25 Years of the BWT: The Past and the Future of an Unusual Compressor

Giovanni Manzini
University of Eastern Piedmont, Alessandria, Italy
Institute of Informatics and Telematics, National Research Council, Pisa, Italy

IEEE Data Compression Conference
Snowbird (UT) March 27th, 2019

1994

Mandela first black South Africa president

May 10, 1994

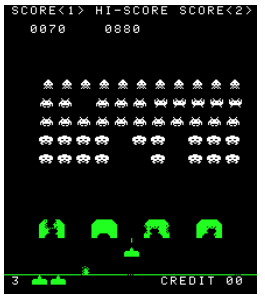
SRC Research Report 124

A Block-sorting Lossless Data Compression Algorithm

M. Burrows and D.J. Wheeler

digital
Systems Research Center
130 Lytton Avenue
Palo Alto, California 94301

1978 (?)



Space Invaders released

- David Wheeler conceives a data compression algorithm based on reversible transformation on the input text, but considers it too slow for practical use

1994

- Mike Burrows improves the speed of the compressor. B&W co-author the technical report describing a “block sorting” lossless data compression algorithm.
- The algorithm splits the input in blocks and computes a reversible transformation that makes the text “more compressible”
- The transformation has been later called the Burrows-Wheeler transform.

The BWT

`swiss·miss·missing`

The BWT

swiss·miss·missing

Consider all rotations of the input text

```
s wiss·miss·missin g
w iss·miss·missing s
i ss·miss·missings w
s s·miss·missingsw i
s ·miss·missingswi s
· miss·missingswis s
m iss·missingswiss ·
i ss·missingswiss· m
s s·missingswiss·m i
s ·missingswiss·mi s
· missingswiss·mis s
m issingswiss·miss ·
i ssingswiss·miss· m
s singswiss·miss·m i
s ingswiss·miss·mi s
i ngswiss·miss·mis s
n gswiss·miss·miss i
g swiss·miss·missi n
```

The BWT

`swiss·miss·missing`

Consider all rotations of the input text

Sort them in lexicographic order

```
· miss·missingswis s
· missingswiss·mis s
g swiss·miss·missi n
i ngswiss·miss·mis s
i ss·miss·missings w
i ss·missingswiss· m
i ssingswiss·miss· m
m iss·missingswiss ·
m issingswiss·miss ·
n gswiss·miss·miss i
s ·miss·missingswi s
s ·missingswiss·mi s
s ingswiss·miss·mi s
s s·miss·missingsw i
s s·missingswiss·m i
s singswiss·miss·m i
s wiss·miss·missin g
w iss·miss·missing s
```

The BWT

swiss·miss·missing

Consider all rotations of the input text

Sort them in lexicographic order

Take the last character of each rotation

ssnswmm··isssiiigs

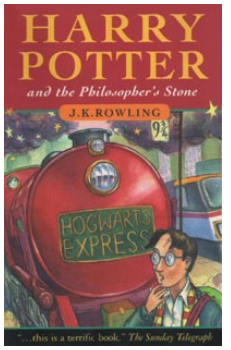
		L
·	miss·missingswis	s
·	missingswiss·mis	s
g	swiss·miss·missi	n
i	ngswiss·miss·mis	s
i	ss·miss·missings	w
i	ss·missingswiss·	m
i	ssingswiss·miss·	m
m	iss·missingswiss	·
m	issingswiss·miss	·
n	gswiss·miss·miss	i
s	·miss·missingswi	s
s	·missingswiss·mi	s
s	ingswiss·miss·mi	s
s	s·miss·missingsw	i
s	s·missingswiss·m	i
s	singswiss·miss·m	i
s	wiss·miss·missin	g
w	iss·miss·missing	s

final char (L)	sorted rotations
a	n to decompress. It achieves compression
o	n to perform only comparisons to a depth
o	n transformation} This section describes
o	n transformation} We use the example and
o	n treats the right-hand side as the most
a	n tree for each 16 kbyte input block, enc
a	n tree in the output stream, then encodes
i	n turn, set $L[i]$ to be the
i	n turn, set $R[i]$ to the
o	n unusual data. Like the algorithm of Man
a	n use a single set of probabilities table
e	n using the positions of the suffixes in
i	n value at a given point in the vector $R
e	n we present modifications that improve t
e	n when the block size is quite large. Ho
i	n which codes that have not been seen in
i	n with ch appear in the {\em same order
i	n with ch. In our exam
o	n with Huffman or arithmetic coding. Bri
o	n with figures given by Bell~\cite{bell}.

Figure 1: Example of sorted rotations. Twenty consecutive rotations from the sorted list of rotations of a version of this paper are shown, together with the final character of each rotation.

- The BWT was computed using a $O(n^2)$ algorithm, fast for non-pathological cases.
- The output of the BWT was compressed using Move-to-front and Huffman/Arithmetic Coding
- The resulting prototype was superior in compression and of similar speed wrt gzip

1997



First Harry Potter novel published

bzip2

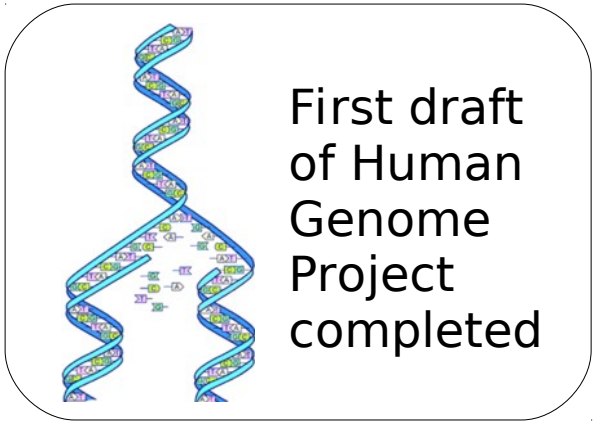
- In 1997 Julian Seward releases the bzip compression tool based on the BWT.
- It was highly optimized, and could be used as a drop-in replacement of gzip, both command line and library version
- Suffix sorting based on Bentley-McIlroy ternary quicksort + tricks O(1) working space, quadratic time worst case

1997-2000

Research on BWT compression was able to:

- Improve compression
- Explain the compression (relate it to known quantities, such as the entropy)
- Improve (de)compression speed, and reduce working space

2000



- First results on compressed indices: the BWT can replace the Suffix Array for exact off-line pattern matching.

R. Grossi, J. Vitter. Compressed Suffix Arrays and Suffix Trees with Applications toText Indexing and String Matching, STOC 00

V. Mäkinen, Compact Suffix Array, CPM 00

P. Ferragina, G. M. Opportunistic data structures with applications, FOCS 00

What does this mean?

- To efficiently search in a 1GB file it was necessary to build a full text index of size 5GB
- Now a BWT-compressed file of size ~300MB was able to store the original text and a full text index
- An unexpected 16x space reduction!

A happy coincidence

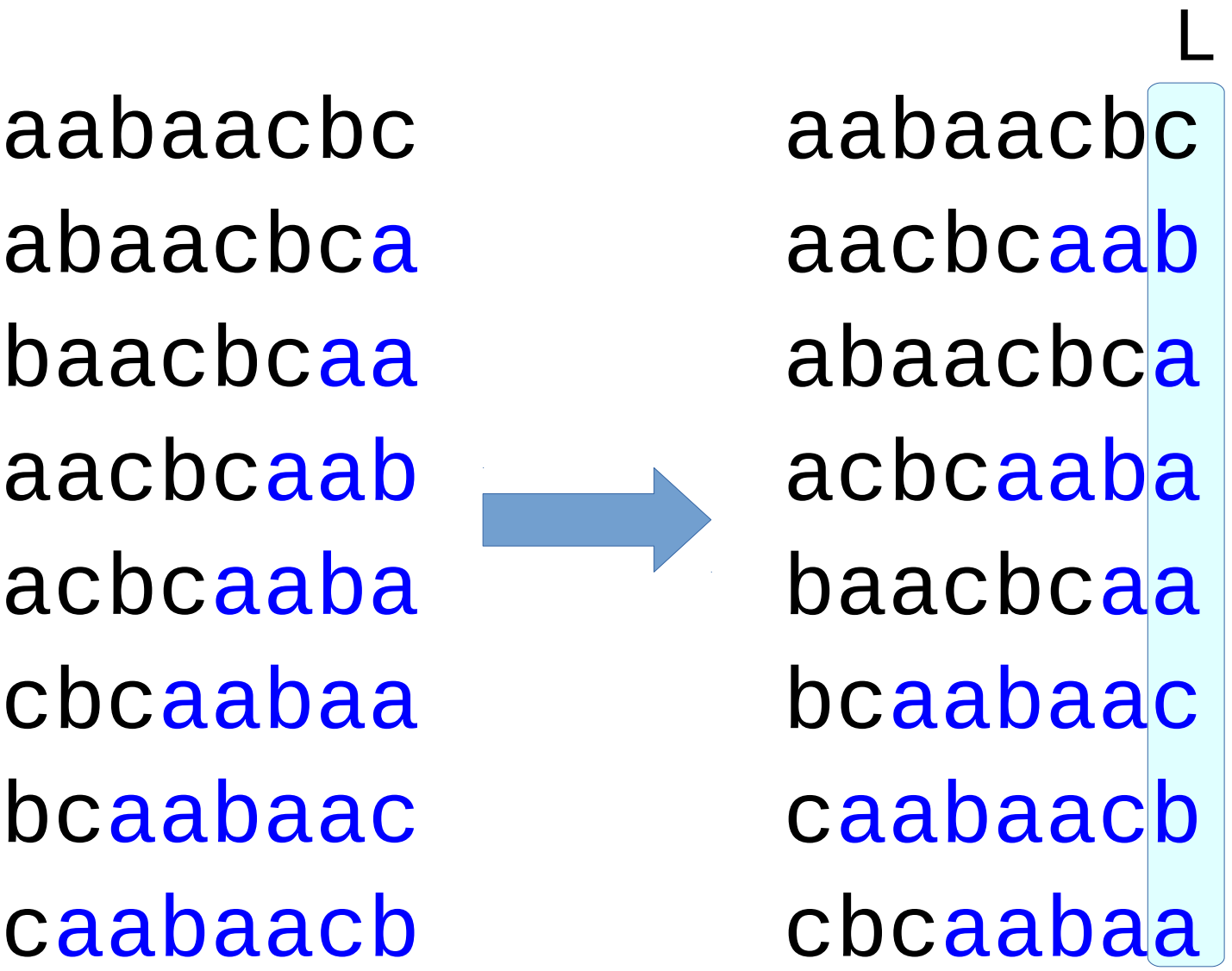
- People was looking for tools to search in the recently assembled human genome
- Traditional full text indices did not fit in RAM
- Thanks to the 16x space reduction, compressed indices did!
- Bowtie [Langmead et al. 09], BWA [Li et al, 09], SOAP2 [Li et al, 09] are extremely popular aligner based on a BWT index.

In the meantime...

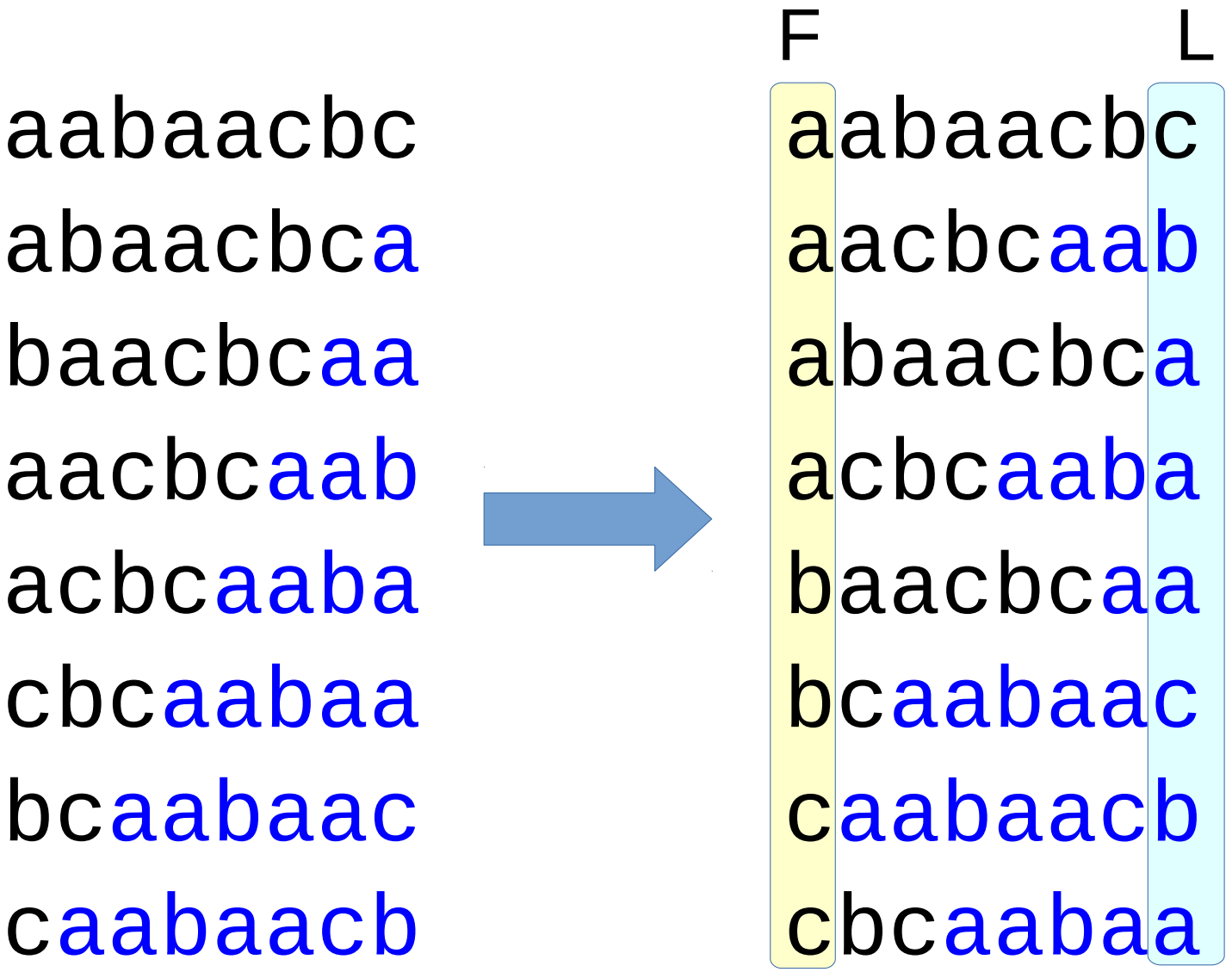
- Researchers were able to devise compressed indices using other compressors (eg LZ77)
- BWT variants designed only for data compression met limited success…
- … while BWT variants for indexing objects different than texts flourished

- BWT variants were devised to compress and index Trees, Graphs, Alignments,…
- Their common trait is that they provide a space efficient solution to an offline matching problem
- We now have a reasonable idea of why this is possible

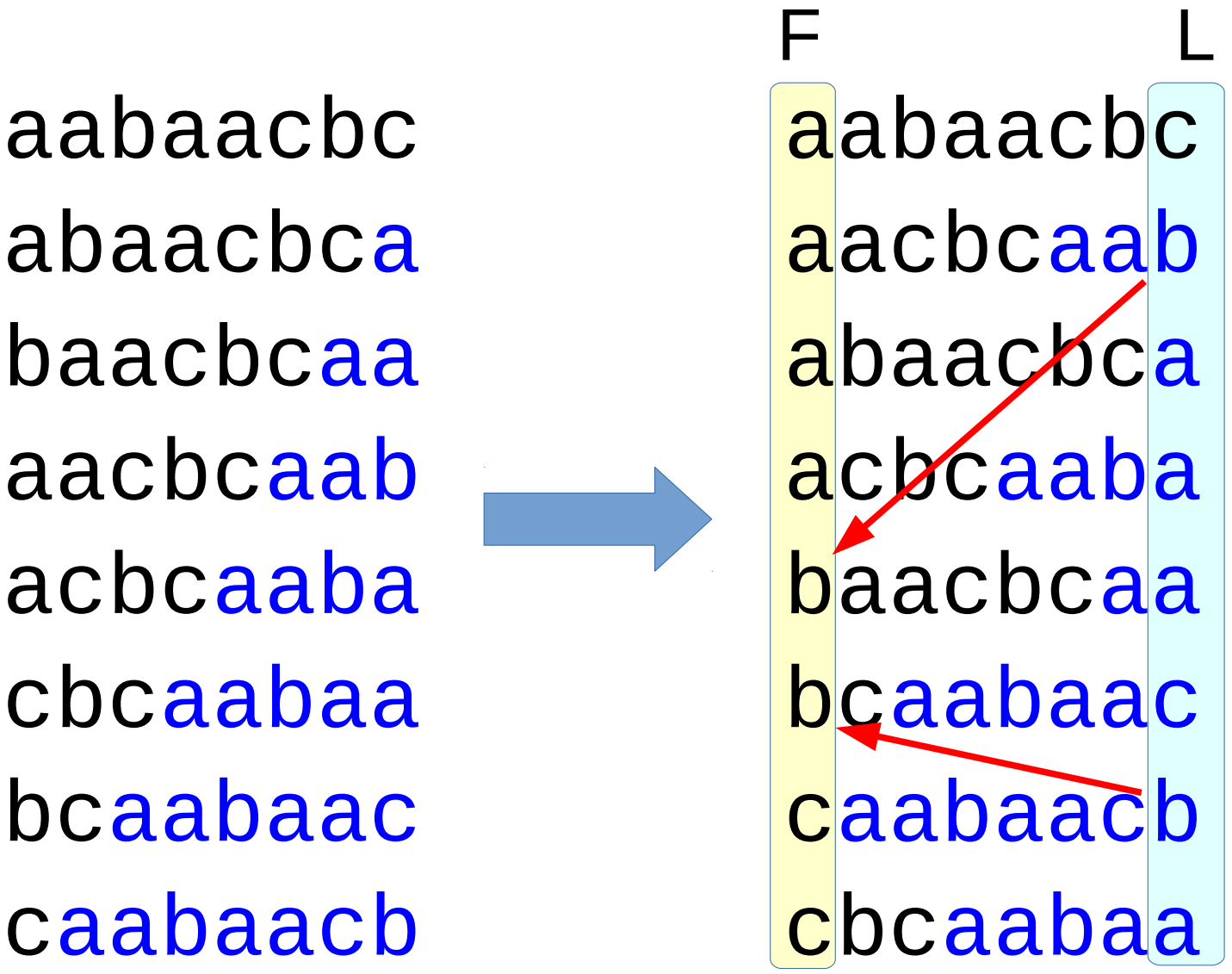
BWT revisited



BWT revisited

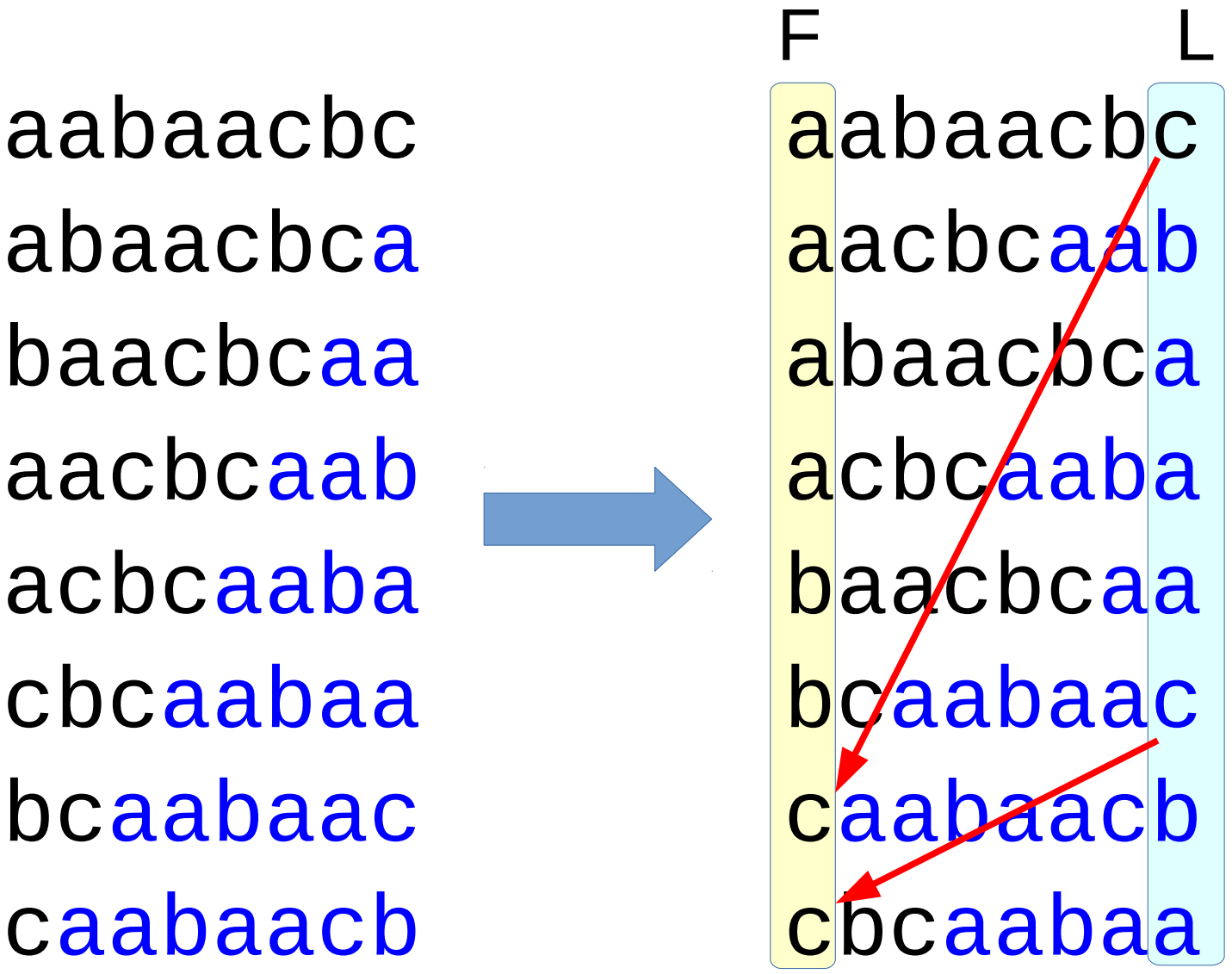


BWT revisited



Last-to-First map is order preserving!

BWT revisited



Last-to-First map is order preserving!

Introducing Wheeler graphs

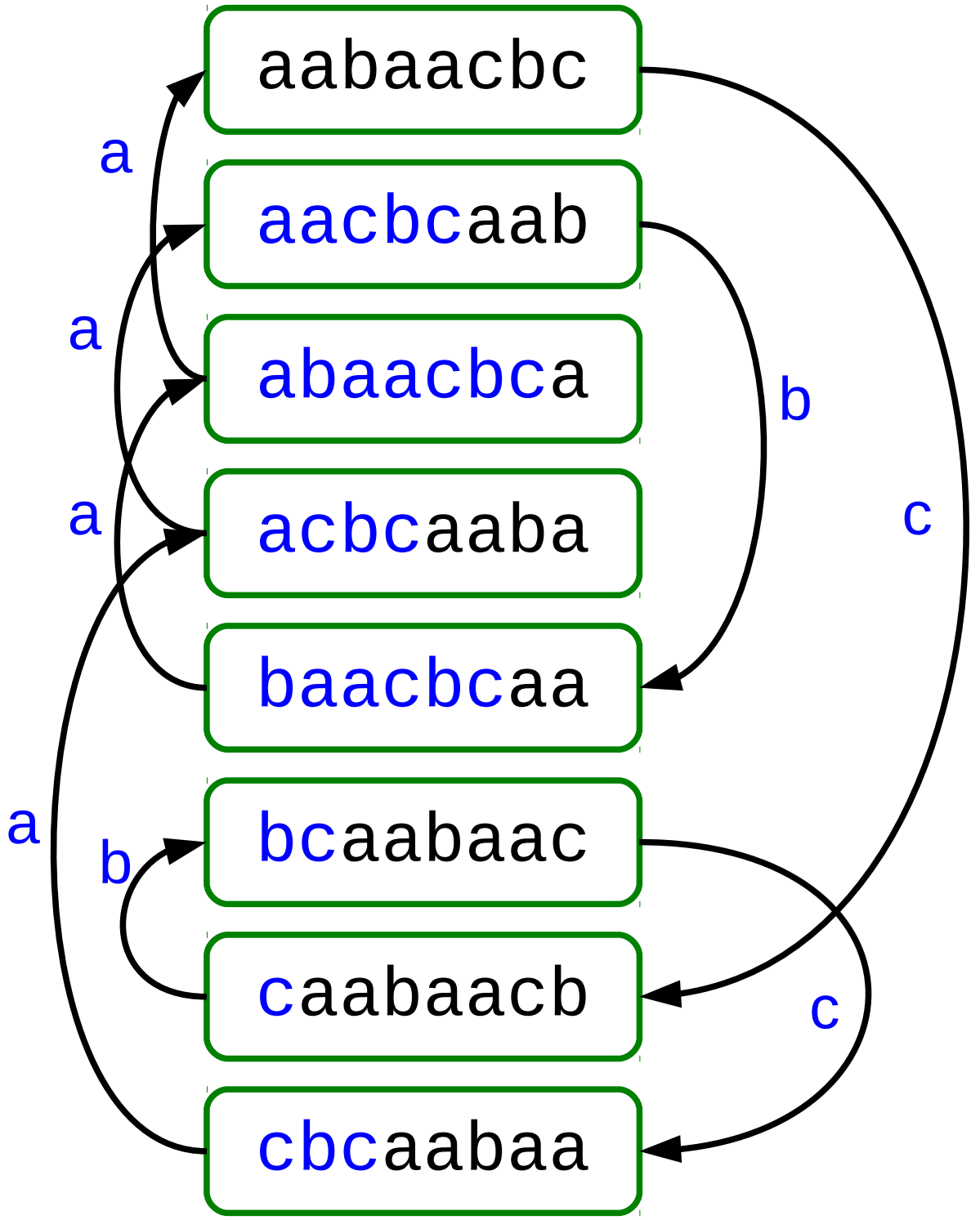
F	L
a	c
a	b
a	a
b	a
b	a
c	a
c	b
c	a

Ordered graph, a node per rotation

Edges according to the LF-map

Edges are order preserving

Wheeler Graph!



Beyond BWT: Wheeler graph

Directed labeled graph with ordered nodes
x(1) x(2) ... x(n)

Each edge has a label over an alphabet A
x(j) = E(x(i),c) (edge x(i) → x(j) with label c)

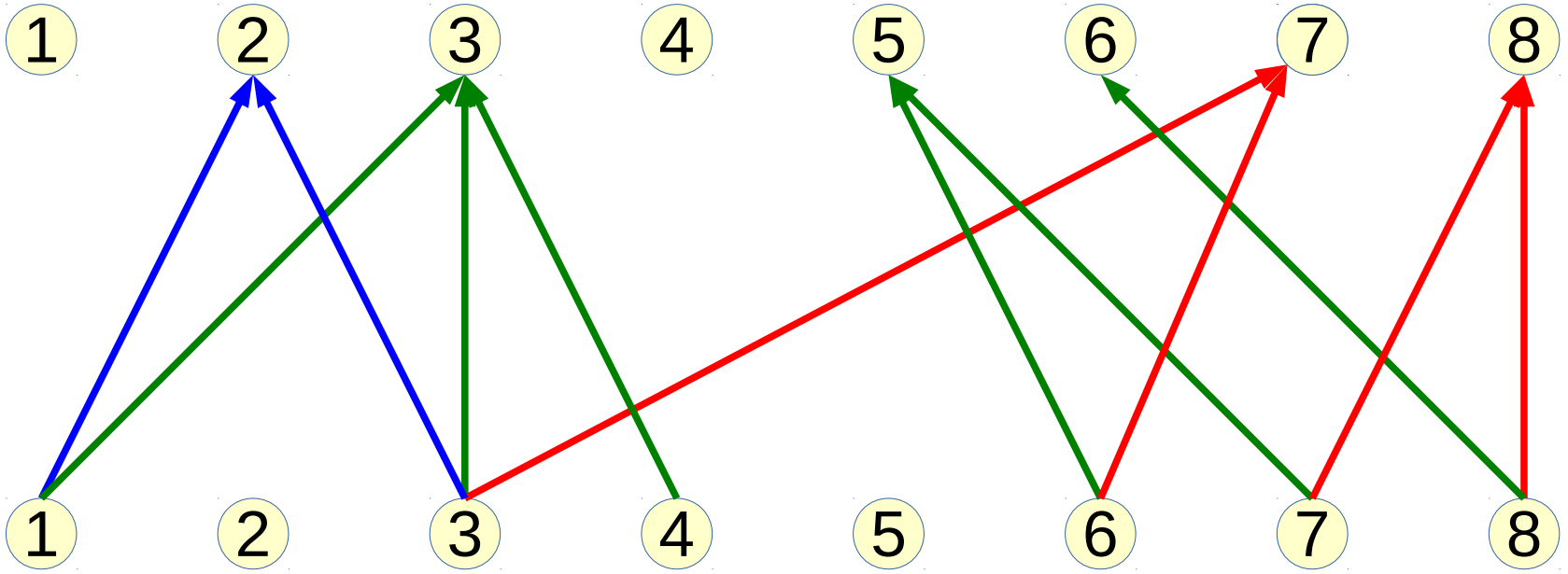
Edge ordering properties: ∀ i,j

a<b ⇒ E(x(i),a) < E(x(j), b)

x(i) < x(j) ⇒ E(x(i),c) ≤ E(x(j),c)

A colorful 8-node Wheeler graph

(Nodes are replicated as sources and sinks)

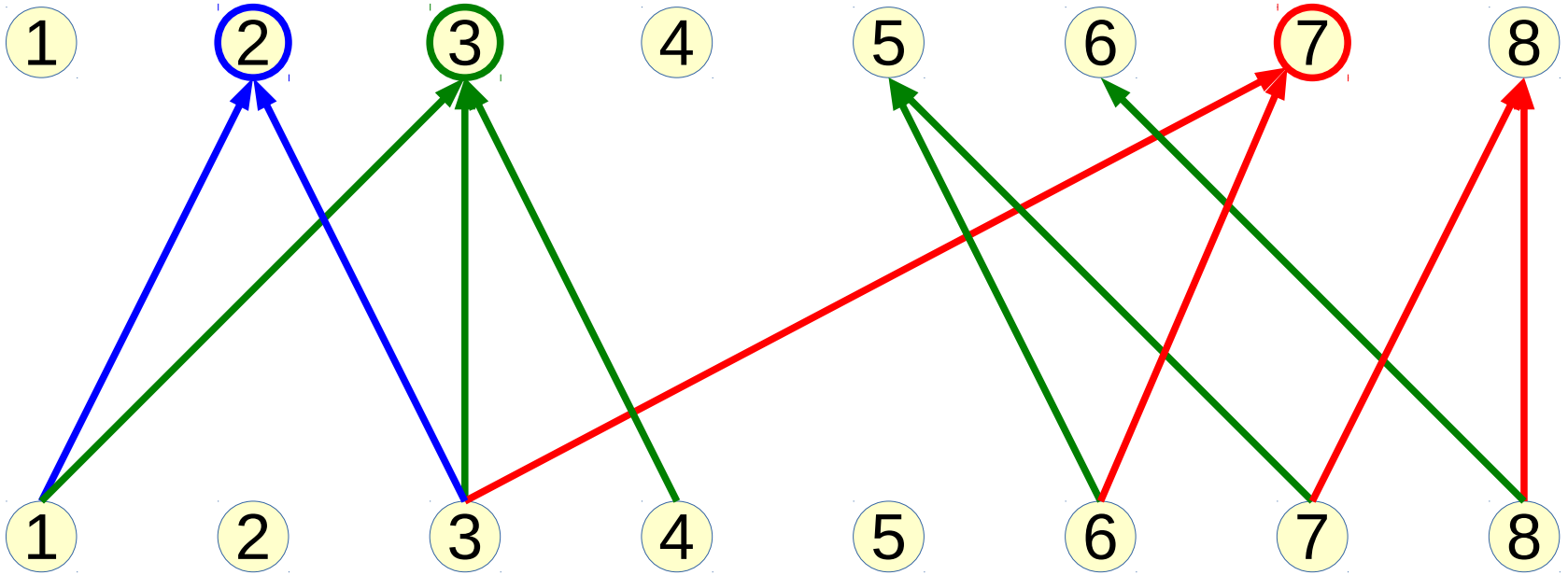


labels: bg bgr g gr gr gr

Succinct representation of a Wheeler graph

[Bowe et al '12]

in-degree unary: `1 001 0001 1 001 01 001 001`



labels: `bg bgr g gr gr gr`

unary out-degree: `001 1 0001 01 1 001 001 001`

Starting nodes: b → 2 g → 3 r → 7

The standard representation for a graph with n nodes and m edges takes O(m log n) bits.

Because of their structure, Wheeler graphs can be represented in O(n+m) bits and still support constant time navigation.

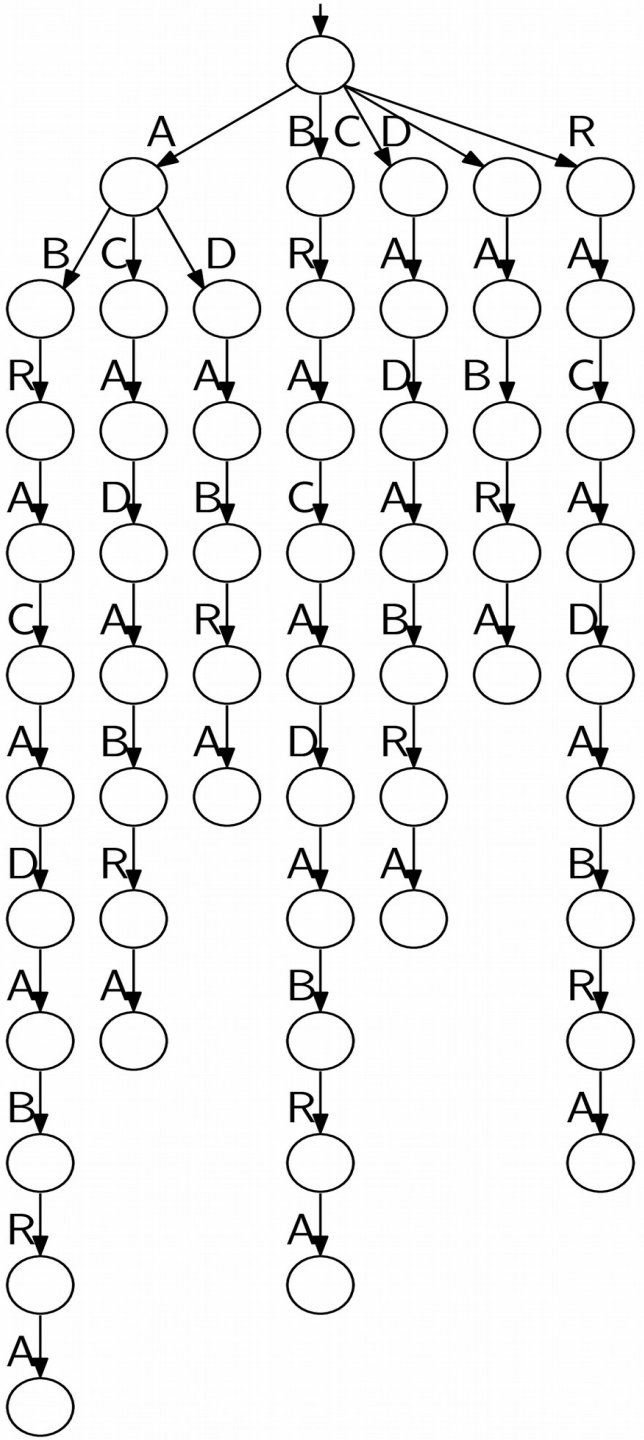
Many BWT-related succinct indices have a Wheeler Graph structure. This explains the "succinct" part

What about indexing?

Searching for substrings of ABRACADABRA

We can use a DFA

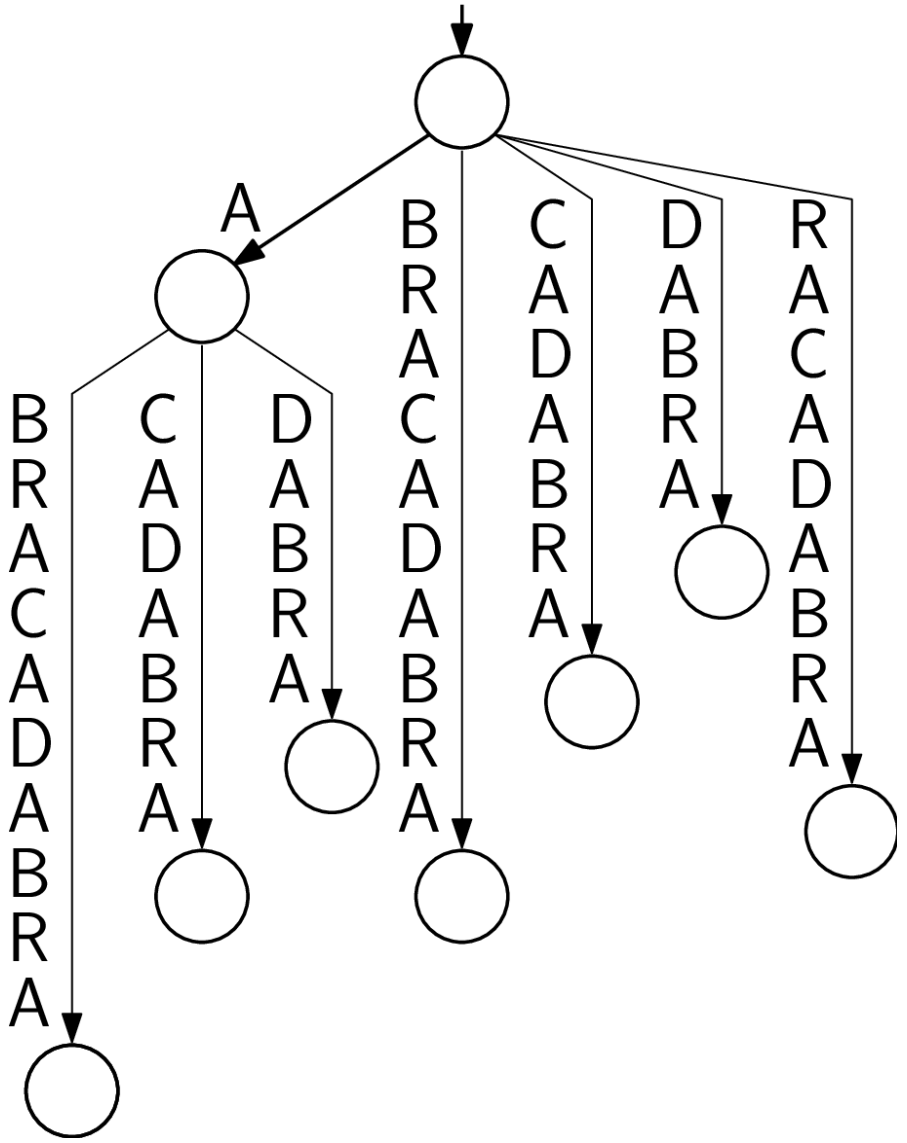
Simple but not space efficient



Searching for substrings of ABRACADABRA

Compacted Trie

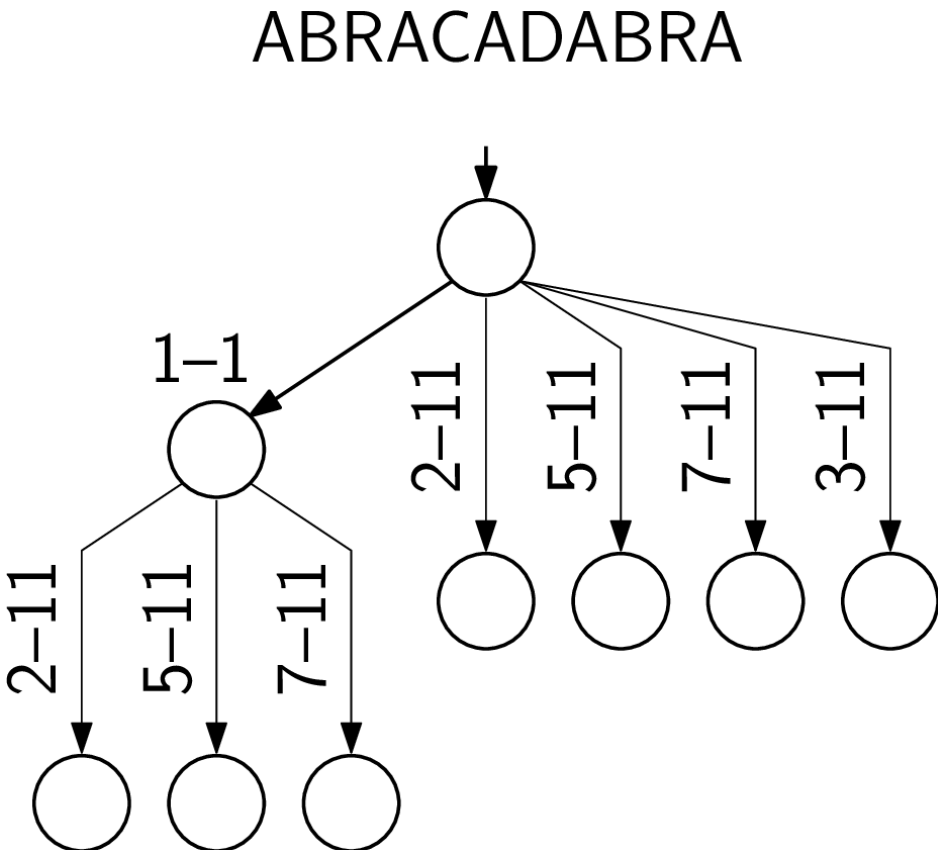
More space efficient!



Searching for substrings of ABRACADABRA

Suffix Tree

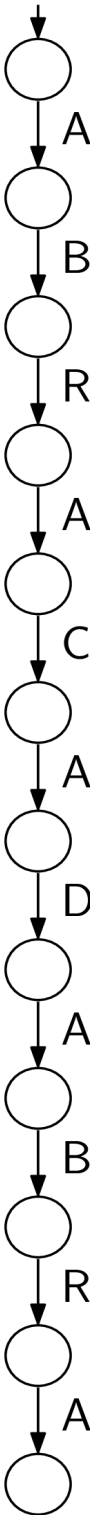
“theoretically” space efficient



Searching for substrings of ABRACADABRA

We can use a NFA!

Every state initial & final
Extremely space efficient!

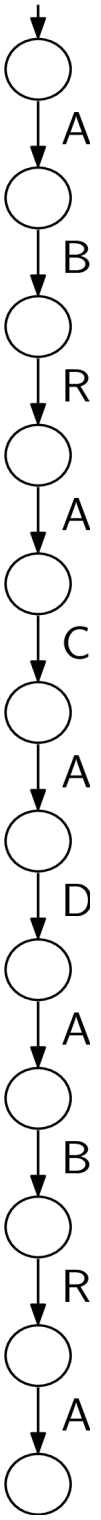


Searching for substrings of ABRACADABRA

We can use a NFA!

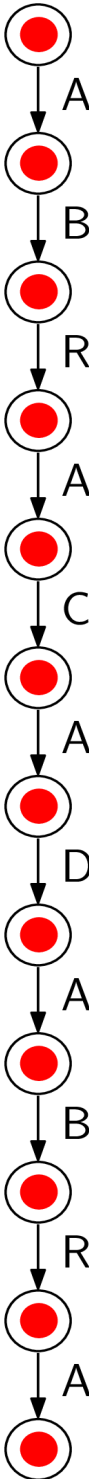
Every state initial & final
Extremely space efficient!

Searching is a headache



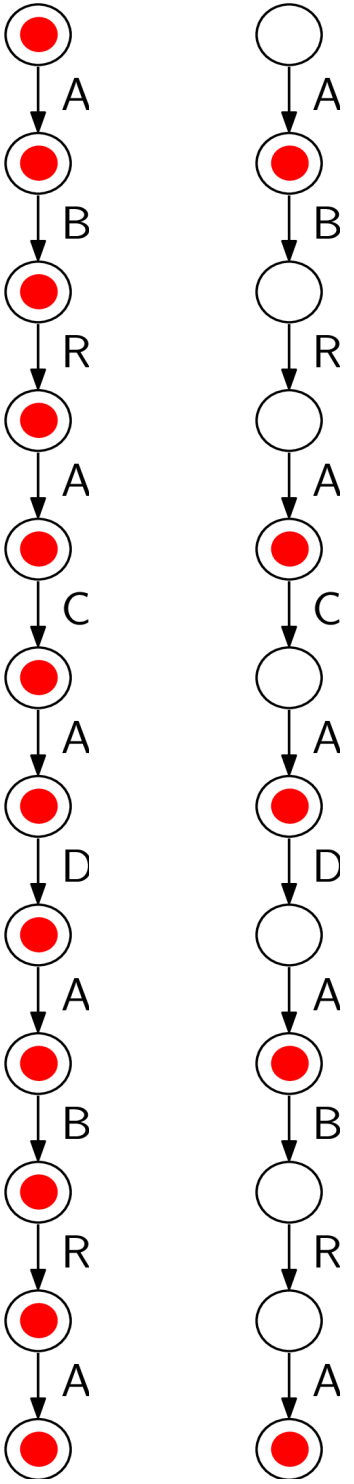
Searching for substrings of ABRACADABRA

Example:
Searching ABR



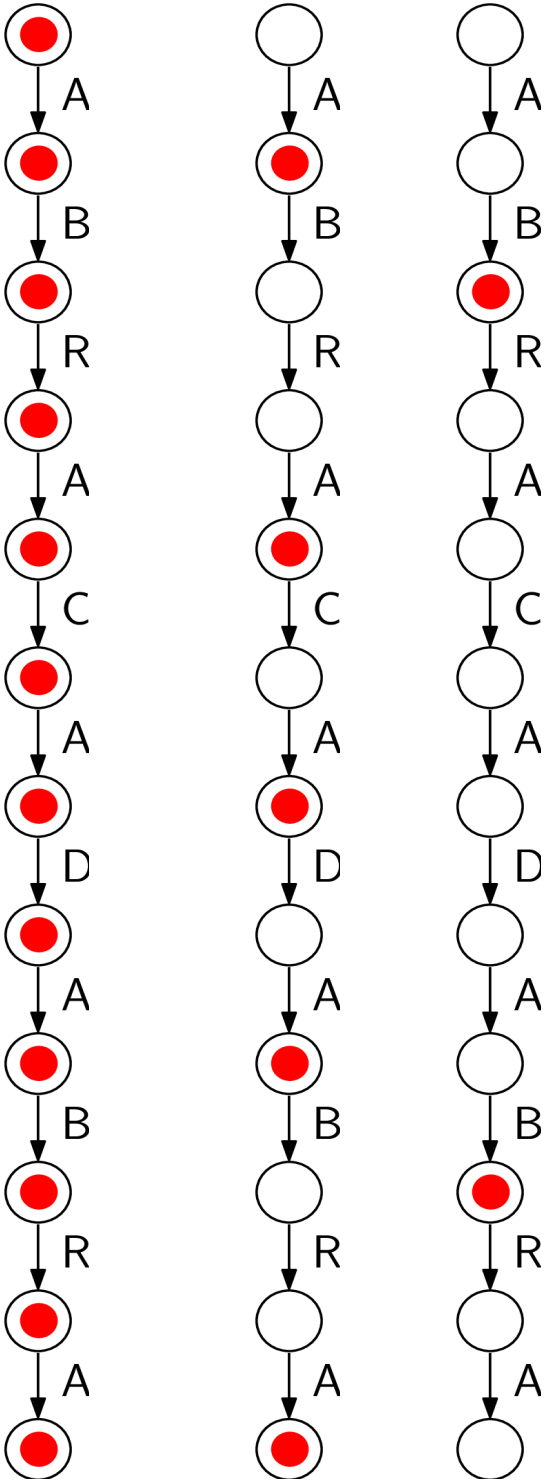
Searching for substrings of ABRACADABRA

Example:
Searching ABR



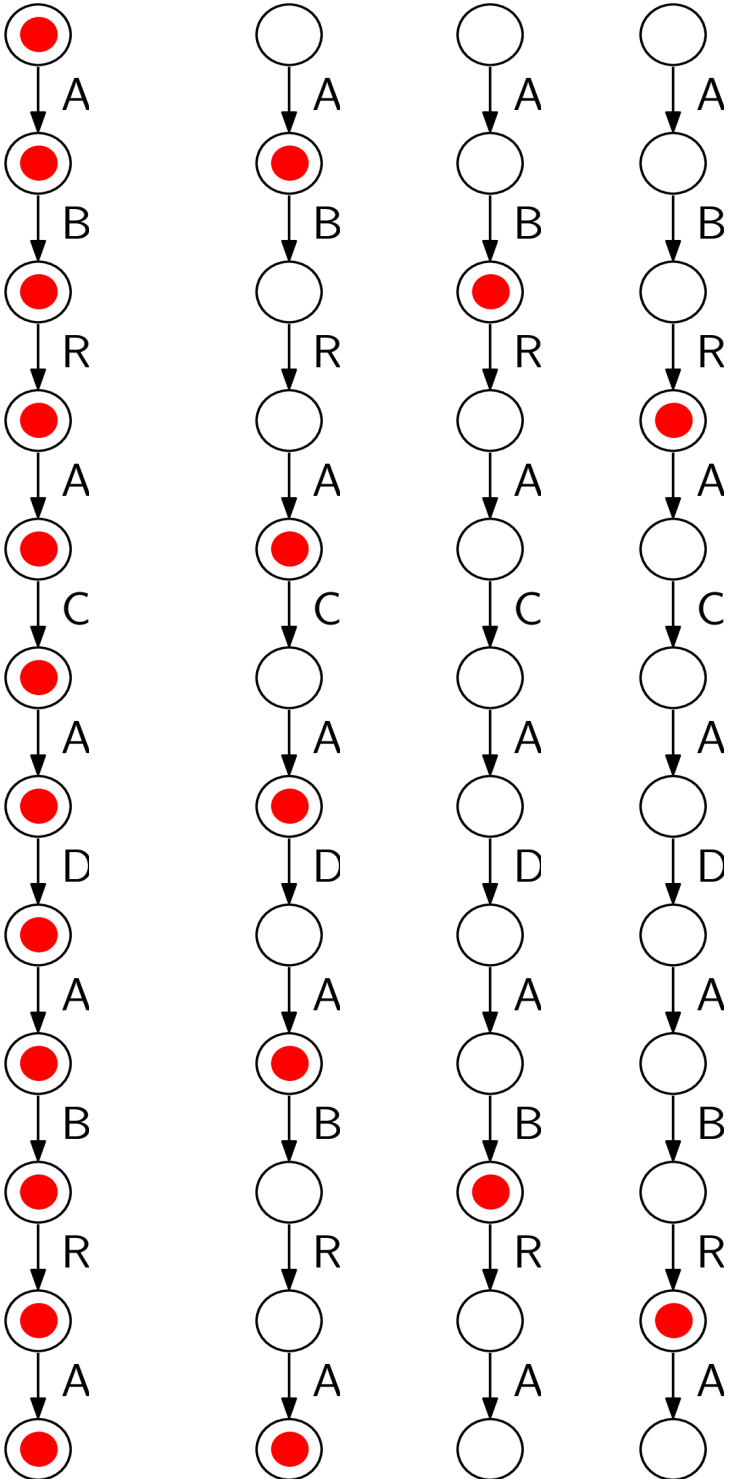
Searching for substrings of ABRACADABRA

Example:
Searching ABR



Searching for substrings of ABRACADABRA

Example:
Searching ABR

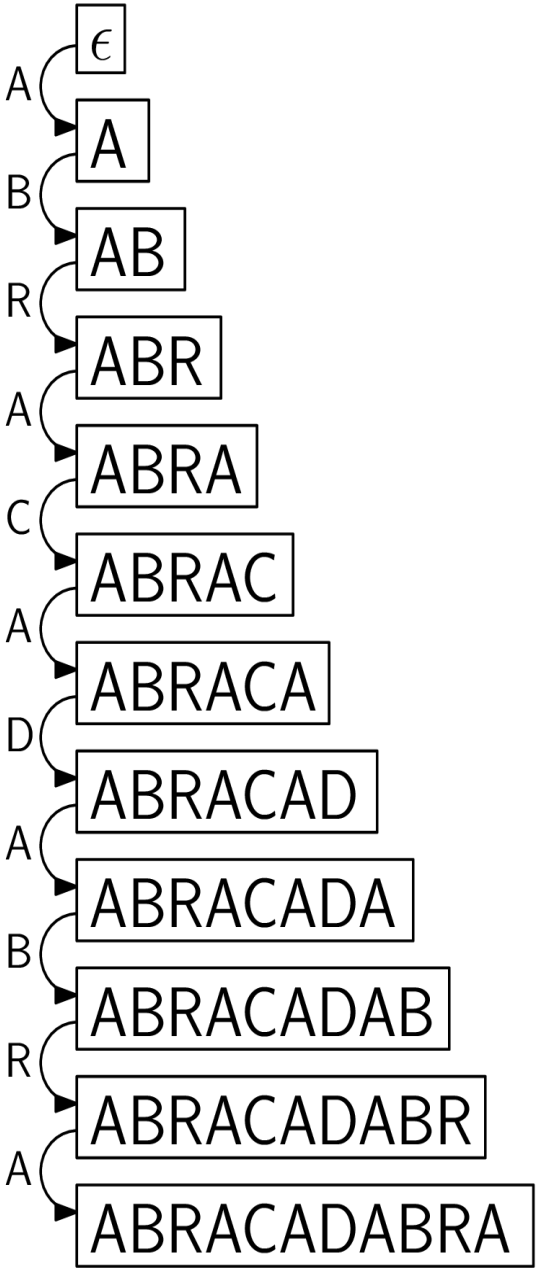


NFAs made simpler

NFA for
ABRACADABRA

NFAs made simpler

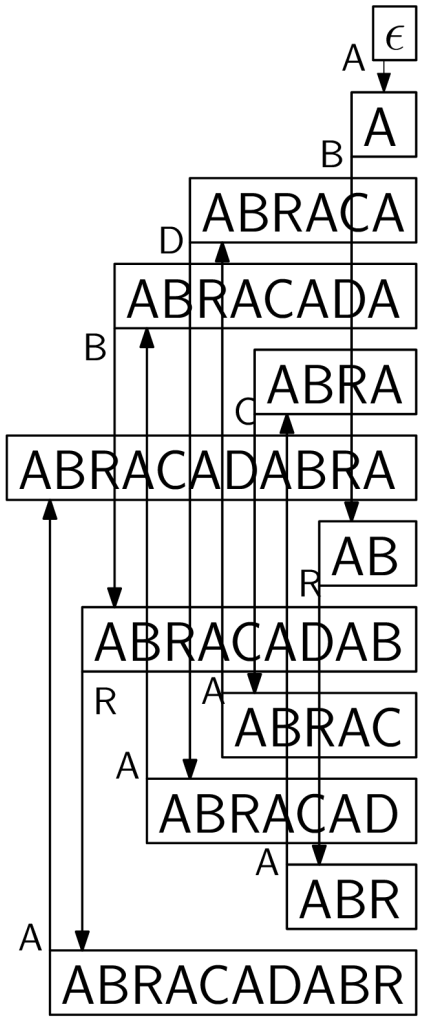
"Naturally" assign
a label to each state



NFAs made simpler

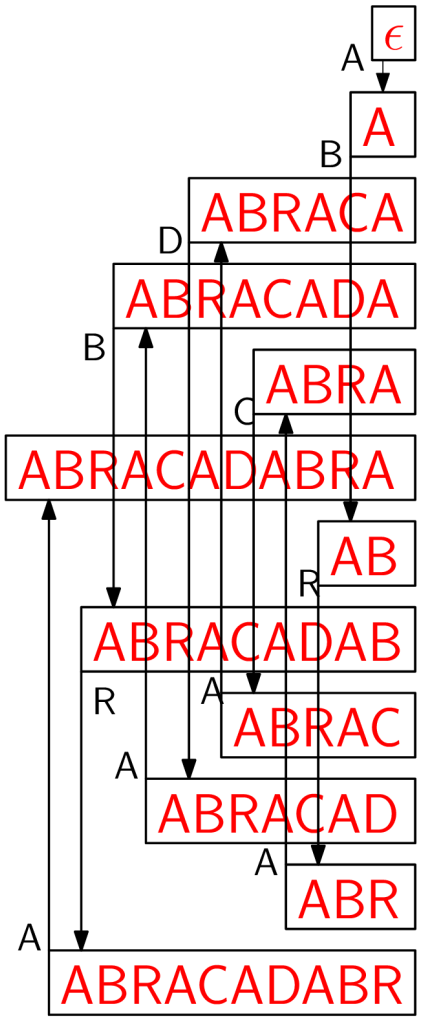
"Naturally" assign a label to each state

Arrange NFA states according to labels



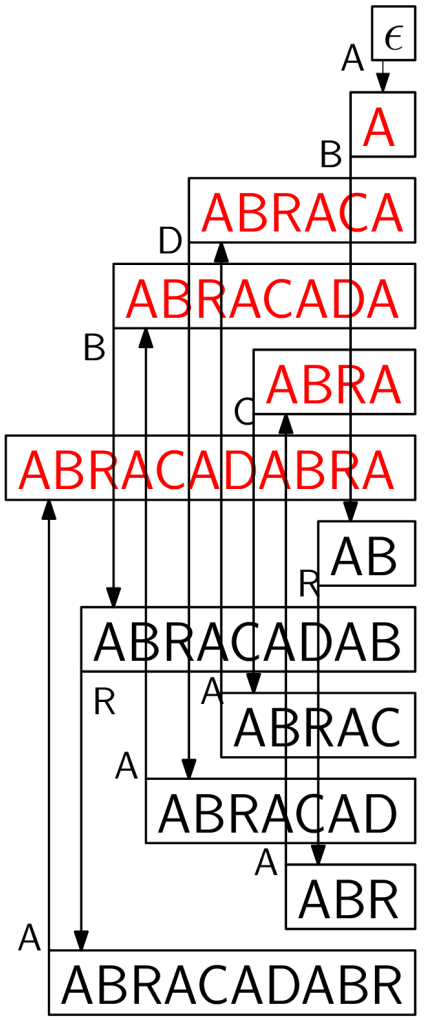
Searching in a sorted NFA

Example:
Searching ABR



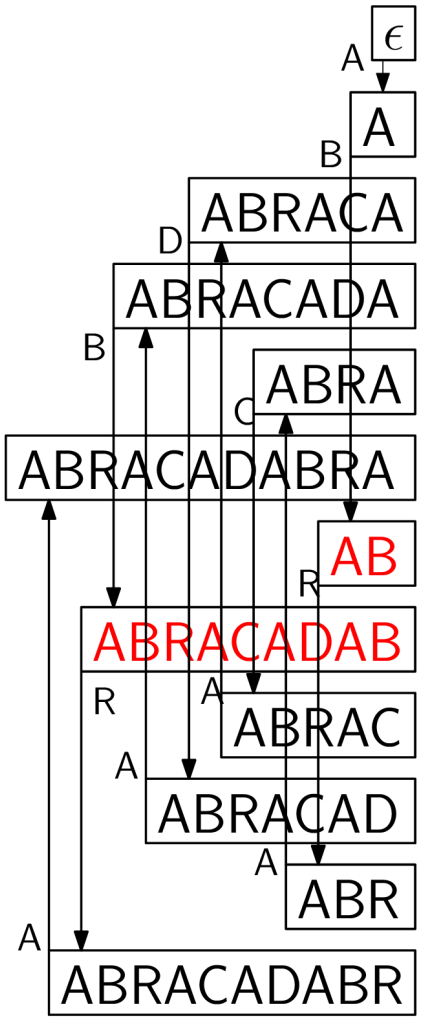
Searching in a sorted NFA

Example:
Searching ABR



Searching in a sorted NFA

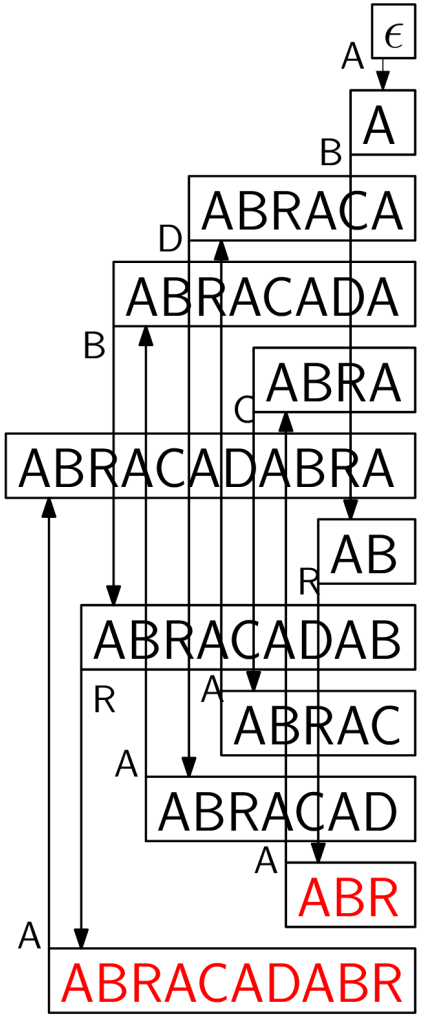
Example:
Searching ABR



Searching in a sorted NFA

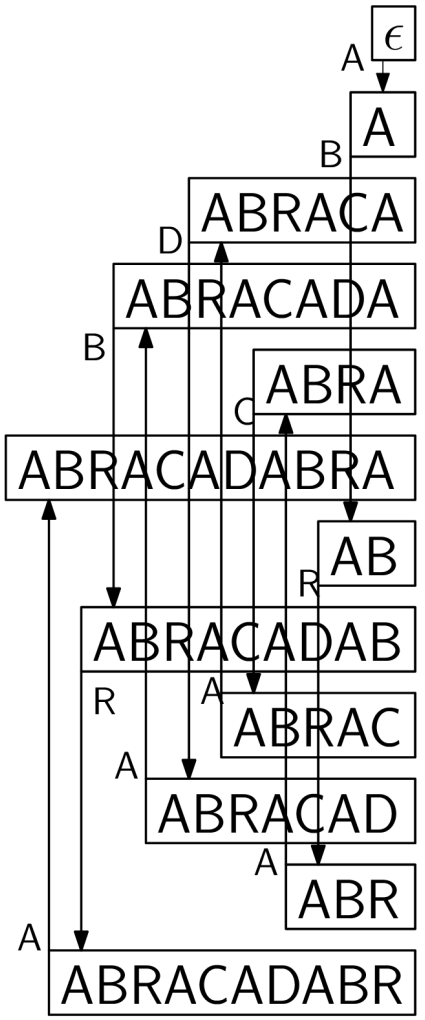
Example:
Searching ABR

Two occurrences found!

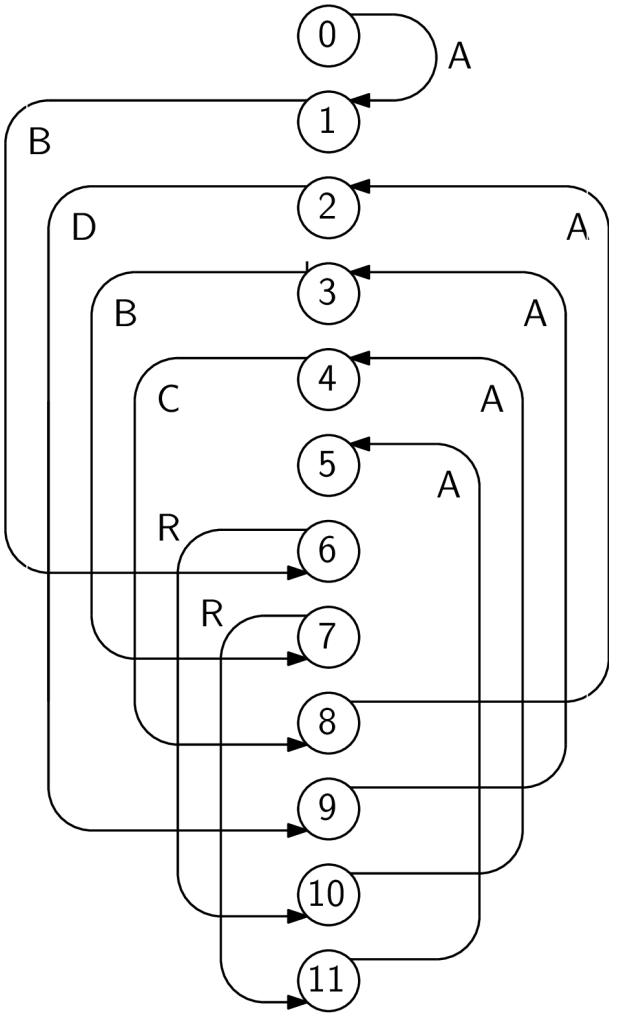


NFAs made simpler

Easy to search sorted NFA



NFAs made simpler

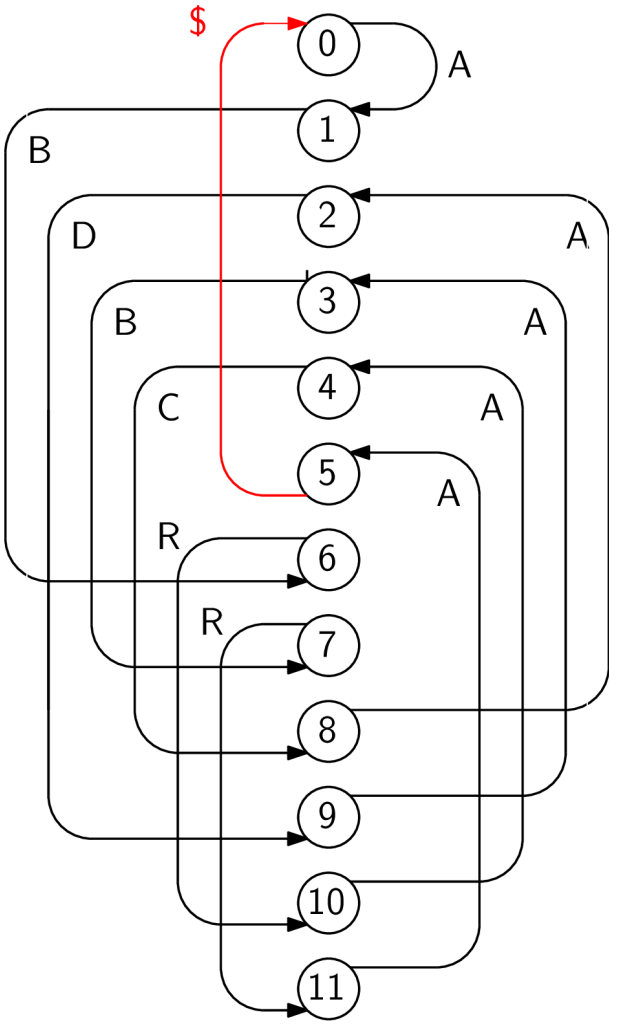


No need to store the state labels

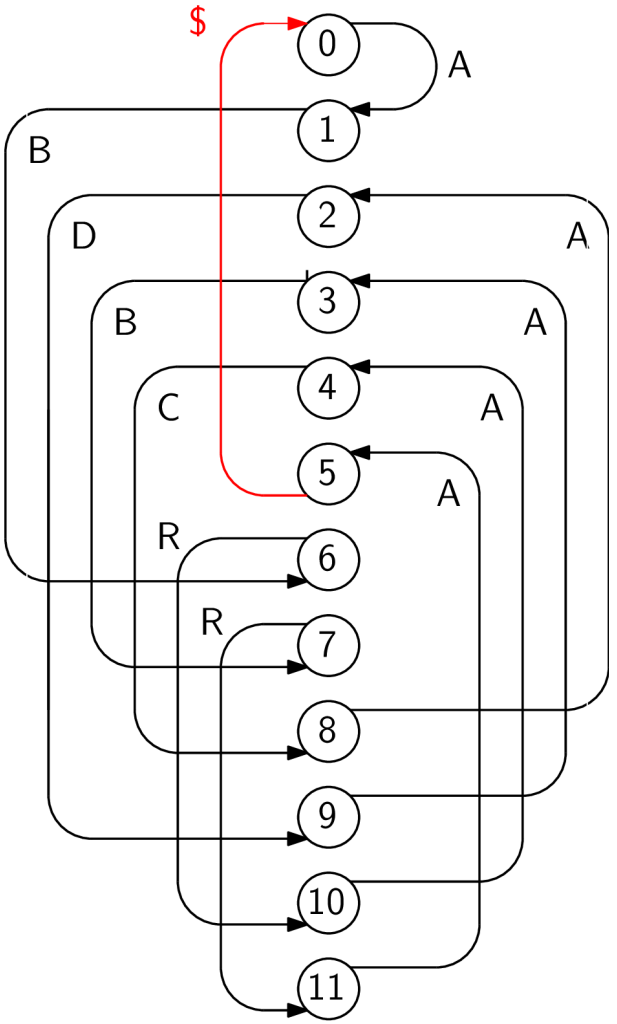
Wheeler Graph!

NFAs made simpler

Add one arc
for symmetry



NFAs made simpler



Add one arc
for symmetry

ABDBC$RRAAAA
is the BWT of
$(ABRACADABRA)^R$

Summing up

By rearranging the NFA states:

- searching becomes easier
- the NFA becomes an easy to navigate Wheeler graph

The BWT can be seen as a succinct representation of the reordered NFA

This trick works for other search problems: many BWT variants can be seen as Wheeler graphs obtained by reordering the states of an appropriate NFA.

2019

UK is leaving EU (maybe)

In 25 years some important issues have been solved, but they usually lead to more difficult open questions

Repetitive collections

- Today we are interested in compressing and indexing large collections of repetitive data
- The plain BWT index is not suitable for repetitive data for the cost of the SA samples
- Using a, so far undetected, property of the BWT [Gagie et al, 18] made a significant progress introducing the r-index. Can we do better?

Space efficient construction

- In 1994 suffix sorting algorithms were either memory hogs or slow for some inputs
- Many improvements in 25 years: we now have O(n) time O(n log σ) space algorithms, but still space for improvements
- External memory BWT/LCP computation for inputs larger than the available RAM (see [WABI 18] for some preliminary results)

BWT as a compressor

- Most of the mainstream compressors released in the last 20 years are based on LZ77 parsing, eg. LZMA, Snappy, Brotli, Zstd
- LZ77 has more “free parameters” and can offer a wide range of compression/speed trade-offs
- In BWT compression we cannot easily trade compression for speed

Sample results (1)

Size	Ratio %	C.MB/s	D.MB/s	Compressor (Binary 42% + Text 58%) Silesia.tar
48616057	22.9	**1.07**	**77.11**	**LzTurbo 49**
48758739	23.0	**2.47**	**81.17**	**lzma 9**
49517150	23.4	0.46	**336.19**	**brotli 11d29**
50861542	24.0	1.68	269.97	lzham 4
51720632	24.4	1.42	**1239.95**	**LzTurbo 39**
52715921	24.9	2.03	602.56	zstd 22
54596837	25.8	**11.80**	38.94	**bzip2**
58008992	27.4	7.96	853.20	zstd 15
59273940	28.0	**59.48**	**1293.41**	**LzTurbo 32**
59581397	28.1	33.48	416.81	brotli 5
60411647	28.5	45.64	798.97	zstd 9
60813803	28.7	1.60	**2002.86**	**LzTurbo 29**
64141404	30.3	**162.02**	1372.34	**LzTurbo 31**
64191258	30.3	65.28	416.81	**brotli 4**
64711652	30.5	0.22	325.27	zopfli
67624724	31.9	62.86	692.87	lzfse
67647204	31.9	9.99	316.72	zlib 9

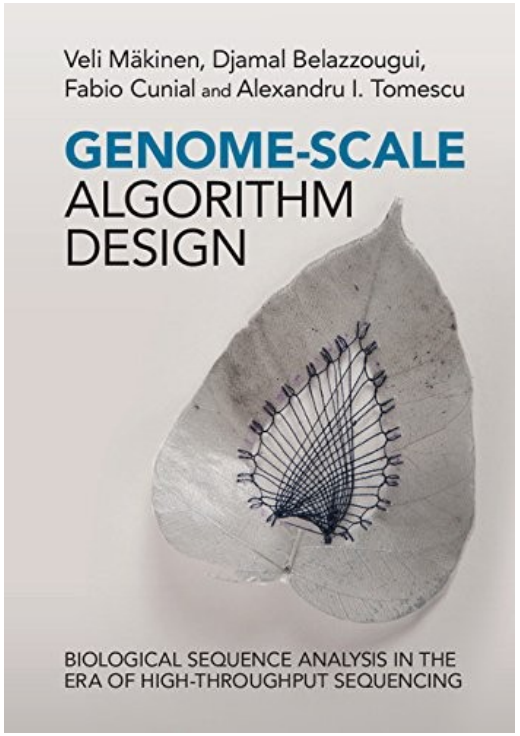
Sample results (2)

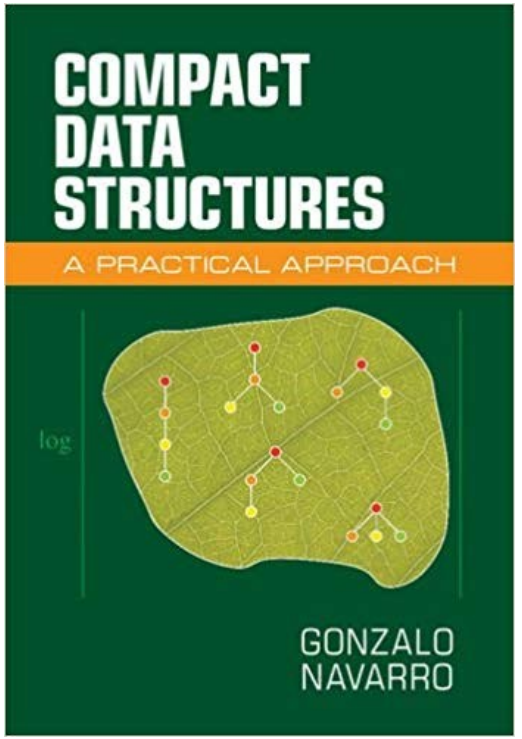
Size	Ratio %	C.MB/s	D.MB/s	Compressor	Text log: NASA_access_log
11355945	5.5	**0.86**	**320.68**	**LzTurbo 49**	
11907661	5.8	**0.99**	**2502.71**	**LzTurbo 39**	
11960483	5.8	**10.13**	67.81	**bzip2**	
12236072	6.0	0.51	1022.47	brotli 11d29	
12617026	6.1	1.36	1348.32	zstd 22	
13598062	6.6	2.68	265.69	lzma 9	
13651218	6.7	1.33	880.25	lzham 4	
14661031	7.1	8.67	1819.99	zstd 15	
15041556	7.3	1.13	**3732.63**	**LzTurbo 29**	
16665926	8.1	**78.89**	1245.90	brotli 5	
17387746	8.5	**117.98**	1375.73	**zstd 9**	
18279979	8.9	**187.64**	2186.17	**LzTurbo 32**	
18654669	9.1	173.25	1227.89	brotli 4	
19085875	9.3	1.50	3527.36	lizard 49	
19545036	9.5	32.75	651.55	zlib 9	

Conclusions

- Block sorting was indeed an unusual compressor
- The Burrows-Wheeler Transform taught us that compression should be (almost) free
- Even after 25 years new properties are discovered and translated to efficient algorithms: r-index, tunneling, Wheeler graphs, …
- There is no lack of challenging problems mainly coming from NGS analysis

References





T. Gagie, G. M, J. Sirén,
Wheeler graphs: A framework for BWT-based data Structures,
Theoretical Computer Science,
Vol 698 (2017)

Thank You