# Introduction to Wheeler Graphs/Automata

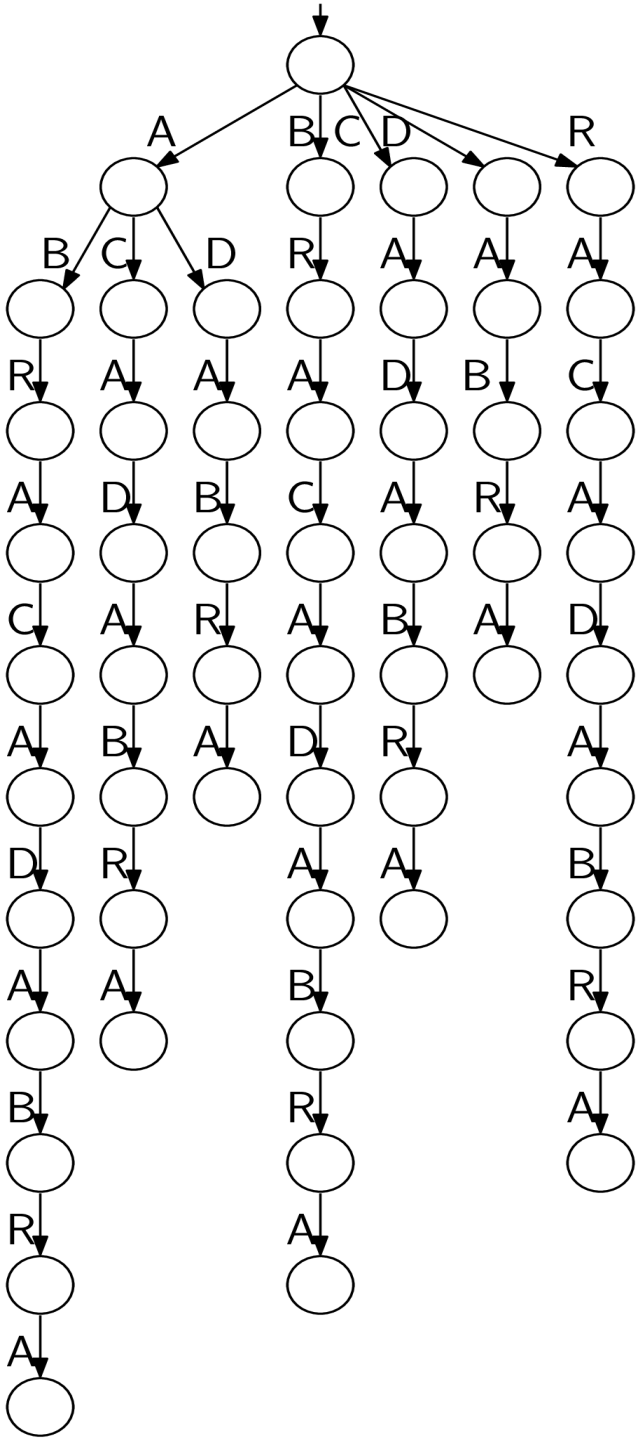
# BWT as a compressed index

- Starting from the year 2000 several researchers observed that the BWT is strictly related to the Suffix Array
- That intuition is somewhat misleading...
- Having seen 20+ years of research on the topic, we can offer an alternative view
- Let’s start with a very basic search problem...

# Searching for substrings of ABRACADABRA

We can use a DFA

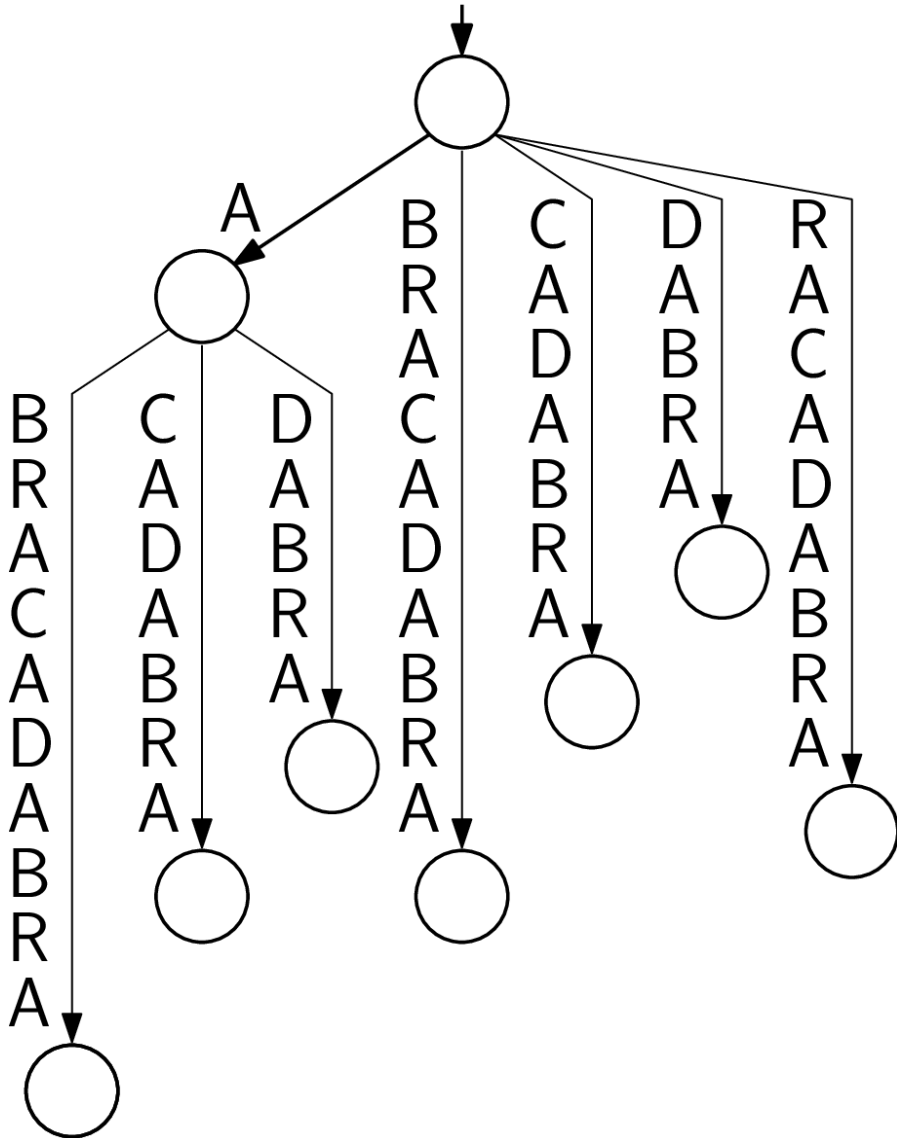
Simple but not space efficient



# Searching for substrings of ABRACADABRA

Compacted Trie

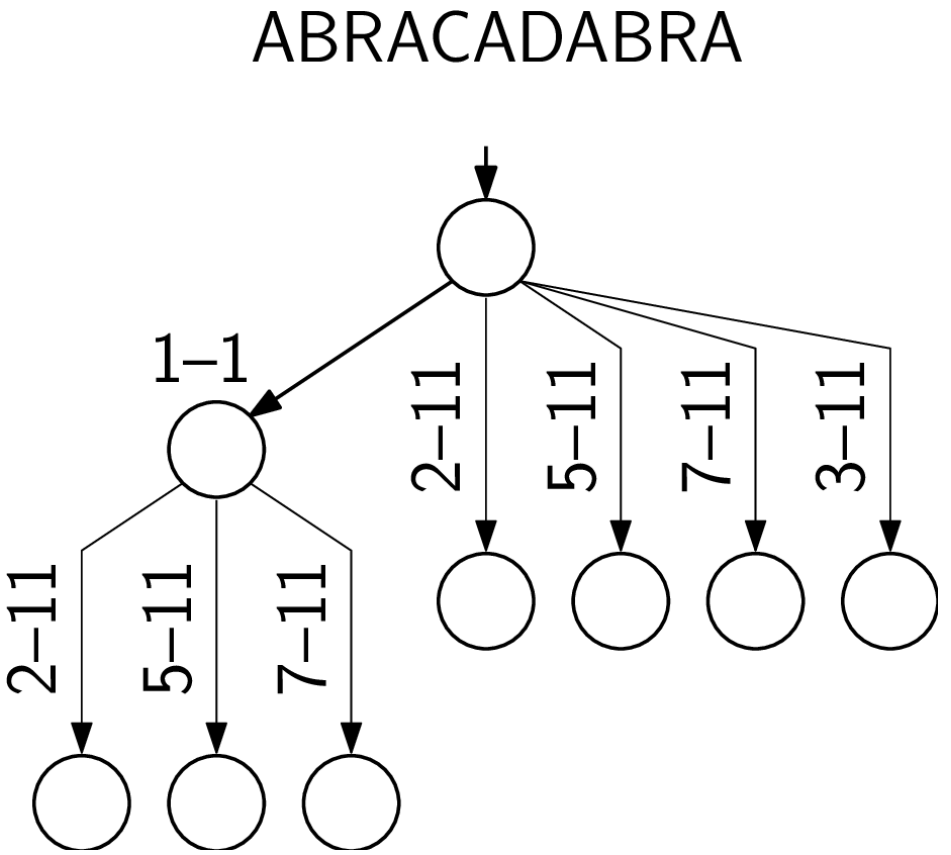
More space efficient!



# Searching for substrings of ABRACADABRA

Suffix Tree

“theoretically” space efficient

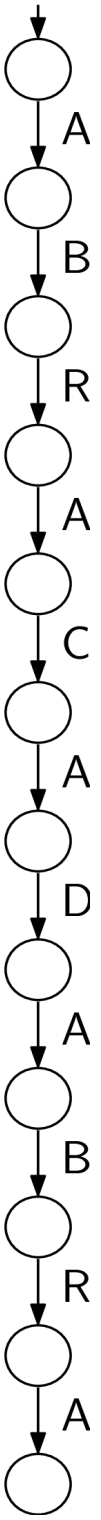


# Searching for substrings of ABRACADABRA

We can use a NFA!

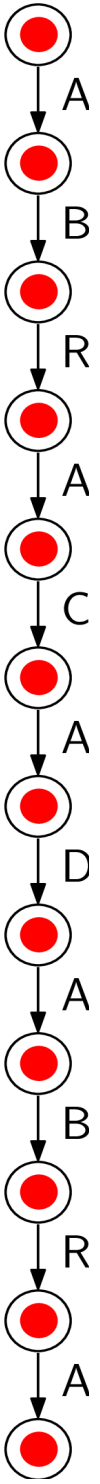
Every state initial & final
Extremely space efficient!

Searching is a headache



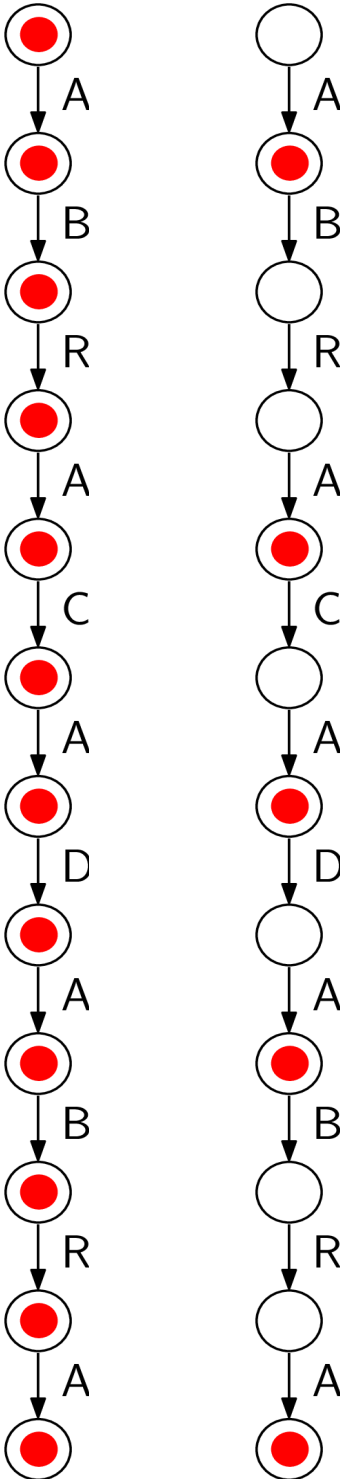
# Searching for substrings of ABRACADABRA

Example:
Searching ABR



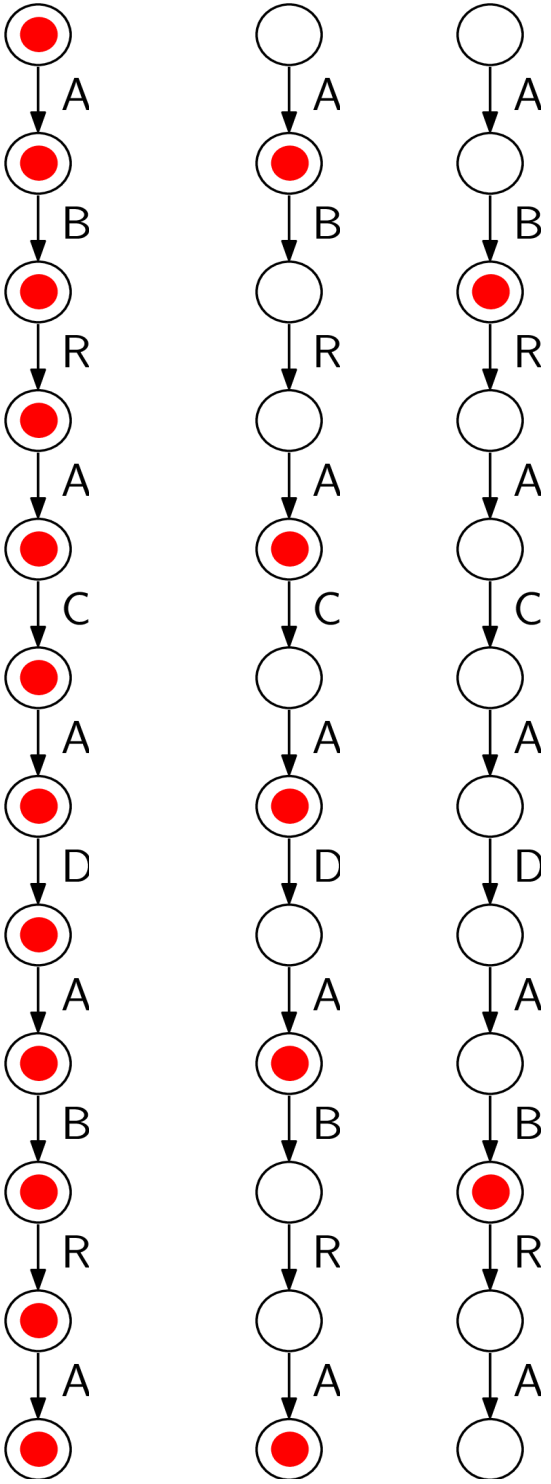
# Searching for substrings of ABRACADABRA

Example:
Searching ABR



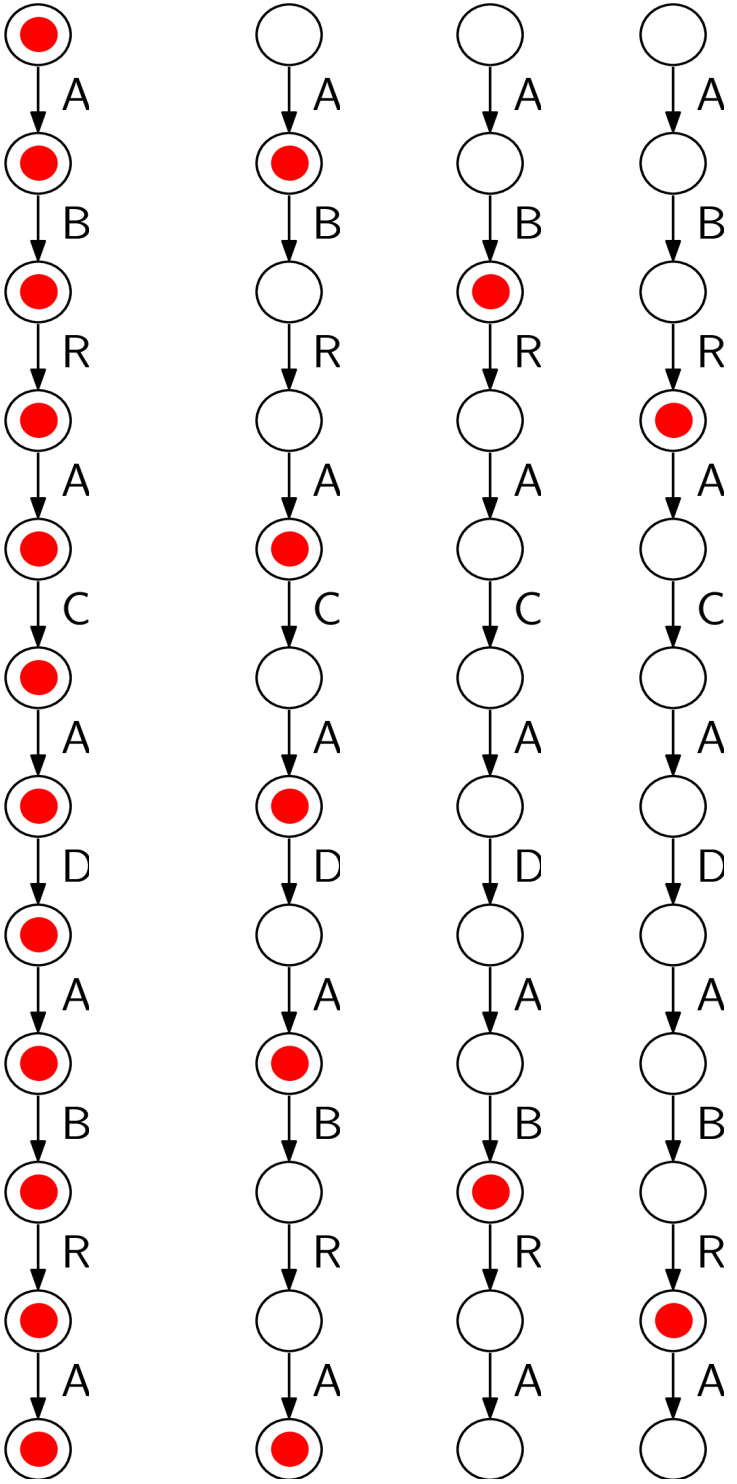
# Searching for substrings of ABRACADABRA

Example:
Searching ABR



# Searching for substrings of ABRACADABRA

Example:
Searching ABR

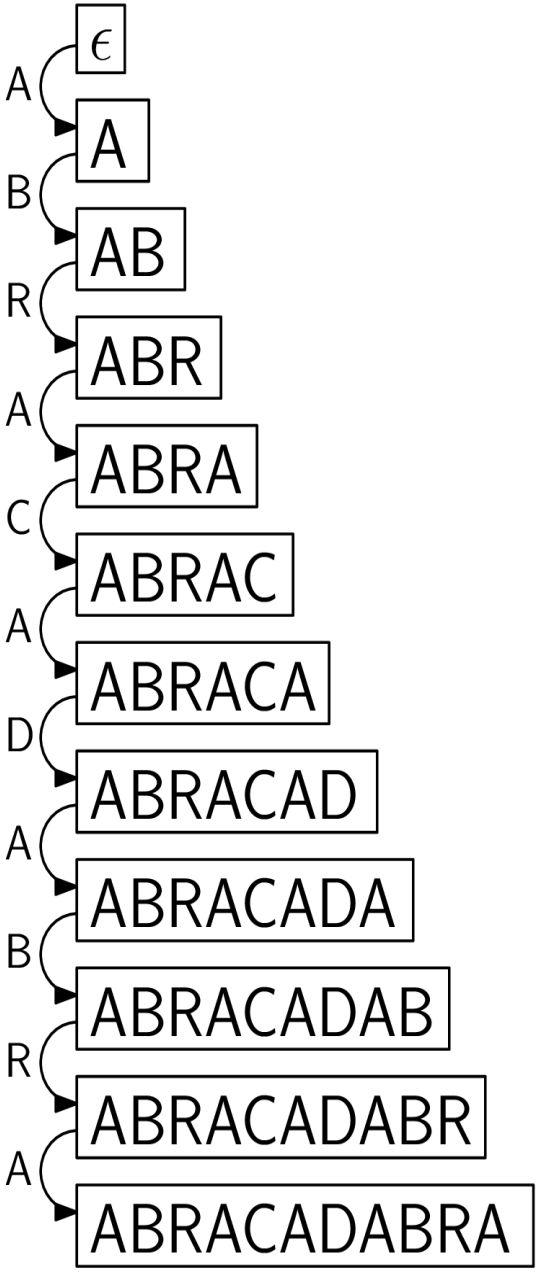


# NFAs made simpler

NFA for
ABRACADABRA

# NFAs made simpler

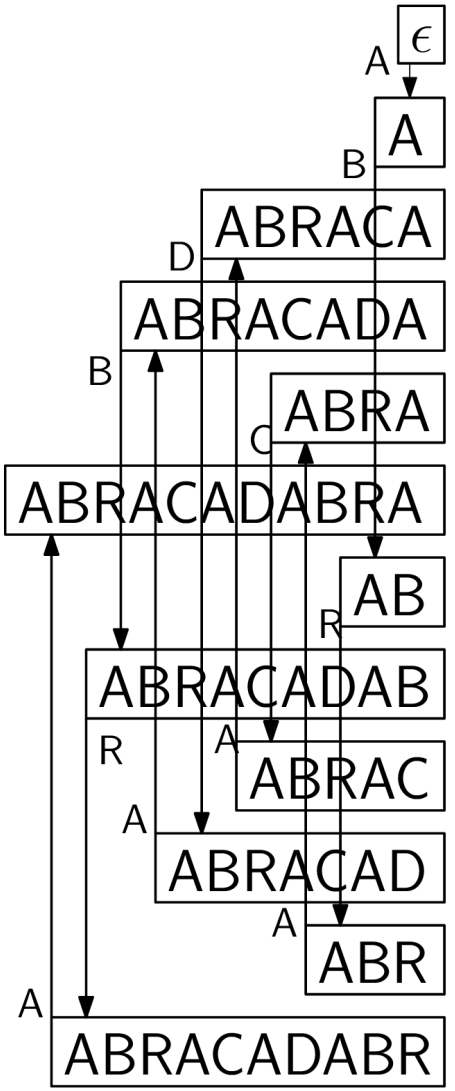
"Naturally" assign
a label to each state



# NFAs made simpler

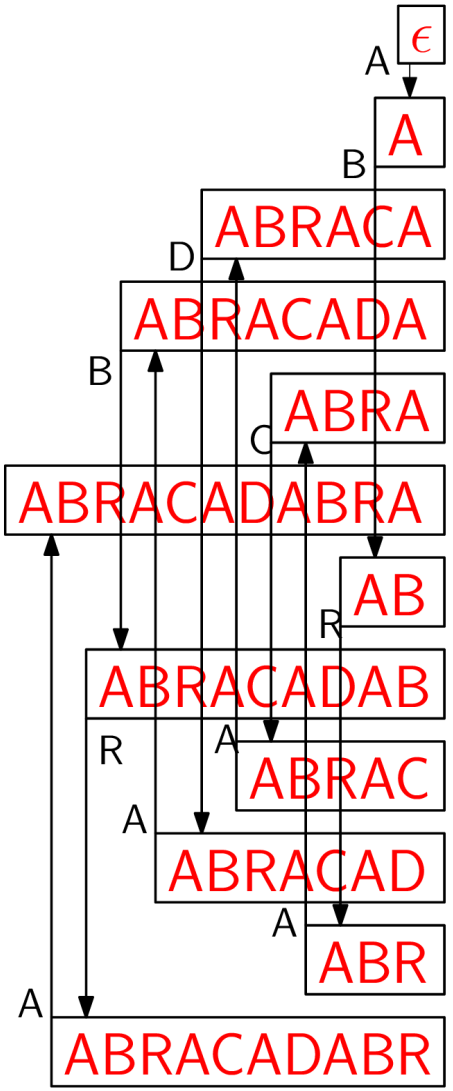
"Naturally" assign a label to each state

Arrange NFA states according to labels



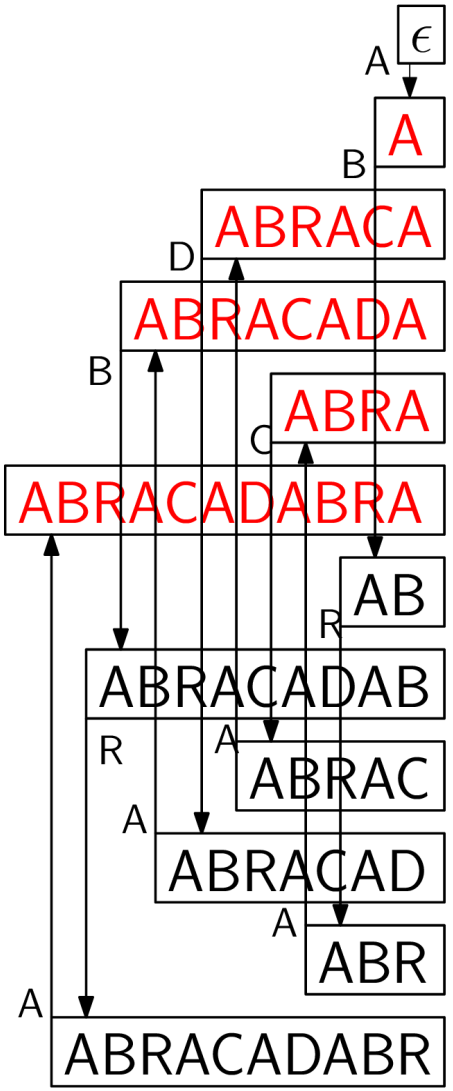
# Searching in a sorted NFA

Example:
Searching ABR



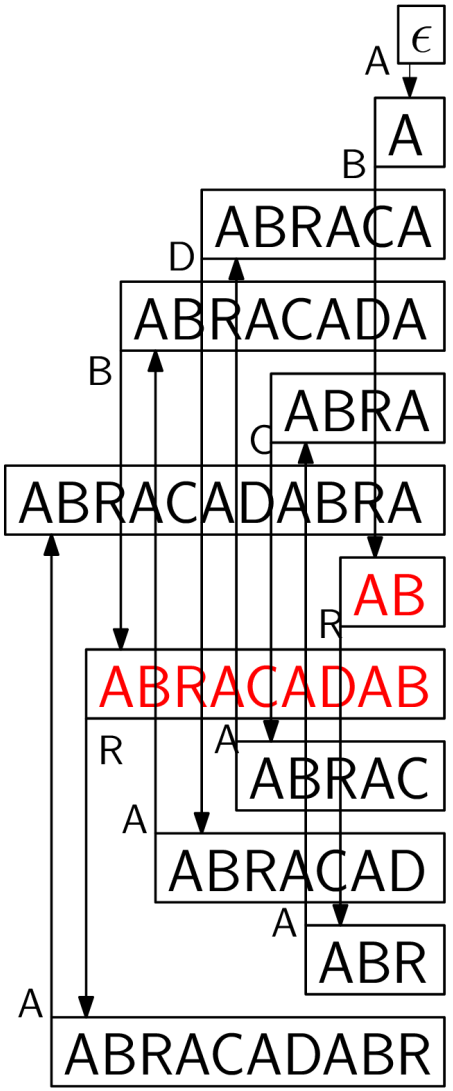
# Searching in a sorted NFA

Example:
Searching ABR



# Searching in a sorted NFA

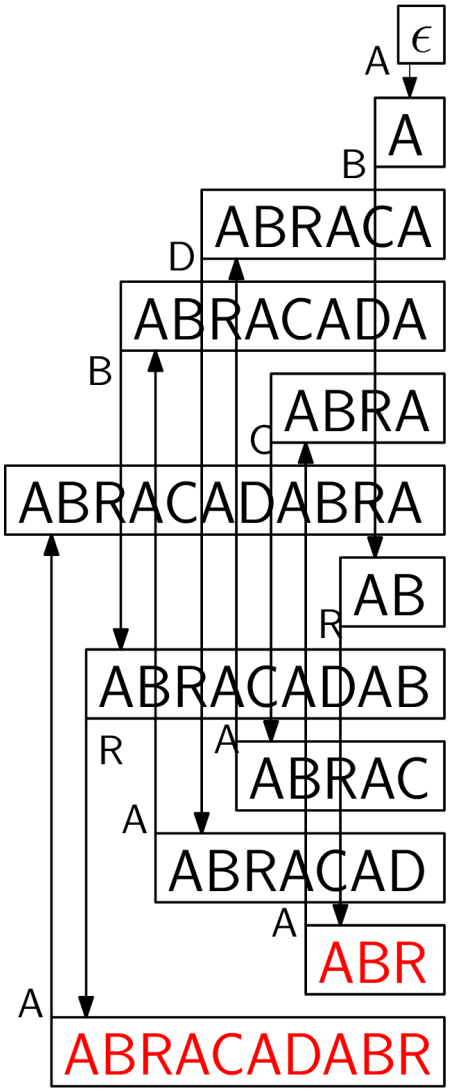
Example:
Searching ABR



# Searching in a sorted NFA

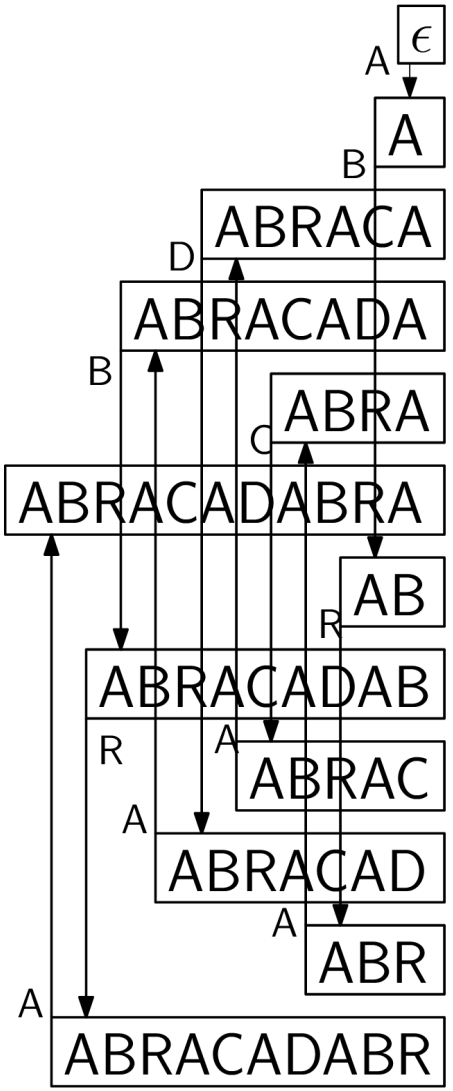
Example:
Searching ABR

Two occurrences found!



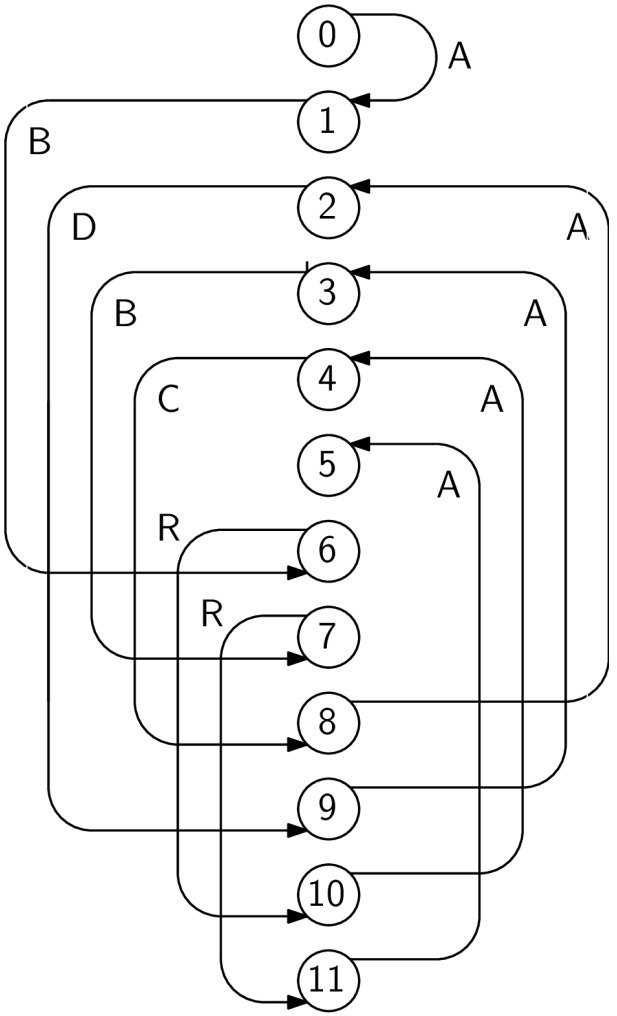
# NFAs made simpler

Easy to search permuted NFA

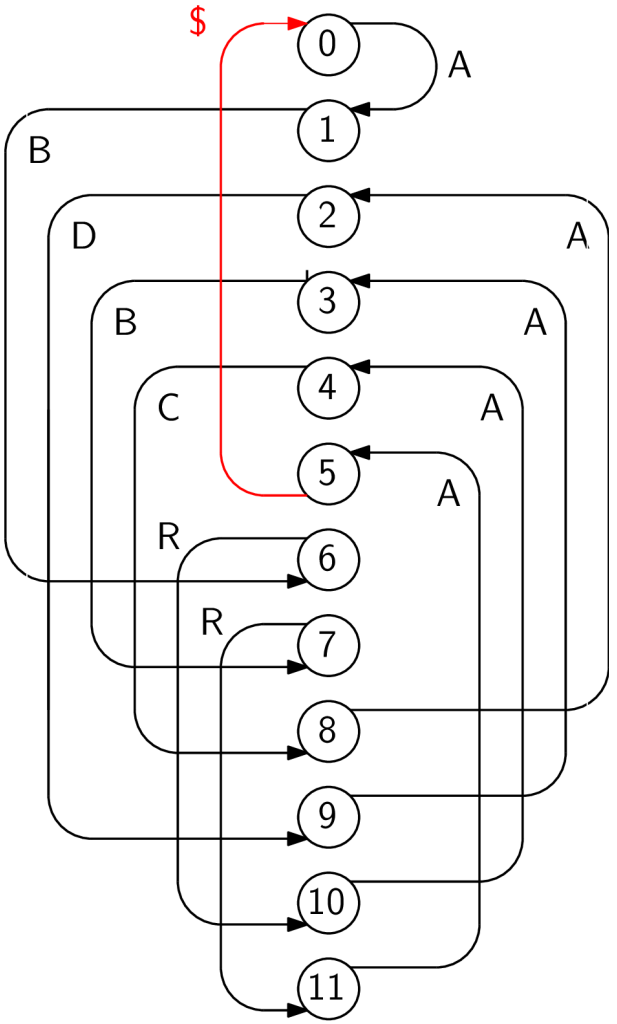


# NFAs made simpler

No need to store the state labels



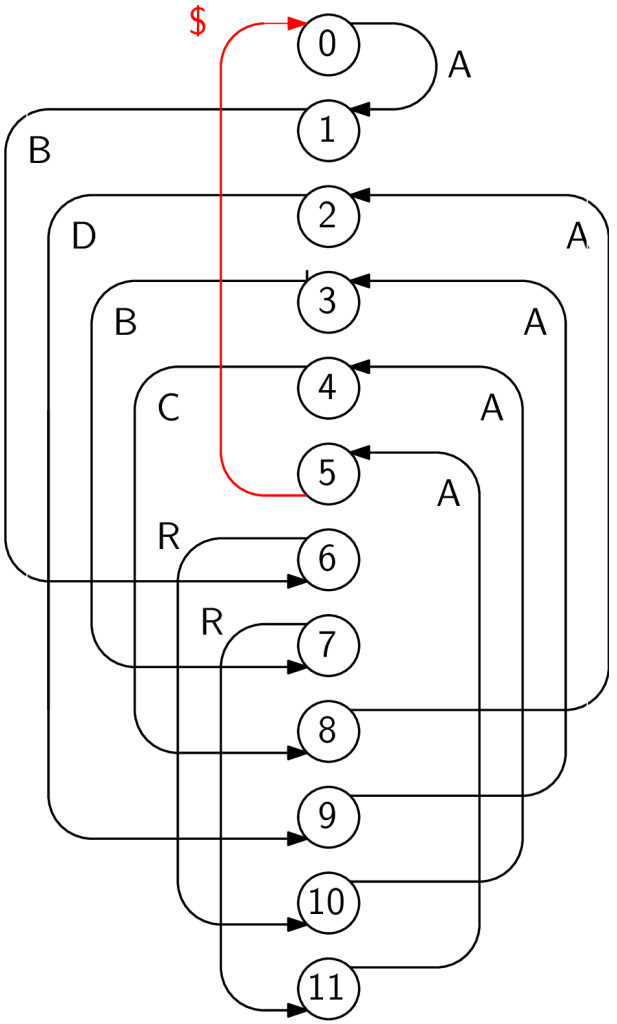
# NFAs made simpler




No need to store the state labels

Add one arc for symmetry

# NFAs made simpler




It suffices to store the edge labels:
ABDBC$RRAAAA

which is the BWT of
$(ABRACADABRA)^R$

# Summing up

The BWT can be seen as:

- a tool to achieve order-k compression using an order-0 encoder (well known)
- a stripped down version of the suffix array (earlier today)
- a tool to compactly represent NFAs in a way which is “search friendly” (now)

# Generalization

- Over the years we got BWT variants for searching and navigating string collections, tries, De Bruijn graphs, alignments, ...
- Most of them are equivalent to a properly reordered NFA
- The graph representing such NFAs have the same egde ordering properties of the BWT

# BWT-NFA representation:

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

- Each node has in-degree 1 and out-degree 1.

- 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: $\forall$ i,j

$a<b \Rightarrow E(x(i),a) < E(x(j), b)$

$x(i) < x(j) \Rightarrow E(x(i),c) \leq E(x(j),c)$

# Wheeler Graph/Automata

- Directed labeled graph with n ordered nodes
  x(1)  x(2) ... x(n)
- Nodes with in-degree 0 are the smallest
- Each edge has a label over an alphabet A
  x(j) = E(x(i),c)  (edge x(i) → x(j) with label c)

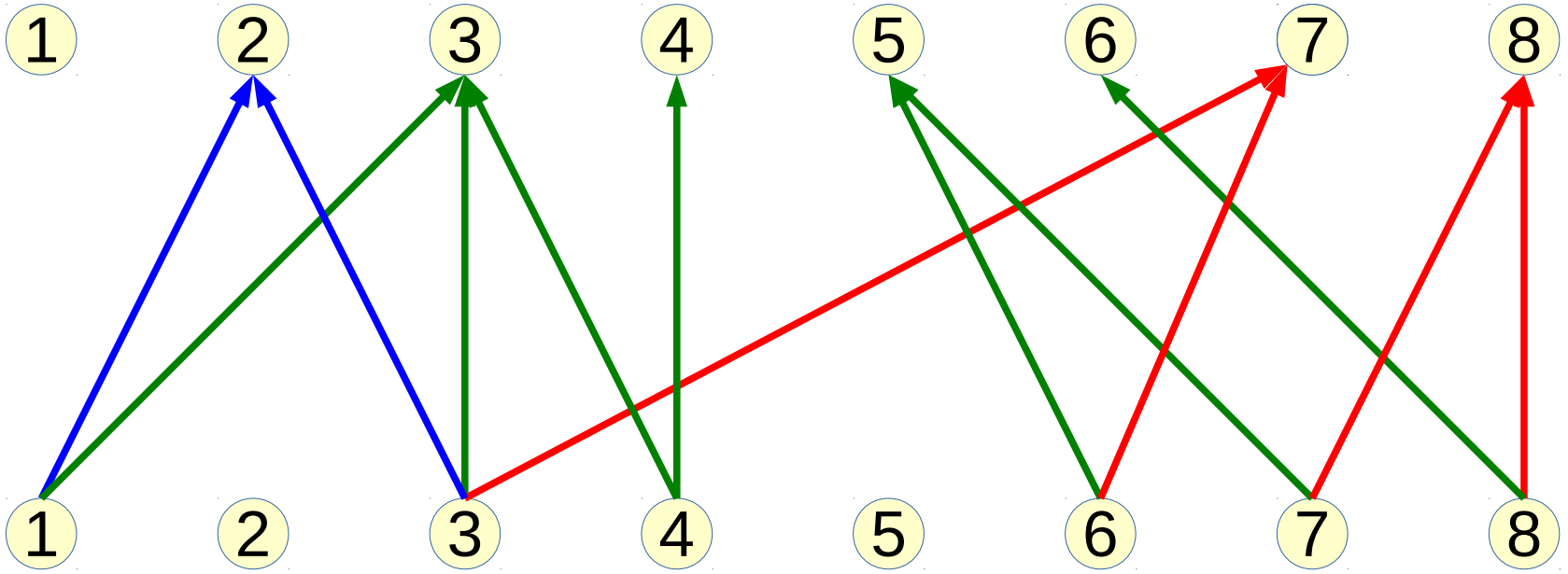
Edge ordering properties: $\forall$ i,j

$a<b \Rightarrow E(x(i),a) < E(x(j), b)$

$x(i) < x(j) \Rightarrow E(x(i),c) \leq E(x(j),c)$

# A colorful 8-node Wheeler graph

(Nodes are replicated as sources and sinks)

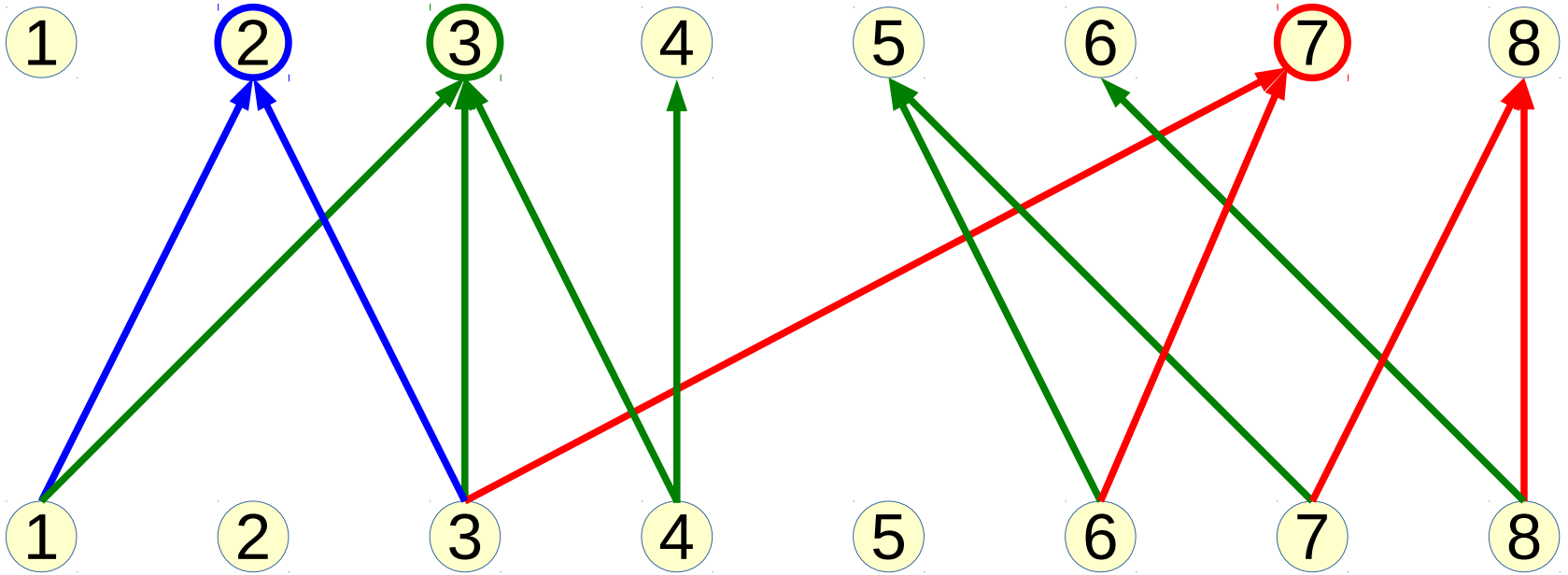



labels: bg bgr gg gr gr gr

# Succinct representation of a Wheeler graph

[Bowe et al '12]

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




labels: `bg bgr gg gr gr gr`

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

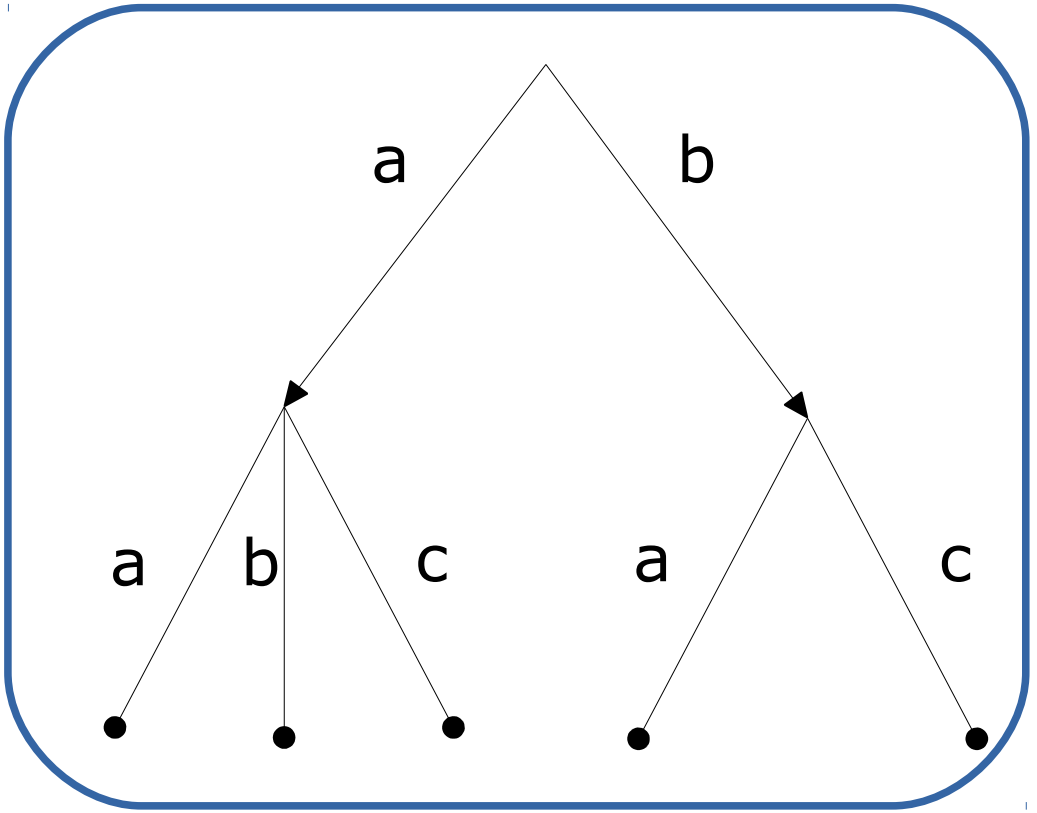
Starting nodes: b → 2  g → 3  r → 7

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.

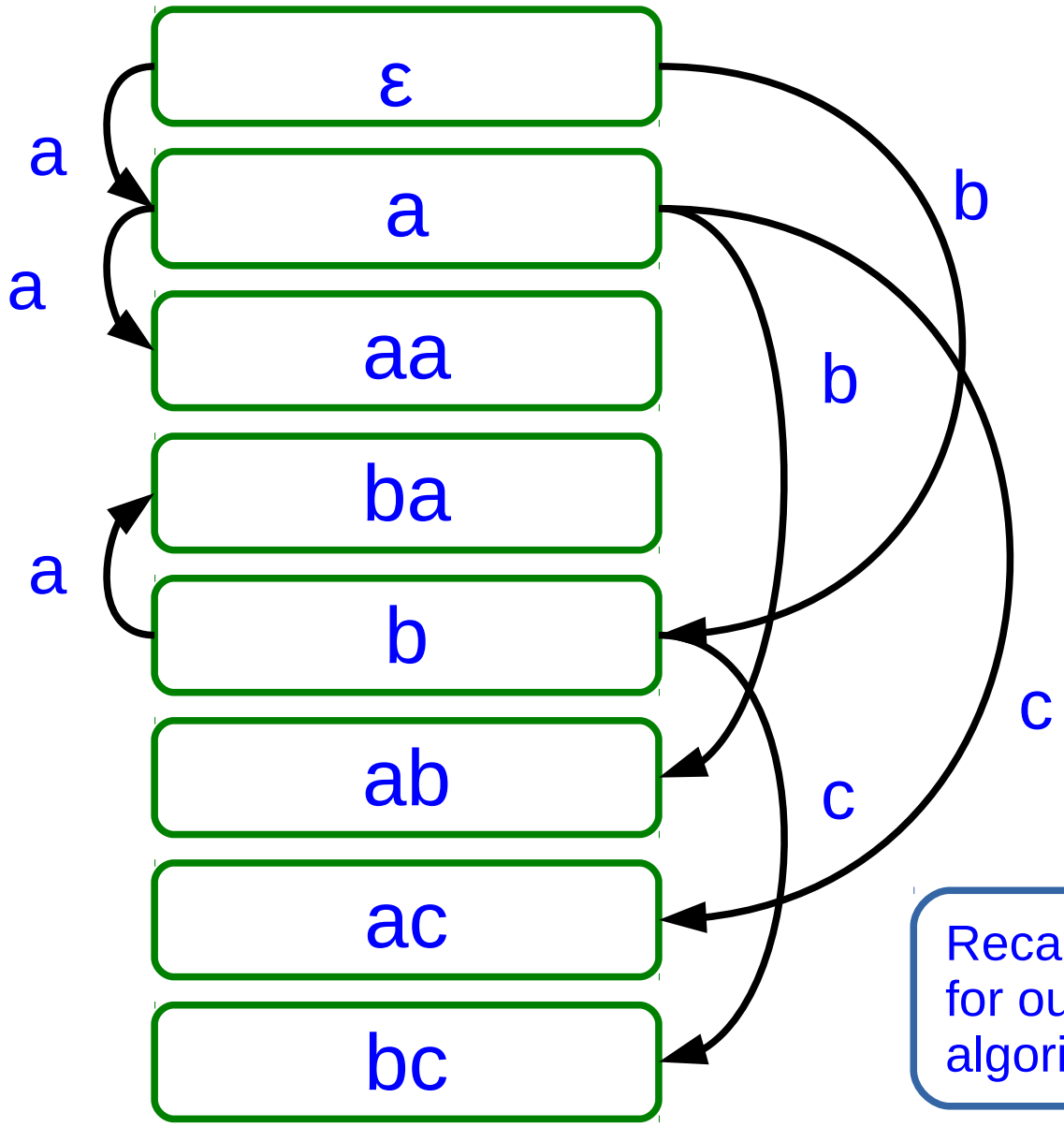
If a NFA has an ordering that makes it a Wheeler graph we can exploit this succinct representation.

Example: given a labeled tree, we want to check whether there exists a path starting from the root spelling a given string p

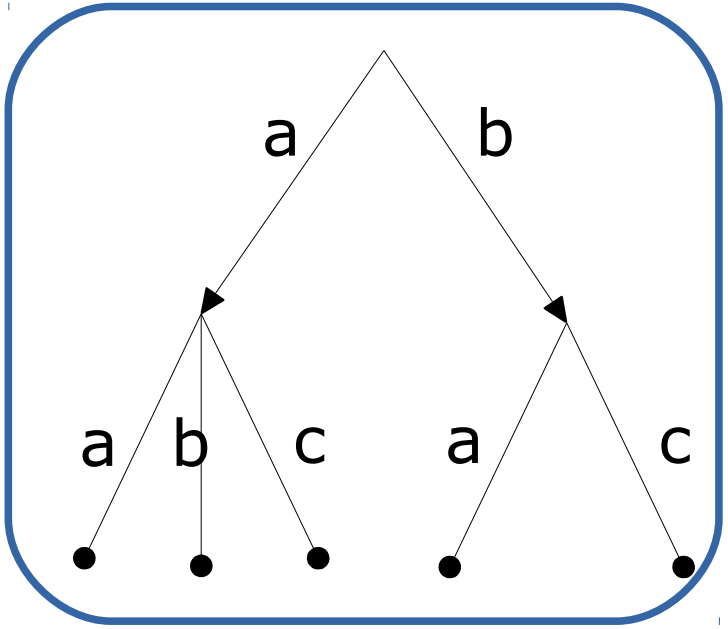



The tree itself is a NFA solving this problem (the root is the start state, all nodes are final)

Reasoning as before we can find an ordering that makes the NFA a Wheeler Automata: we automatically get a compact representation with efficient navigation and search




The original tree




Recall: state labels are for our convenience, the algorithms don't use them

# References


Historical reference:

T. Gagie, G. Manzini, J. Sirén,

Wheeler graphs: A framework for BWT-based data Structures

Theoretical Computer Science, Vol 698, 2017

Clear exposition and pointers to recent results:

N. Prezza,

Subpath Queries on Compressed Graphs: A Survey

Algorithms (MDPI), Vol. 14, 2021