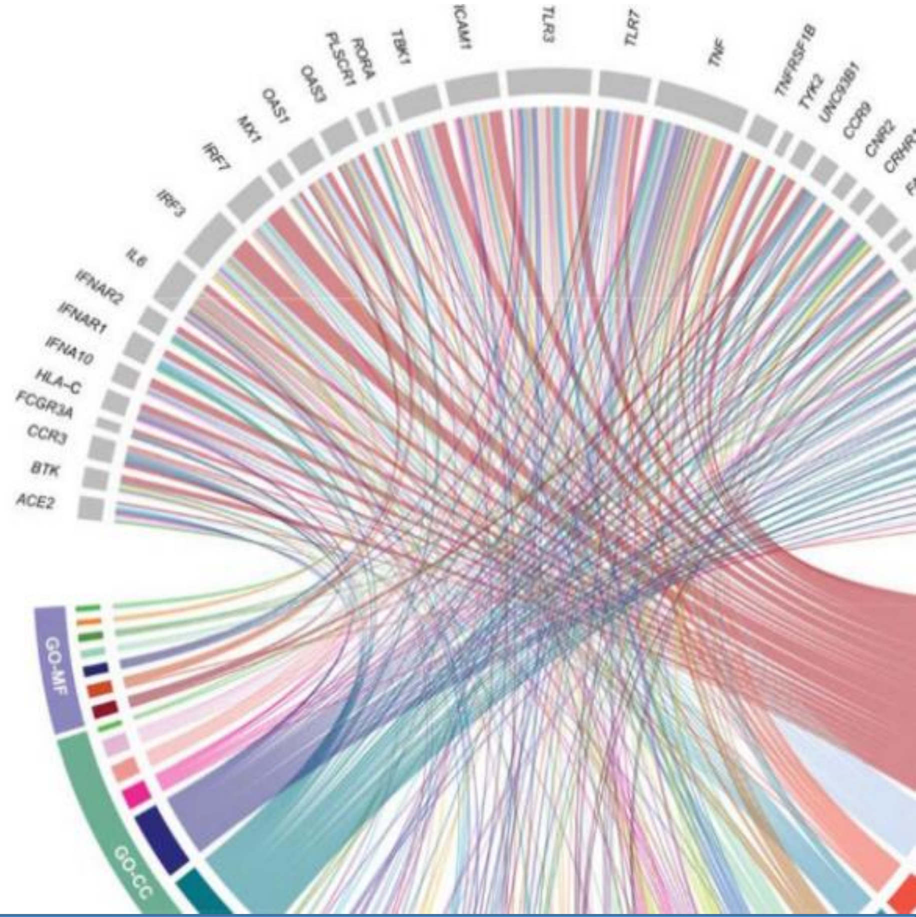
Tècniques i Eines Bioinformàtiques

FM-Index

Santiago Marco-Sola (santiago.marco@upc.edu)

Màster en Enginyeria Informàtica, UPC
Departament of Computer Science
Facultat d'Informàtica de Barcelona (FIB), UPC



Course 2024-25

Acknowledgements

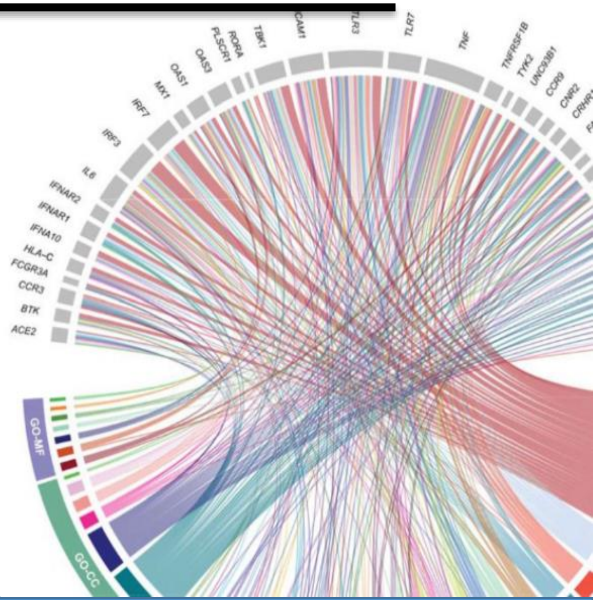
Many pictures and materials are taken from **Ben Langmead's course**.

Course heavily inspired in:

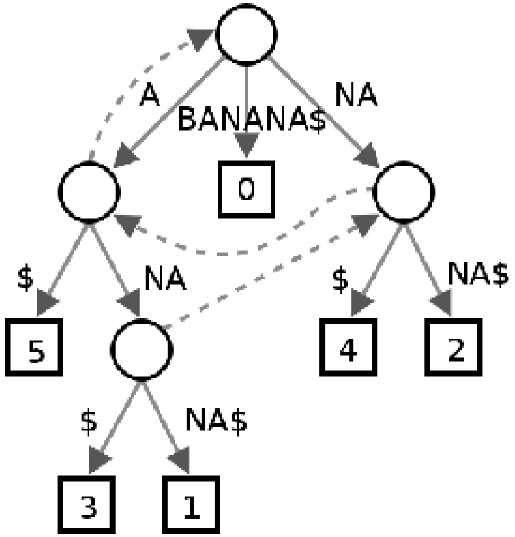
- **Genome-Scale Algorithm Design**. Veli Mäkinen, Djamal Belazzougui, Fabio Cunial, Alexandru I. Tomescu. Cambridge University Press.
- **Algorithms on Strings, Trees, and Sequences**. Dan Gusfield. Cambridge University Press.
- **An Introduction to Bioinformatics Algorithms.** Neil C. Jones, Pavel A. Pevzner. MIT Press.

1

FM-Index


IRF7
MX1
OAS1
IRF3
IL6
IFNAR2
IFNAR1
IFNA10
HLA-C
FCGR3A
CCR3
BTK
ACE2
TLR3
TLR7
TNF
GO-MF
GO-CC

FM Index: small memory footprint



Suffix tree

≥ 45 GB

6	$
5	A$
3	ANA$
1	ANANA$
0	BANANA$
4	NA$
2	NANA$

Suffix array

≥ 12 GB

$BANAN**A**
A$BANA**N**
ANA$BA**N**
ANANA$**B**
BANANA**$**
NA$BAN**A**
NANA$B**A**

FM Index

~1.5 GB

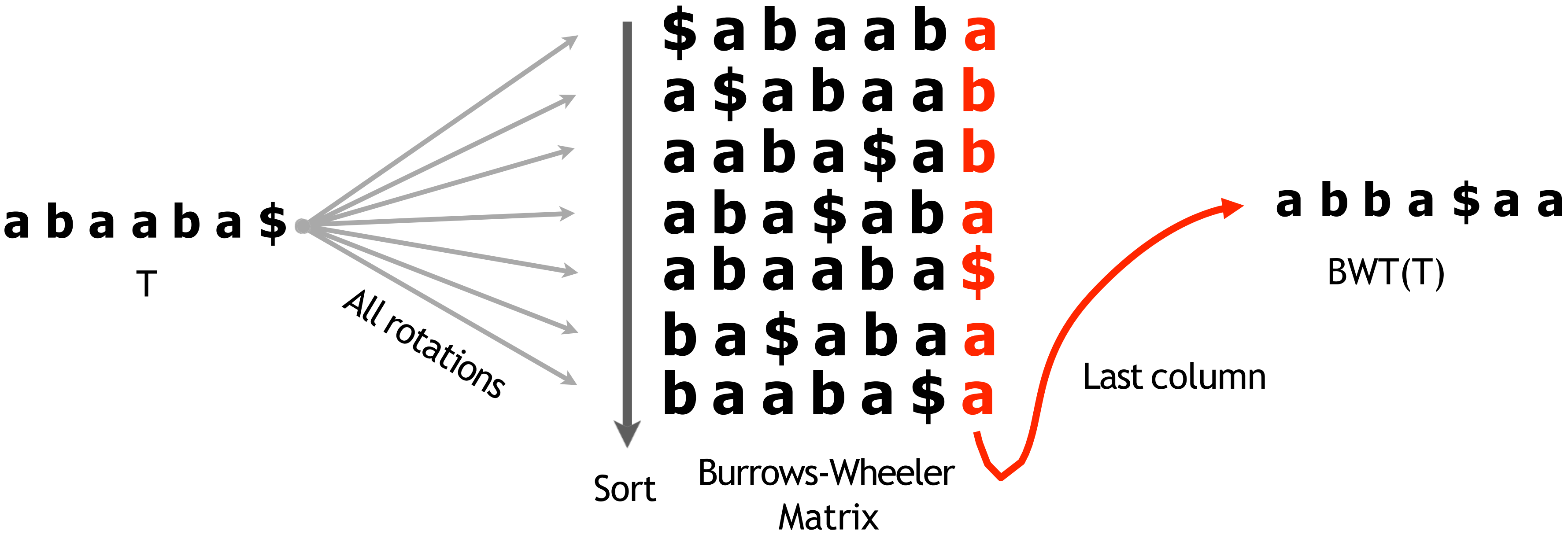
Suffix index bounds

	Suffix tree	Suffix array	FM Index
Time: Does P occur?	$O(n)$	$O(n \log m)$	$O(n)$
Time: Count k occurrences of P	$O(n + k)$	$O(n \log m)$	$O(n)$
Time: Report k locations of P	$O(n + k)$	$O(n \log m + k)$	$O(n + k)$
Space	$O(m)$	$O(m)$	$O(m)$
Needs T?	*yes*	*yes*	*no*
Bytes per input character	>15	~4	~0.5

$m = |T|, n = |P|, k = \#\text{ occurrences of } P \text{ in } T$

Burrows-Wheeler Transform

Reversible permutation of the characters of a string, used originally for compression



a b a a b a $

T

$ a b a a b a
a $ a b a a b
a a b a $ a b
a b a $ a b a
a b a a b a $
b a $ a b a a
b a a b a $ a

Burrows-Wheeler Matrix

a b b a $ a a

BWT(T)

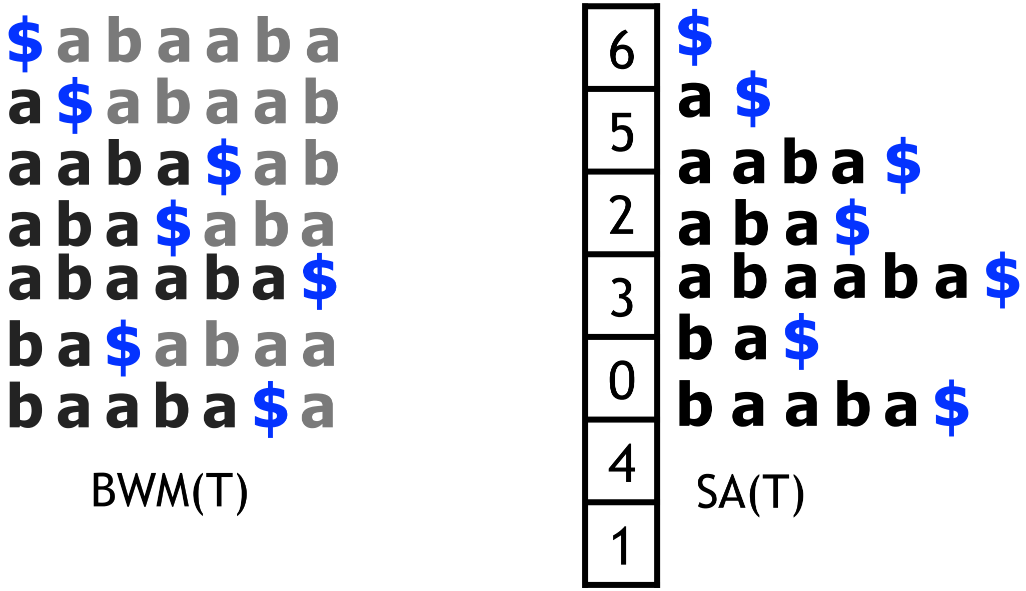
Burrows M, Wheeler DJ: A block sorting lossless data compression algorithm. *Digital Equipment Corporation, Palo Alto, CA* 1994, Technical Report 124; 1994

Burrows-Wheeler Transform

In fact, this gives us a new definition / way to construct BWT(T):

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

"BWT = characters just to the left of the suffixes in the suffix array"



$abaaba
a$abaab
aaba$ab
aba$aba
abaaba$
ba$abaa
baaba$a

BWM(T)

6 $
5 a$
2 aaba$
3 aba$
0 abaaba$
4 ba$
1 baaba$

SA(T)

Burrows-Wheeler Transform

We've seen how BWT is useful for compression:

- Sorts characters by right-context, making a more compressible string

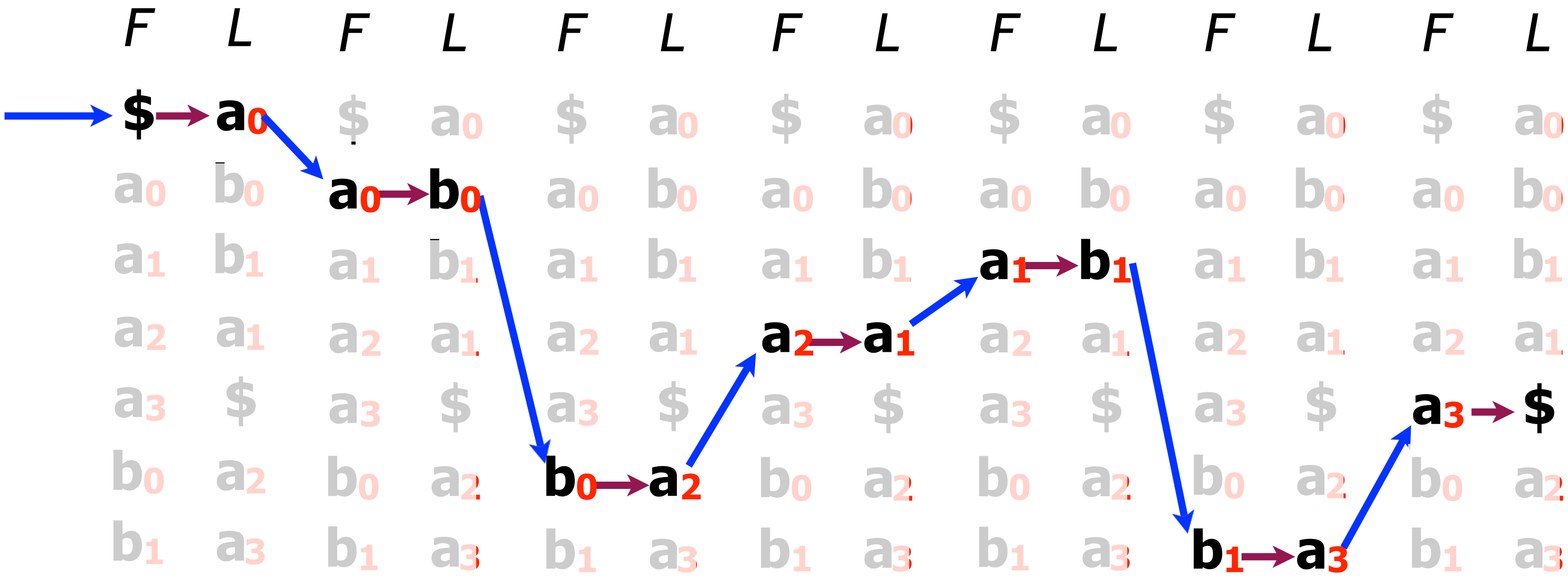
And how it's reversible:

- Repeated applications of LF Mapping, recreating *T* from right to left

How is it used as an index?

Burrows-Wheeler Transform: reversing

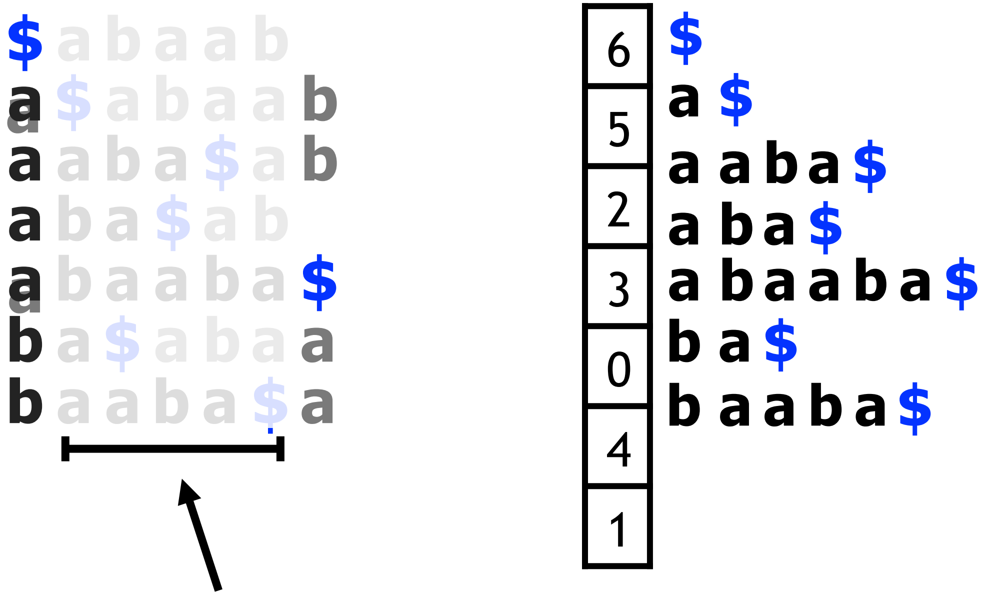
Another way to visualize:



T: $a_3 b_1 a_1 a_2 b_0 a_0 \$$

FM Index: querying

Can we query like the suffix array?



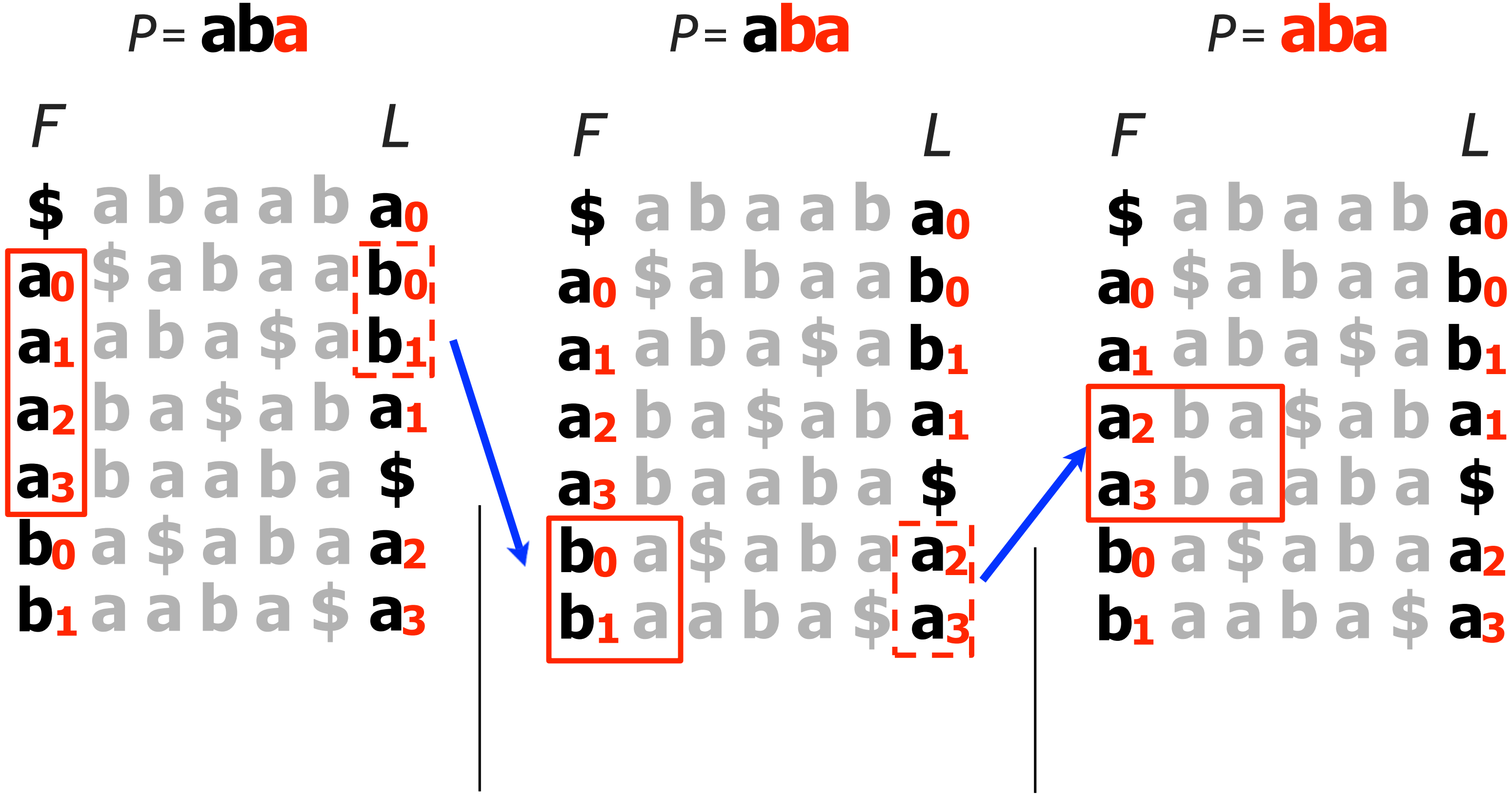
```
$ a b a a b
a $ a b a a b
a a b a $ a b
a b a $ a b
a b a a b a $
b a $ a b a a
b a a b a $ a
```

6	$
5	a $
2	a a b a $
3	a b a $
0	a b a a b a $
4	b a $
1	b a a b a $

We don’t have these columns, and we don’t have T.
Binary search not possible.

FM Index

$|T| = m$, $|P| = n$



$P =$ **aba**

F						L
$	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

$P =$ **aba**

F						L
$	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

$P =$ **aba**

F						L
$	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

2 $O(1)$ rank calculations

2 $O(1)$ rank calculations

Determining of P occurs in T in FM Index is $O(n)$ time

FM Index: small memory footprint

Paolo Ferragina, and Giovanni Manzini. "Opportunistic data structures with applications." *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*. IEEE, 2000.

FM Index described here is simplified version of what's described in paper

Also discussed in paper: compressing BWT(*T*) for further savings (and selectively decompression portions of it at query time)