Construction of a De Bruijn Graph for Assembly from a Truncated Suffix Tree

Bastien Cazaux, Thierry Lecroq, Eric Rivals

LIRMM & IBC, Montpellier - LITIS Rouen

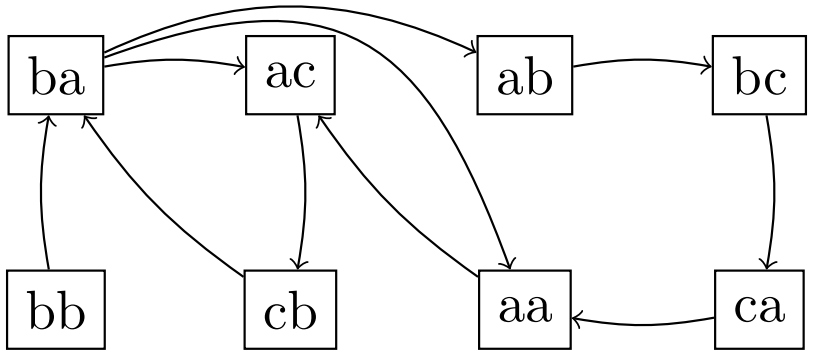
Mars 3, 2015

De Bruijn Graph for assembly

$$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$$

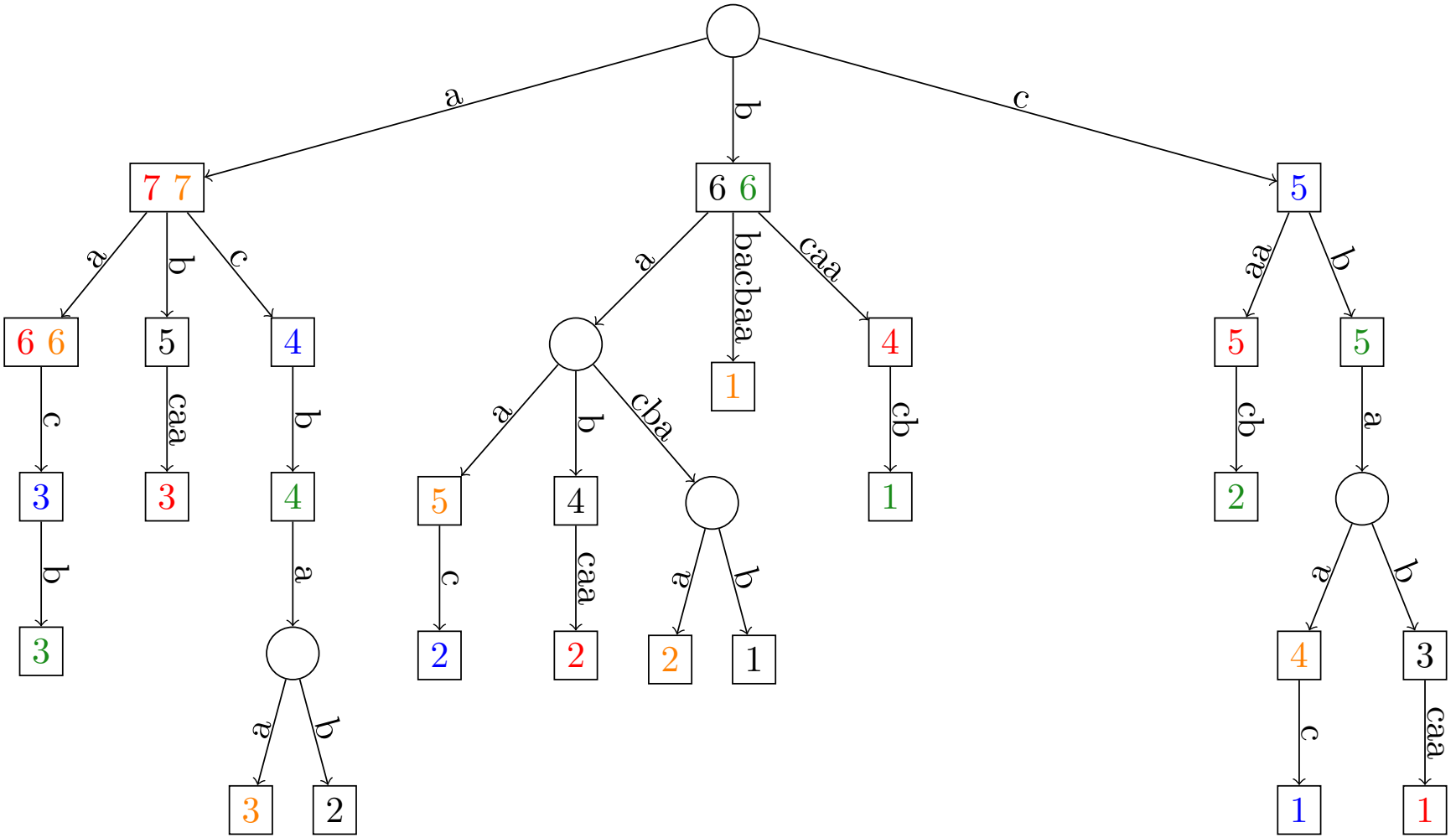
De Bruijn Graph for assembly

$$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$$



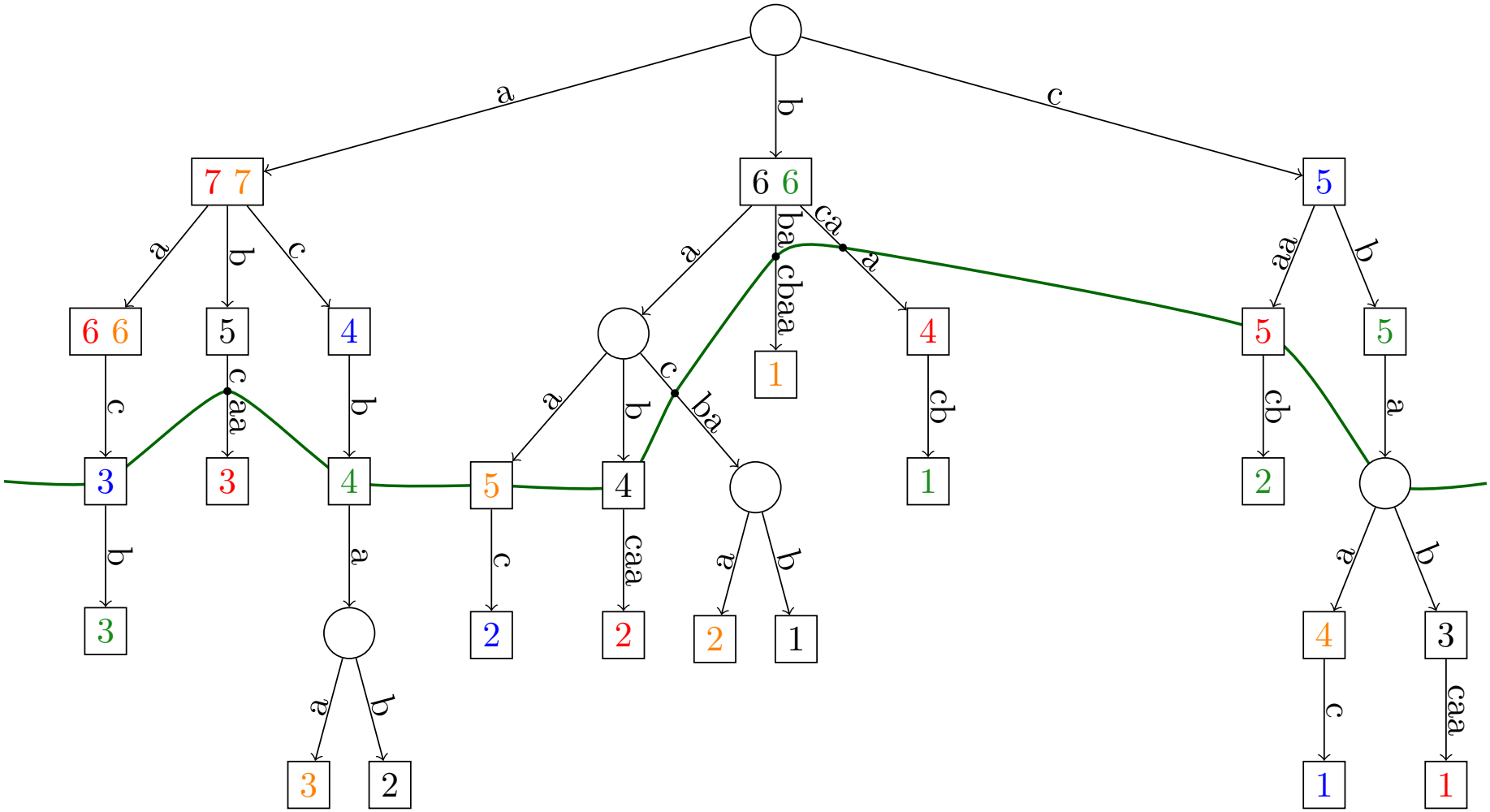
Generalized Suffix Tree (GST)

$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$



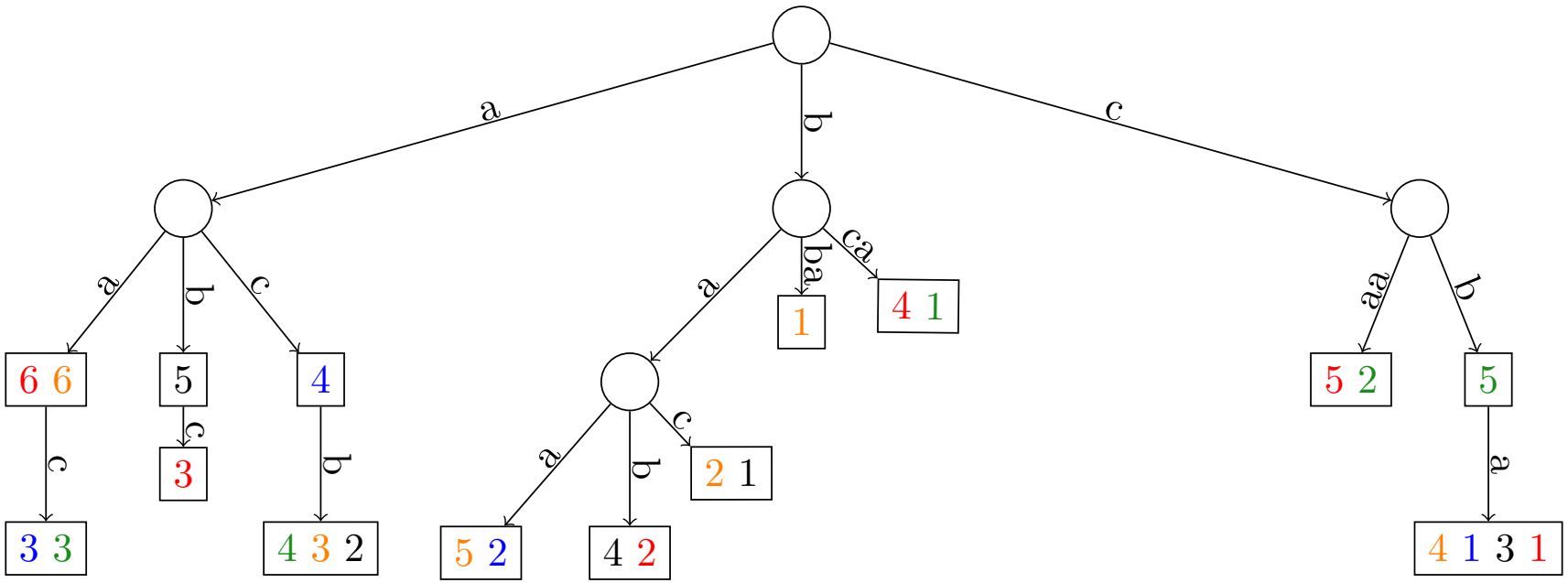
Generalized Suffix Tree with cut

$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$



Truncated Suffix Tree (TST)

$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$



Motivation

- De Bruijn Graph is largely used in *de novo* genome assembly. [Pevzner et al., 2001]
- One builds a suffix tree before the assembly for some applications, for instance for the error correction. [Salmela, 2010]
- There exist algorithms to build directly the De Bruijn Graph [Onodera et al., 2013] [Rodland, 2013] and the Contracted De Bruijn Graph [Cazaux et al., 2014][Chikhi et al., 2014].

Indexing data structures

- Numerous data structures: suffix tree, affix tree, suffix table, factor automata, etc.
- to index one or several texts (generalized index)
- functionnally equivalent

Indexing data structures

- Numerous data structures: suffix tree, affix tree, suffix table, factor automata, etc.
- to index one or several texts (generalized index)
- functionnally equivalent

Result: We can directly build the assembly De Bruijn graph

in the classical or contracted form
from an indexing data structures.[Cazaux et al., 2014]

Indexing data structures

- Numerous data structures: suffix tree, affix tree, suffix table, factor automata, etc.
- to index one or several texts (generalized index)
- functionnally equivalent

Result: We can directly build the assembly De Bruijn graph

in the classical or contracted form
from an indexing data structures.[Cazaux et al., 2014]

Question: How to do it without using more space than necessary?

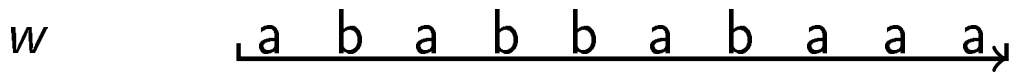
Chain of suffix-dependant strings and Tree

String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.

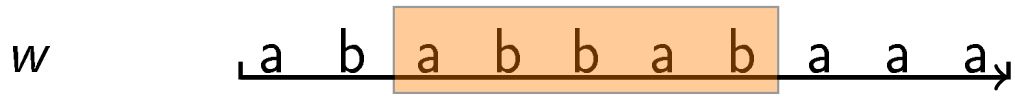


String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.

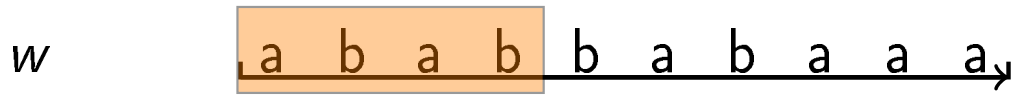


String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.

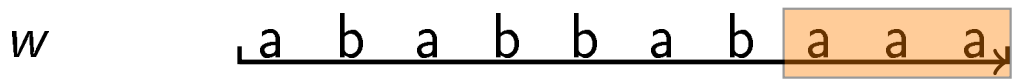


String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.

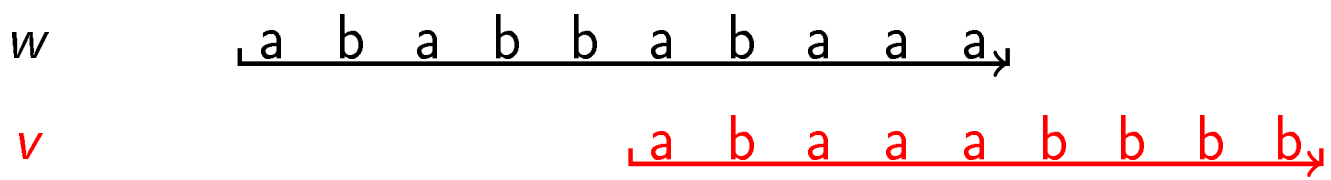


String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.

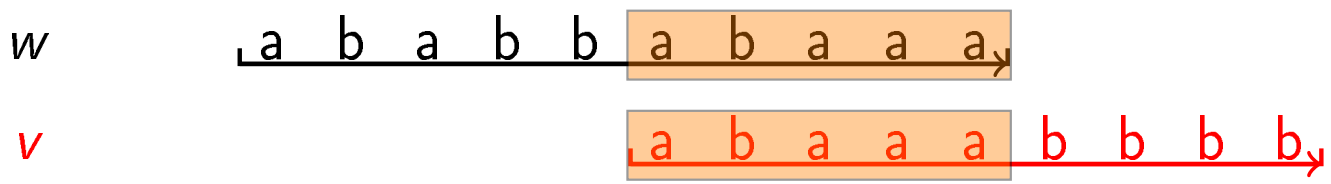


String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.

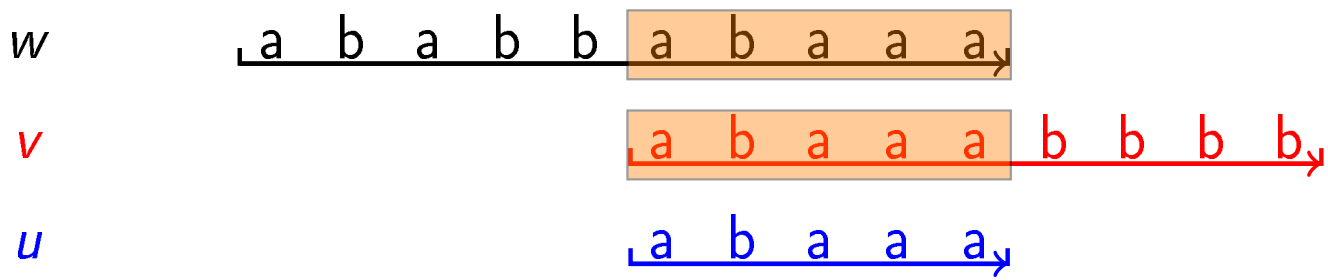


String

Definition [Gusfield 1997]

Let w a string.

- a **substring** of w is a string included in w,
- a **prefix** of w is a substring which begins w and
- a **suffix** is a substring which ends w.
- an **overlap** between w and v is a suffix of w which is also a prefix of v.



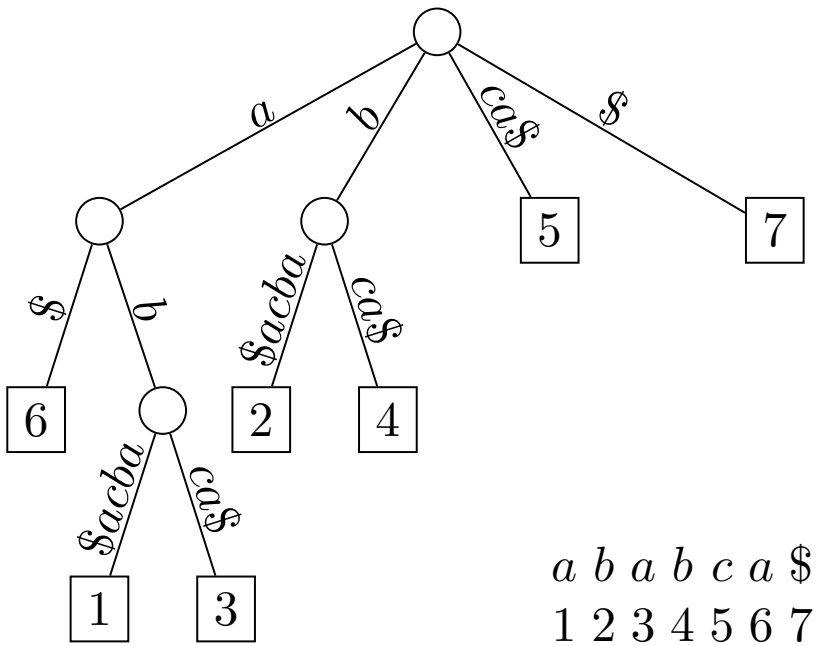
Norm of a set of words

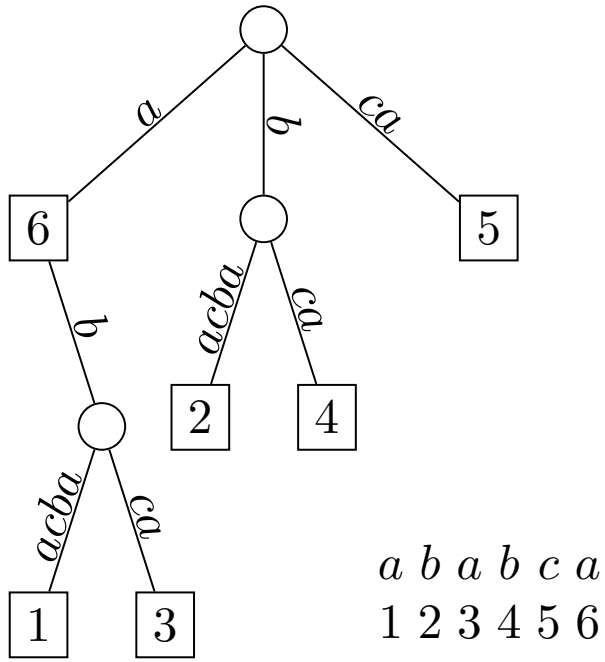
$$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$$

$$||R|| = \sum_{w_i \in R} |w_i|$$

$$||R|| = 7 + 5 + 6 + 7 + 6 = 31$$

Suffix Tree





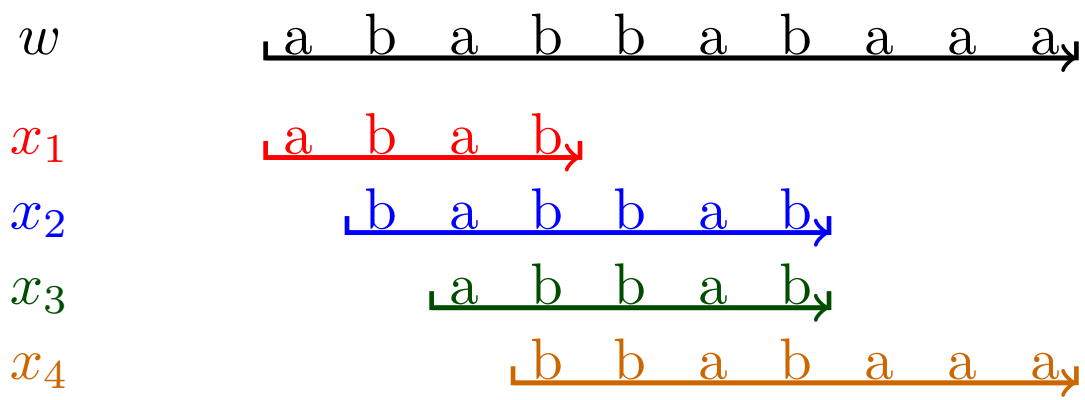
Theorem

The GST of a set of words R takes linear space in $||R||$.

Chain of suffix-dependant strings

Definition

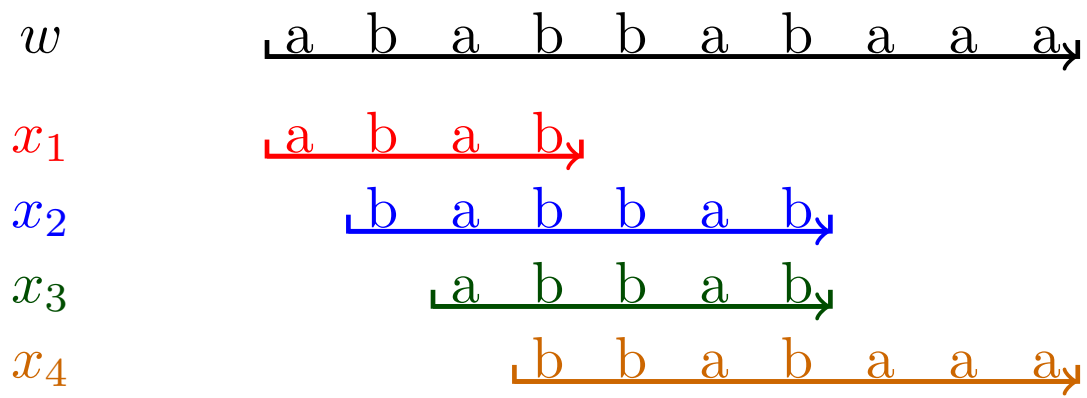
- A string x is said to be *suffix-dependant* of another string y if $x[2..|x|]$ is prefix of y.
- Let w be a string and m be a positive integer smaller than $|w| - 1$. A m-tuple of m strings $(x_1, \ldots, x_m)$ is a *chain of suffix-dependant strings of* w if x_1 is a prefix of w and for each $i \in [2, m]$, x_i is a prefix of $w[i, |w|]$ such that $|x_i| \geq |x_{i-1}| - 1$.

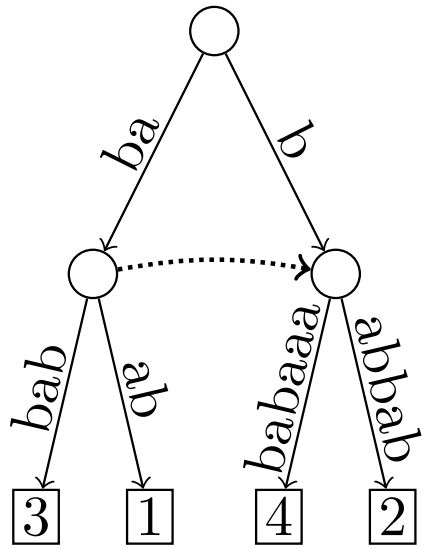


$T(S)$ tree

Definition

Let $R = \{w_1, \ldots, w_n\}$ be a set of strings and $S = \{C_1, \ldots, C_n\}$ a set of tuples such that for $i \in [1, n]$, C_i is a chain of suffix dependant strings of w_i. $T(S)$ is the tree of the contracted Aho-Corasick tree of S.





Linear construction of $T(S)$

Theorem

For a set of chains of suffix-dependant strings S of a set of strings R, we can construct $T(S)$ in $O(||R||)$ time and space.

Application to well known structures

Example

Let $R = \{w_1, \ldots, w_n\}$ be a set of strings and $S = \{C_1, \ldots, C_n\}$ a set of tuples such that for $i \in [1, n]$, C_i is a chain of suffix dependant strings of w_i.

- For $n = 1$, an $S = \{C_1\}$ the tuple of suffixes of w_1, $T(S)$ is the Contracted Suffix Tree of R,
- For C_i the tuple of of suffixes of w_i for all $i \in [1, n], T(S)$ is the Generalised Contracted Suffix Tree of R.
- We can construct the Truncated Suffix Tree of [Peng et al., 2003]
- We can construct the Generalised Truncated Suffix Tree of [Schulz et al., 2008]

Truncated Suffix Tree (TST)

Our Truncated Suffix Tree

Definitions

For a set of words $R = \{w_1, w_2, \ldots, w_n\}$ and an integer $k > 0$, we define the following notation.

1. $F_k(R)$ is the set of substrings of length k of words of R.
2. $Suff_k(R)$ is the set of suffixes of length k of words of R.
3. For all $i \in [1, |R|]$ and $j \in [1, |w_i| - k + 1]$, $A_{k,i}$ denotes the tuple such that its j^{th} element is defined by

$$A_{k,i}[j] := \begin{cases} w_i[j, j+k] & \text{if } j \leq |w_i| - k \\ w_i[j, |w_i|] & \text{otherwise.} \end{cases}$$

4. and finally A_k is the set of these tuples: $A_k := \bigcup_{i=1}^{n} A_{k,i}$.

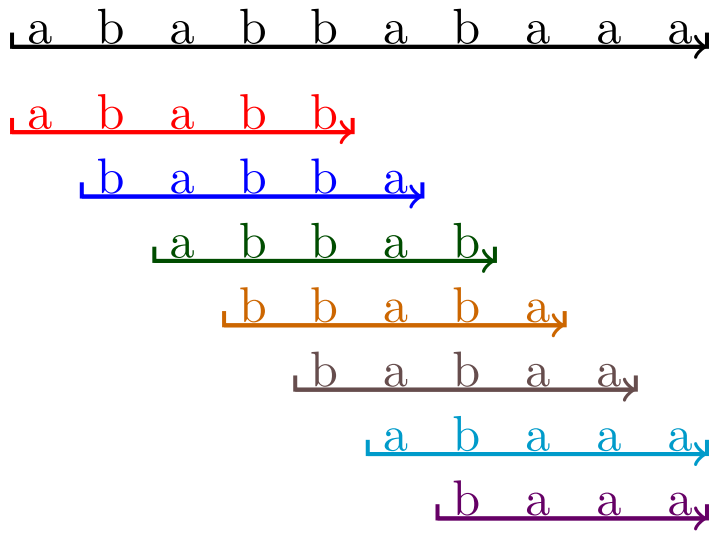
Example of TST

Proposition

1. $A_{k,i}$ is a chain of suffix-dependant strings of w_i.
2. Moreover, $\{w \in A_{k,i} \mid A_{k,i} \in A_k\} = F_{k+1}(R) \cup Suff_k(R)$.

For $R = \{ababbabaaa\}$, we have

$A_4 = \{(ababb, babba, abbab, bbaba, babaa, abaaa, baaa)\}$.



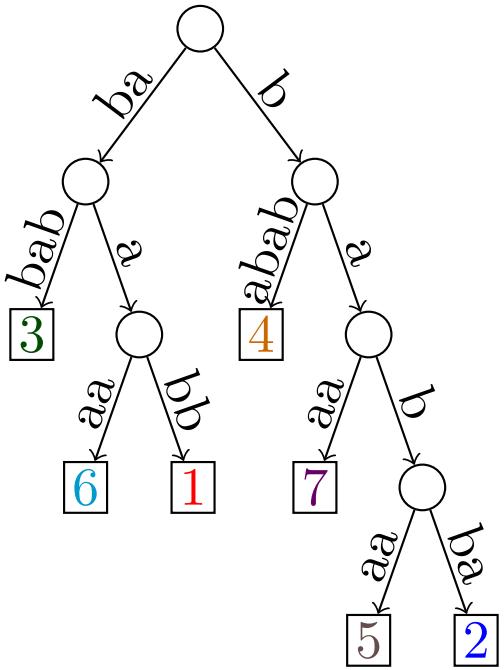
Linear construction of $T(A_k)$

Corollary

We can construct $T(A_k)$ in $O(||R||)$ time and space.

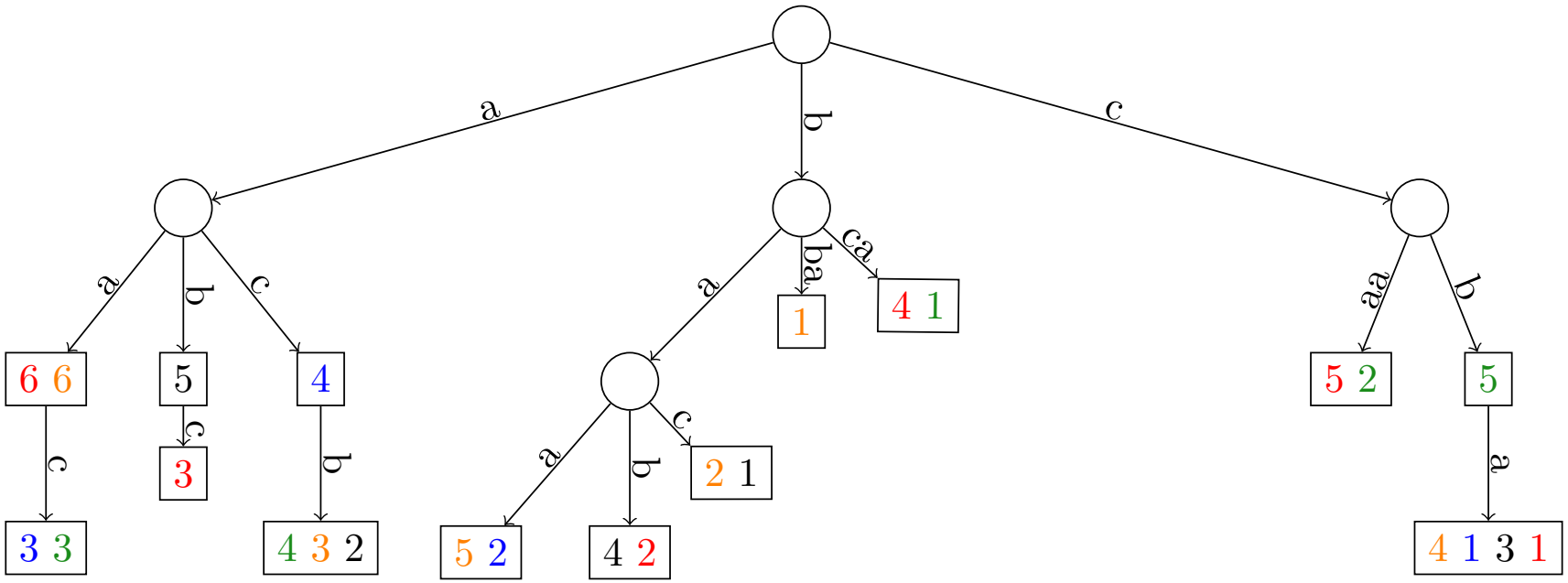
For $R = \{ababbabaaa\}$, we have

$A_4 = \{(ababb, babba, abbab, bbaba, babaa, abaaa, baaa)\}$.



Example of Truncated Suffix Tree

$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$



De Bruin Graph via the TST

Example of construction: De Bruijn Graph DBG_2

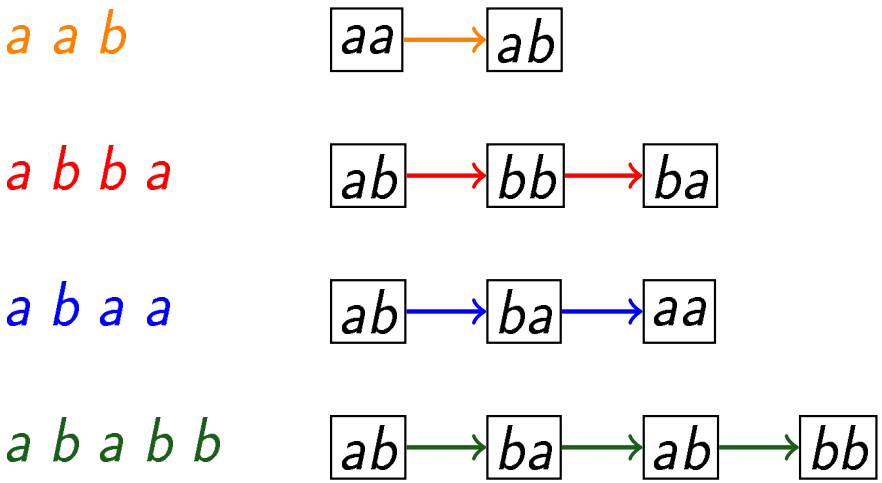
a a b

a b b a

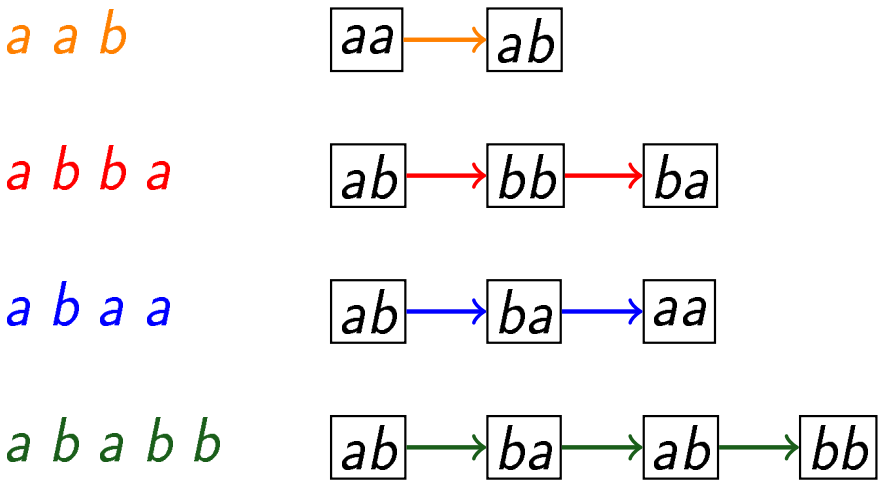
a b a a

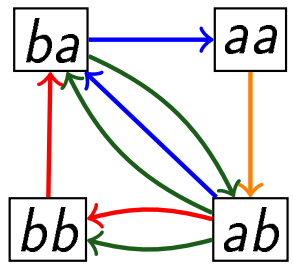
a b a b b

Example of construction: De Bruijn Graph DBG_2

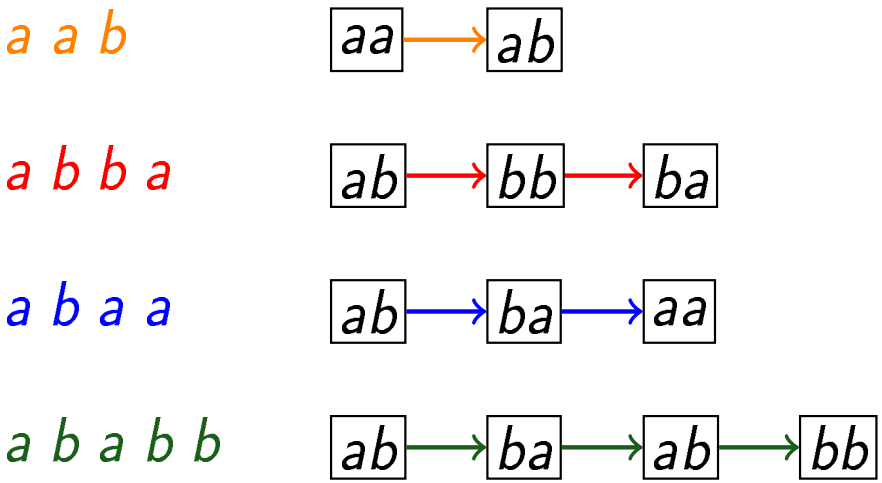


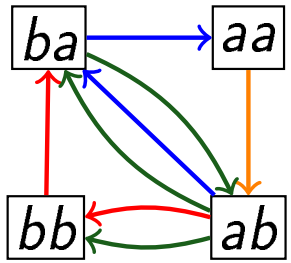
Example of construction: De Bruijn Graph DBG_2

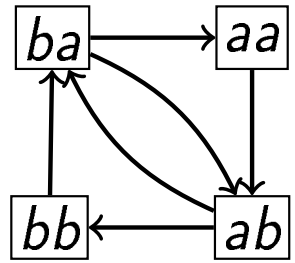




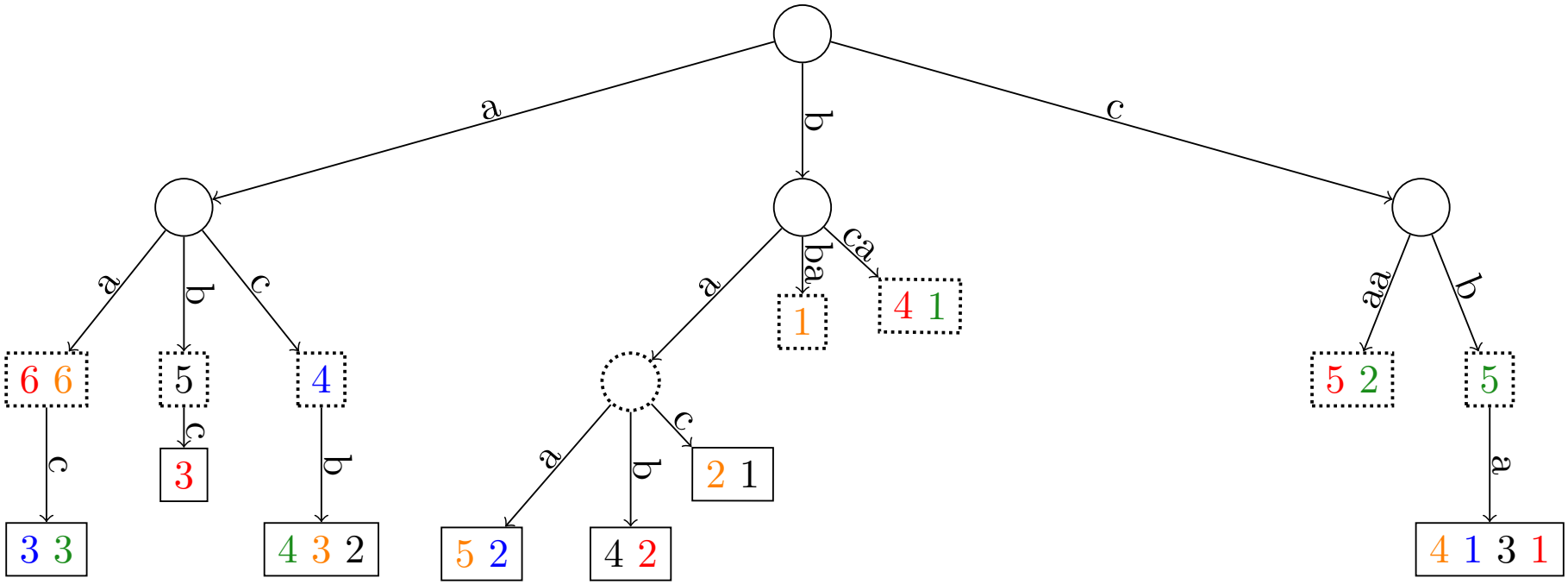
Example of construction: De Bruijn Graph DBG_2





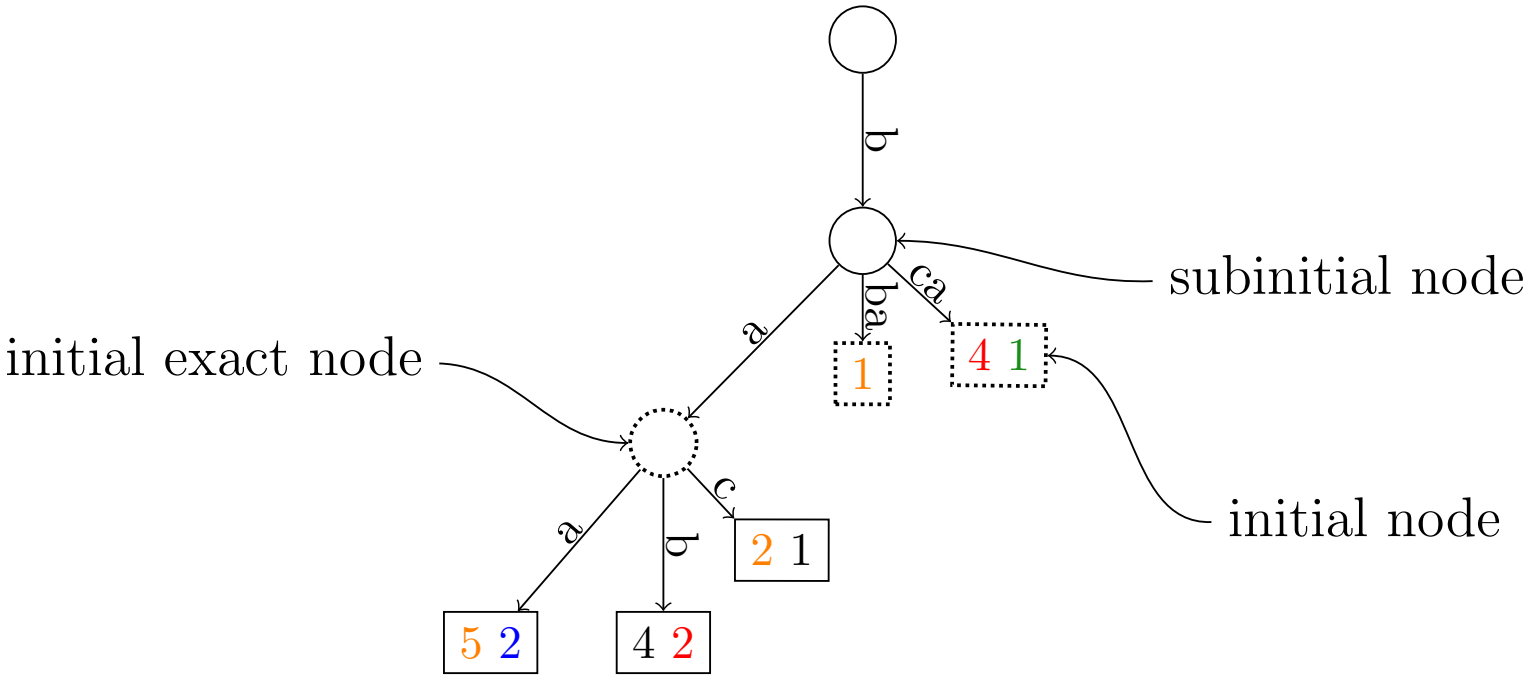


Truncated Suffix Tree (TST)



$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$ and $k = 2$

Truncated Suffix Tree (TST)



$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$ and $k = 2$

Nodes of the de Bruijn Graph

Notation: $Init(R)$

Let $Init(R)$ denote the set of initial nodes of the TST of R.

Property: node correspondence

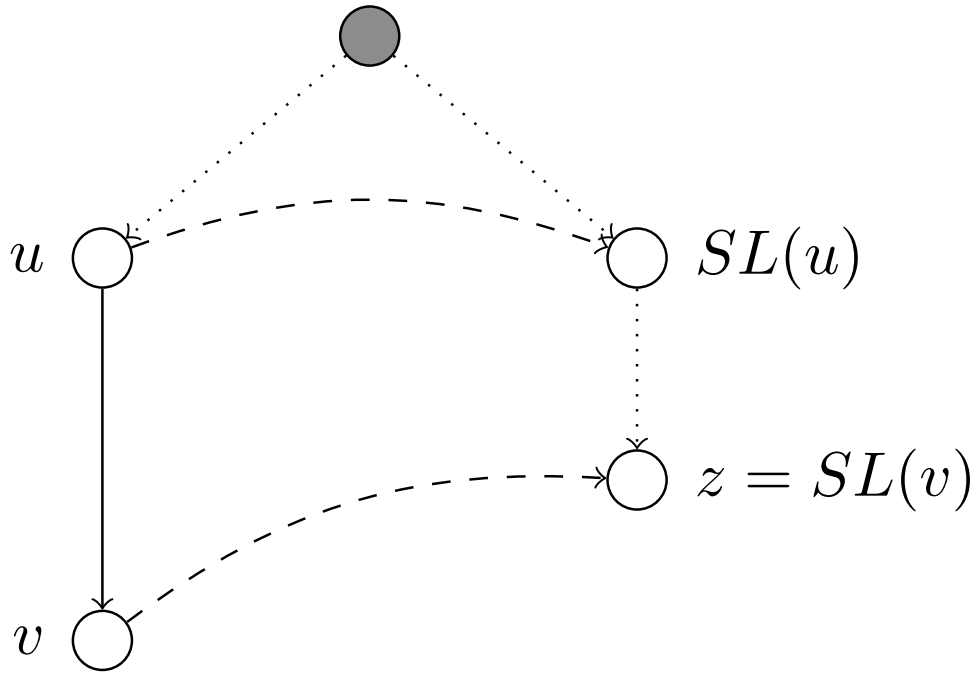
The set of k-mers of DBG_k of R is isomorphic to $Init(R)$.

Arcs of the de Bruijn Graph

Idea

1. Take an initial node v
2. follow its suffix link to node z (lose the first letter of its k-mer)
3. if needed, go the children of z to find its extensions
4. check whether the extensions are valid

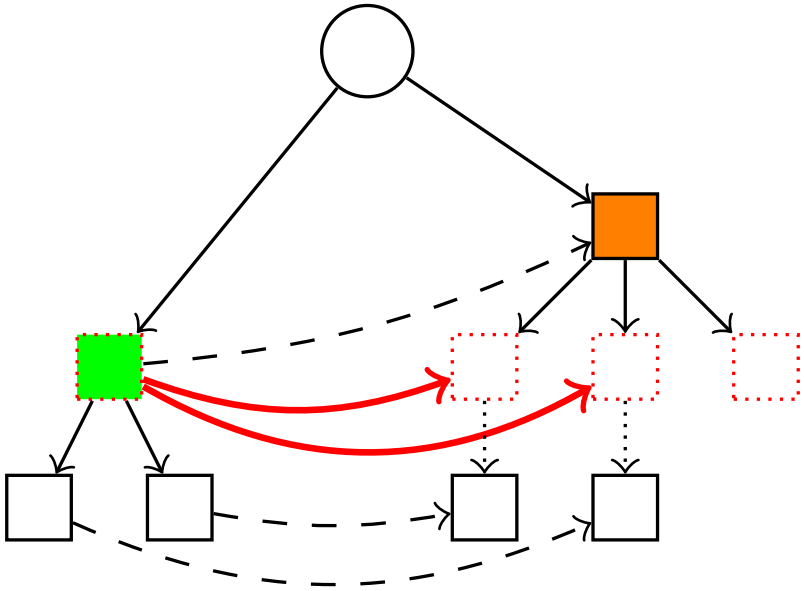
Let v be an initial node, u its father, and z the node pointed at by the suffix link of v.



Kinship property of suffix links in suffix trees

Let v be a node of suffix tree. If it exists, the suffix link of v belongs to the sub-tree of the suffix link of $p(v)$.

Example of construction of arcs of DBG_k



v is initial exact
with a several children

DBG construction

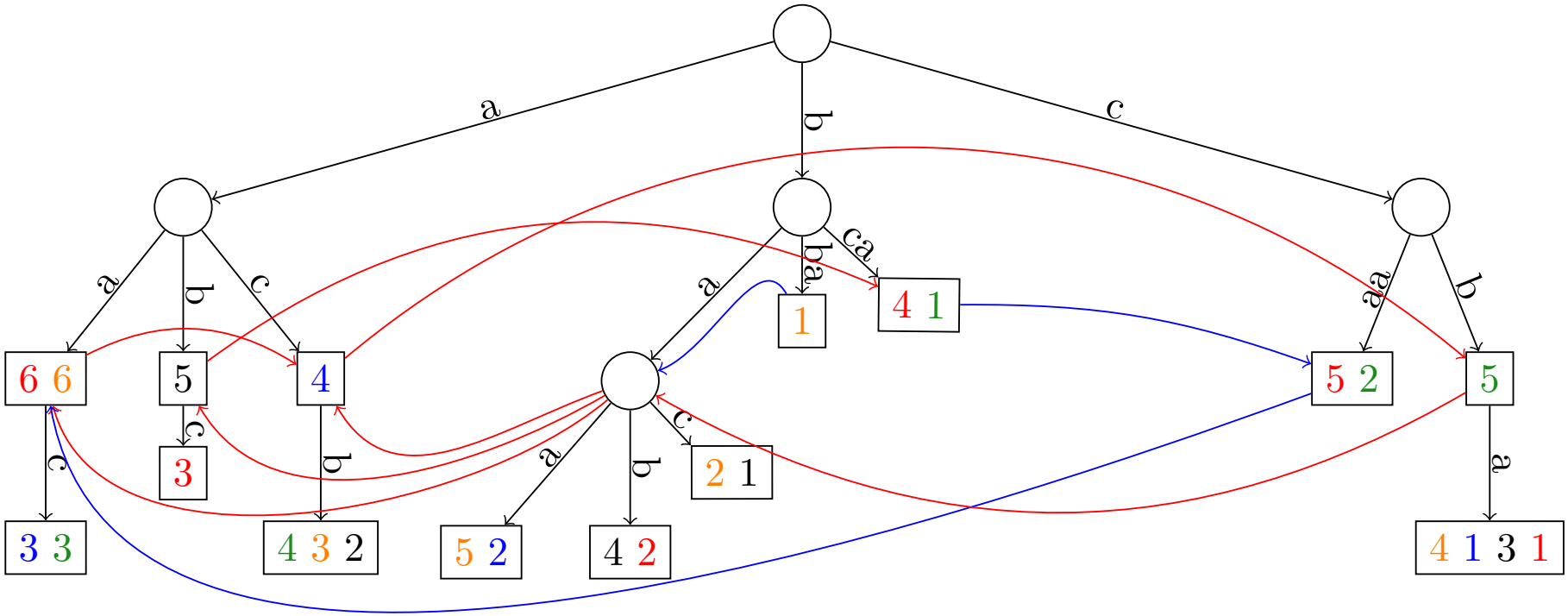
Theorem

Given the TST of a set of words R.
The construction of the De Bruijn Graph takes linear time in $||R||$.

Proof

All different cases of the typology are processed in constant time.

DBG_2 of R embedded in the TST of R



$R = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$ and $k = 2$

Linear space construction

Theorem

Given the TST of a set of words R.
The construction of the De Bruijn Graph takes linear space in the size of the De Bruijn Graph.

Proof

The size of the TST is linear in the size of the De Bruijn Graph of the same order.

Conclusion

Conclusion

An algorithm that builds the De Bruijn Graph

from a Truncated Suffix Tree

in linear time in the size of the input and

in linear space in the size of the output.

Funding and acknowledgments


Thanks for your attention

Questions?