From text indexing to regular language indexing

One FLAT World Seminar

November 26, 2025

Nicola Prezza, Ca’ Foscari University of Venice, Italy

Università
Ca’Foscari
Venezia

Overview

1. Sorting and indexing

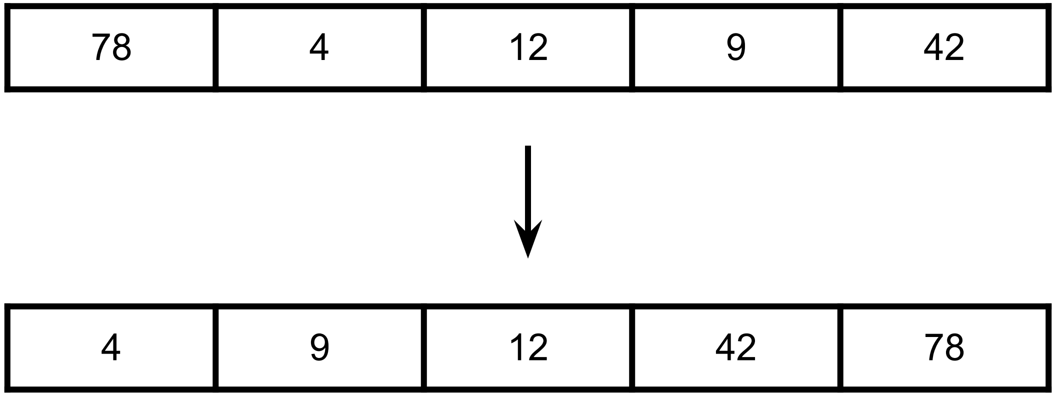
2. Wheeler automata / languages

3. p-sortable automata / languages

Sorting and indexing

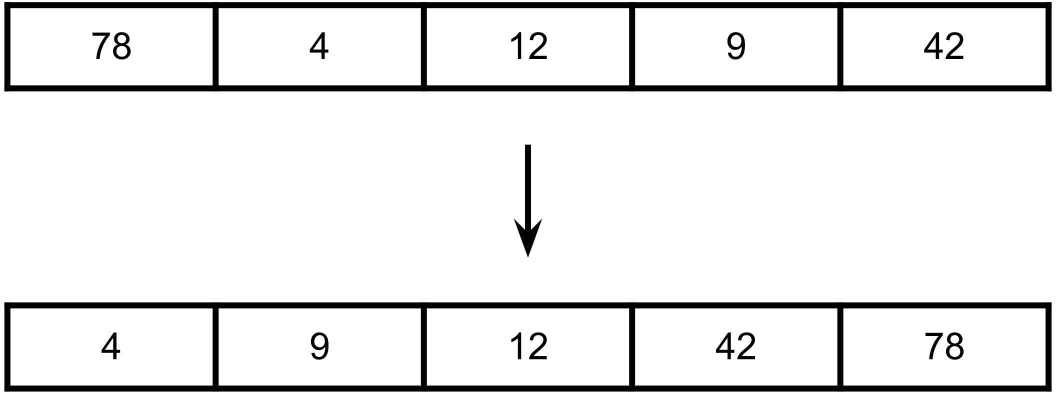
Sorting

Sorting is the algorithmic process of ordering the elements of a given set according to a specific order.



Sorting

Sorting is the algorithmic process of ordering the elements of a given set according to a specific order.

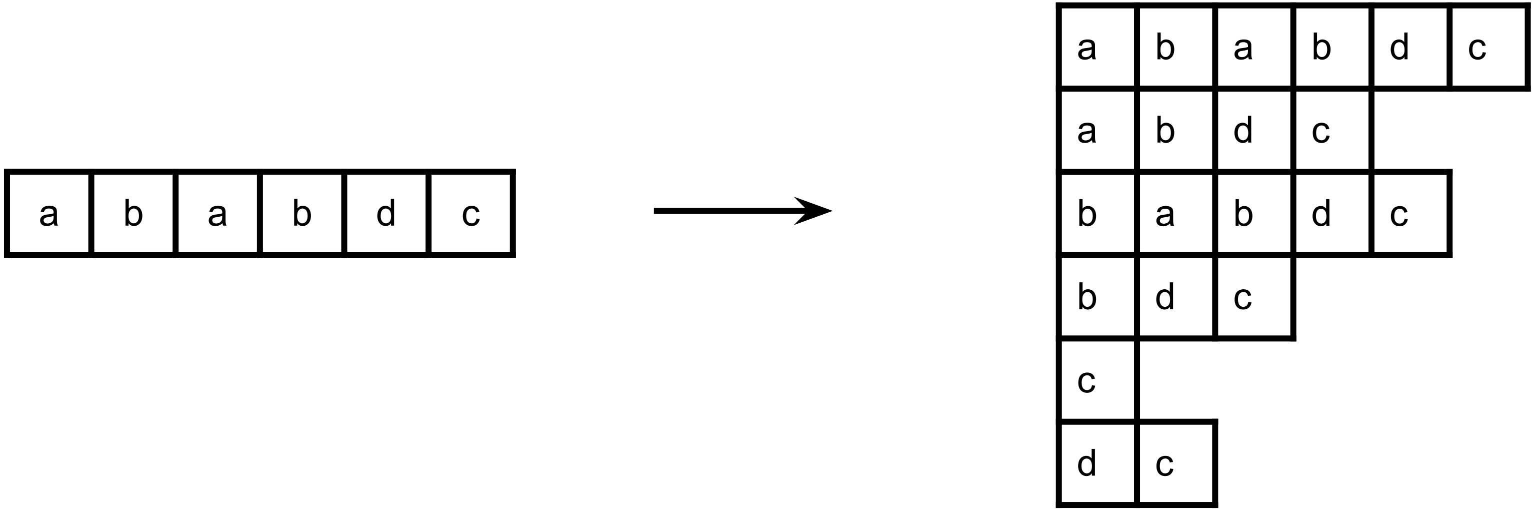


Benefits: the sorted list is

- Searchable (binary search)
- More compressible (store just the differences between consecutive integers)

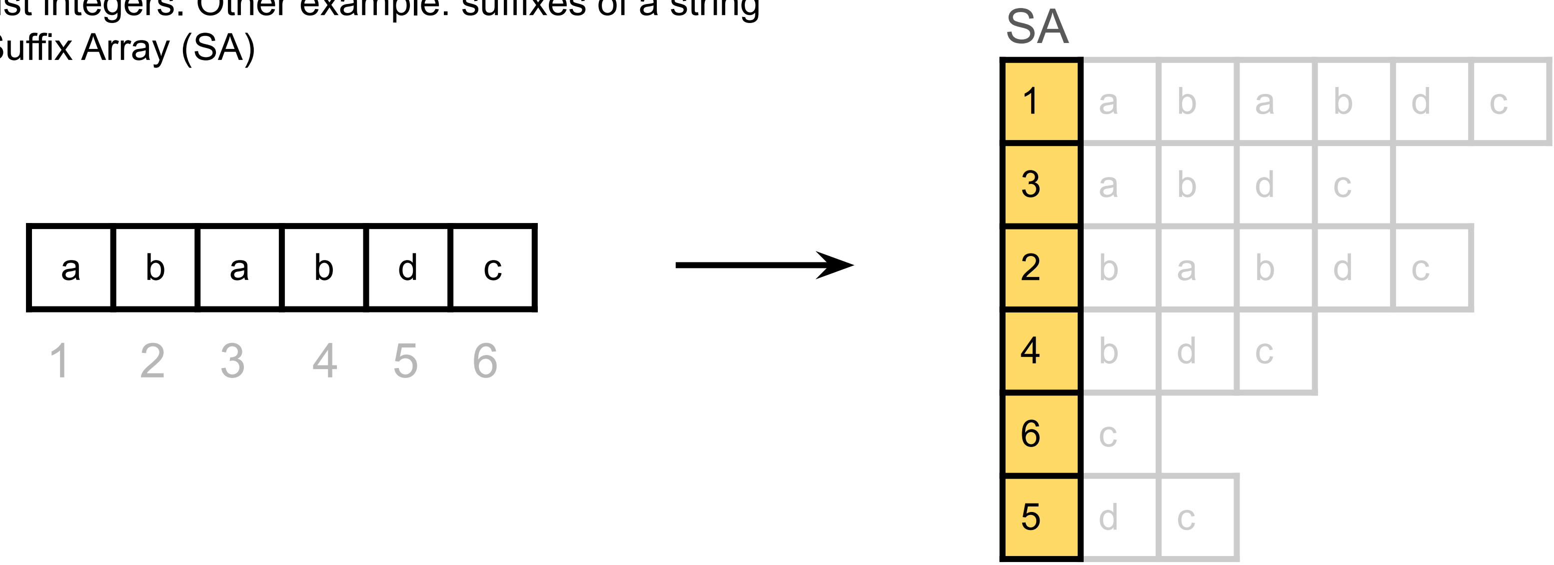
Sorting

Not just integers. Other example: suffixes of a string



Sorting

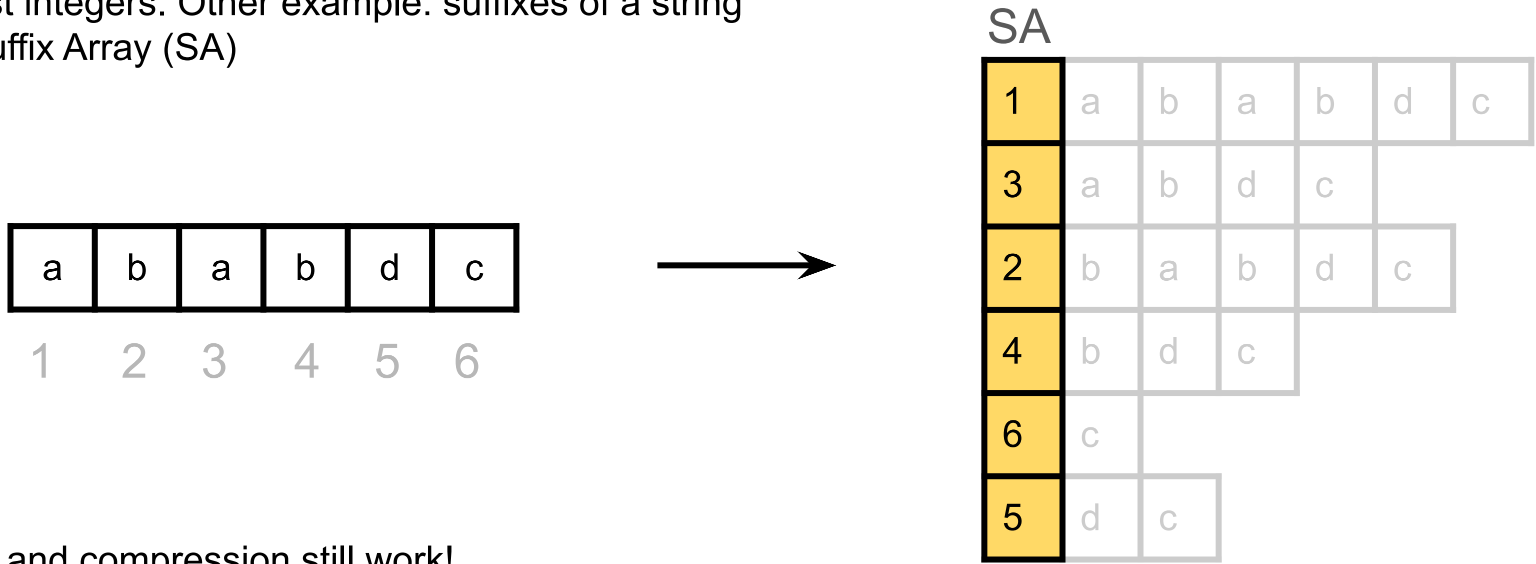
Not just integers. Other example: suffixes of a string
The Suffix Array (SA)



Manber, Udi, and Gene Myers. "Suffix arrays: a new method for on-line string searches." *siam Journal on Computing* 22.5 (1993): 935-948.

Text indexing

Not just integers. Other example: suffixes of a string
The Suffix Array (SA)



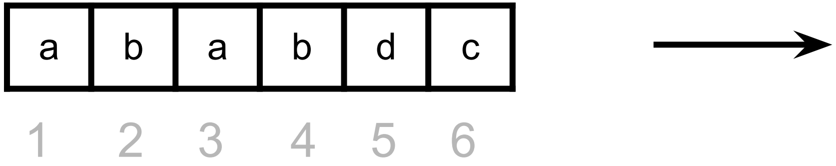
Indexing and compression still work!

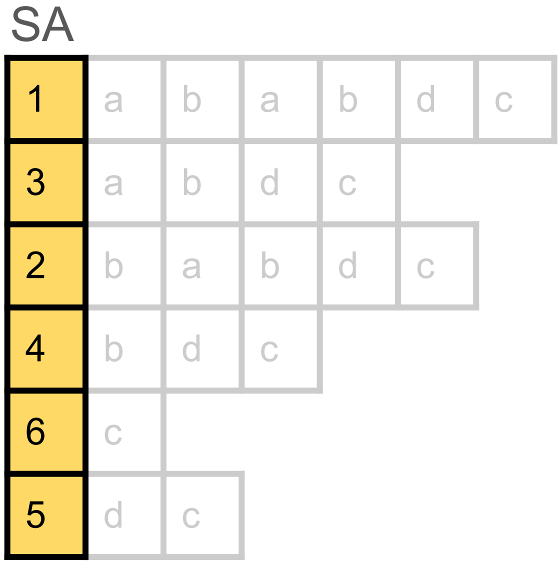
- **Indexing**: suffixies prefixed by a word (e.g. “ab”) form a range. Can be found by binary search on SA.

Manber, Udi, and Gene Myers. "Suffix arrays: a new method for on-line string searches." *siam Journal on Computing* 22.5 (1993): 935-948.

Text indexing

Not just integers. Other example: suffixes of a string
The Suffix Array (SA)





Indexing and compression still work!

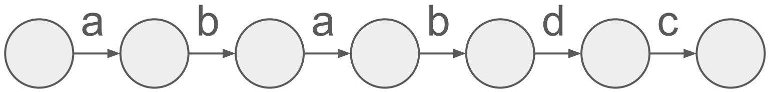
- **Indexing**: suffixies prefixed by a word (e.g. “ab”) form a range. Can be found by binary search on SA.
- SA can be **compressed** while still supporting fast pattern matching.

Roberto Grossi, Jeffrey Scott Vitter, 2000. Compressed Suffix Arrays and Suffix Trees with Applications to Text Indexing and String Matching. STOC.
Paolo Ferragina, Giovanni Manzini, 2000. Opportunistic Data Structures with Applications. FOCS.

Text indexing

Observations:

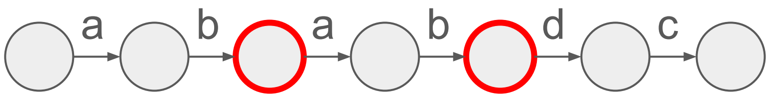
- A string is a very simple kind of NFA



Text indexing

Observations:

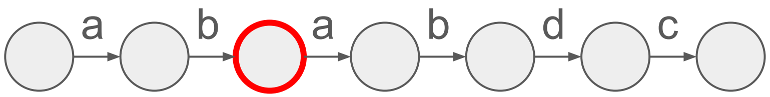
- A string is a very simple kind of NFA
- “pattern matching on NFA” ≡ find states reached by given string (“ab” in the example)



Text indexing

Observations:

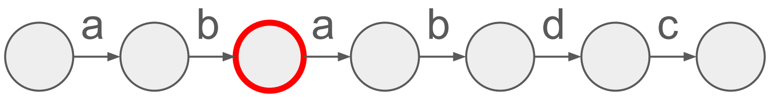
- A string is a very simple kind of NFA
- “pattern matching on NFA” ≡ find states reached by given string (“ab” in the example)
- Membership ≡ pattern matching from the source



Text indexing

Observations:

- A string is a very simple kind of NFA
- “pattern matching on NFA” ≡ find states reached by given string (“ab” in the example)
- Membership ≡ pattern matching from the source



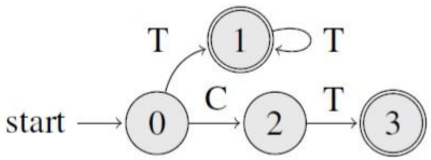
Can we generalize suffix sorting to regular languages?
Ideally: queries in O(|P|) time, where P is the query string.

Problem definition

Problem (regular language indexing)

Given a NFA, build a (small) data structure supporting efficiently the following queries:

- Count: given string P, return the number of states reached by a path labeled P
- Locate: given string P, return the states reached by a path labeled P
- Membership: given string P, is a final state reached by a path labeled P starting in the source?



Example:

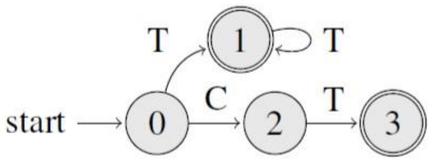
- count("T") = 2, locate("T") = {1,3}
- count("TT") = 1, locate("T") = {1}
- count("CT") = 1, locate("T") = {3}
- membership("TTT") = true
- membership("C") = false

Problem definition

Problem (regular language indexing)

Given a NFA, build a (small) data structure supporting efficiently the following queries:

- Count: given string P, return the number of states reached by a path labeled P
- Locate: given string P, return the states reached by a path labeled P
- Membership: given string P, is a final state reached by a path labeled P starting in the source?



Example:

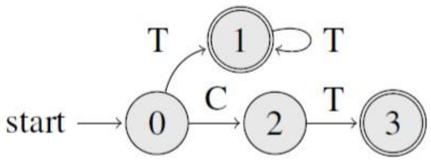
- count(“T”) = 2, locate(“T”) = {1,3}
- count(“TT”) = 1, locate(“T”) = {1}
- count(“CT”) = 1, locate(“T”) = {3}
- membership(“TTT”) = true
- membership(“C”) = false

Problem definition

Problem (regular language indexing)

Given a NFA, build a (small) data structure supporting efficiently the following queries:

- Count: given string P, return the number of states reached by a path labeled P
- Locate: given string P, return the states reached by a path labeled P
- Membership: given string P, is a final state reached by a path labeled P starting in the source?

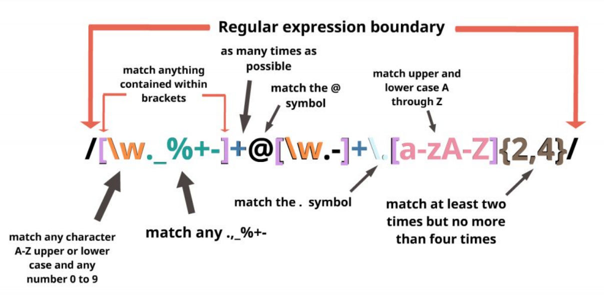


Example:

- count(“T”) = 2, locate(“T”) = {1,3}
- count(“TT”) = 1, locate(“T”) = {1}
- count(“CT”) = 1, locate(“T”) = {3}
- membership(“TTT”) = true
- membership(“C”) = false

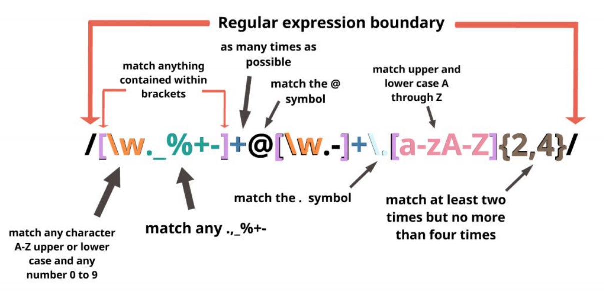
Applications

- Regular expression matching (e.g. linux `grep`) (matching a regex on text)

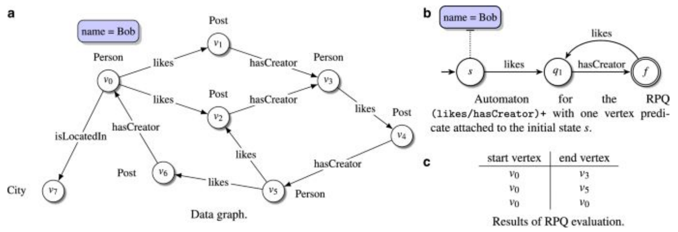


Applications

- Regular expression matching (e.g. linux `grep`) (matching a regex on text)

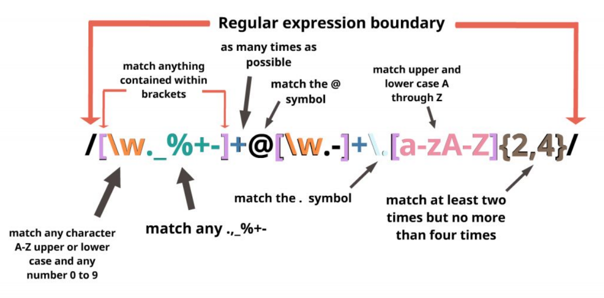


- Graph databases (e.g. regular path queries) (matching a regex on a graph)

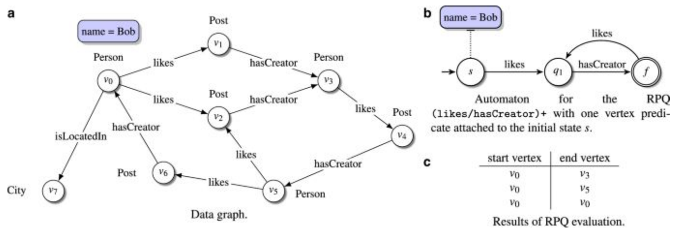


Applications

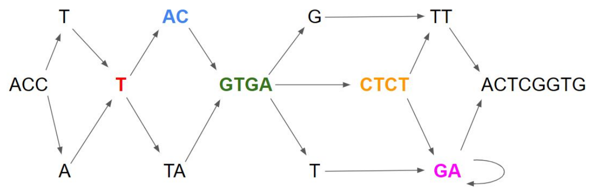
- Regular expression matching (e.g. linux `grep`) (matching a regex on text)



- Graph databases (e.g. regular path queries) (matching a regex on a graph)



- Bioinformatics (pattern matching on pan-genome graphs). (matching a text on graph)



Lower bounds

Unfortunately, unless the Strong Exponential Time Hypothesis (SETH) fails:

- Matching a regular expression E on text T requires $\Omega(|E| \cdot |T|)$ time [Backurs, Indyk FOCS ‘16]

Arturs Backurs, Piotr Indyk. "Which regular expression patterns are hard to match?." *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016.

Lower bounds

Unfortunately, unless the Strong Exponential Time Hypothesis (SETH) fails:

- Matching a regular expression E on text T requires $\Omega(|E| \cdot |T|)$ time [Backurs, Indyk FOCS ‘16]
- Regular path queries (emptiness) require quadratic time: size of graph times size of NFA for the regex [Casel, Schmid, LMCS’23]

Arturs Backurs, Piotr Indyk. "Which regular expression patterns are hard to match?." *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016.

Katrin Casel, Markus L. Schmid. "Fine-grained complexity of regular path queries." *Logical Methods in Computer Science* 19 (2023).

Lower bounds

Unfortunately, unless the Strong Exponential Time Hypothesis (SETH) fails:

- Matching a regular expression E on text T requires $\Omega(|E| \cdot |T|)$ time [Backurs, Indyk FOCS ‘16]
- Regular path queries (emptiness) require quadratic time: size of graph times size of NFA for the regex [Casel, Schmid, LMCS’23]
- Solving pattern matching queries on labeled graphs requires $\Omega(m \cdot n)$ time, where m is the graph’s size and n is the query length [Equi et al. TALG’23]

Arturs Backurs, Piotr Indyk. "Which regular expression patterns are hard to match?." *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2016.

Katrin Casel, Markus L. Schmid. "Fine-grained complexity of regular path queries." *Logical Methods in Computer Science* 19 (2023).

Massimo Equi, Veli Mäkinen, Alexandru Tomescu, Roberto Grossi (2023). On the complexity of string matching for graphs. *ACM Transactions on Algorithms*, *19*(3), 1-25.

Not hard on all NFA!

Although these problems are hard in general, for some NFA they are easy: paths, trees, de Bruijn graphs, …

Next slides: what is the most general class of “indexable/sortable” NFA?

- The class of Wheeler NFA / languages
- Generalization to arbitrary NFA

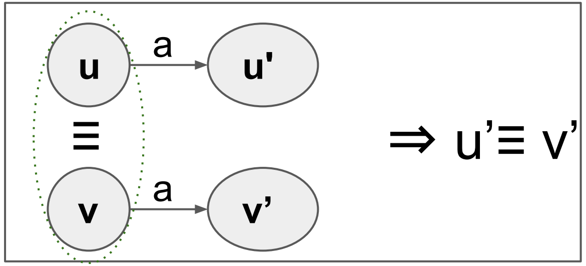
Wheeler automata
and
Wheeler languages

Myhill-Nerode relation

- We take our first steps from a central object in finite automata theory: the MN equivalence relation
- Intuition: we will turn the MN relation into an order, and use it to index the NFA

Myhill-Nerode relation

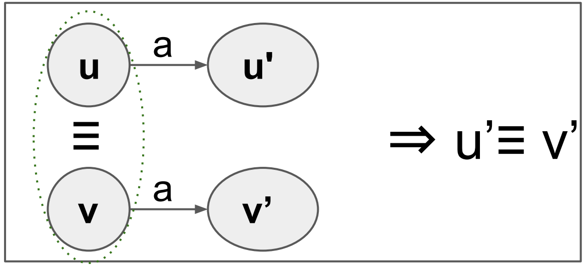
- We take our first steps from a central object in finite automata theory: the MN equivalence relation
- Intuition: we will turn the MN relation into an order, and use it to index the NFA
- MN equivalence between states: coarsest equivalence relation satisfying* :

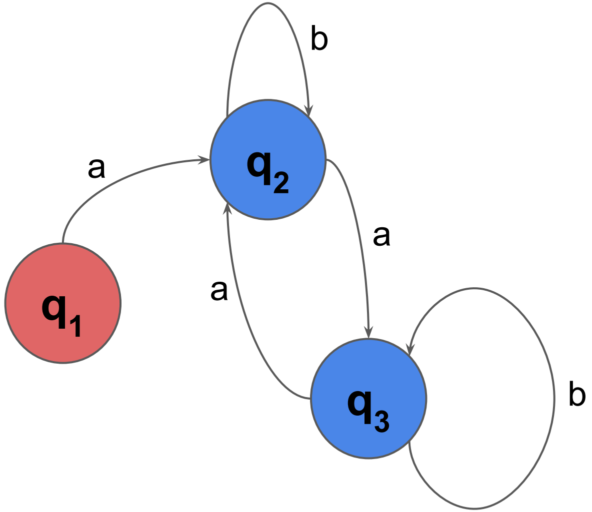


* For simplicity: (1) take all states to be final. (2) nonexistent transitions go into a (virtual) non-final sink.

Myhill-Nerode relation

- We take our first steps from a central object in finite automata theory: the MN equivalence relation
- Intuition: we will turn the MN relation into an order, and use it to index the NFA
- MN equivalence between states: coarsest equivalence relation satisfying* :





* For simplicity: (1) take all states to be final. (2) nonexistent transitions go into a (virtual) non-final sink.

Switching to an order

… but to index, we need an **order**. What if we turn ≡ (equivalence relation) into a **total order** ≤?

Equivalence relation ≡

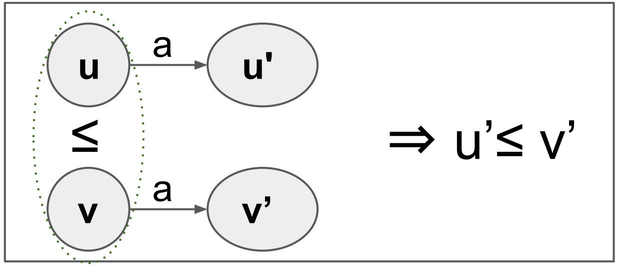
reflexive:	$x \equiv x$
symmetric:	$x \equiv y \Leftrightarrow y \equiv x$
transitive:	$x \equiv y \land y \equiv z \Rightarrow x \equiv z$

Total order ≤

reflexive:	$x \leq x$
antisymmetric:	$x \leq y \land y \leq x \Rightarrow x = y$
transitive:	$x \leq y \land y \leq z \Rightarrow x \leq z$
strongly connected:	$x \leq y \lor y \leq x$

Switching to an order

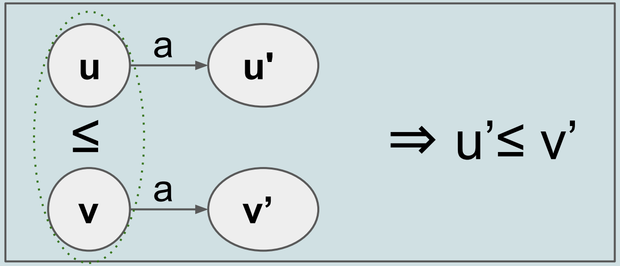
… but to index, we need an **order**. What if we turn ≡ (equivalence relation) into a **total order** ≤?



Ordered Automata

We obtain:

Def: ordered automaton (OA)
An OA is a NFA for which there exists a total order ≤ satisfying:



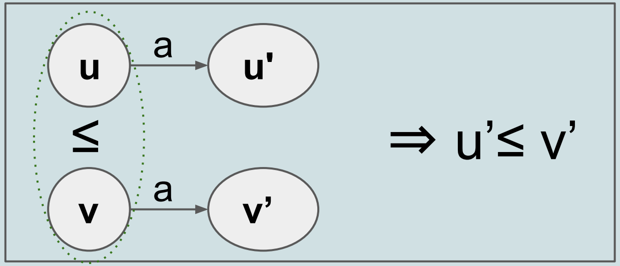
* distinction between final/non final states does not matter in this definition. We allow incomplete NFA.

Shyr, H.J. and Thierrin, G., 1974. Ordered automata and associated languages. *Tamkang J. Math*, *5*(1)

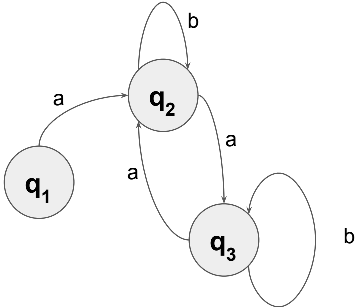
Ordered Automata

We obtain:

Def: ordered automaton (OA)
An OA is a NFA for which there exists a total order ≤ satisfying:



Such a total order does not always exist!



* distinction between final/non final states does not matter in this definition. We allow incomplete NFA.

Shyr, H.J. and Thierrin, G., 1974. Ordered automata and associated languages. *Tamkang J. Math*, *5*(1)

Ordered Automata

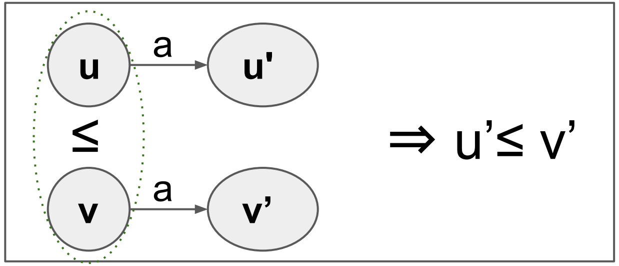
- Interestingly, Shyr and Thierrin prove that OA recognize only star-free languages.

Shyr, H.J. and Thierrin, G., 1974. Ordered automata and associated languages. *Tamkang J. Math*, *5*(1)

Ordered Automata

- Interestingly, Shyr and Thierrin prove that OA recognize only star-free languages.

- However, OA are not good enough for indexing: states reached by the same letter are not necessarily contiguous in the order. We miss a base case! let’s add it …

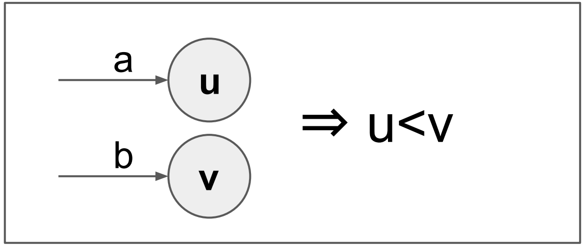


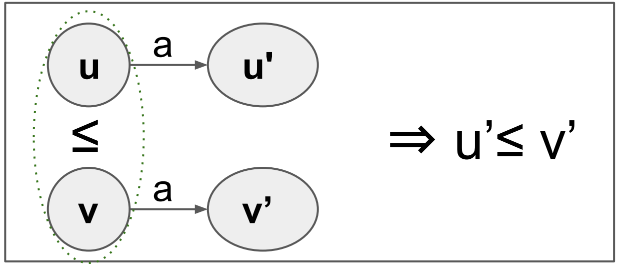
Shyr, H.J. and Thierrin, G., 1974. Ordered automata and associated languages. *Tamkang J. Math*, *5*(1)

Ordered Automata

- Interestingly, Shyr and Thierrin prove that OA recognize only star-free languages.

- However, OA are not good enough for indexing: states reached by the same letter are not necessarily contiguous in the order. We miss a base case! let’s add it …



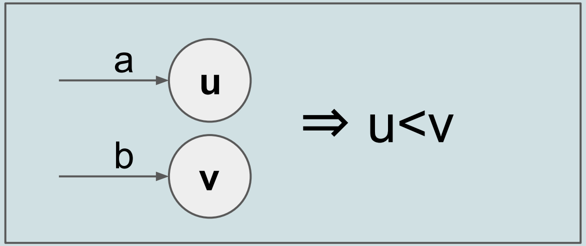


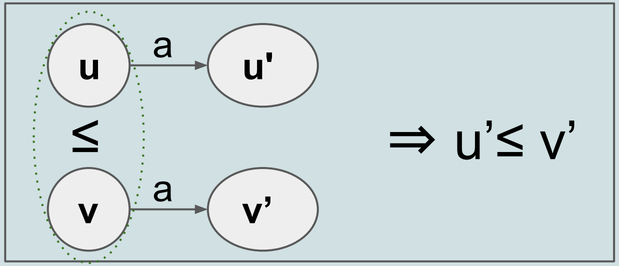
Shyr, H.J. and Thierrin, G., 1974. Ordered automata and associated languages. *Tamkang J. Math*, *5*(1)

Wheeler Automata

Def: Wheeler Automaton (WNFA)

A NFA is said to be Wheeler iff there exists a total order ≤ of its states satisfying the following two axioms:



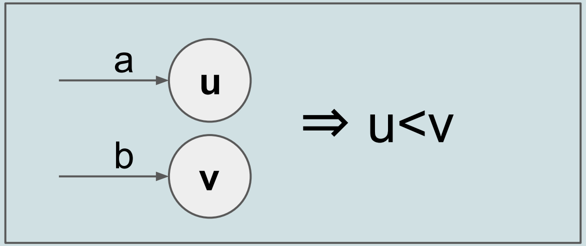


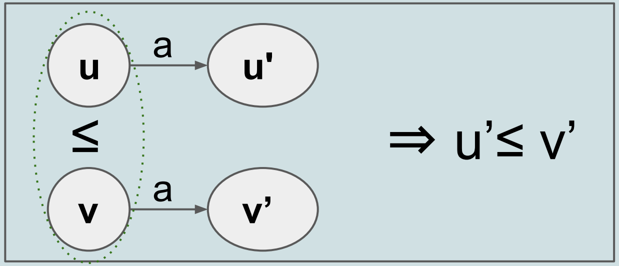
Gagie, Manzini, Sirén, 2017. "Wheeler graphs: A framework for BWT-based data structures." TCS

Wheeler Automata

Def: Wheeler Automaton (WNFA)

A NFA is said to be Wheeler iff there exists a total order ≤ of its states satisfying the following two axioms:





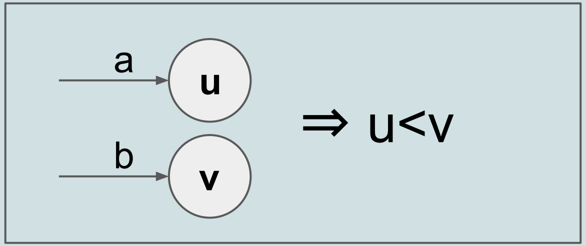
* We assume that, if the source state has in-degree zero, then it comes first in the order (equiv: dummy incoming edge labeled #)

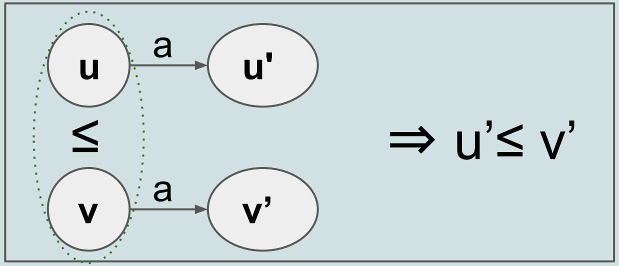
Gagie, Manzini, Sirén, 2017. "Wheeler graphs: A framework for BWT-based data structures." TCS

Wheeler Automata

Def: Wheeler Automaton (WNFA)

A NFA is said to be Wheeler iff there exists a total order ≤ of its states satisfying the following two axioms:





* We assume that, if the source state has in-degree zero, then it comes first in the order (equiv: dummy incoming edge labeled #)

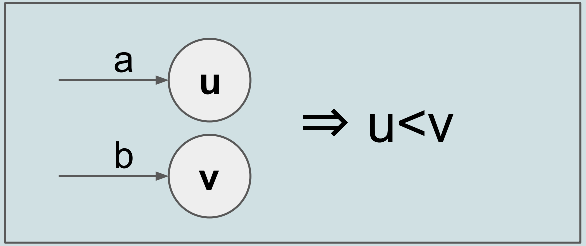
** Axiom 1 implies *input consistency*: all incoming edges of a given state bear the same label. Not restrictive.

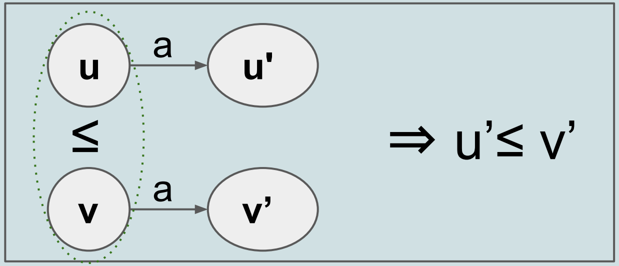
Gagie, Manzini, Sirén, 2017. "Wheeler graphs: A framework for BWT-based data structures." TCS

Wheeler Automata

Def: Wheeler Automaton (WNFA)

A NFA is said to be Wheeler iff there exists a total order ≤ of its states satisfying the following two axioms:





* We assume that, if the source state has in-degree zero, then it comes first in the order (equiv: dummy incoming edge labeled #)

** Axiom 1 implies *input consistency*: all incoming edges of a given state bear the same label. Not restrictive.

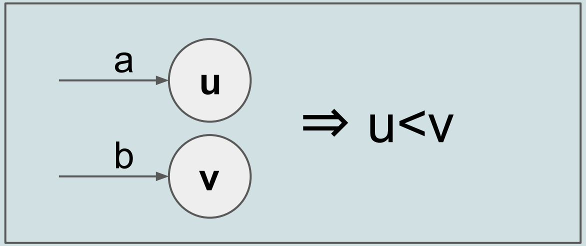
*** Note: WNFA ⊂ OA ⇒ *Wheeler languages* are star-free

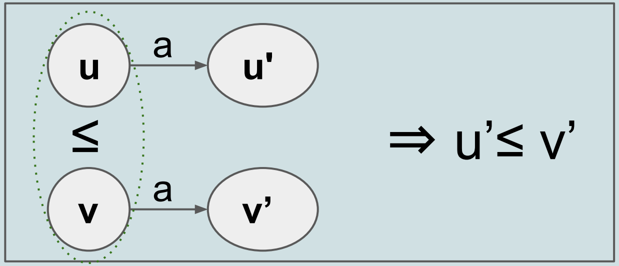
Gagie, Manzini, Sirén, 2017. "Wheeler graphs: A framework for BWT-based data structures." TCS

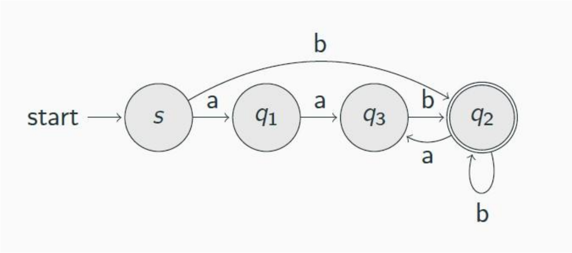
Wheeler Automata

Def: Wheeler Automaton (WNFA)

A NFA is said to be Wheeler iff there exists a total order ≤ of its states satisfying the following two axioms:





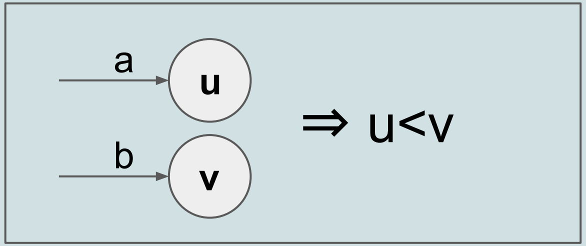


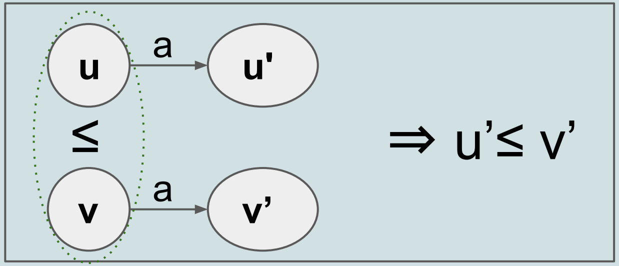
Gagie, Manzini, Sirén, 2017. "Wheeler graphs: A framework for BWT-based data structures." TCS

Wheeler Automata

Def: Wheeler Automaton (WNFA)

A NFA is said to be Wheeler iff there exists a total order ≤ of its states satisfying the following two axioms:





Axiom (1) makes things much more interesting w.r.t. Ordered Automata! Next slides:

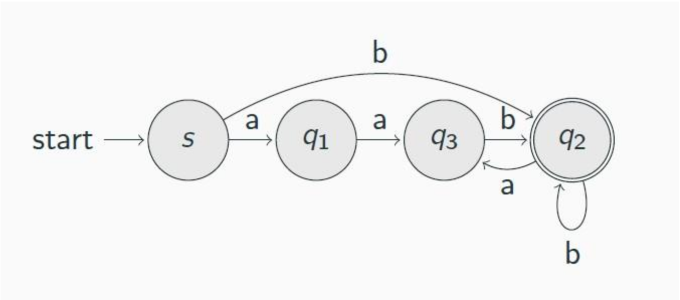
- Efficient encoding
- Linear-time queries
- Wheeler languages

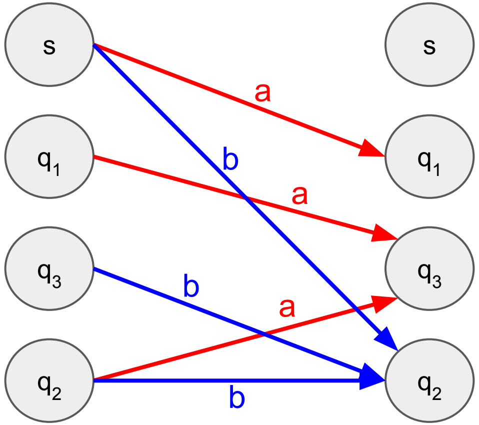
Gagie, Manzini, Sirén, 2017. "Wheeler graphs: A framework for BWT-based data structures." TCS

Bipartite representation

Useful visualization of the Wheeler order: **bipartite representation**.

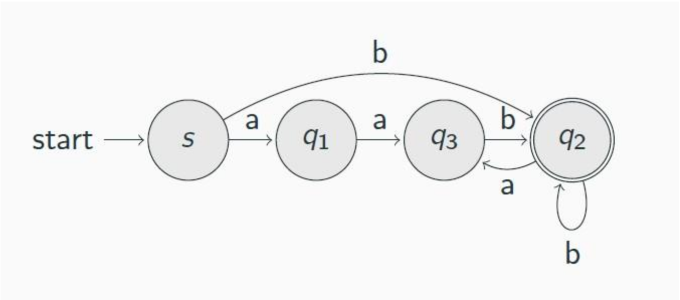
- Build a bipartite graph: two copies of the nodes, sorted by the candidate order.
- Same edges of the input NFA, but drawn left-to-right.
- The order is Wheeler ⇔ Same-letter edges must not cross.

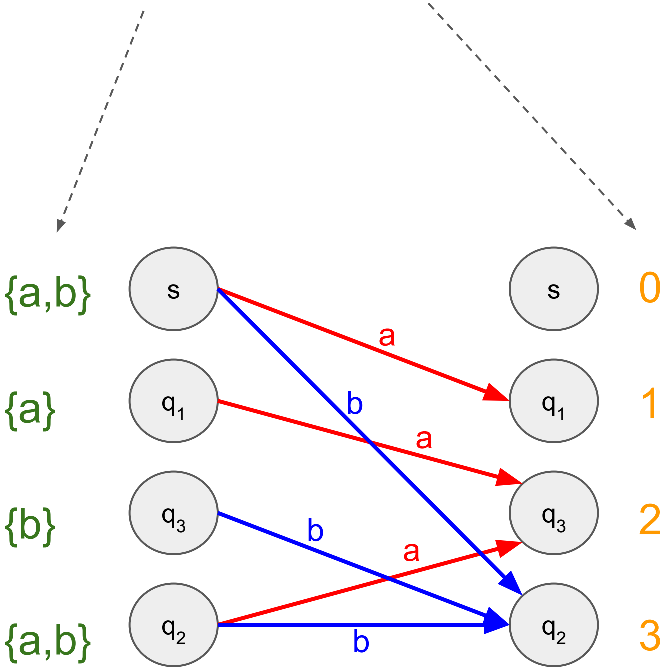




Efficient encoding

⇒ we can store the WNFA in O(1) bits per edge*! just store out-going labels and in-degrees



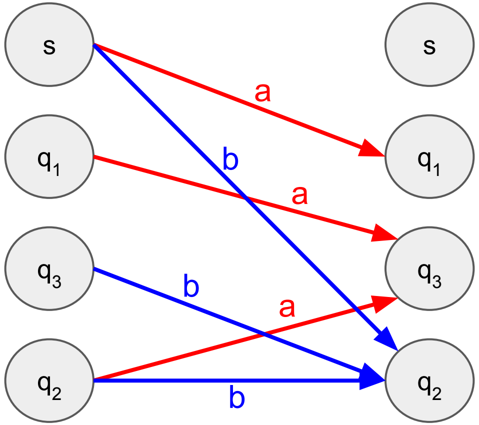


*assuming constant-size alphabet for simplicity

Pattern matching

Also:

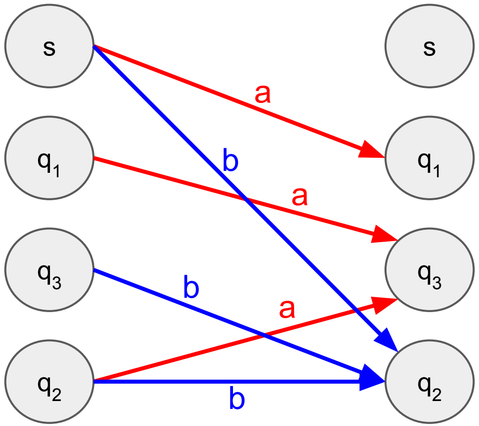
same-letter edges must not cross
⇒
Nodes reached by a given string P form a *range* in the total order



Pattern matching

Also:

same-letter edges must not cross
⇒
Nodes reached by a given string P form a *range* in the total order



Consequences:

- WNFAs generalize known indexes on strings, trees, de Bruijn graphs…
- Indexed pattern matching/membership in optimal O(|P|) time

Pattern matching

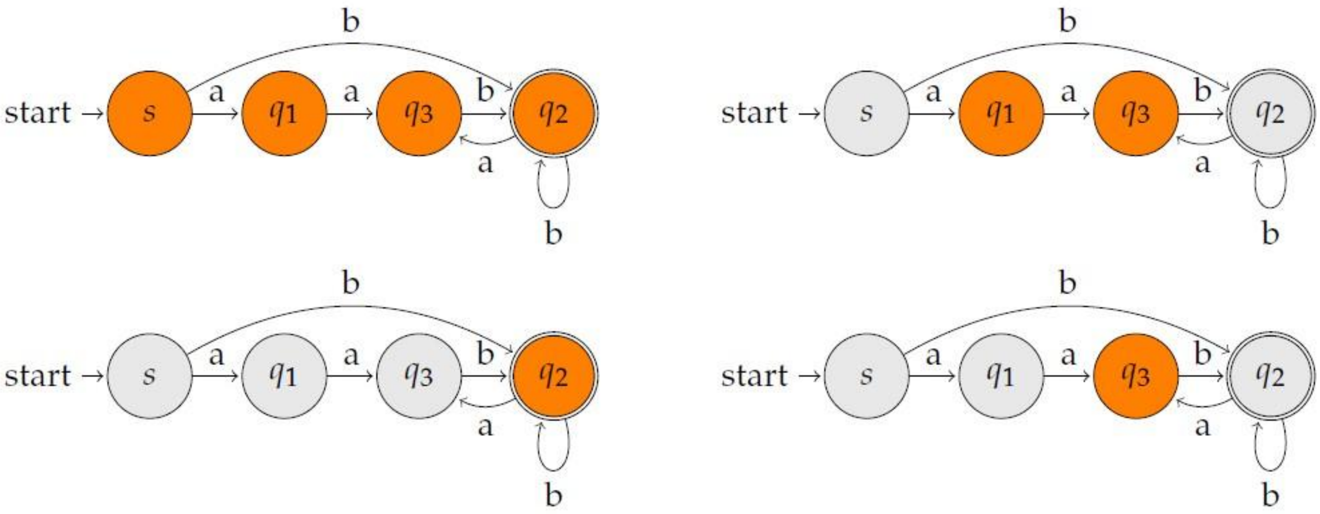
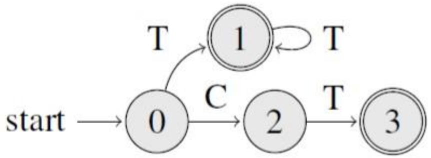


Figure 11. Searching nodes reached by a path labeled "aba" in a Wheeler graph. Top left: we begin with the nodes reached by the empty string (full range). Top right: range obtained from the previous one following edges labeled 'a'. Bottom left: range obtained from the previous one following edges labeled 'b'. Bottom right: range obtained from the previous one following edges labeled 'a'. This last range contains all nodes reached by a path labeled "aba"

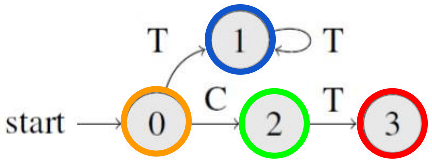
Wheeler languages



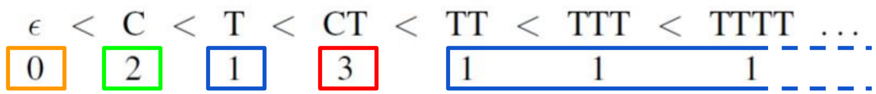
minimum DFA for L = TT* | CT

ϵ	<	C	<	T	<	CT	<	TT	<	TTT	<	TTTT	...
0		2		1		3		1		1		1	

Wheeler languages

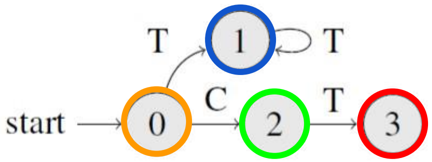


minimum DFA for L = TT* | CT

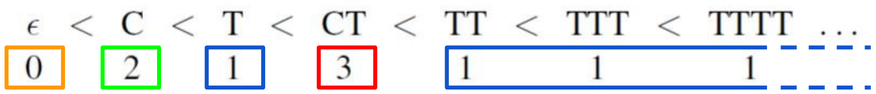


Finite number of Myhill-Nerode intervals in co-lex order

Wheeler languages



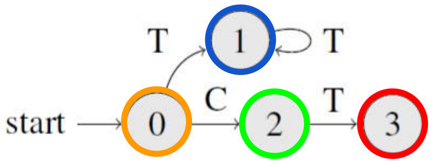
minimum DFA for L = TT* | CT



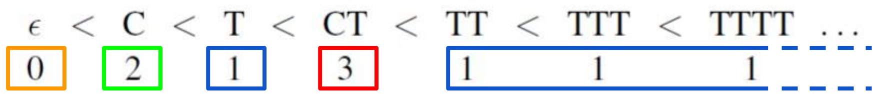
Finite number of Myhill-Nerode intervals in co-lex order ≡ **Wheeler language**

[1] J. Alanko, G. D'Agostino, A. Policriti, and N. P. *Regular Languages meet Prefix Sorting*. SODA 2020.

Wheeler languages



minimum DFA for L = TT* | CT



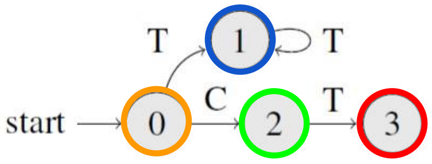
Finite number of Myhill-Nerode intervals in co-lex order ≡ **Wheeler language**

Theorem [1] Myhill-Nerode theorem for W. languages. The following are equivalent:

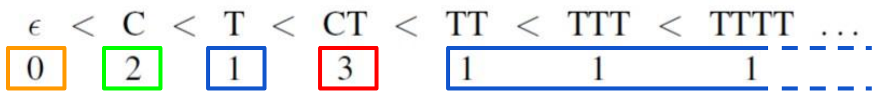
1. *A regular language L is Wheeler*
2. *L is recognized by a Wheeler NFA*
3. *L is recognized by a Wheeler DFA*
4. *The Myhill-Nerode equivalence classes of L form a **finite number of intervals in co-lex order**.*

[1] J. Alanko, G. D'Agostino, A. Policriti, and N. P. *Regular Languages meet Prefix Sorting*. SODA 2020.

Wheeler languages



minimum DFA for L = TT* | CT



Finite number of Myhill-Nerode intervals in co-lex order ≡ **Wheeler language**

Theorem [1] Myhill-Nerode theorem for W. languages. The following are equivalent:

1. *A regular language L is Wheeler*
2. *L is recognized by a Wheeler NFA*
3. *L is recognized by a Wheeler DFA*
4. *The Myhill-Nerode equivalence classes of L form a **finite number of intervals in co-lex order**.*

In fact, given a WNFA we can always build an equivalent WDFA of at most twice the size!

[1] J. Alanko, G. D'Agostino, A. Policriti, and N. P. *Regular Languages meet Prefix Sorting*. SODA 2020.

Overview of algorithmic results on Wheeler languages

Overview of algorithmic results

m = size of input A
M = size of output

output / input	NFA	2-NFA	DFA	WNFA	WDFA
W. order of A if A is W					
min. equivalent WDFA if L(A) is W					
equivalent sorted WNFA if L(A) is W					
is L(A) Wheeler?					
Is A Wheeler?					

Overview of algorithmic results

m = size of input A
M = size of output

output \ input	NFA	2-NFA	DFA	WNFA	WDFA
W. order of A if A is W	NP-C [2]				
min. equivalent WDFA if L(A) is W	PSPACE-H [1]				
equivalent sorted WNFA if L(A) is W					
is L(A) Wheeler?	PSPACE-C [1]				
Is A Wheeler?	NP-C [2]				

[1] Giovanna D’Agostino, Davide Martincigh, and Alberto Policriti. Ordering regular languages: a danger zone. ICTCS 2021.
[2] Daniel Gibney and Sharma V Thankachan. On the complexity of recognizing wheeler graphs. Algorithmica 2022.

Overview of algorithmic results

m = size of input A
M = size of output

output \ input	NFA	2-NFA	DFA	WNFA	WDFA
W. order of A if A is W	NP-C [2]	$O(m^2)$ [3]			
min. equivalent WDFA if L(A) is W	PSPACE-H [1]	?			
equivalent sorted WNFA if L(A) is W					
is L(A) Wheeler?	PSPACE-C [1]				
Is A Wheeler?	NP-C [2]	$O(m^2)$ [3]			

[1] Giovanna D’Agostino, Davide Martincigh, and Alberto Policriti. Ordering regular languages: a danger zone. ICTCS 2021.
[2] Daniel Gibney and Sharma V Thankachan. On the complexity of recognizing wheeler graphs. Algorithmica 2022.
[3] J. Alanko, G. D’Agostino, A. Policriti, and N. P. Regular Languages meet Prefix Sorting. SODA 2020.

Overview of algorithmic results

m = size of input A
M = size of output

<table>
<tr><th>input
output</th><th>NFA</th><th>2-NFA</th><th>DFA</th><th>WNFA</th><th>WDFA</th></tr>
<tr><td>W. order of A
if A is W</td><td>NP-C [2]</td><td>$O(m^2)$ [3]</td><td>$O(m)$ [3]</td><td></td><td></td></tr>
<tr><td>min. equivalent WDFA
if L(A) is W</td><td rowspan="2">PSPACE-H [1]</td><td rowspan="3">?</td><td rowspan="2">O(M log M)
[ongoing]</td><td></td><td></td></tr>
<tr><td>equivalent sorted WNFA
if L(A) is W</td><td></td><td></td></tr>
<tr><td>is L(A) Wheeler?</td><td>PSPACE-C [1]</td><td>$\Theta(m^2)$ [4]</td><td rowspan="2" colspan="2"></td></tr>
<tr><td>Is A Wheeler?</td><td>NP-C [2]</td><td>$O(m^2)$ [3]</td><td>$O(m)$ [3]</td></tr>
</table>

[1] Giovanna D’Agostino, Davide Martincigh, and Alberto Policriti. Ordering regular languages: a danger zone. ICTCS 2021.
[2] Daniel Gibney and Sharma V Thankachan. On the complexity of recognizing wheeler graphs. Algorithmica 2022.
[3] J. Alanko, G. D’Agostino, A. Policriti, and N. P. Regular Languages meet Prefix Sorting. SODA 2020.
[4] Ruben Becker, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Alberto Policriti, and Nicola Prezza. Optimal wheeler language recognition. SPIRE 2023

Overview of algorithmic results

m = size of input A
M = size of output

<table>
<tr><th>output \ input</th><th>NFA</th><th>2-NFA</th><th>DFA</th><th>WNFA</th><th>WDFA</th></tr>
<tr><td>W. order of A
if A is W</td><td>NP-C [2]</td><td>$O(m^2)$ [3]</td><td>$O(m)$ [3]</td><td>NP-C [2]</td><td></td></tr>
<tr><td>min. equivalent WDFA
if L(A) is W</td><td rowspan="2">PSPACE-H [1]</td><td rowspan="3">?</td><td rowspan="2">$O(M \log M)$
[ongoing]</td><td>polytime [3,6]</td><td></td></tr>
<tr><td>equivalent sorted WNFA
if L(A) is W</td><td>$O(m \log m)$ [5]</td><td></td></tr>
<tr><td>is L(A) Wheeler?</td><td>PSPACE-C [1]</td><td>$\Theta(m^2)$ [4]</td><td colspan="2" rowspan="2">trivial</td></tr>
<tr><td>Is A Wheeler?</td><td>NP-C [2]</td><td>$O(m^2)$ [3]</td><td>$O(m)$ [3]</td></tr>
</table>

[1] Giovanna D’Agostino, Davide Martincigh, and Alberto Policriti. Ordering regular languages: a danger zone. ICTCS 2021.
[2] Daniel Gibney and Sharma V Thankachan. On the complexity of recognizing wheeler graphs. Algorithmica 2022.
[3] J. Alanko, G. D’Agostino, A. Policriti, and N. P. Regular Languages meet Prefix Sorting. SODA 2020.
[4] Ruben Becker, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Alberto Policriti, and Nicola Prezza. Optimal wheeler language recognition. SPIRE 2023
[5] Ruben Becker, Manuel Cáceres, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Francisco Olivares, Nicola Prezza. Sorting Finite Automata via Partition Refinement. ESA 2023.
[6] Jarno Alanko, Nicola Cotumaccio, Nicola Prezza. Linear-time Minimization of Wheeler DFAs. DCC 2022.

Overview of algorithmic results

m = size of input A
M = size of output

<table>
<tr><th>output \ input</th><th>NFA</th><th>2-NFA</th><th>DFA</th><th>WNFA</th><th>WDFA</th></tr>
<tr><td>W. order of A
if A is W</td><td>NP-C [2]</td><td>$O(m^2)$ [3]</td><td>$O(m)$ [3]</td><td>NP-C [2]</td><td>$O(m)$ [3]</td></tr>
<tr><td>min. equivalent WDFA
if L(A) is W</td><td rowspan="2">PSPACE-H [1]</td><td rowspan="3">?</td><td rowspan="2">$O(M \log M)$
[ongoing]</td><td>polytime [3,6]</td><td>$O(m)$ [6]</td></tr>
<tr><td>equivalent sorted WNFA
if L(A) is W</td><td>$O(m \log m)$ [5]</td><td>$O(m)$ [3]</td></tr>
<tr><td>is L(A) Wheeler?</td><td>PSPACE-C [1]</td><td>$\Theta(m^2)$ [4]</td><td colspan="2" rowspan="2">trivial</td></tr>
<tr><td>Is A Wheeler?</td><td>NP-C [2]</td><td>$O(m^2)$ [3]</td><td>$O(m)$ [3]</td></tr>
</table>

[1] Giovanna D'Agostino, Davide Martincigh, and Alberto Policriti. Ordering regular languages: a danger zone. ICTCS 2021.
[2] Daniel Gibney and Sharma V Thankachan. On the complexity of recognizing wheeler graphs. Algorithmica 2022.
[3] J. Alanko, G. D'Agostino, A. Policriti, and N. P. Regular Languages meet Prefix Sorting. SODA 2020.
[4] Ruben Becker, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Alberto Policriti, and Nicola Prezza. Optimal wheeler language recognition. SPIRE 2023
[5] Ruben Becker, Manuel Cáceres, Davide Cenzato, Sung-Hwan Kim, Bojana Kodric, Francisco Olivares, Nicola Prezza. Sorting Finite Automata via Partition Refinement. ESA 2023.
[6] Jarno Alanko, Nicola Cotumaccio, Nicola Prezza. Linear-time Minimization of Wheeler DFAs. DCC 2022.

Generalizing:
p-sortable automata and languages

Searching a partially-ordered set

Classic algorithmic result:

Given a set of objects and a partial order < on the set such that width(<) = p, then:

- Searching the set requires at least p log (n/p) operations
- There exists a data structure supporting search in time O(p log (n/p))

Intuition: decompose < into p totally-sorted chains (Dilworth's theorem), run binary search on each chain.

Searching a partially-ordered set

Classic algorithmic result:

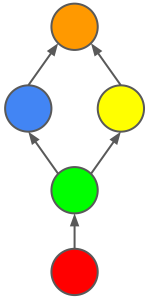
Given a set of objects and a partial order < on the set such that width(<) = p, then:

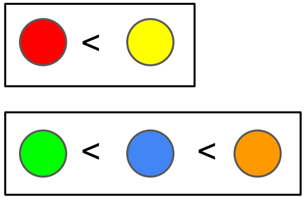
- Searching the set requires at least p log (n/p) operations
- There exists a data structure supporting search in time O(p log (n/p))

Intuition: decompose < into p totally-sorted chains (Dilworth's theorem), run binary search on each chain.

Hasse diagram of <

width(<) = 2





A chain decomposition into p=2 chains

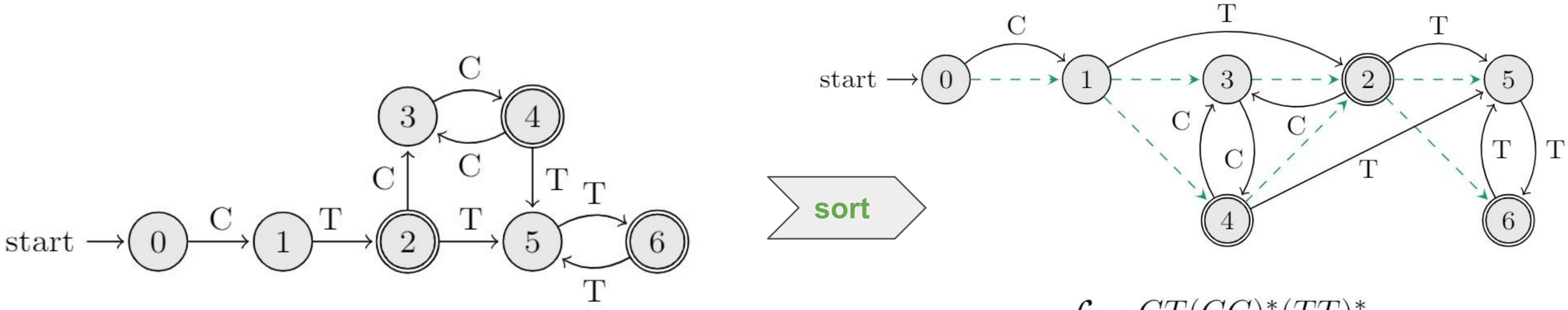
Generalization to arbitrary NFA: co-lex orders

For arbitrary NFAs: same idea of the Wheeler case, but do not require that $<$ is total.

Generalization to arbitrary NFA: co-lex orders

For arbitrary NFAs: same idea of the Wheeler case, but do not require that < is total.

Any NFA admits a ***partial co-lex order*** of its nodes.
We are interested in a **minimum-width** one (Wheeler NFA are the case width=1)

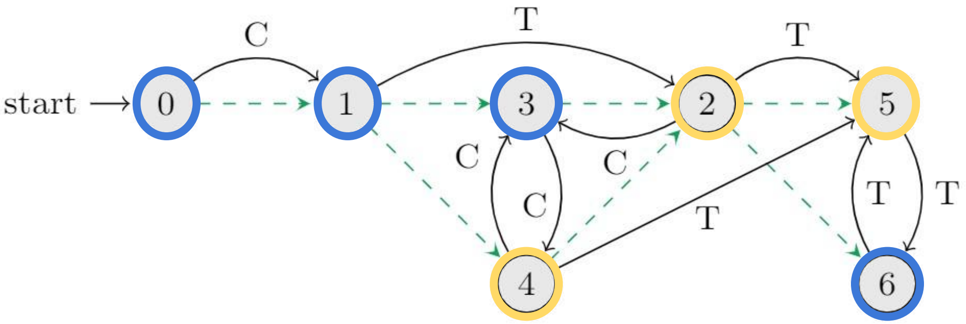


$$\mathcal{L} = CT(CC)^*(TT)^*$$

- - - → Hasse diagram

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

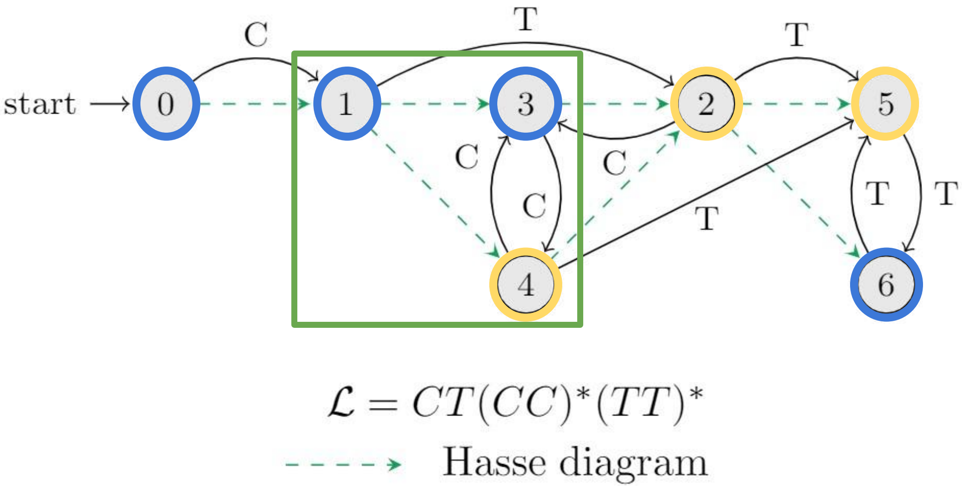


$$\mathcal{L} = CT(CC)^*(TT)^*$$

Hasse diagram

A possible chain partitioning (yellow, blue) of the partial order.

co-lex orders



A possible chain partitioning (yellow, blue) of the partial order.

Indexing ≡ states reached by any string (in the example, “C”) always form a *convex set in the partial order*.

co-lex orders

Let n = number of states.

Results. $p = \text{width}(<)$ is an important parameter for NFAs:

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

Let n = number of states.

Results. p = width(<) is an important parameter for NFAs:

- NFA of size n and width p

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

Let n = number of states.

Results. p = width(<) is an important parameter for NFAs:

- NFA of size n and width p ⇒ powerset construction

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

Let n = number of states.

Results. p = width(<) is an important parameter for NFAs:

- NFA of size n and width p ⇒ powerset construction ⇒ DFA of size ≤ $(\mathbf{n-p+1)\cdot 2^p}$ and width ≤ 2^p *

*consequence: NFA equivalence / universality (PSPACE-complete) are FPT w.r.t. p

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

Let n = number of states.

Results. p = width(<) is an important parameter for NFAs:

- NFA of size n and width p ⇒ powerset construction ⇒ DFA of size ≤ $\mathbf{(n-p+1)\cdot 2^p}$ and width ≤ 2^p *
- NFA compression: **log(p) + O(1)** bits per edge (rather than O(log n))

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

Let n = number of states.

Results. p = width(<) is an important parameter for NFAs:

- NFA of size n and width p ⇒ powerset construction ⇒ DFA of size ≤ **(n-p+1)•2^p** and width ≤ 2^p *
- NFA compression: **log(p) + O(1)** bits per edge (rather than O(log n))
- NFA membership / pattern matching: O(**p^2 loglog p**) time per character (rather than graph's size)

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

co-lex orders

Let n = number of states.

Results. p = width(<) is an important parameter for NFAs:

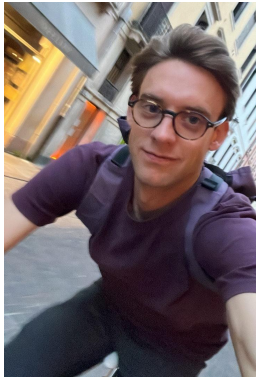
- NFA of size n and width p ⇒ powerset construction ⇒ DFA of size ≤ $(\mathbf{n-p+1})\cdot\mathbf{2^p}$ and width ≤ 2^p *
- NFA compression: **log(p) + O(1)** bits per edge (rather than O(log n))
- NFA membership / pattern matching: O($\mathbf{p^2}$ **loglog p**) time per character (rather than graph's size)
- Fast index construction with state-of-the-art algorithms.

Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, N. P., 2023. *Co-lexicographically Ordering Automata and Regular Languages - Part I*. JACM

Team & Funding

Funded by ERC StG “REGINDEX: Compressed indexes for regular languages with applications to computational pan-genomics” grant nr 101039208.

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.







Università Ca’Foscari Venezia

European Research Council
Established by the European Commission