Directed Acyclic Word Graphs and Their Efficient Implementations

Shunsuke Inenaga

Department of Computer Science
University of Helsinki

Short CV

Shunsuke Inenaga

March 2003	PhD Kyushu University, Japan “String Processing Algorithms”
April – Sept. 2003	Posdoc of Japan Science Technology Agency (JST)
Sept. 2003 -	Posdoc of Dep. of Computer Science, University of Helsinki

http://www.i.kyushu-u.ac.jp/~s-ine/

Pattern Matching Problem

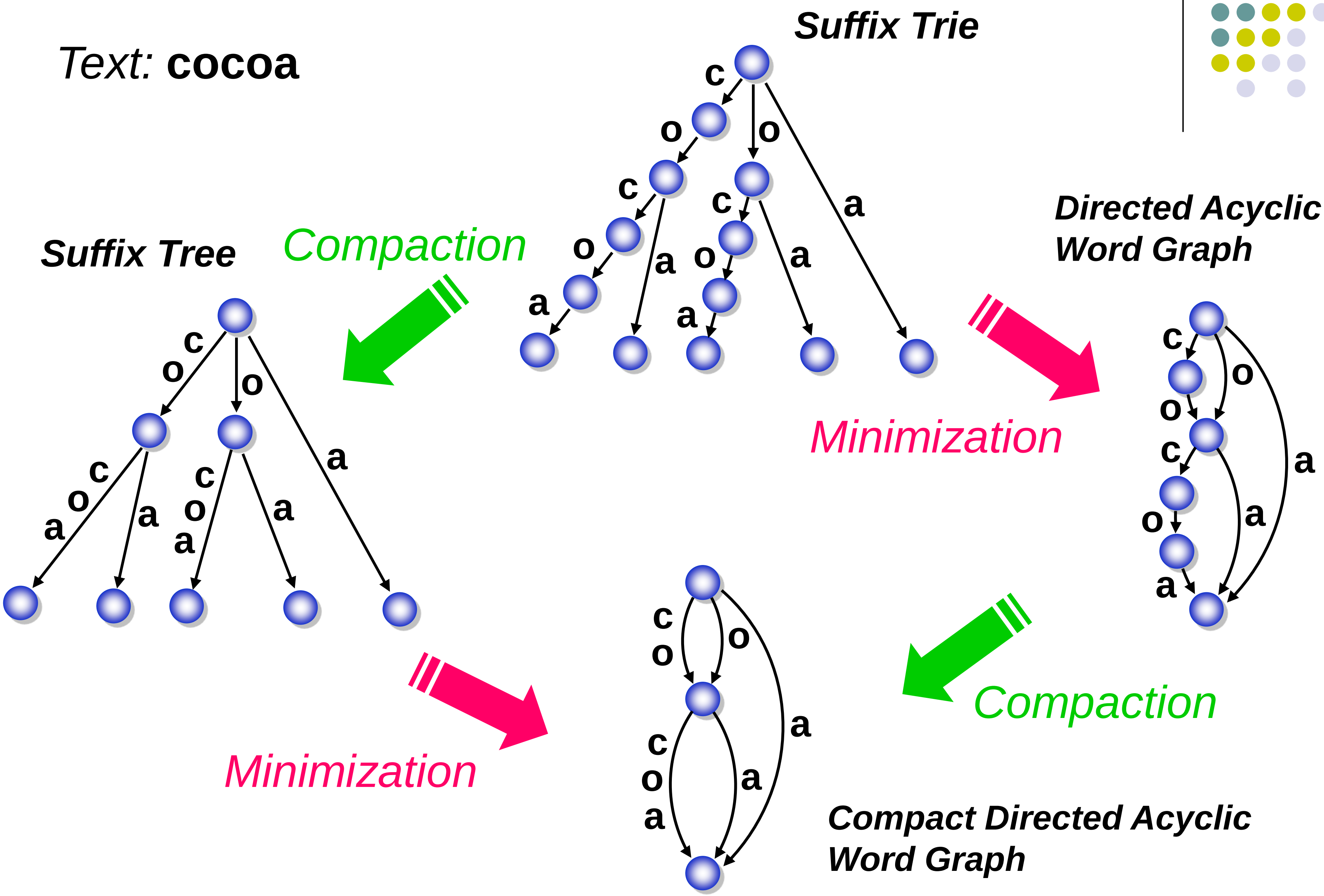
*Input: text string **T** and pattern string **P**.*

*Output: all occurrences of **P** in **T**.*

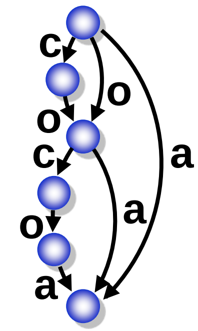
When T is fixed ***Text Indexing Structures***

Text Indexing Structures

Text: **cocoa**



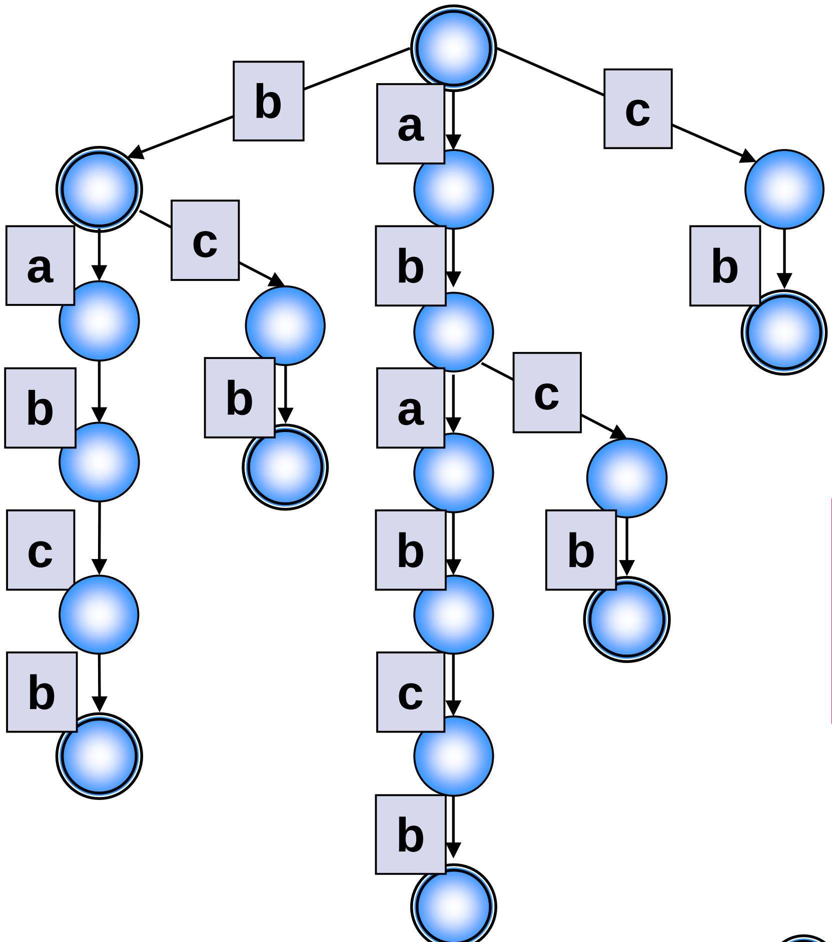
Directed Acyclic Word Graphs



The Directed Acyclic Word Graph (DAWG) of string w *is* ***the smallest DFA that accepts all suffixes of*** w.

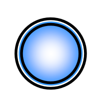
DAWG = Suffix Automaton

Conversion of Suffix Trie to DAWG



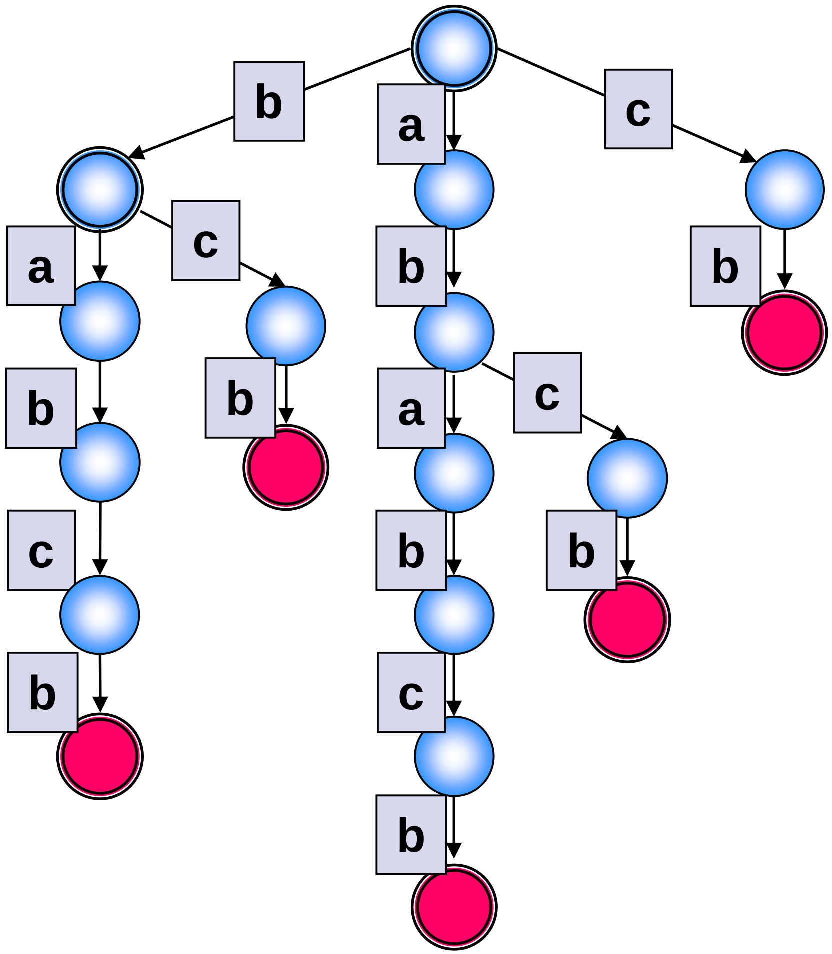
ababcb

Merge isomorphic subtrees



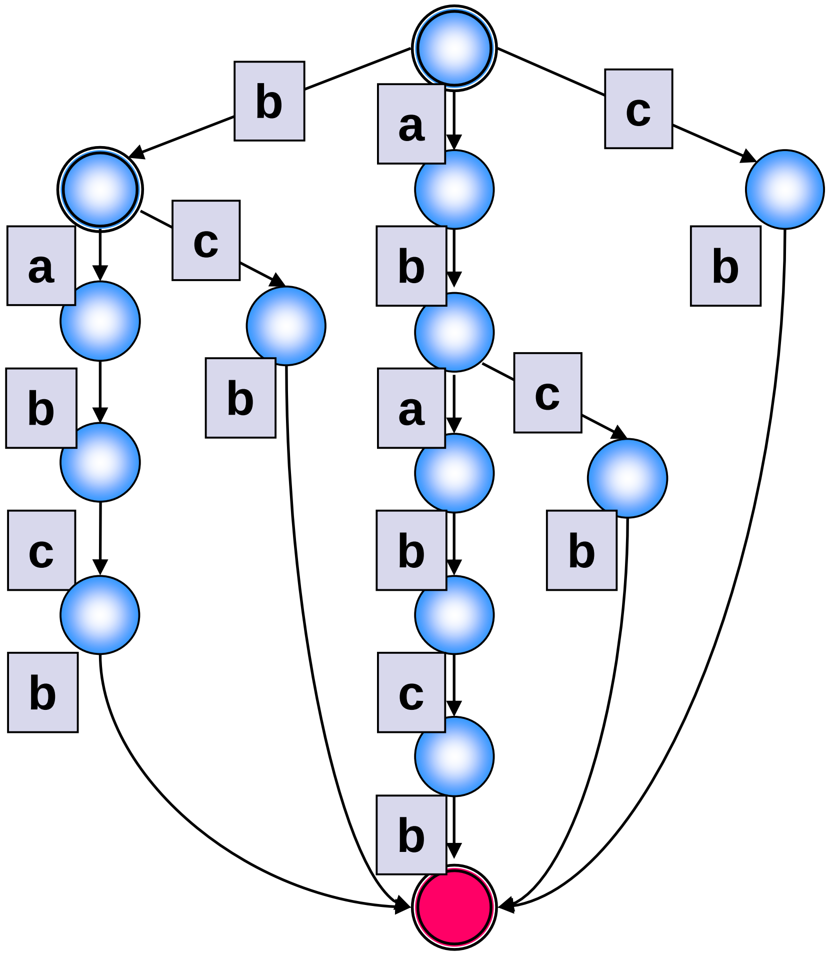
accepting node for suffixes

Conversion to DAWG



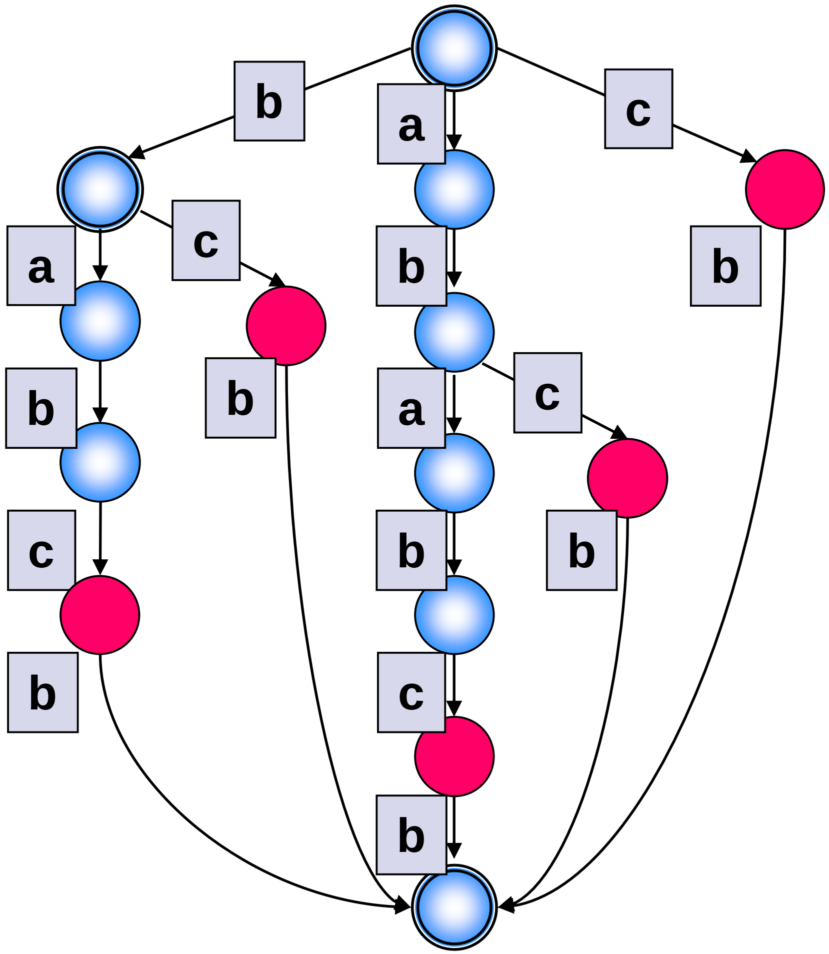
ababcb

Conversion to DAWG [Cont.]



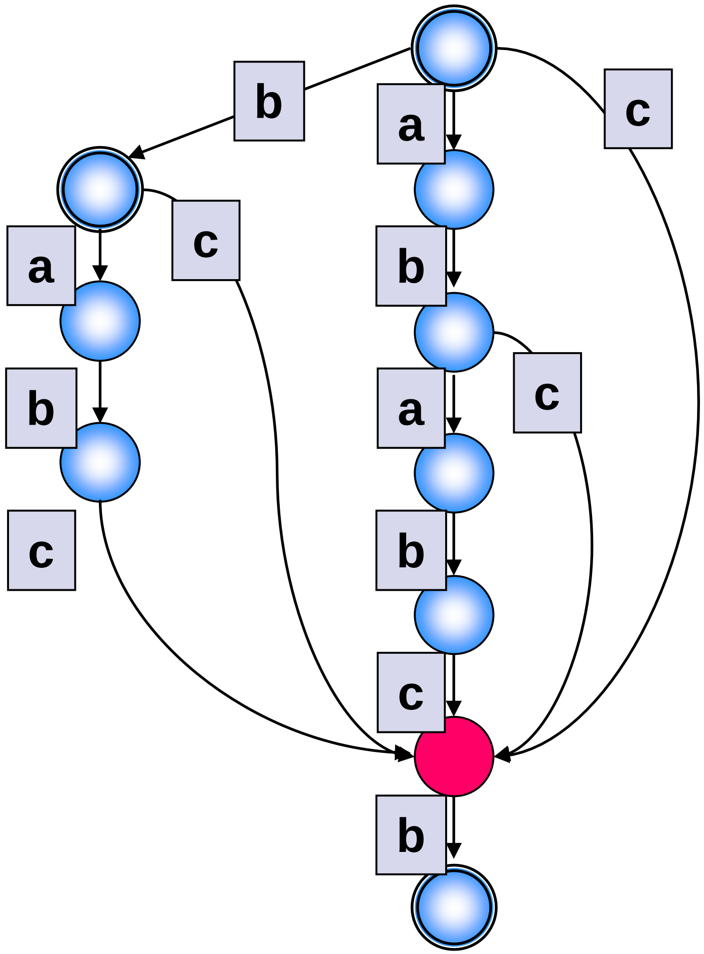
ababcb

Conversion to DAWG [Cont.]



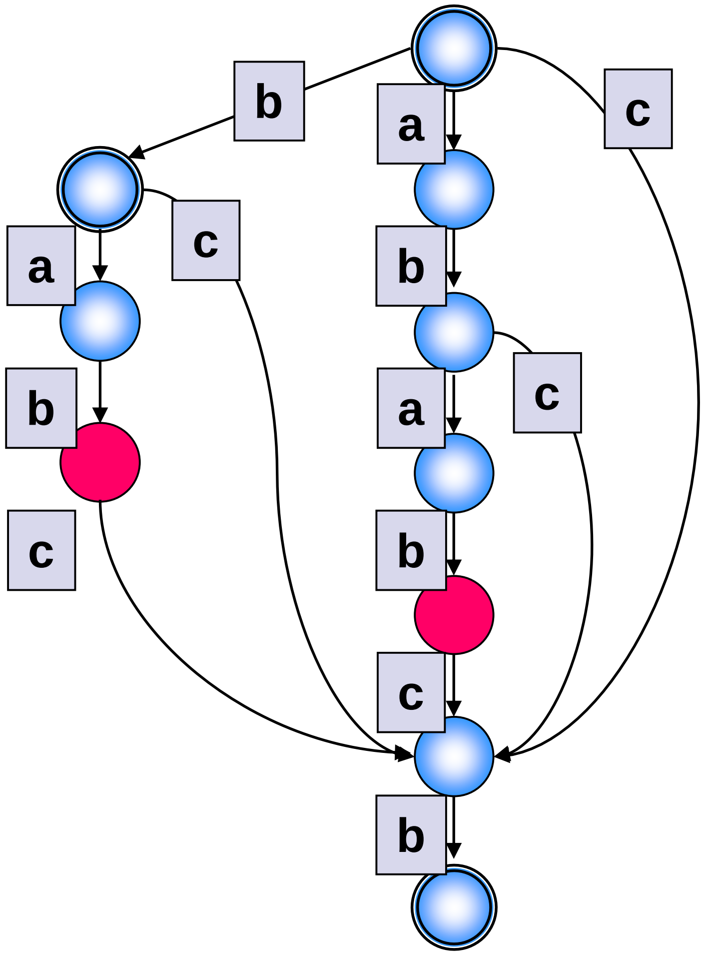
ababcb

Conversion to DAWG [Cont.]



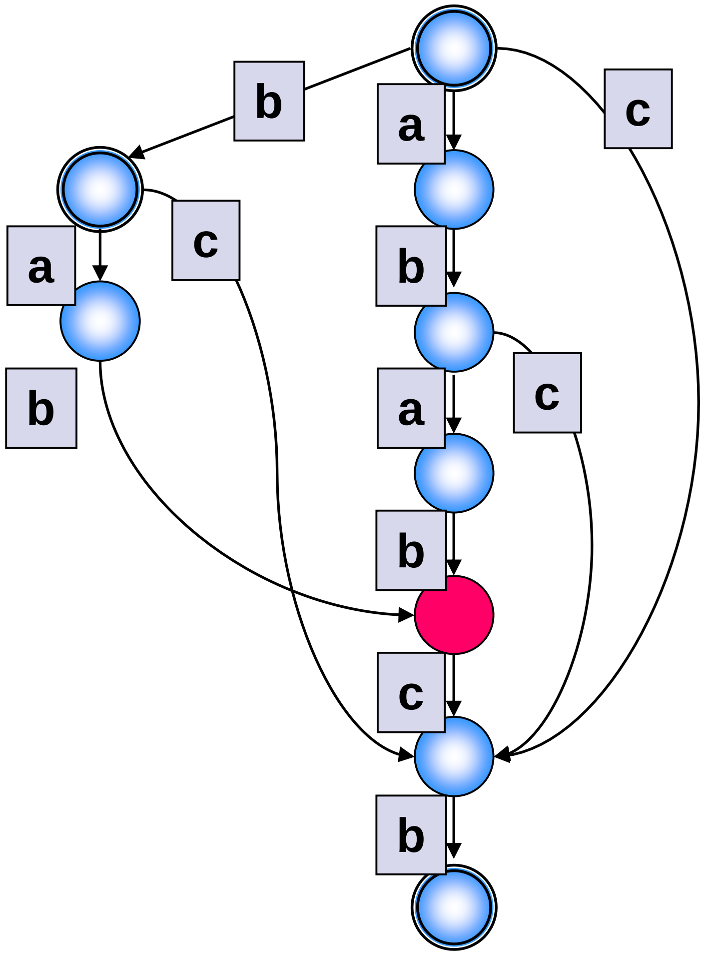
ababcb

Conversion to DAWG [Cont.]



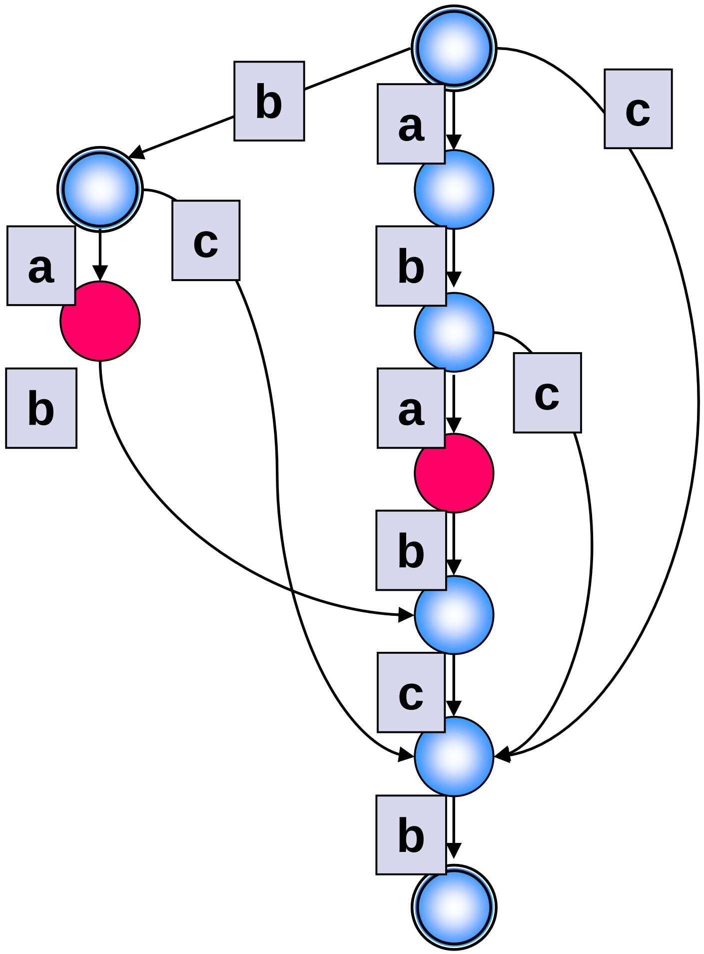
ababcb

Conversion to DAWG [Cont.]



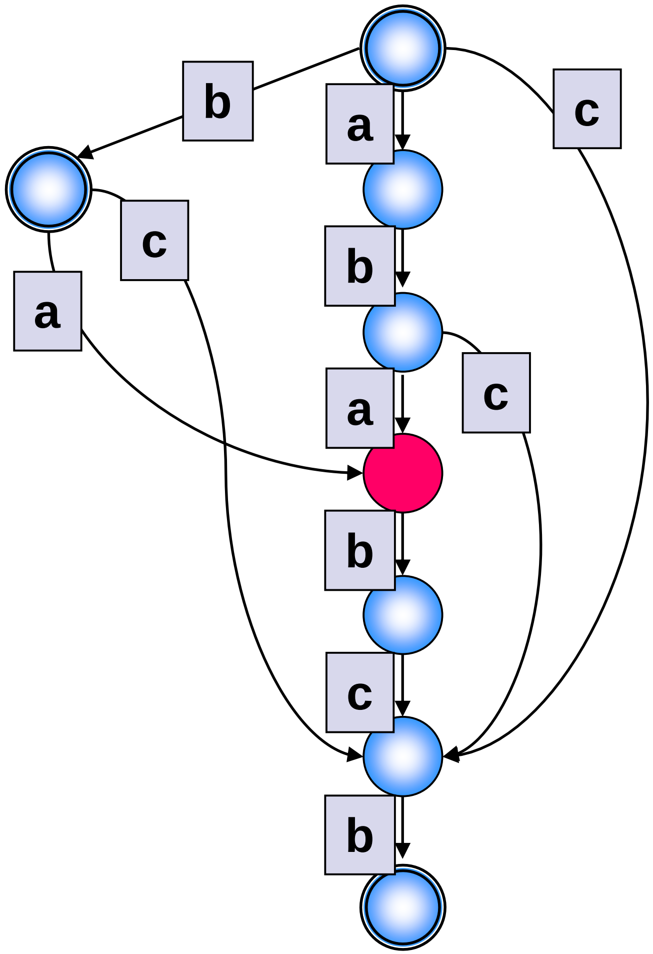
ababcb

Conversion to DAWG [Cont.]



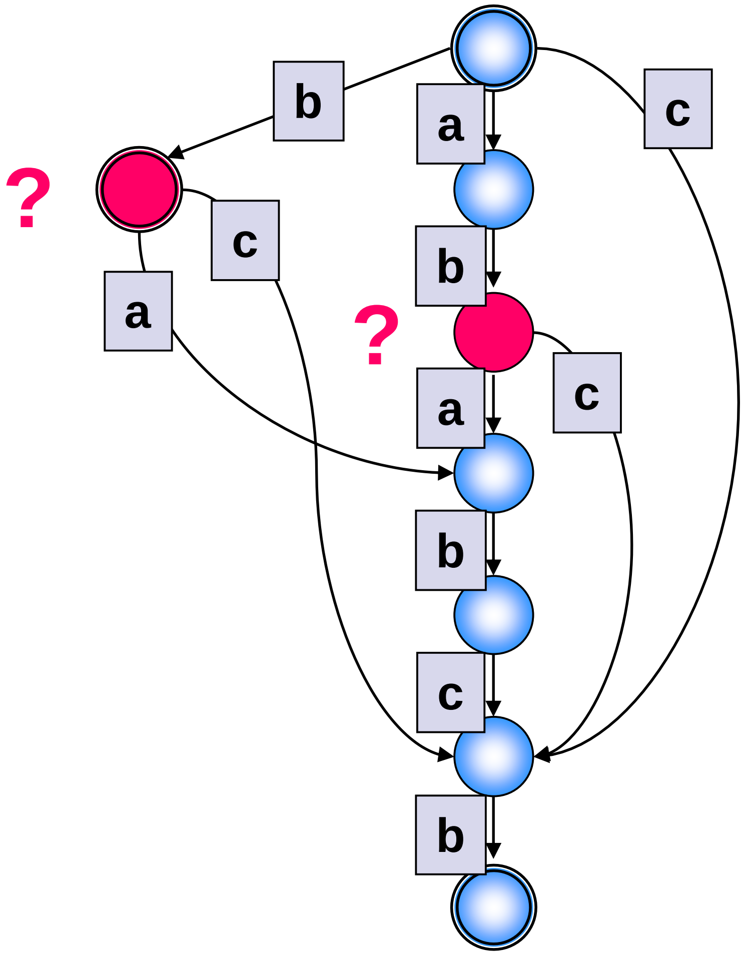
ababcb

Conversion to DAWG [Cont.]



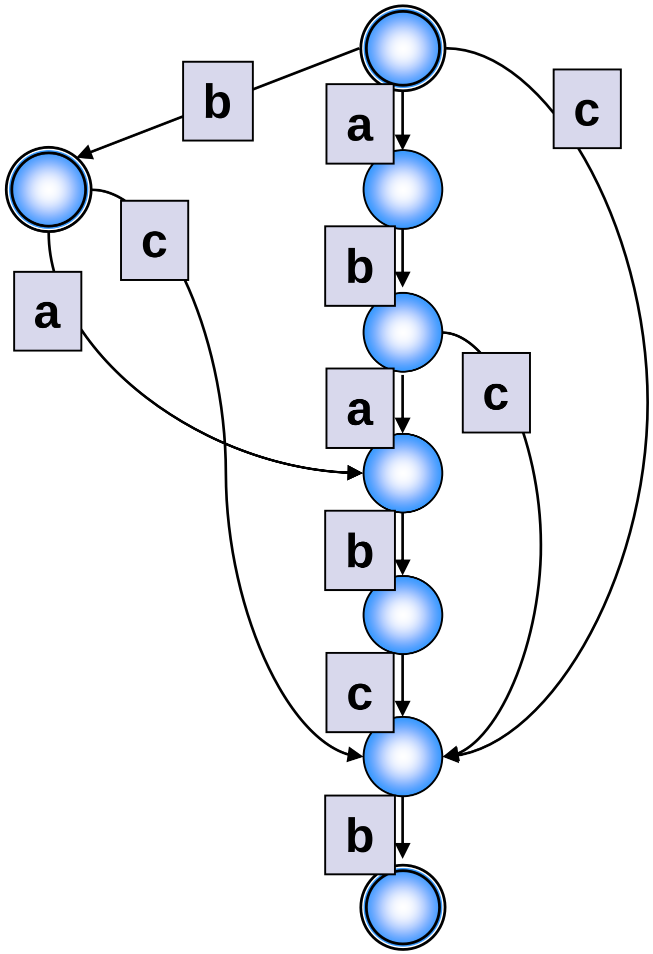
ababcb

Conversion to DAWG [Cont.]



ababcb

DAWG Completed



ababcb

Equivalence Class on DAWG

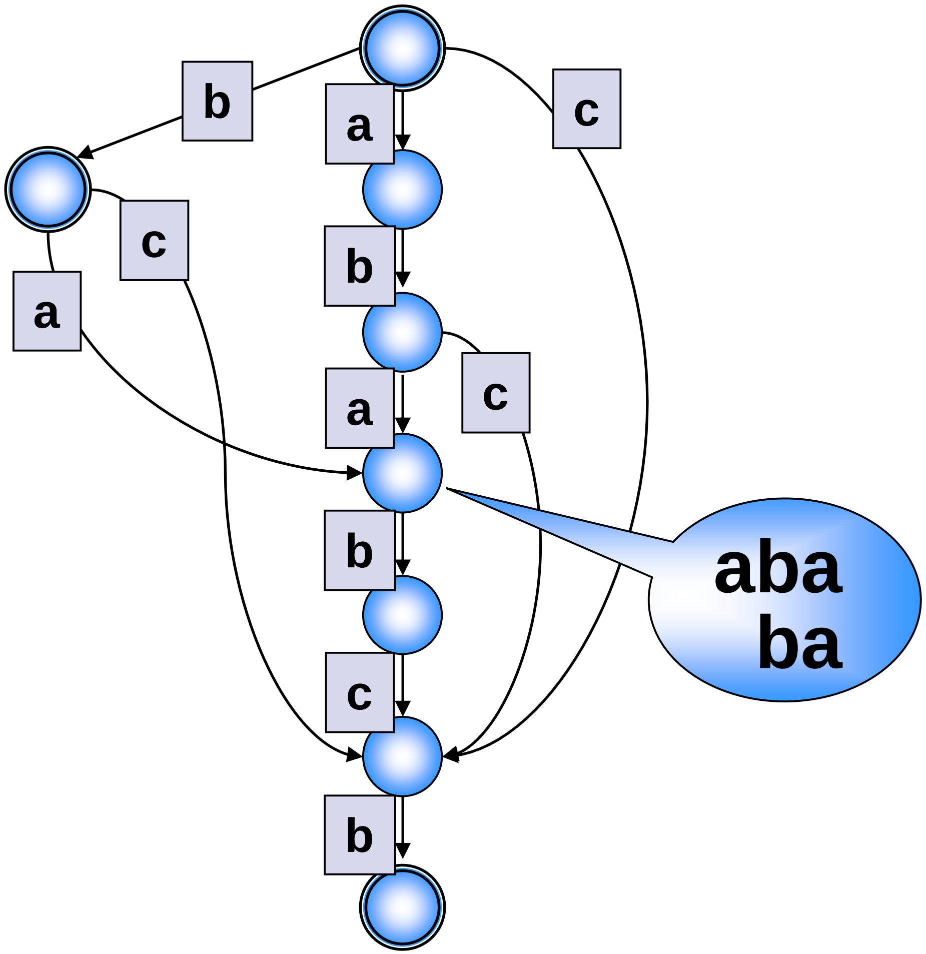
substrings x,y *of* w *are recognized by the same node of* $\mathbf{DAWG}(w)$

$\Updownarrow$ iff

$$EndPos_w(x) = EndPos_w(y)$$

$EndPos_w(x)$: *the set of all positions of* w *where* x *ends*

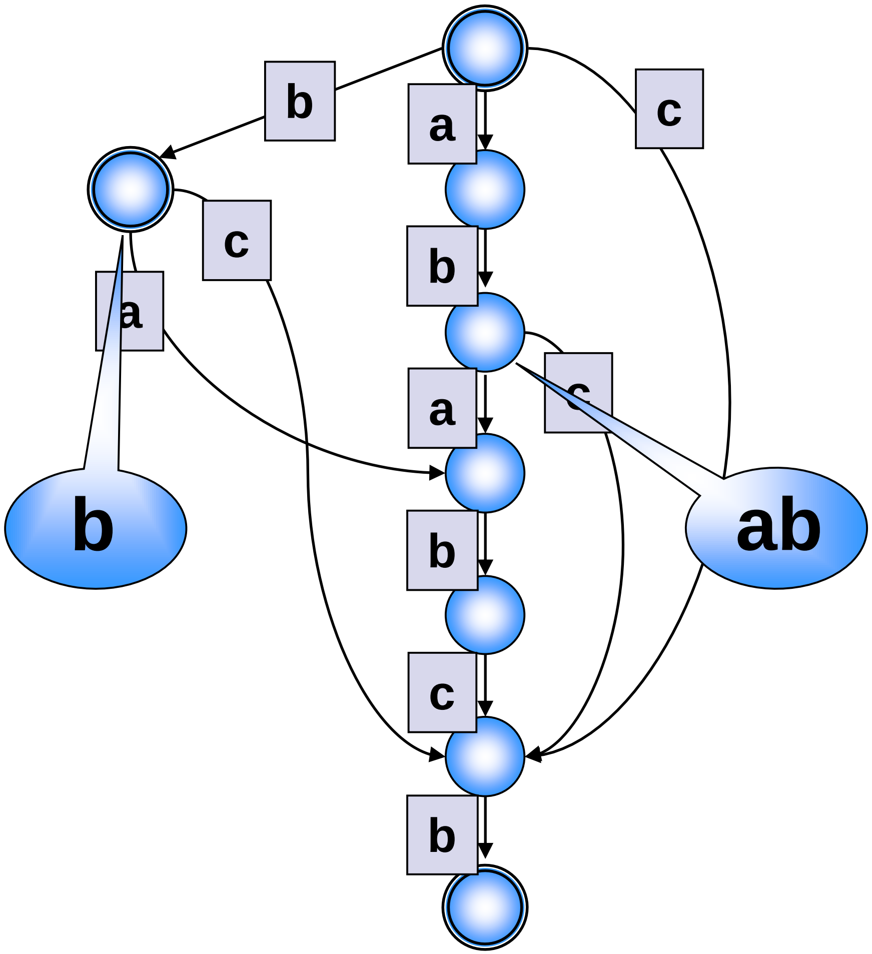
Equivalence Class on DAWG [Cont.]



ababcb
1 2 3 4 5 6

$EndPos_w(\textbf{aba}) = \{3\}$
$EndPos_w(\textbf{ba}) = \{3\}$

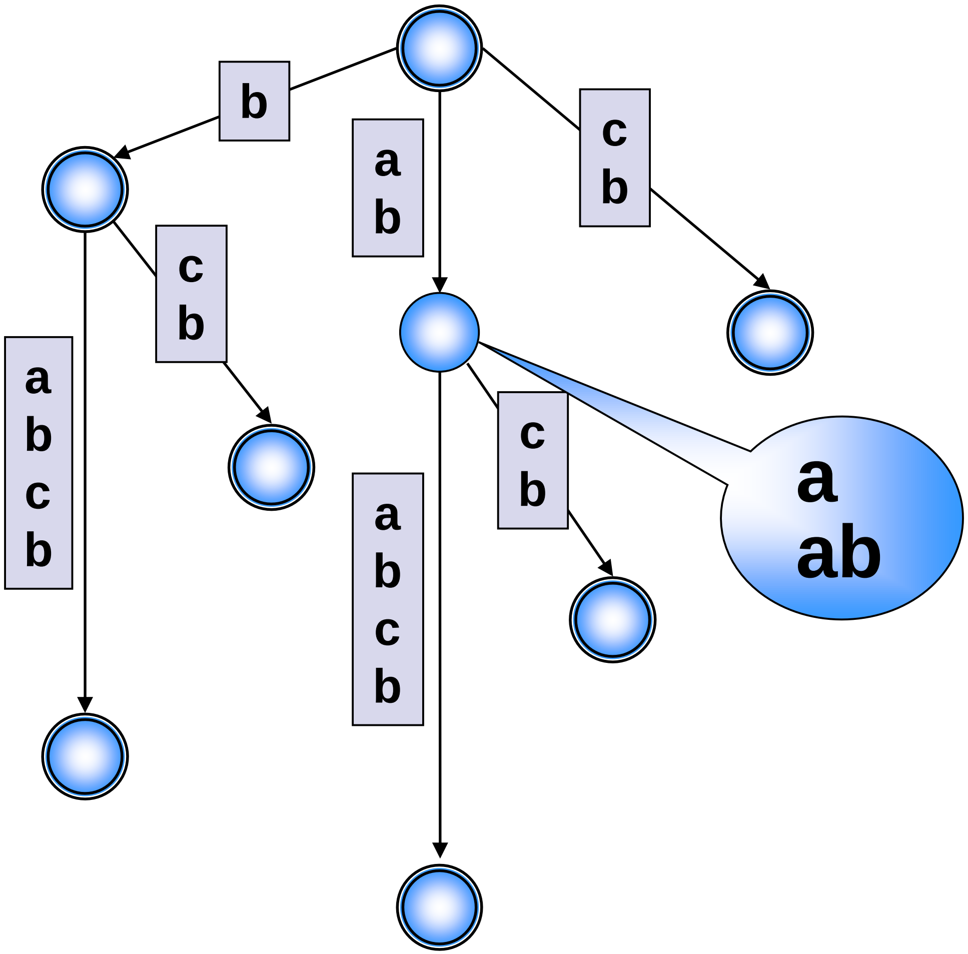
Equivalence Class on DAWG [Cont.]



ababcb
1 2 3 4 5 6

$EndPos_w(\mathbf{ab}) = \{2,4\}$
$EndPos_w(\mathbf{b}) = \{2,4,6\}$

Equivalence Class on Suffix Tree



ababcb
1 2 3 4 5 6

$BegPos_w(\text{a}) = \{1,2\}$
$BegPos_w(\text{ab}) = \{1,2\}$

Size of DAWG

Theorem. (Blumer et al. 1985)

Let $\boldsymbol{n} = |\boldsymbol{w}|$.
If $\boldsymbol{n} > \mathbf{1}$, $\mathbf{DAWG}(\boldsymbol{w})$ *has at most* $\boldsymbol{2n\text{-}1}$ *nodes and* $\boldsymbol{3n\text{-}3}$ *edges.*

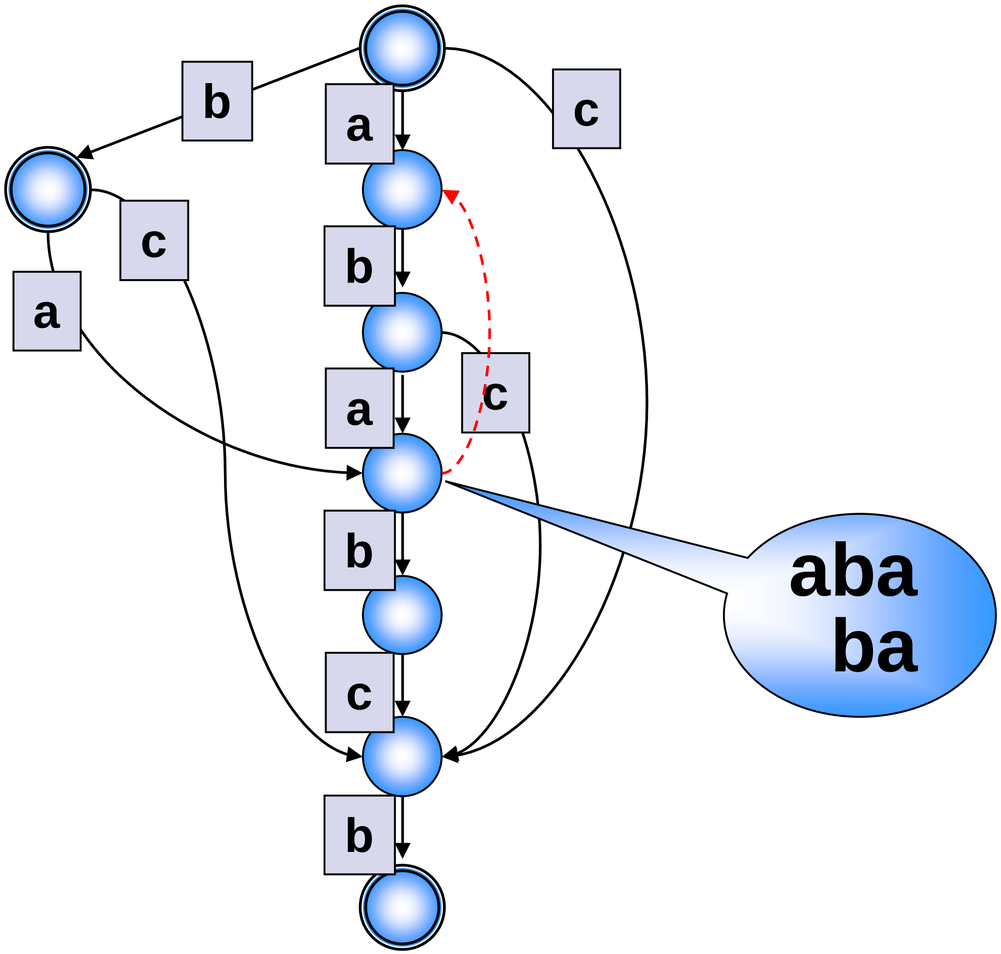
If $\boldsymbol{n} = \mathbf{1}$, $\mathbf{DAWG}(\boldsymbol{w})$ *has exactly* **2** *nodes and* $\mathbf{1}$ *edge.*

Construction of DAWG

Theorem. (Blumer et al. 1985)

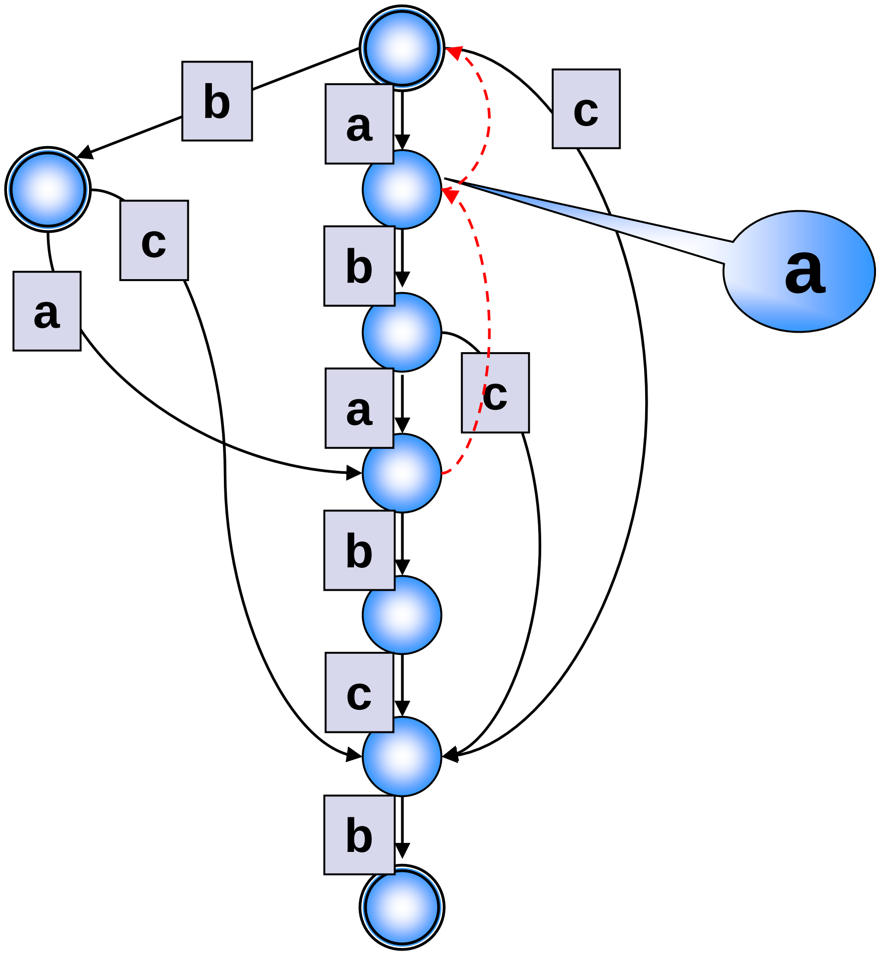
Assume Σ *is fixed.*
For any string w *of length* n,
$DAWG(w)$ *can be constructed on-line*
and in $O(n)$ *time and space.*

Suffix Links



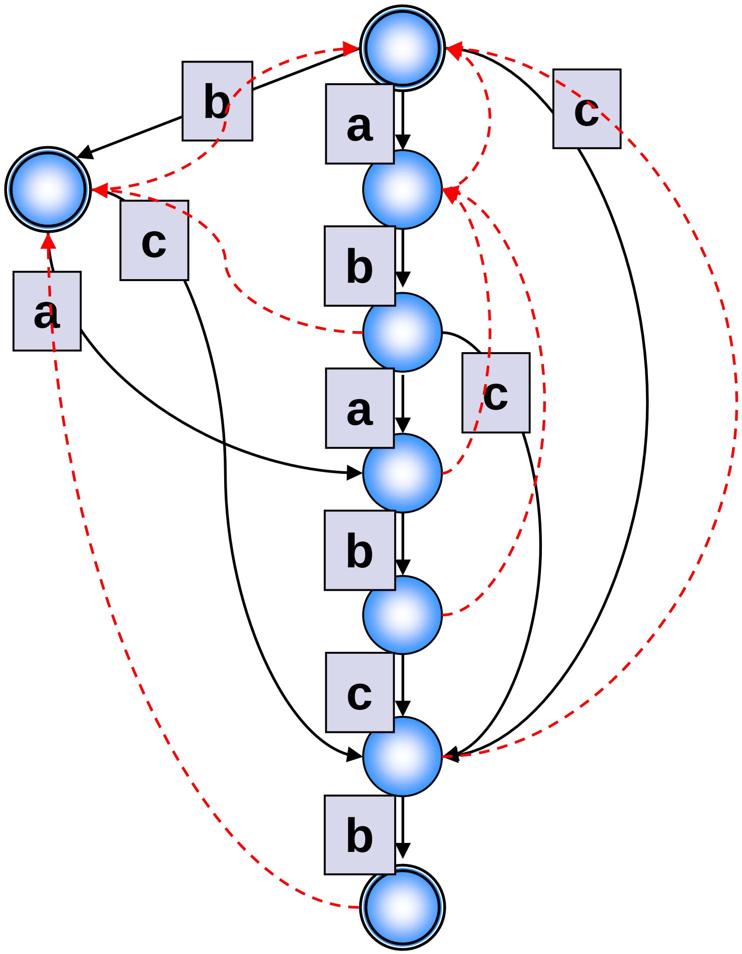
ababcb

Suffix Links



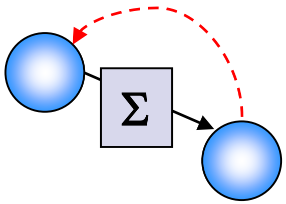
ababcb

Suffix Links



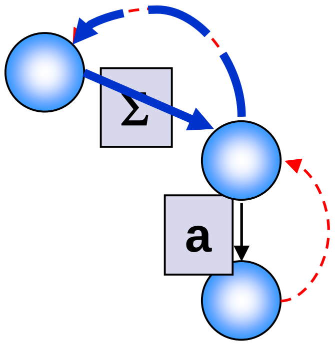
ababcb

On-line Construction of DAWG



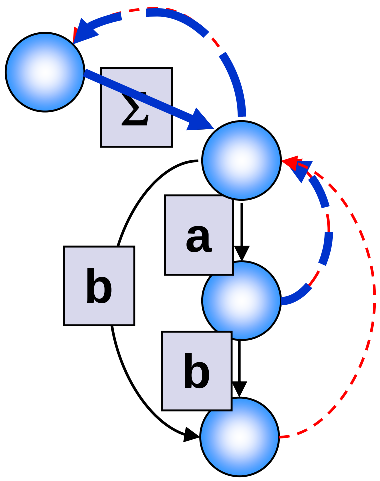
ababcb

On-line Construction of DAWG



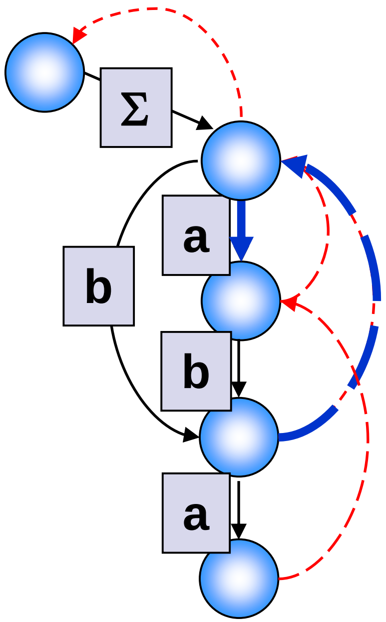
ababcb

On-line Construction of DAWG



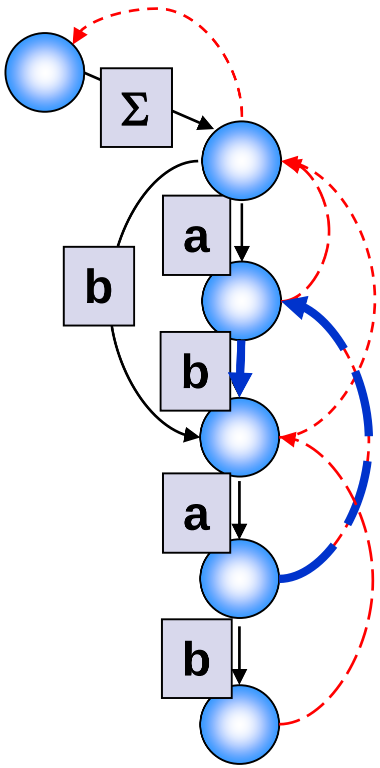
ababcb

On-line Construction of DAWG



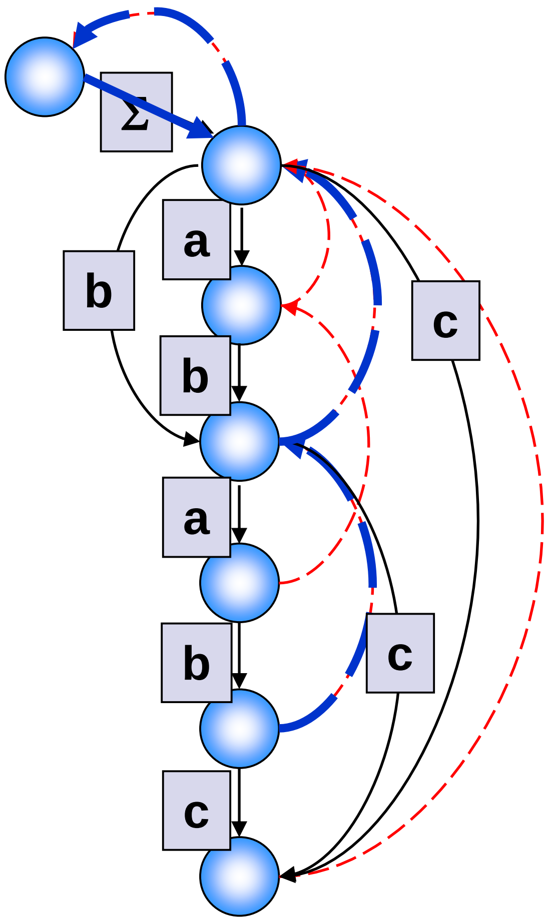
ababcb

On-line Construction of DAWG



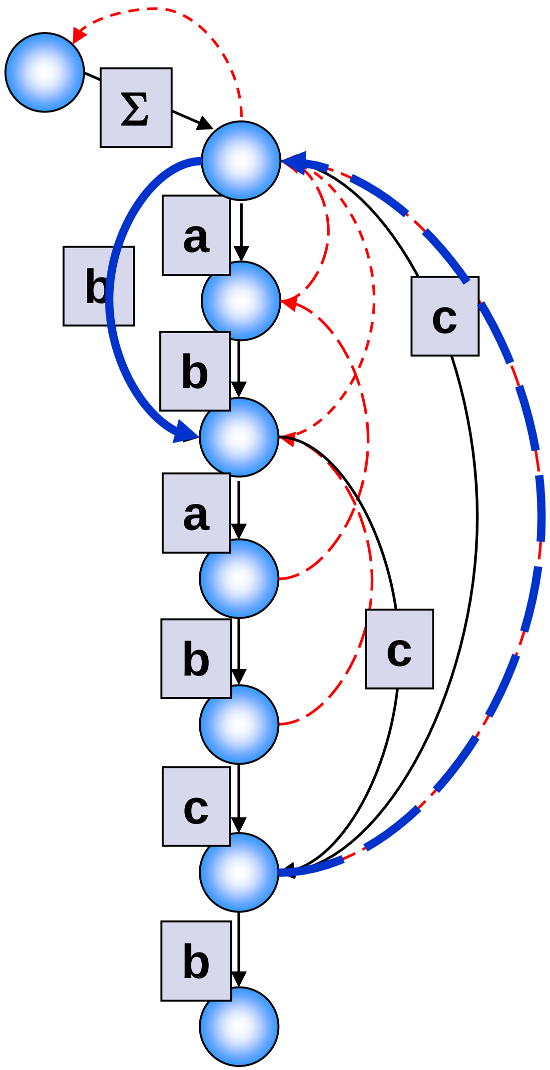
ababcb

On-line Construction of DAWG



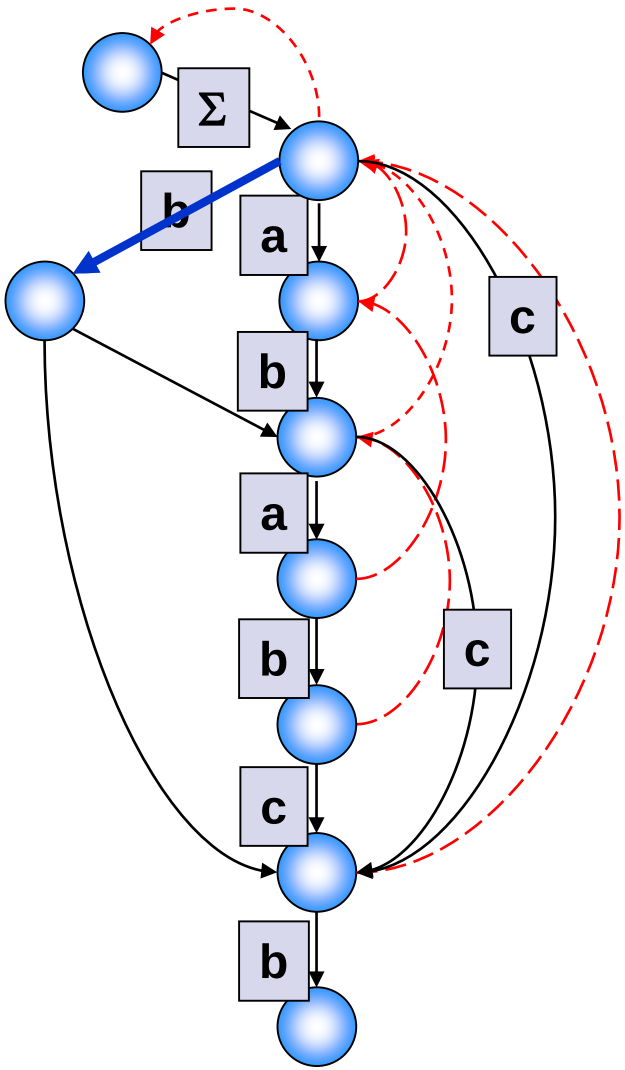
ababcb

On-line Construction of DAWG



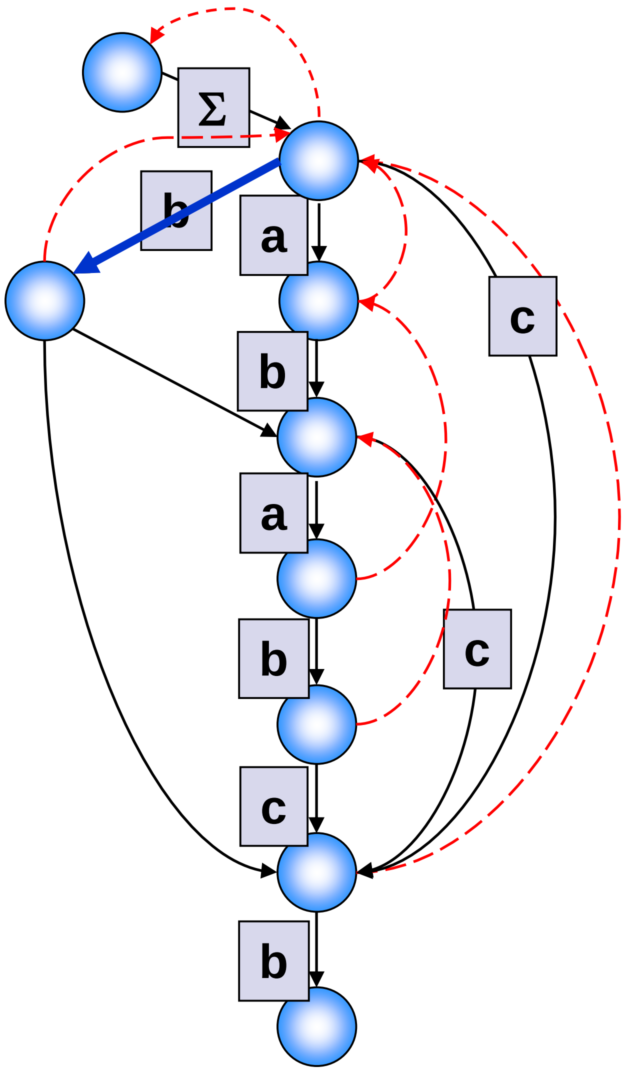
ababcb

On-line Construction of DAWG



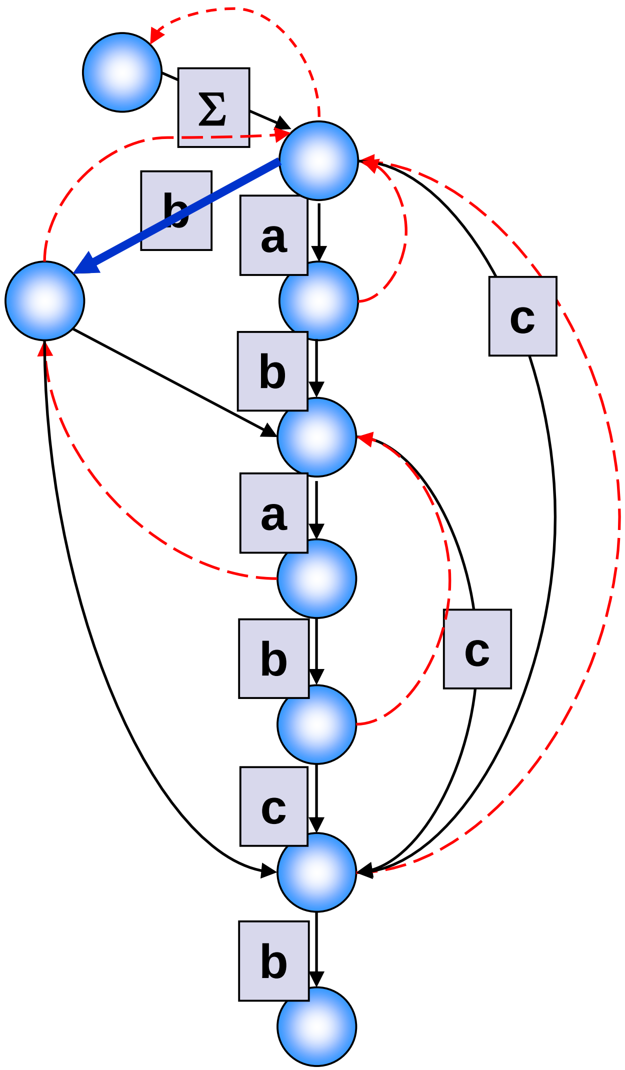
ababcb

On-line Construction of DAWG



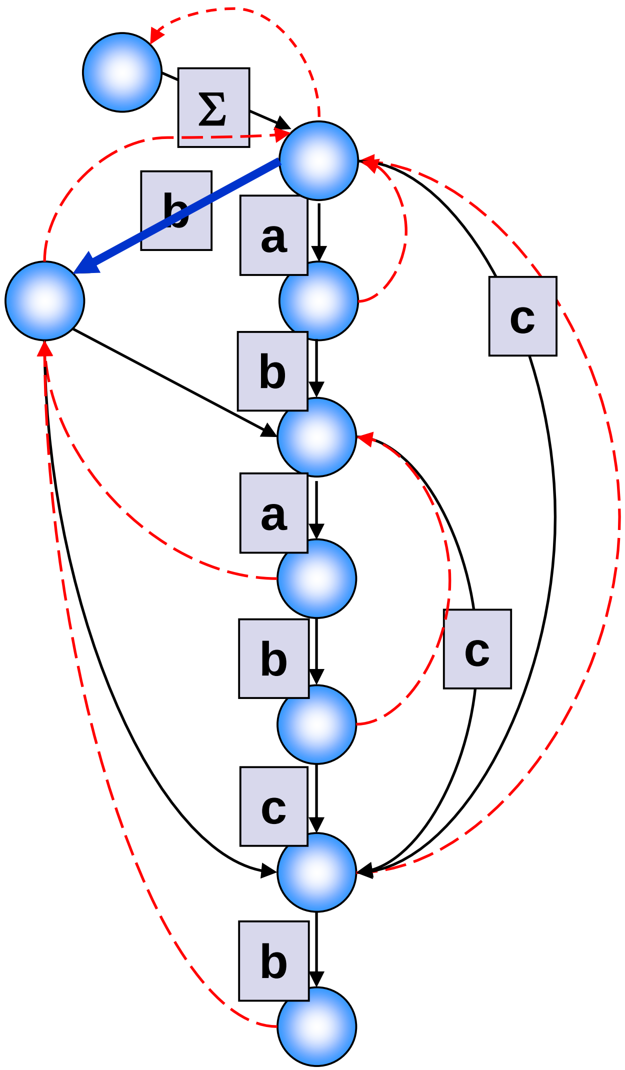
ababcb

On-line Construction of DAWG



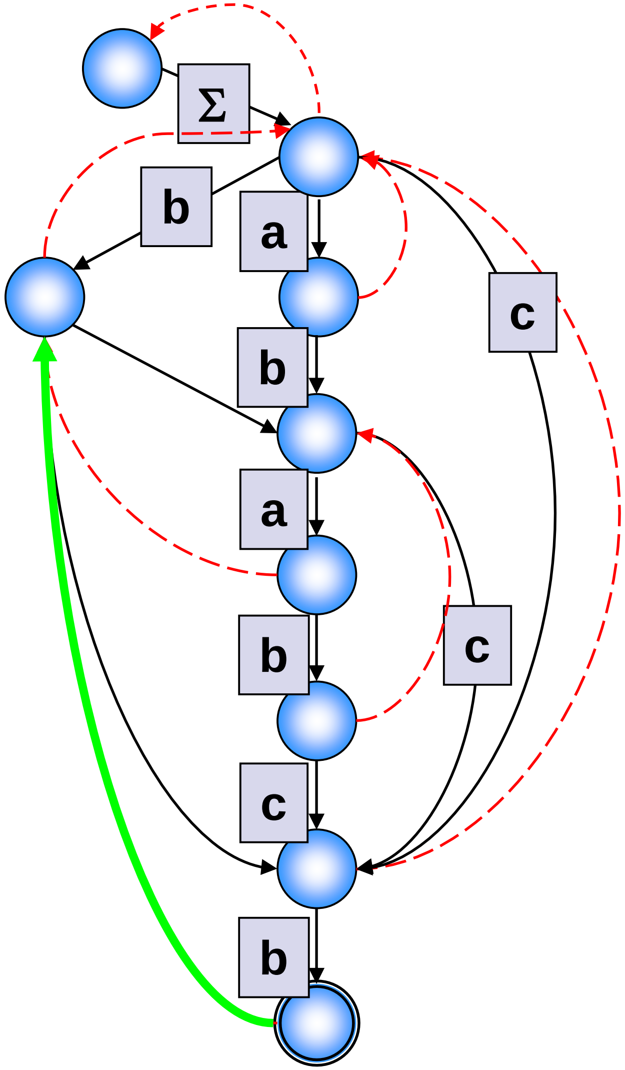
ababcb

On-line Construction of DAWG



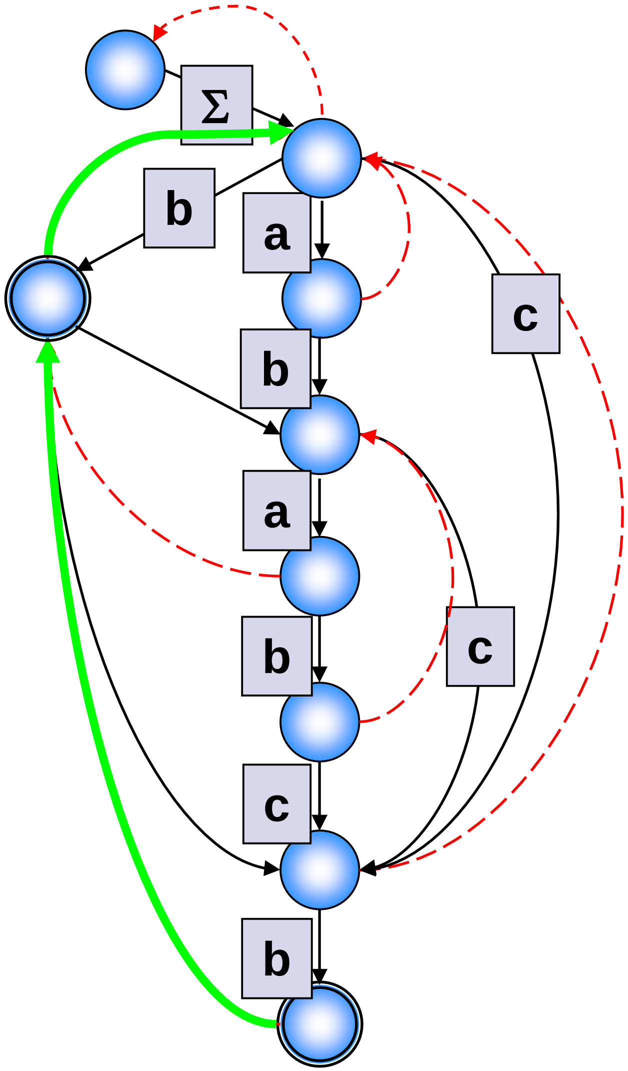
ababcb

Marking Accepting Nodes



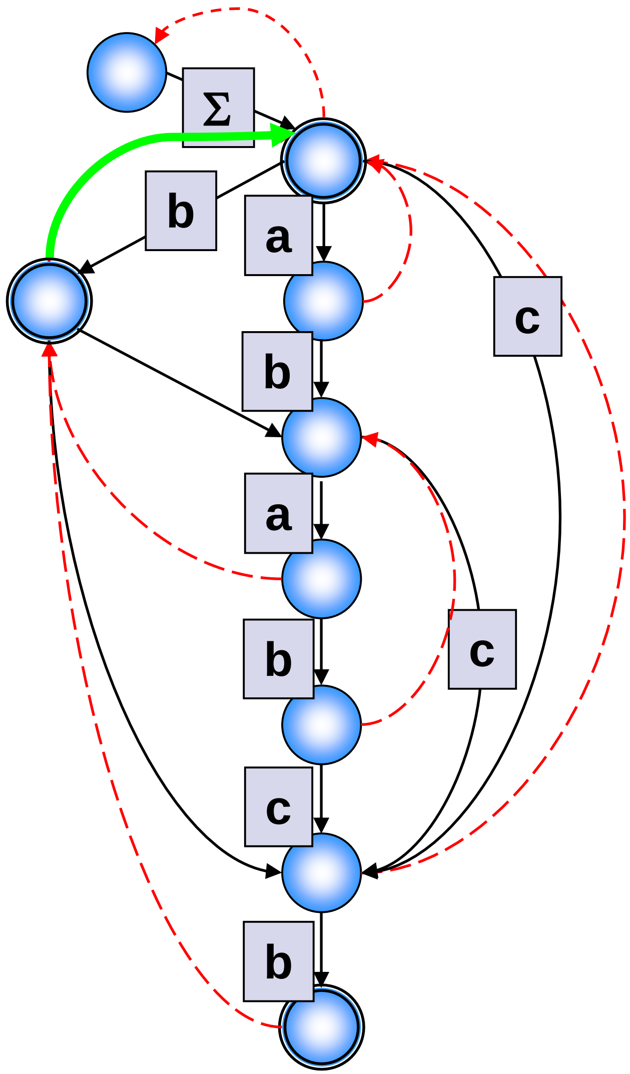
ababcb

Marking Accepting Nodes



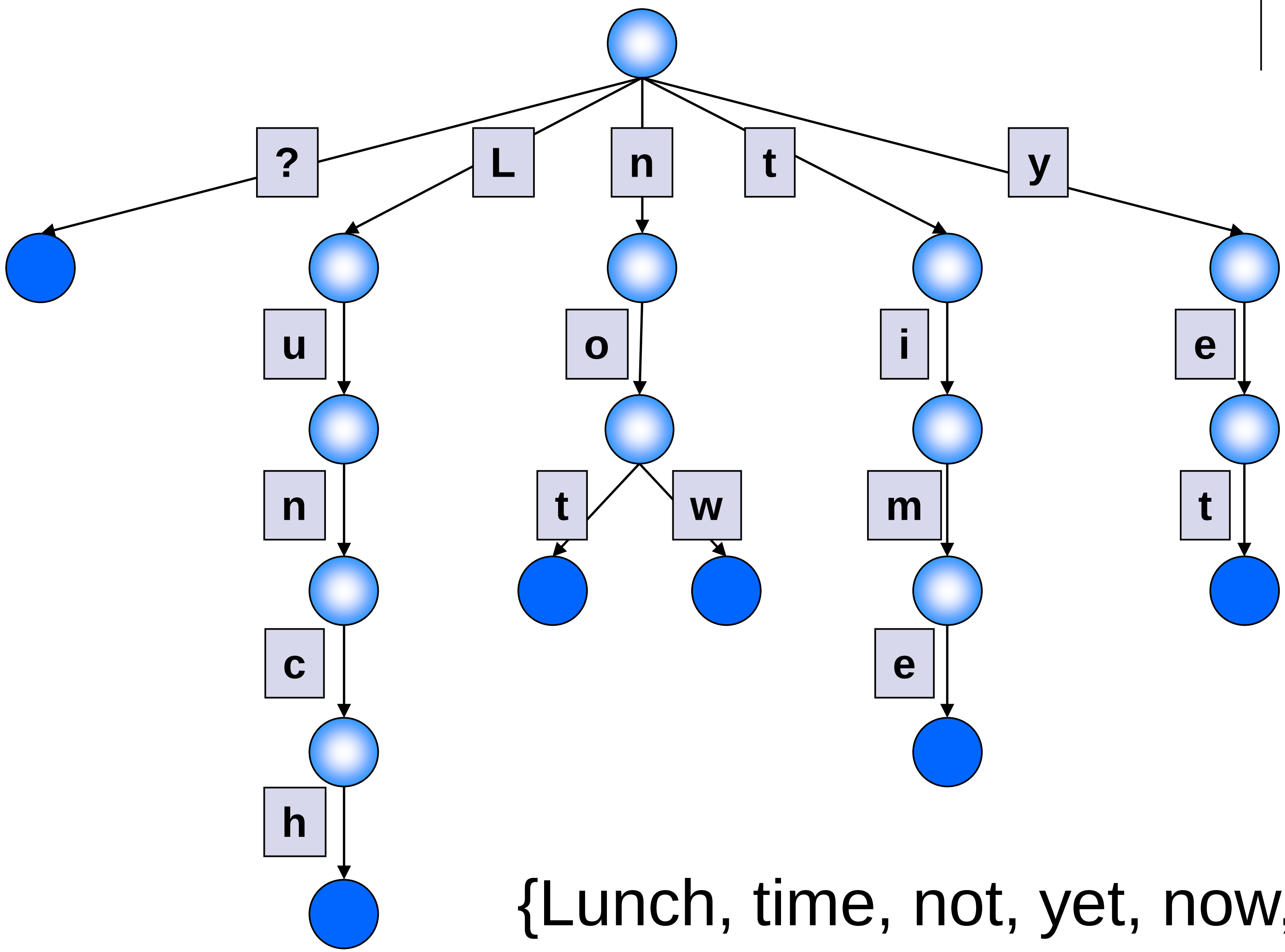
ababcb

Marking Accepting Nodes



ababcb

Search Tree of a Set of Strings



{Lunch, time, not, yet, now, ?}

Table Implementation

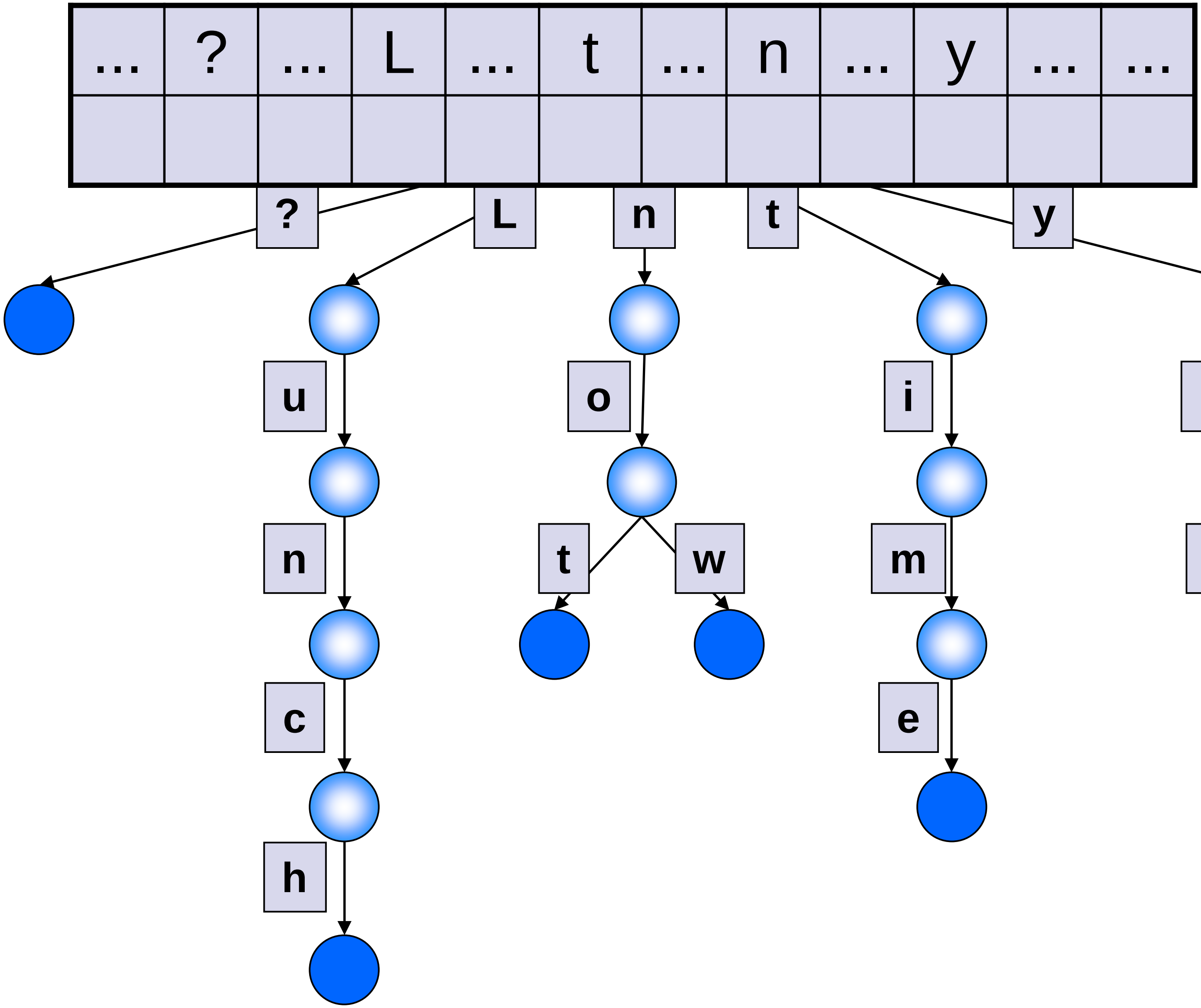
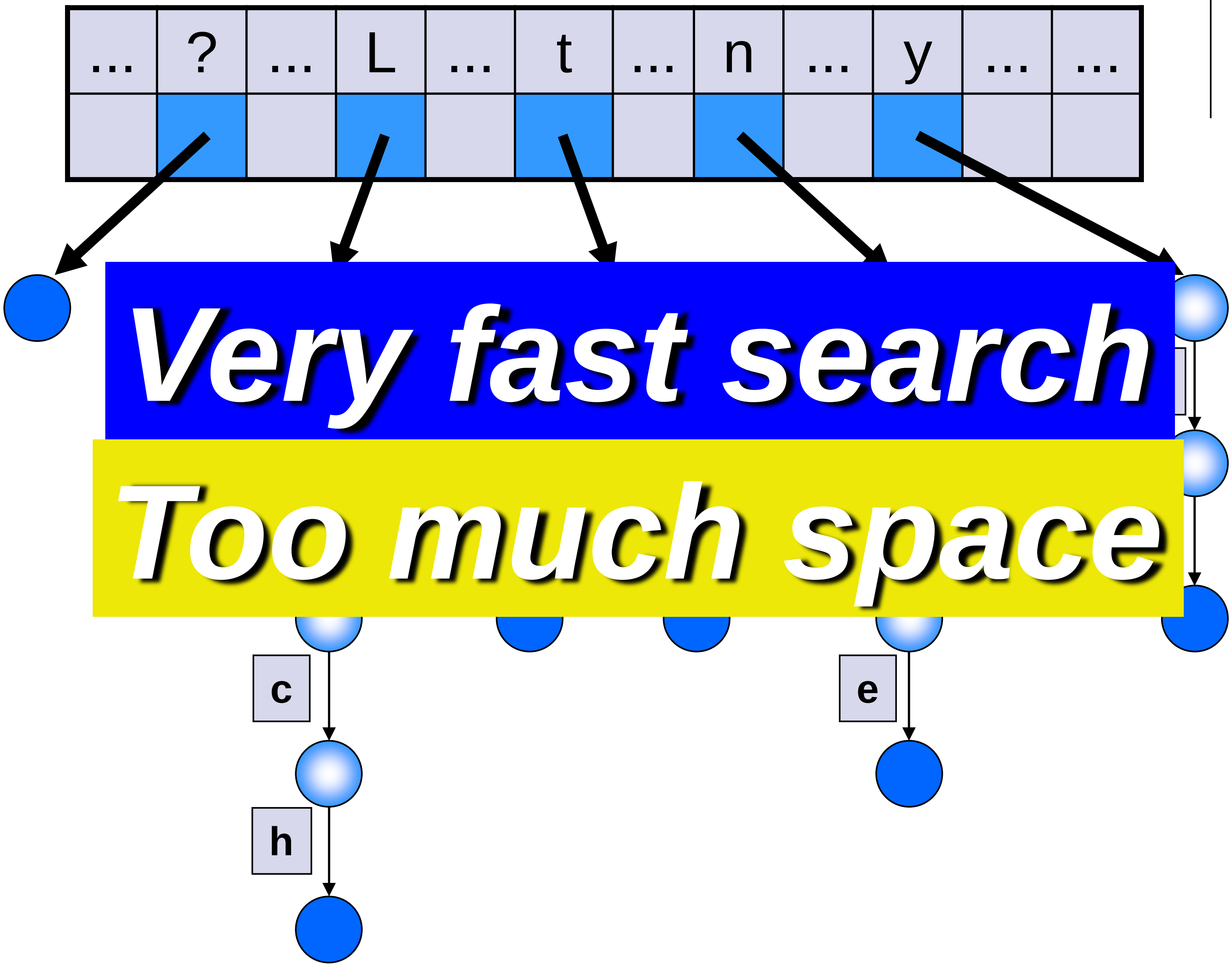


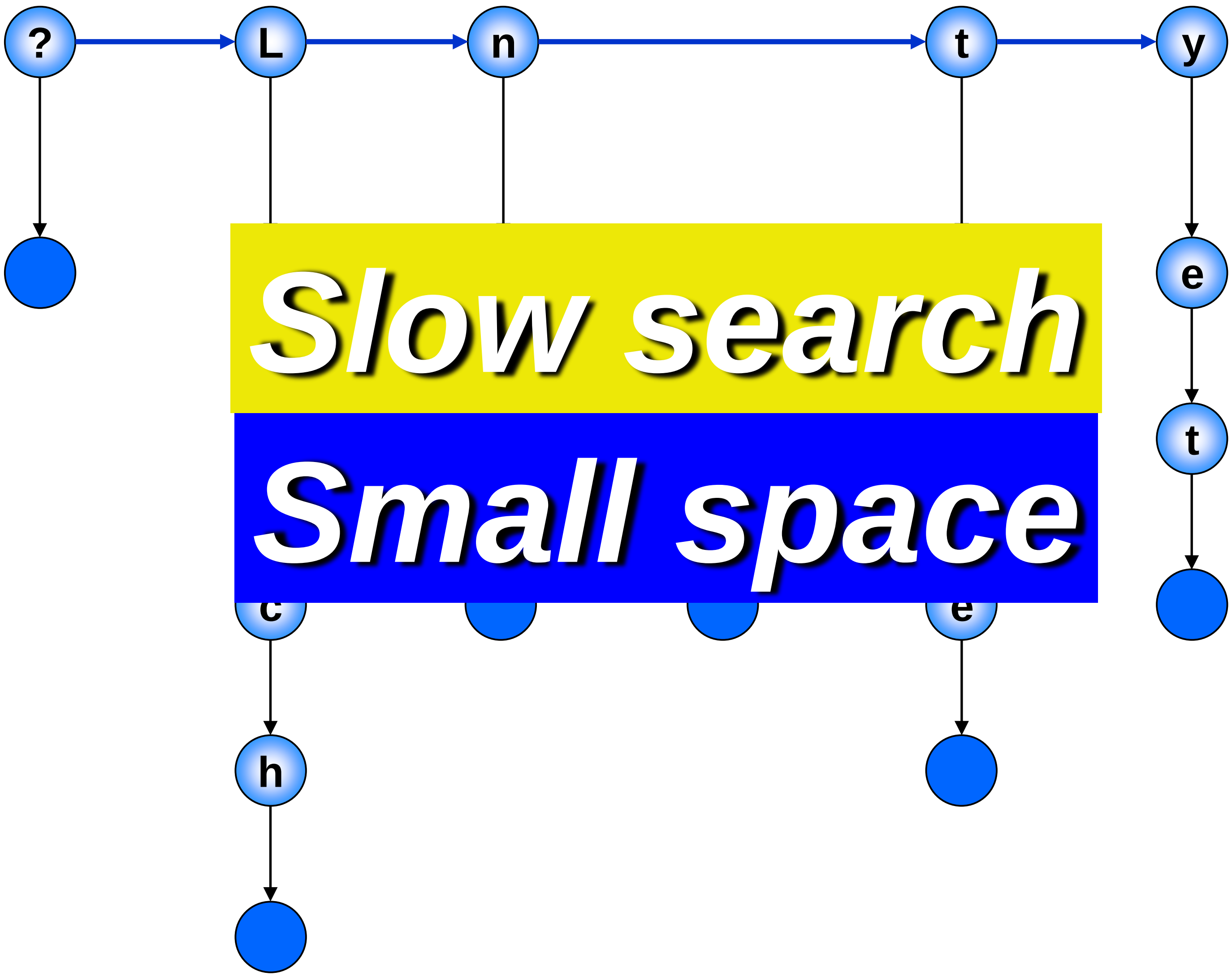
Table Implementation



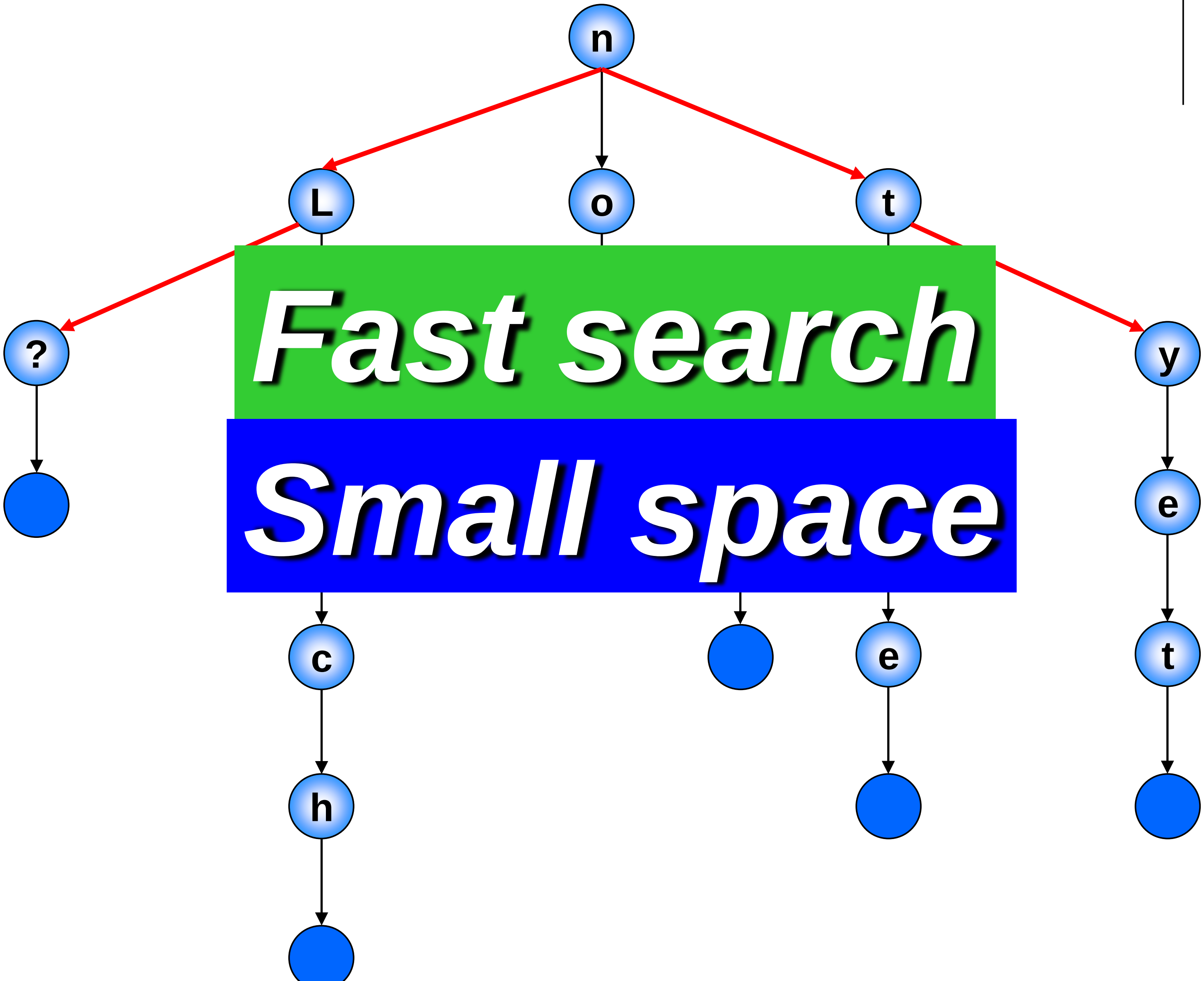
Very fast search

Too much space

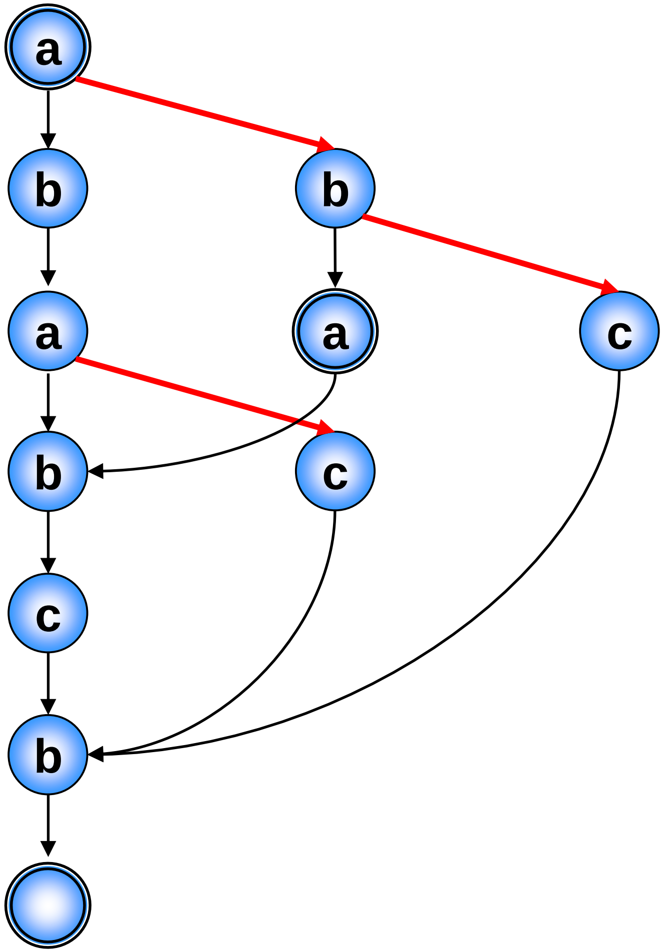
Linked List Implementation



Ternary Search Tree



Ternary Directed Acyclic Word Graph



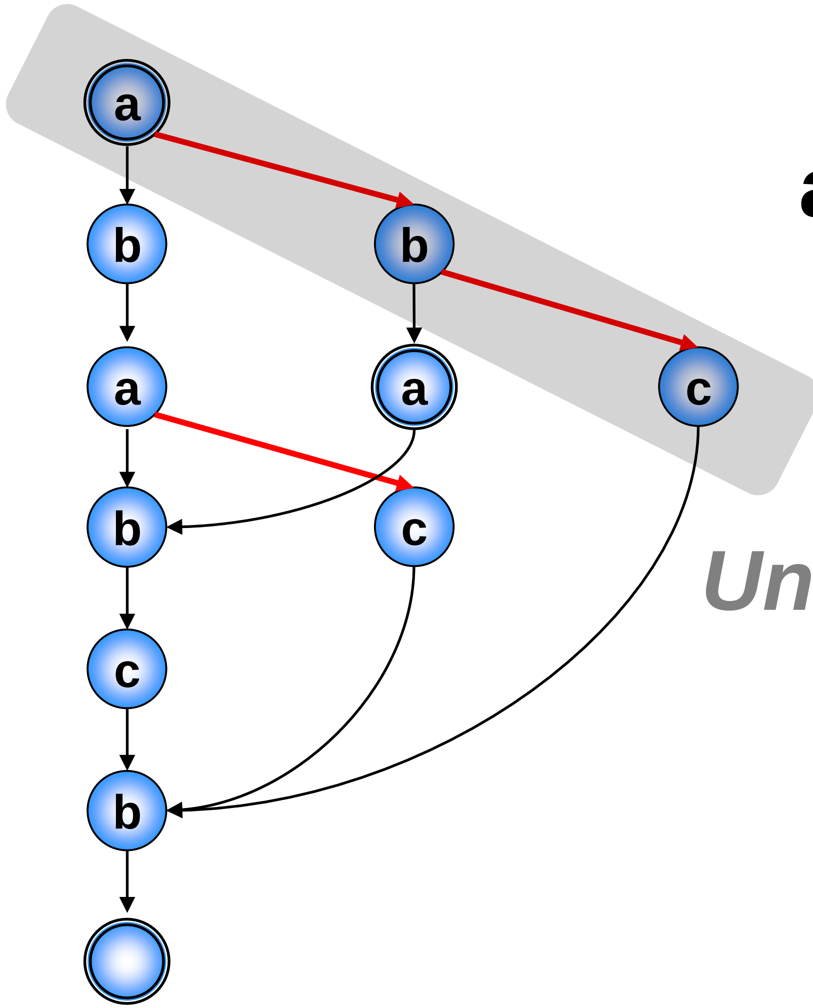
ababcb

Construction of TDAWG

Theorem. (Miyamoto et al. 2003)

For any string $\boldsymbol{w}$ *of length* $\boldsymbol{n}$*,*
$\boldsymbol{TDAWG(w)}$ *can be constructed on-line and in* $\boldsymbol{O(n|\Sigma|)}$ *time and* $\boldsymbol{O(n)}$ *space.*

Ternary Directed Acyclic Word Graph

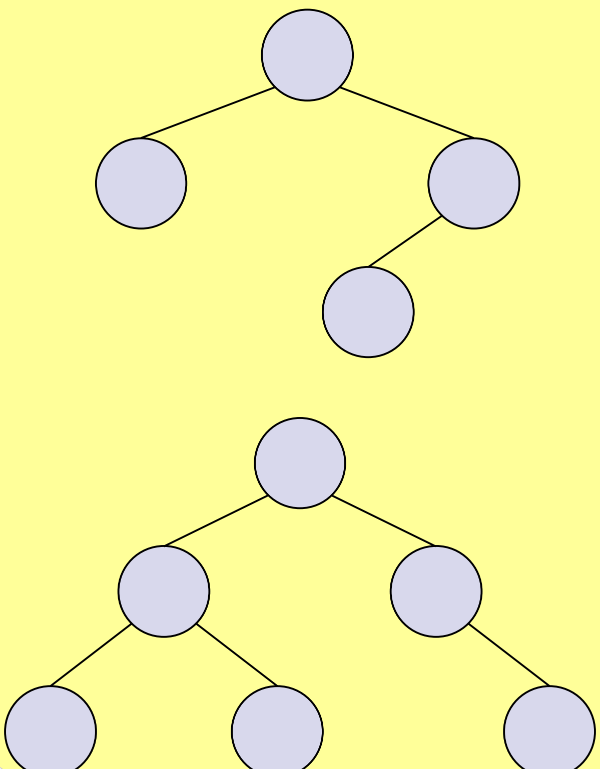


ababcb

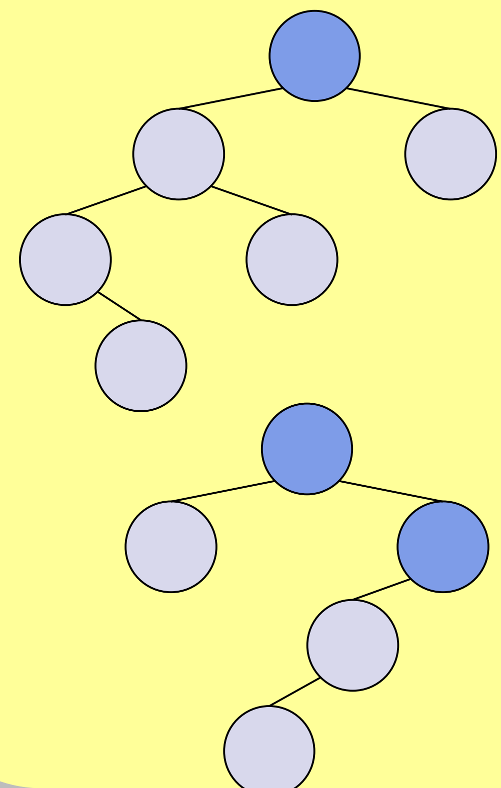
Unbalanced!

Using AVL Tree

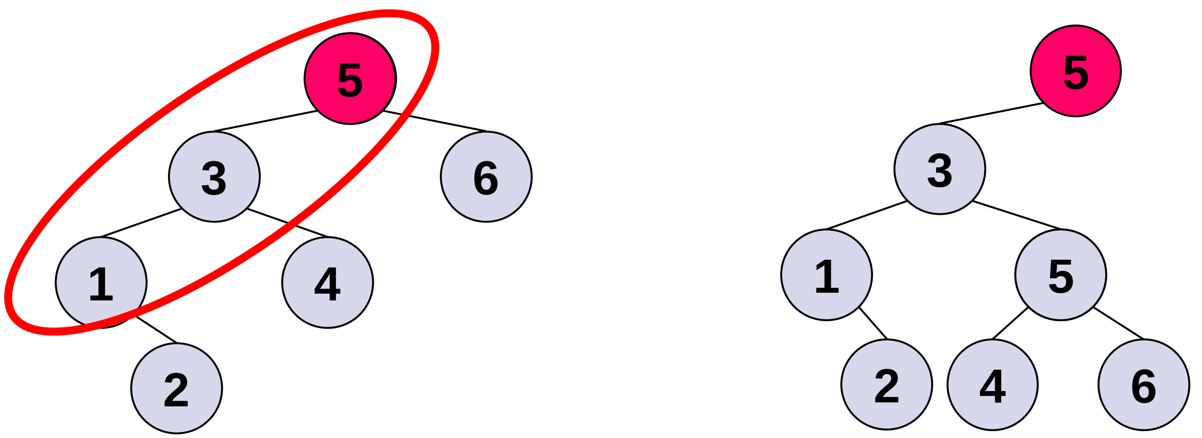
AVL Balanced



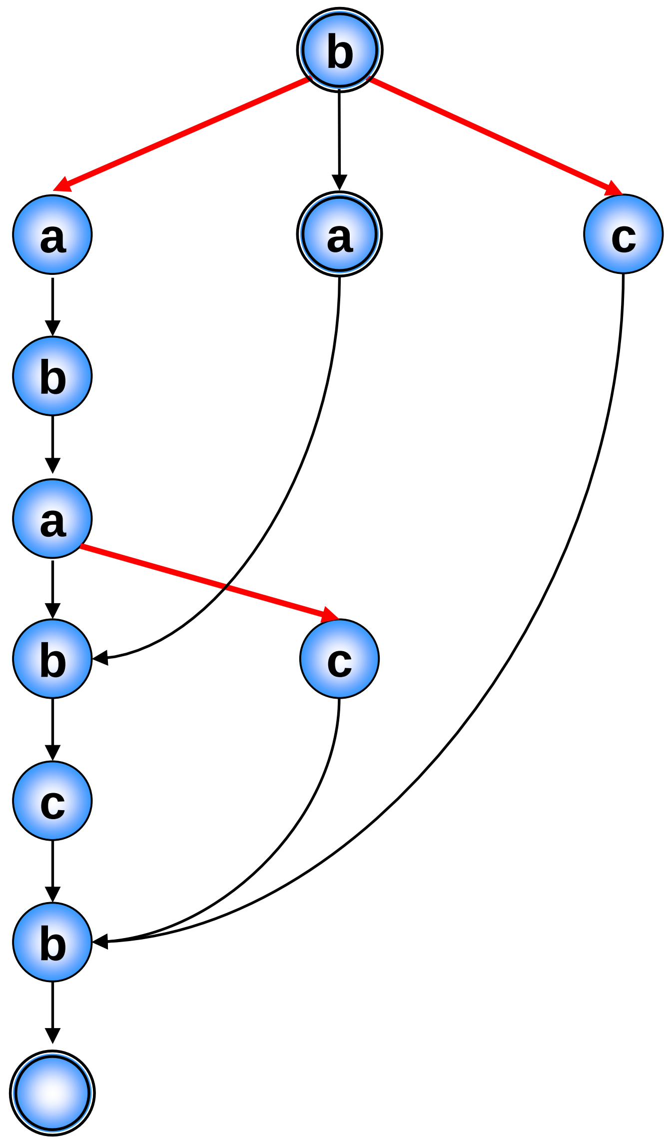
Not AVL Balanced



Node Rotation



AVL DAWG



ababcb

Construction of AVL TDAWG

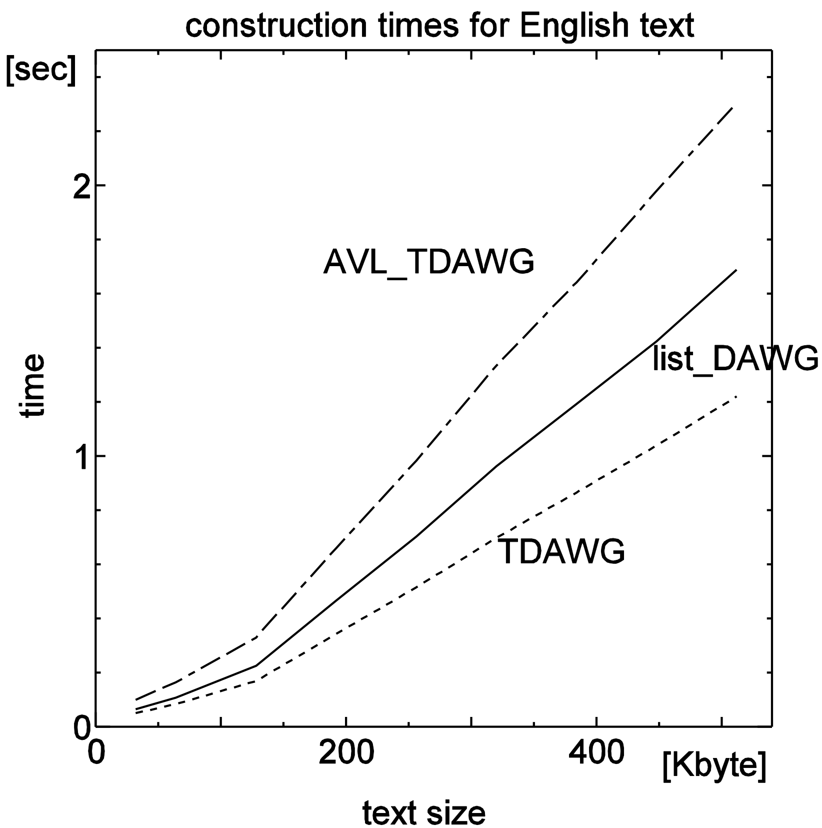
Theorem. (Miyamoto et al. 2003)

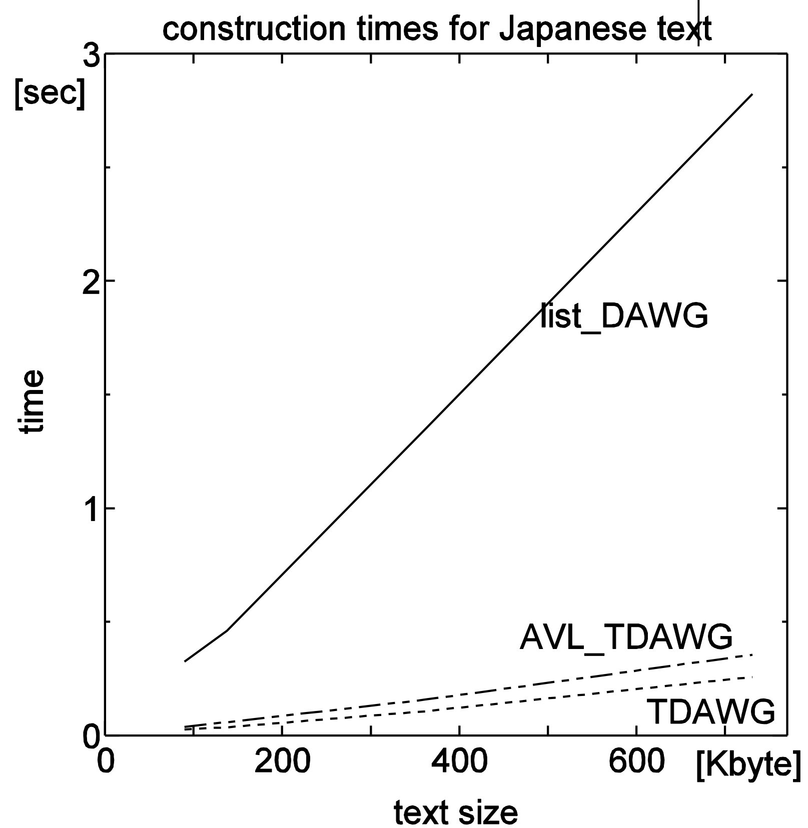
For any string $\boldsymbol{w}$ of length $\boldsymbol{n}$, $\boldsymbol{AVL\_TDAWG(w)}$ can be constructed on-line and in $\mathbf{O(n\log|\Sigma|)}$ time and $\mathbf{O(n)}$ space.

Complexities

	space	search time	
		average case	worst case
table	$O(\|w\| \cdot \|\Sigma\|)$	$O(\|p\|)$	$O(\|p\|)$
linked list	$O(\|w\|)$	$O(\|p\| \cdot \|\Sigma\|)$	$O(\|p\| \cdot \|\Sigma\|)$
TDAWG	$O(\|w\|)$	$O(\|p\| \cdot \log \|\Sigma\|)$	$O(\|p\| \cdot \|\Sigma\|)$
AVL-TDAWG	$O(\|w\|)$	$O(\|p\| \cdot \log \|\Sigma\|)$	$O(\|p\| \cdot \log \|\Sigma\|)$

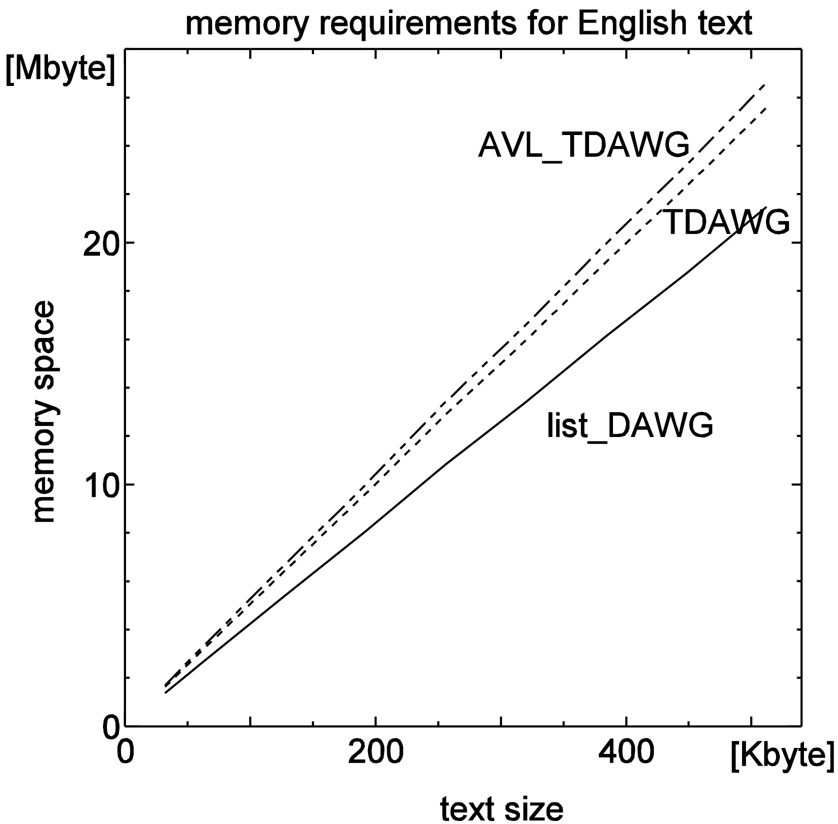
Experiment – Construction Times

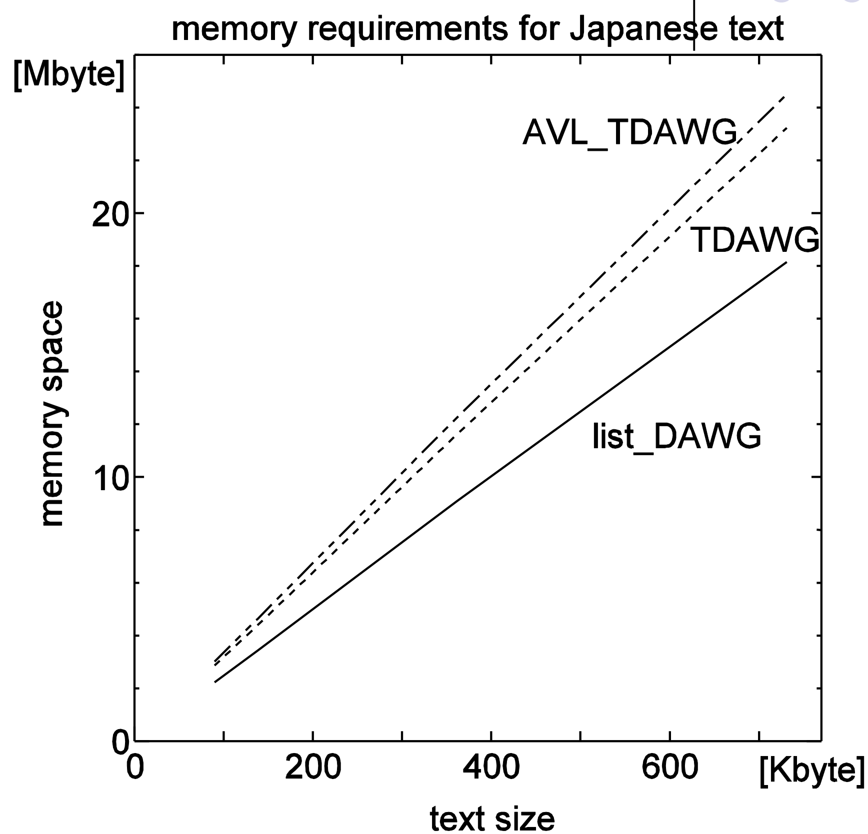




English **Japanese**

Experiment – Memory Space

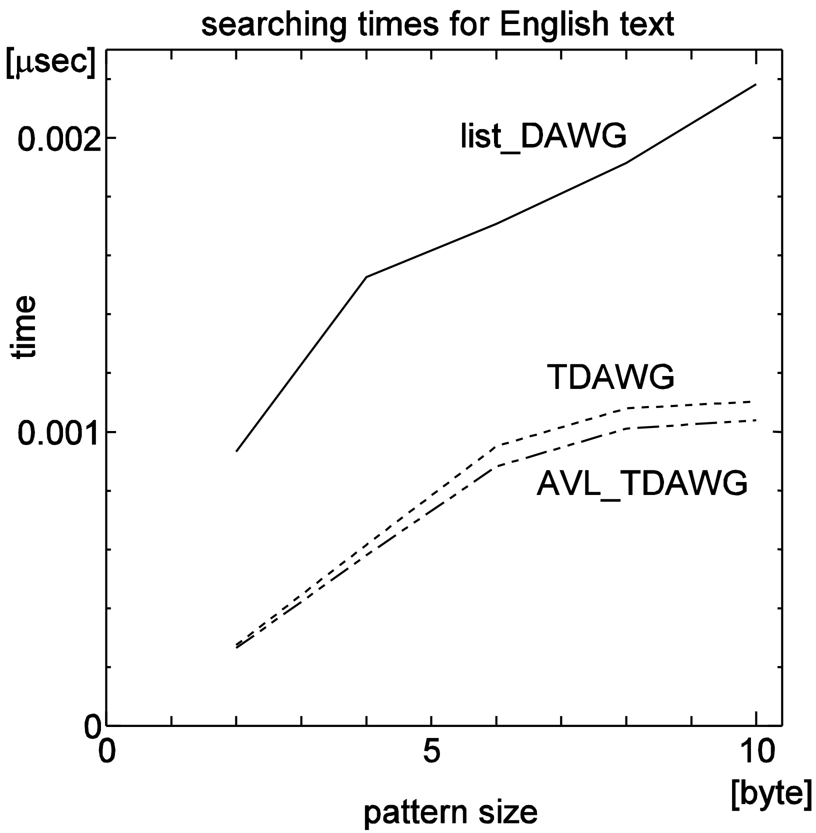


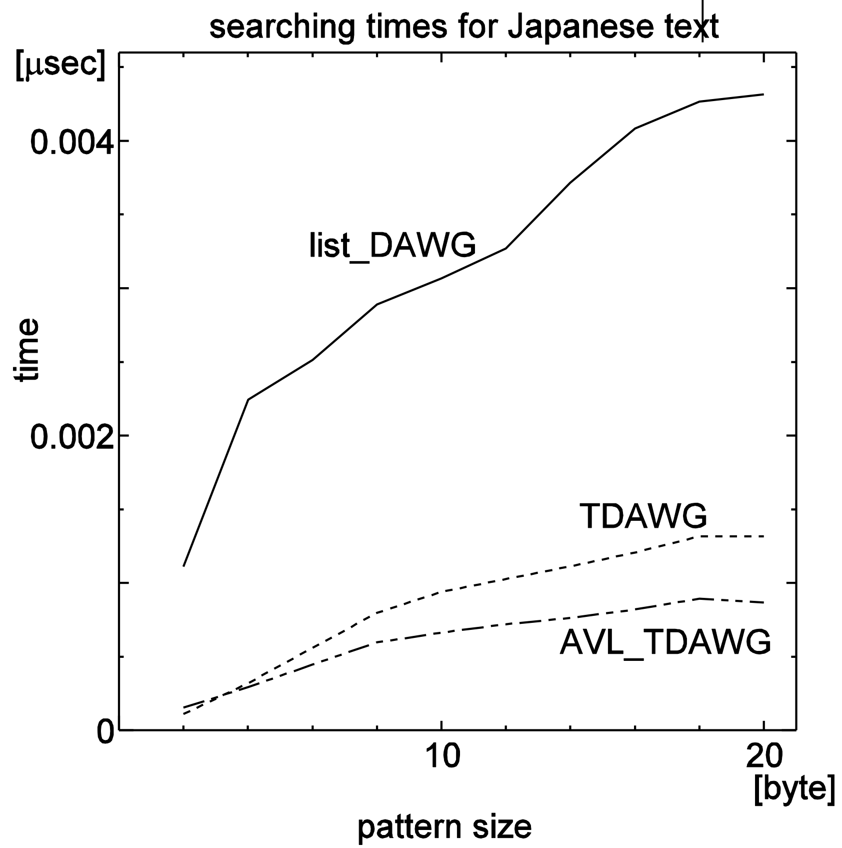


English

Japanese

Experiment – Search Times





English

Japanese

Citations

A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht,
B. M. Chen, and J. Seiferas.
The smallest automaton recognizing the subwords of a text.
Theoretical Computer Science, 40:31-55, 1985.

M. Crochemore,
Transducers and repetitions.
Theoretical Computer Science, 45(1): 63-86, 1986

S. Miyamoto, S. Inenaga, M. Takeda, and A. Shinohara
Ternary Directed Acyclic Word Graphs.
8th Int. Conference on Implementation and Application of Automata (CIAA 2003), Springer-Verlag LNCS 2759, pp. 121-130.
Also Accepted to Theoretical Computer Science