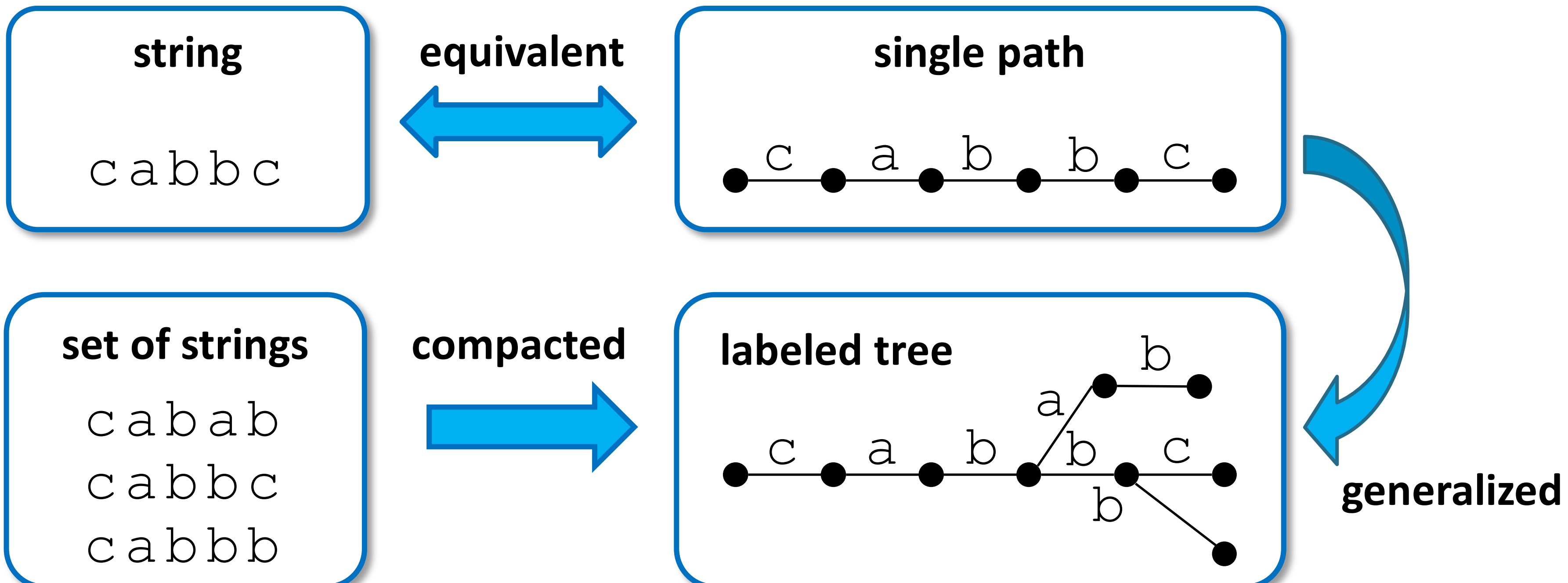
LATIN 2020

Suffix Trees, DAWGs and CDAWGs for Forward and Backward Tries

Shunsuke Inenaga
Kyushu University, Japan

Labeled Trees

- A **string** is a sequence of characters, which is equivalent to a single path where each edge is labeled.
- A **labeled tree** is a generalization of a string which has branches, and it can also be seen as a compact representation of a set of strings.



Labeled Tree Indexing Problem

- We deal with the indexing version of the pattern matching problem on labeled trees (a.k.a. tries).

Problem

Preprocess input: A trie $\mathcal{T}$.

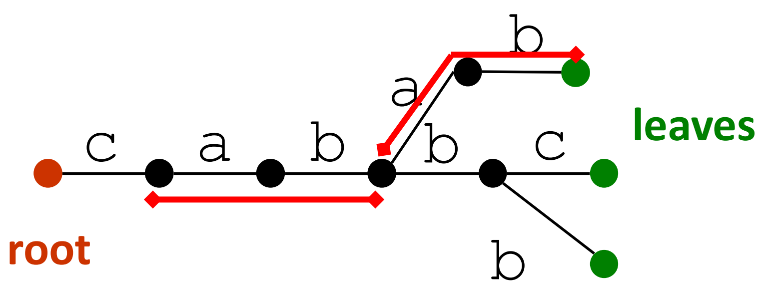
Query input: A pattern string P.

Query output: Every sub-path of $\mathcal{T}$ that matches P.

- We consider two version of tries:
  Forward Tries: paths are read from root to leaves.
  Backward Tries: paths are read from leaves to root.

Indexing Forward/Backward Tries

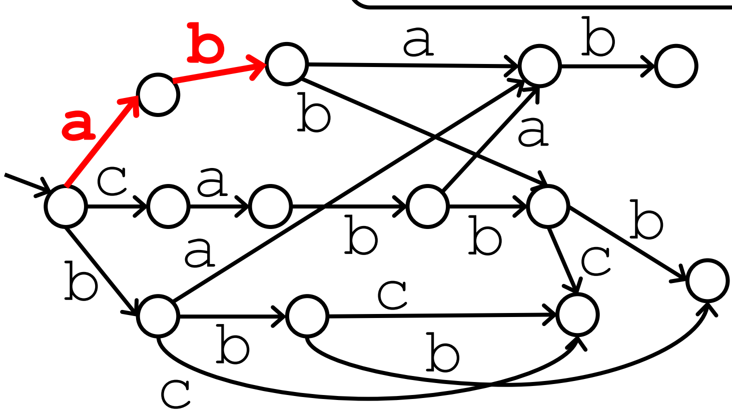
Forward Trie $\mathcal{T}$



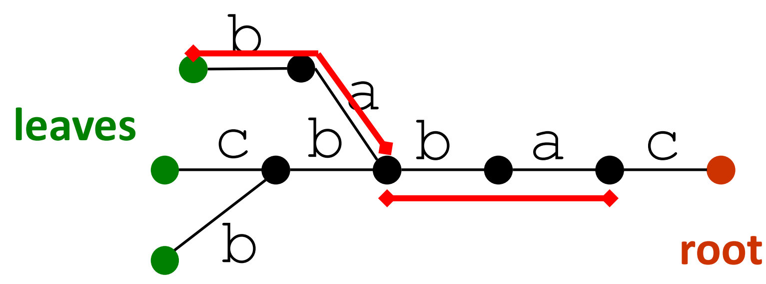
Pattern $P = \texttt{ab}$

DAWG($\mathcal{T}$)

Partial DFA that accepts all substrings of $\mathcal{T}$.



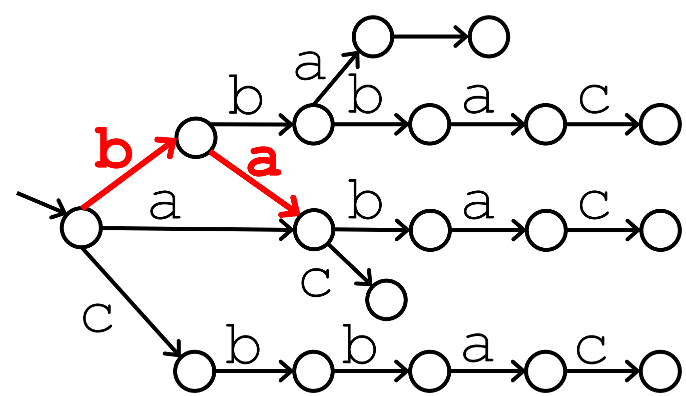
Backward Trie $\mathcal{T}^R$



Pattern $P^R = \texttt{ba}$

DAWG($\mathcal{T}^R$)

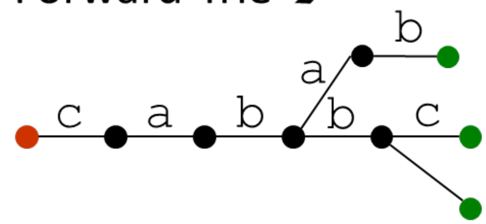
Partial DFA that accepts all substrings of $\mathcal{T}^R$.



Indexing Forward Tries

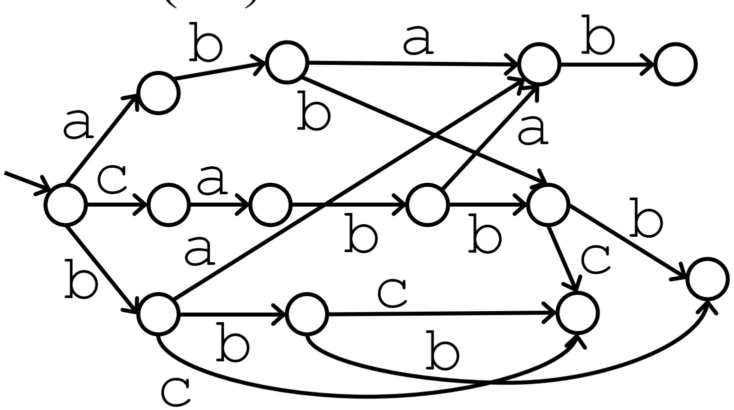
Three kinds of indexing structures for Forward Trie $\mathcal{T}$.

Forward Trie $\mathcal{T}$



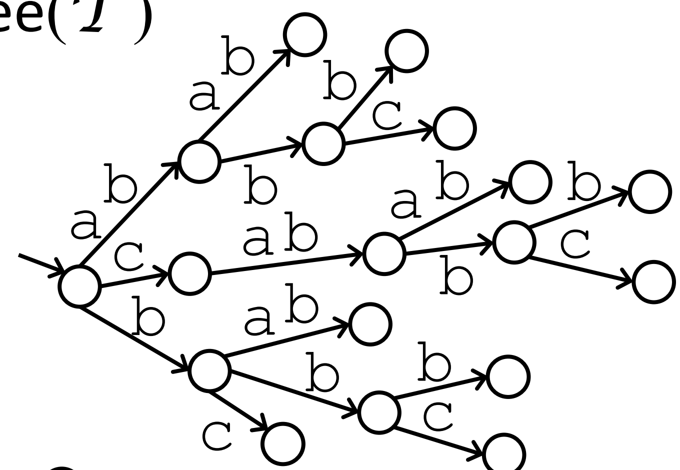
DAWG($\mathcal{T}$)

Partial DFA that accepts all substrings of $\mathcal{T}$.



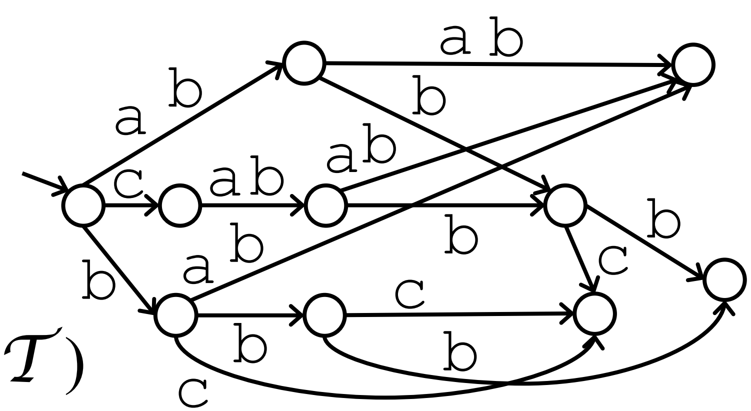
SuffixTree($\mathcal{T}$)

Compact trie representing all substrings of $\mathcal{T}$.



Compact DAG representing all substrings of $\mathcal{T}$.

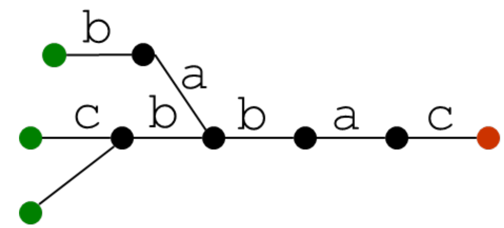
CDAWG($\mathcal{T}$)



Indexing Backward Tries

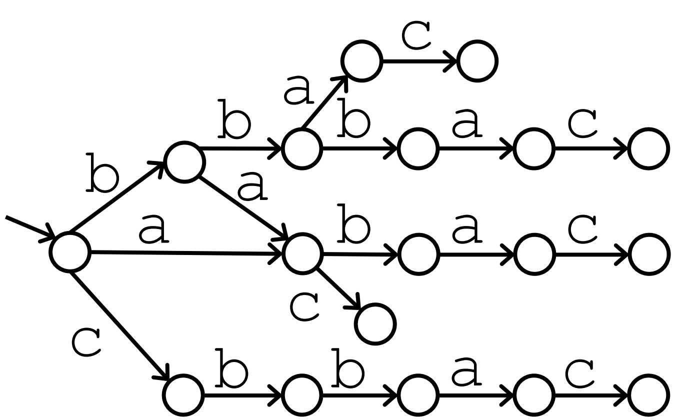
Three kinds of indexing structures for Backward Trie $\mathcal{T}^R$.

Backward Trie $\mathcal{T}^R$



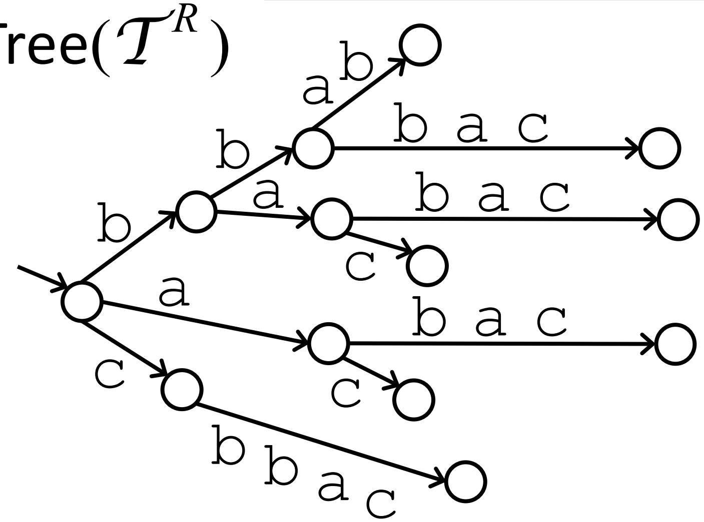
DAWG($\mathcal{T}^R$)

Partial DFA that accepts all substrings of $\mathcal{T}^R$.



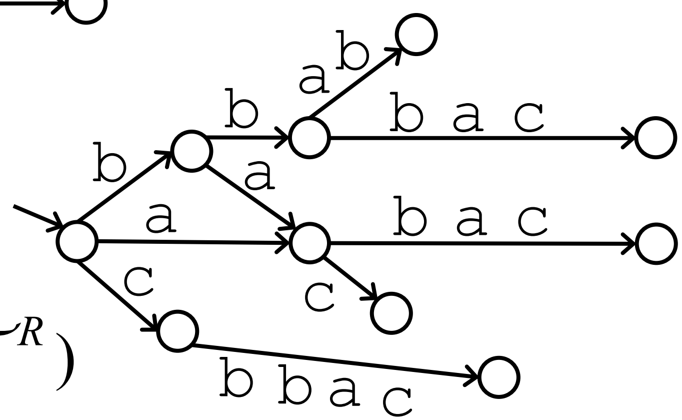
SuffixTree($\mathcal{T}^R$)

Compact trie representing all substrings of $\mathcal{T}^R$.



Compact DAG representing all substrings of $\mathcal{T}^R$.

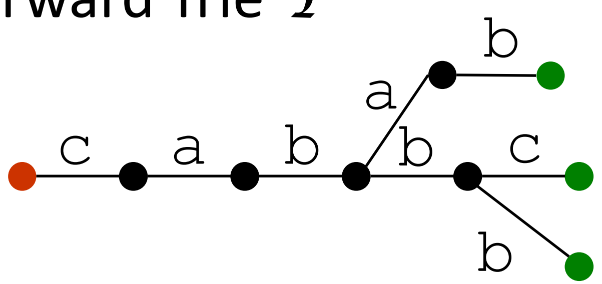
CDAWG($\mathcal{T}^R$)



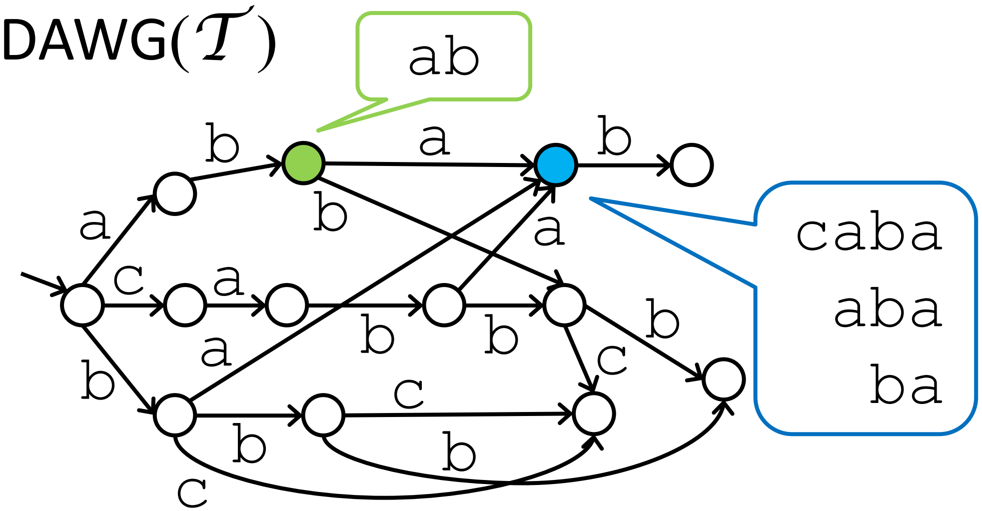
DAWGs (Directed Acyclic Word Graphs)

- A substring (i.e. sub-path) X of a forward trie $\mathcal{T}$ is said to be **left-maximal** if (1) there are two distinct characters a, b such that both aX and bX are substrings of $\mathcal{T}$, or (2) X has an occurrence begging at the root of $\mathcal{T}$.

Forward Trie $\mathcal{T}$



DAWG($\mathcal{T}$)

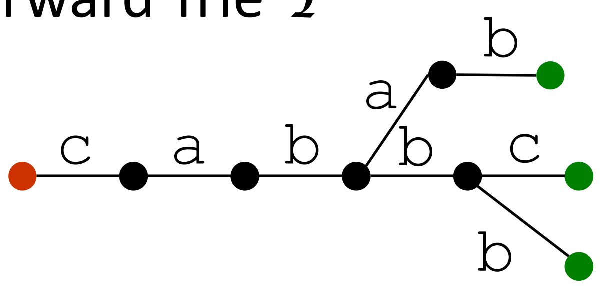


- This generalizes Blumer et al.'s DAWGs for strings to trees.
- DAWGs for backward tries $\mathcal{T}^R$ are defined similarly.

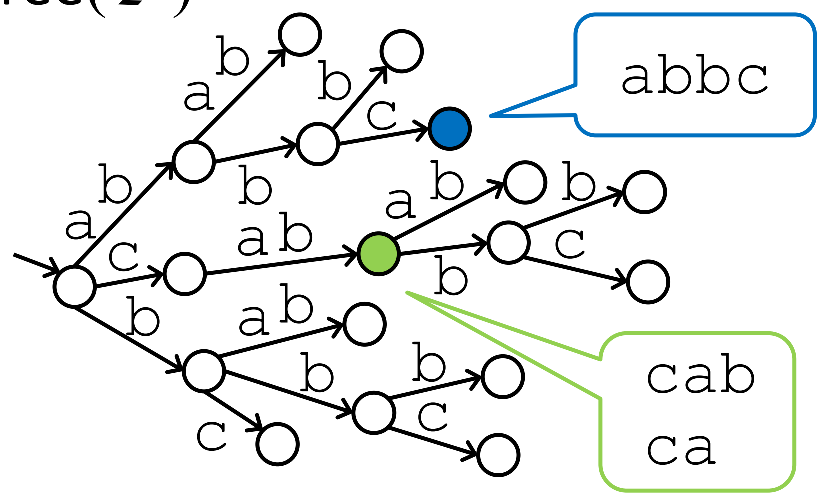
Suffix Trees

- A substring (i.e. sub-path) X of a forward trie $\mathcal{T}$ is said to be **right-maximal** if (1) there are two distinct characters a, b such that both Xa and Xb are substrings of $\mathcal{T}$, or (2) X has an occurrence ending at a leaf of $\mathcal{T}$.

Forward Trie $\mathcal{T}$



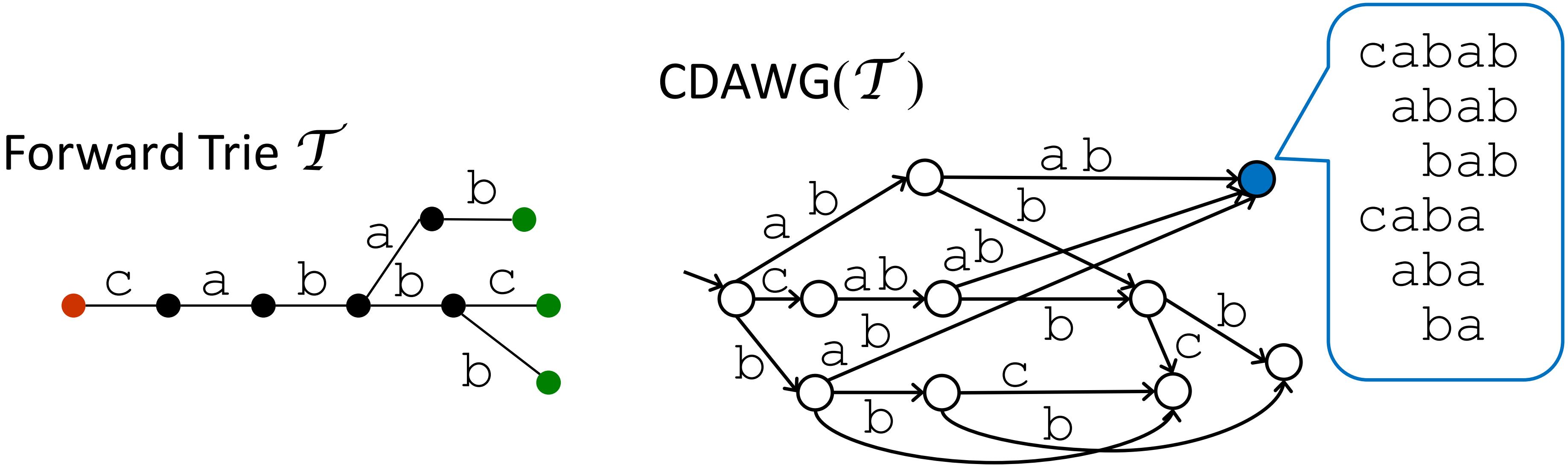
SuffixTree($\mathcal{T}$)



- This generalizes Weiner's suffix trees for strings to trees.
- Suffix trees for backward tries $\mathcal{T}^R$ are defined similarly.

CDAWGs (Compact DAWGs)

- A substring (i.e. sub-path) X of a forward trie $\mathcal{T}$ is said to be **bi-maximal** if X is both left-maximal and right-maximal in $\mathcal{T}$.



- Intuitively, CDAWGs are mixture of DAWGs and Suffix Trees.
- This generalizes Blumer et al.'s CDAWGs for strings to trees.
- CDAWGs for backward tries $\mathcal{T}^R$ are defined similarly.

Sizes of Indexing Structures for Tries

Size Bounds of Indexing Structures for Tries (**Existing Work**)

upper bounds

	Forward Trie $\mathcal{T}$		Backward Trie $\mathcal{T}^R$	
Index. structures	# nodes	# edges	# nodes	# edges
DAWG	$O(n)$	–	–	–
CDAWG	–	–	–	–
Suffix Tree	–	–	$O(n)$	$O(n)$
Suffix Array	–		$n-1$	

n is # of nodes in the input trie.

Sizes of Indexing Structures for Tries

Size Bounds of Indexing Structures for Tries (**This Work**)

upper bounds

	Forward Trie $\mathcal{T}$		Backward Trie $\mathcal{T}^R$	
Index. structures	# nodes	# edges	# nodes	# edges
DAWG	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$
CDAWG	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\mathbf{2}\boldsymbol{n}\mathbf{-4}$
Suffix Tree	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\mathbf{2}\boldsymbol{n}\mathbf{-4}$
Suffix Array	$\mathbf{O}(\boldsymbol{n}^\mathbf{2})$		$n-1$	

n is # of nodes in the input trie.

Note: For a string (i.e. path tree) with n characters, the sizes of these indexing structures are all $\mathrm{O}(n)$.

Matching Upper/Lower Bounds

Size Bounds of Indexing Structures for Tries (**This Work**)

upper bounds

	Forward Trie $\mathcal{T}$		Backward Trie $\mathcal{T}^R$	
Index. structures	# nodes	# edges	# nodes	# edges
DAWG	$\mathbf{2n-3}$	$\mathbf{O(n^2)}$	$\mathbf{O(n^2)}$	$\mathbf{O(n^2)}$
CDAWG	$\mathbf{2n-3}$	$\mathbf{O(n^2)}$	$\mathbf{2n-3}$	$\mathbf{2n-4}$
Suffix Tree	$\mathbf{O(n^2)}$	$\mathbf{O(n^2)}$	$\mathbf{2n-3}$	$\mathbf{2n-4}$
Suffix Array	$\mathbf{O(n^2)}$		$n-1$	

lower bounds

	Forward Trie $\mathcal{T}$		Backward Trie $\mathcal{T}^R$	
Index. structures	# nodes	# edges	# nodes	# edges
DAWG	$\mathbf{2n-3}$	$\mathbf{\Omega(n^2)}$	$\mathbf{\Omega(n^2)}$	$\mathbf{\Omega(n^2)}$
CDAWG	$\mathbf{2n-3}$	$\mathbf{\Omega(n^2)}$	$\mathbf{2n-3}$	$\mathbf{2n-4}$
Suffix Tree	$\mathbf{\Omega(n^2)}$	$\mathbf{\Omega(n^2)}$	$\mathbf{2n-3}$	$\mathbf{2n-4}$
Suffix Array	$\mathbf{\Omega(n^2)}$		$n-1$	

Linear-size Indexing for Forward Tries

Size Bounds of Indexing Structures for Tries (**This Work**)

upper bounds

	Forward Trie $\mathcal{T}$		Backward Trie $\mathcal{T}^R$	
Index. structures	# nodes	# edges	# nodes	# edges
DAWG	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\boxed{\boldsymbol{\Theta(n^2)}}$	$\boldsymbol{\Theta(n^2)}$	$\boldsymbol{\Theta(n^2)}$
CDAWG	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\boldsymbol{\Theta(n^2)}$	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\mathbf{2}\boldsymbol{n}\mathbf{-4}$
Suffix Tree	$\boldsymbol{\Theta(n^2)}$	$\boldsymbol{\Theta(n^2)}$	$\mathbf{2}\boldsymbol{n}\mathbf{-3}$	$\mathbf{2}\boldsymbol{n}\mathbf{-4}$
Suffix Array	$\boldsymbol{\Theta(n^2)}$		$n-1$	

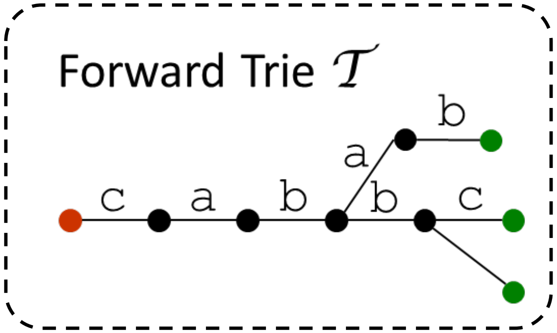
Theorem [This Work]

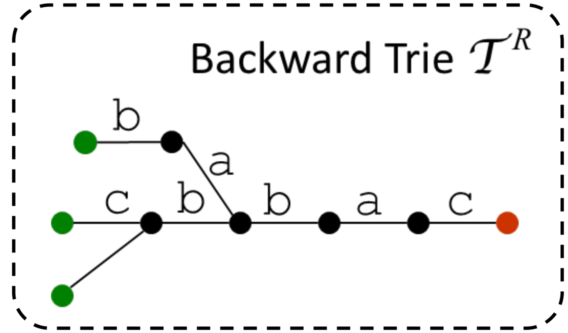
There exists an <u>**$\mathrm{O}(n)$-size compact representation**</u> of the DAWG for forward trie $\mathcal{T}$ which can be built in $\mathrm{O}(n)$ time.

Also, this compact representation supports bidirectional pattern matching queries on the trie in $\mathrm{O}(m \log \sigma + occ)$ time.

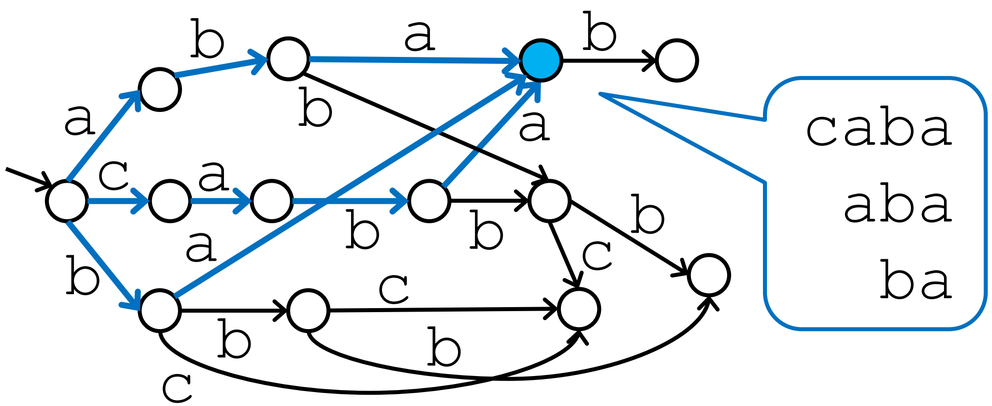
n: # nodes in $\mathcal{T}$, m: pattern length, σ: alphabet size, occ: # pattern occurrences

Connections of DAWG($\mathcal{T}$) and SuffixTree($\mathcal{T}^R$)

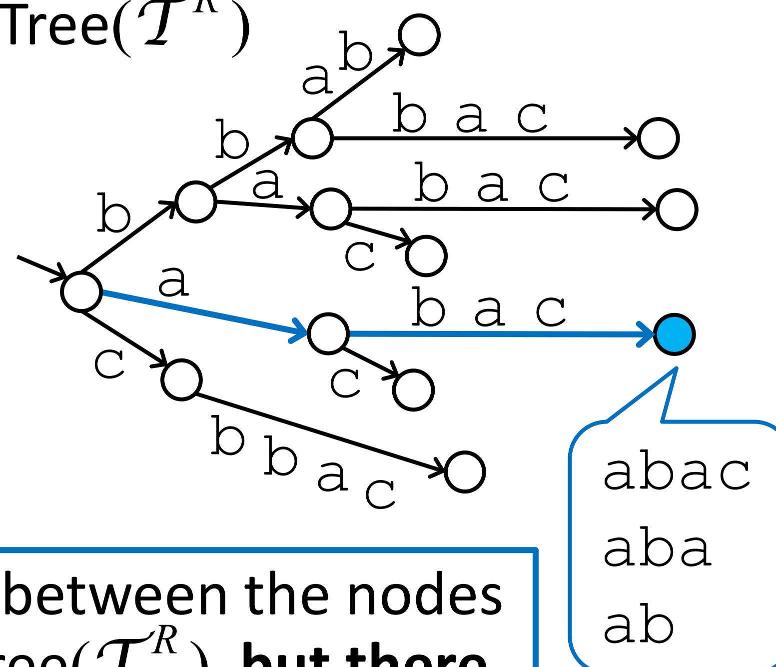




DAWG($\mathcal{T}$)



SuffixTree($\mathcal{T}^R$)



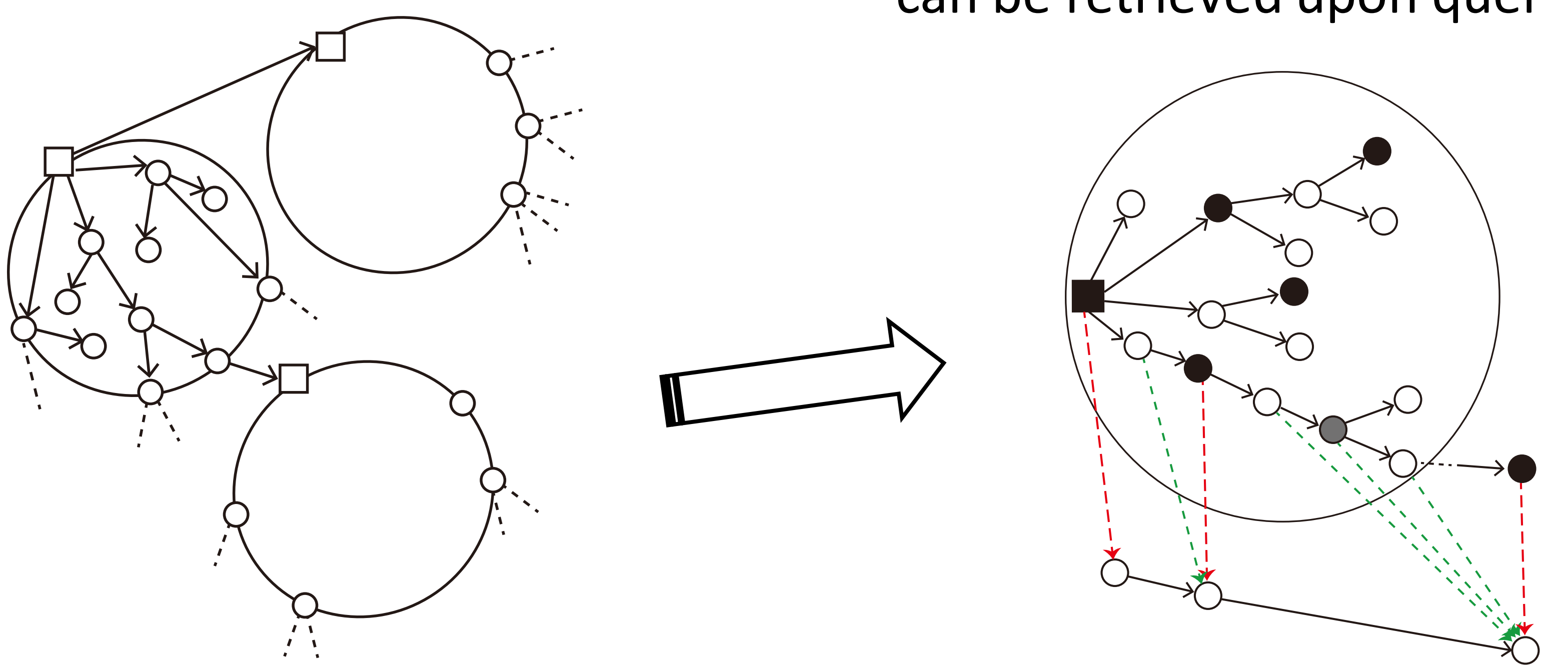
There is a one-to-one correspondence between the nodes of DAWG($\mathcal{T}$) and the nodes of SuffixTree($\mathcal{T}^R$), **<u>but there is no such correspondence between their edges.</u>**

Simulating DAWG($\mathcal{T}$) edges with SuffixTree($\mathcal{T}^R$)

We can simulate all the $O(n^2)$ edges of DAWG($\mathcal{T}$) with SuffixTree($\mathcal{T}^R$) using only $O(n)$ space.

Decompose SuffixTree($\mathcal{T}^R$) into $O(n/\sigma)$ clusters, $O(\sigma)$-size each, where σ is the alphabet size.

Store carefully-selected DAWG edges in each cluster, so that the other DAWG edges can be retrieved upon query.



Conclusions and Open Question

- We have shown a complete perspective on the size bounds of classical indexing structures for forward tries and backward tries.
- We can simulate the DAWG for a forward trie with the suffix tree for a backward trie, using $O(n)$ space.
- Can we simulate the CDAWG for a forward trie with the CDAWG for a backward trie, using $O(n)$ space?

	Forward Trie $\mathcal{T}$		Backward Trie $\mathcal{T}^R$	
Index. structures	# nodes	# edges	# nodes	# edges
DAWG	$2n-3$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$
CDAWG	$2n-3$	$\Theta(n^2)$	$2n-3$	$2n-4$
Suffix Tree	$\Theta(n^2)$	$\Theta(n^2)$	$2n-3$	$2n-4$
Suffix Array	$\Theta(n^2)$		$n-1$	