From Indexing Data Structures to de Bruijn Graphs

Bastien Cazaux, Thierry Lecroq, Eric Rivals

LIRMM & IBC, Montpellier - LITIS Rouen

June 15, 2014

Motivation

- De Bruijn Graph is largely used in *de novo* 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] [Rødland, 2013]
- None is able to build the Contracted De Bruijn Graph directly.

Indexing data structures

- Numerous data structures: suffix tree, affix tree, suffix array, etc.
- to index one or several texts (generalized index)
- functionally equivalent to
- compressed versions (CSA, FM-index, CST, etc)

Relation between indexing structures and assembly graphs

- Generalised Suffix Tree (GST)
- indexes all factors of a set of texts
- is functionally equivalent to other indexes (SA, CSA, FM-index)

Relation between indexing structures and assembly graphs

- Generalised Suffix Tree (GST)
- indexes all factors of a set of texts
- is functionally equivalent to other indexes (SA, CSA, FM-index)

Question: How to directly build, e.g., the assembly De Bruijn graph?

in classical or contracted form?

Generalized Suffix Tree (GST)

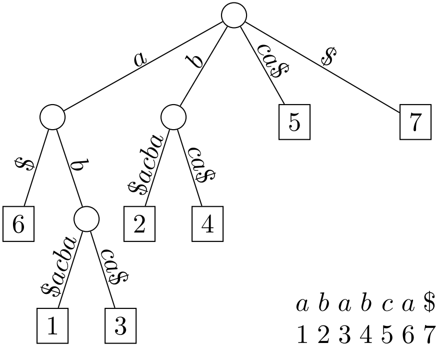
Example

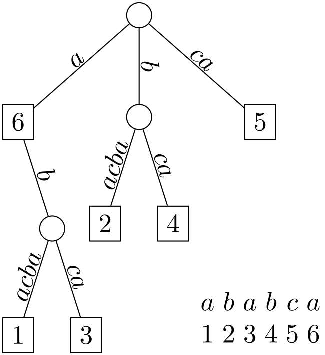
$$S = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$$

$$||S|| = \sum_{s_i \in S} |s_i|$$

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

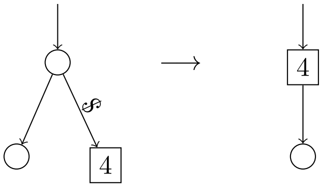
Suffix Tree


a
b
ca$
$
5
7
$
b
$acba
ca$
6
2
4
$acba
ca$
1
3
a b a b c a $
1 2 3 4 5 6 7


a
b
ca
6
5
b
acba
ca
2
4
acba
ca
1
3
a b a b c a
1 2 3 4 5 6

Feature

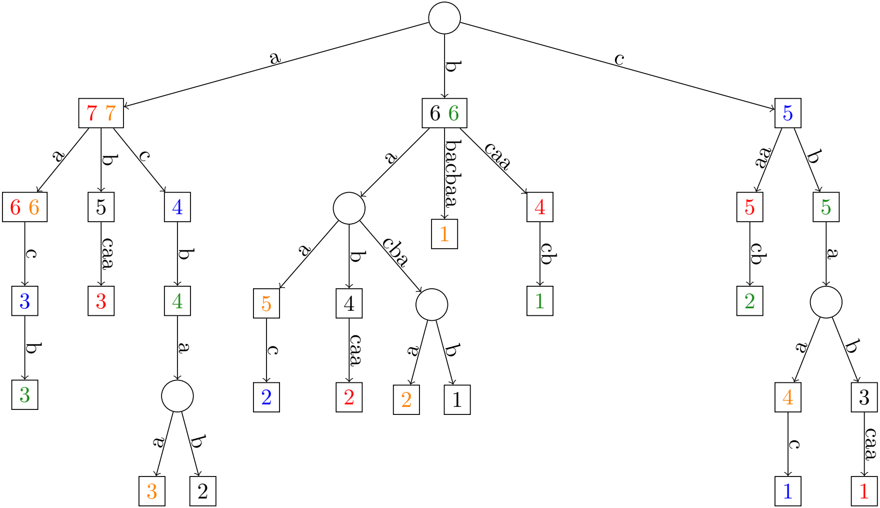
1. take in input a set of words
2. no use an end marking special symbol
   i.e. no hypothesis requiring that a suffix is not the prefix of another suffix



Theorem

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

Generalized Suffix Tree (GST)



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

De Bruijn Graphs

De Bruijn Graph

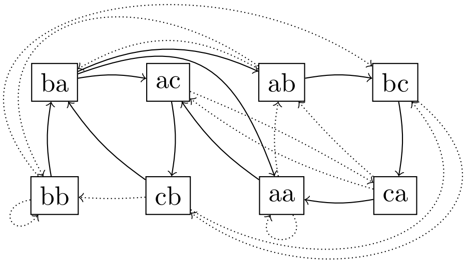
The assembly De Bruijn Graph (DBG_k^+)

Let k be a positive integer satisfying $k \geq 2$ and
$S = \{s_1, \ldots, s_n\}$ be a set of n words.
The De Bruijn graph of order k of S, denoted by DBG_k^+, is a graph such that

- the nodes are the k-mers of S and
- an arc links two k-mers of S if there exists an integer i such that these k-mers start at successive positions in s_i.

Remark: the arc definition implies that the two k-mers overlap by $k-1$ positions.

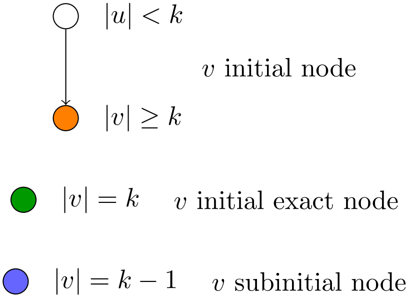
De Bruijn Graph for assembly



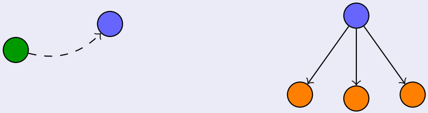
$S = \{bbacbaa, cbaac, bacbab, cbabcaa, bcaacb\}$

From the Generalized Suffix Tree to the DBG

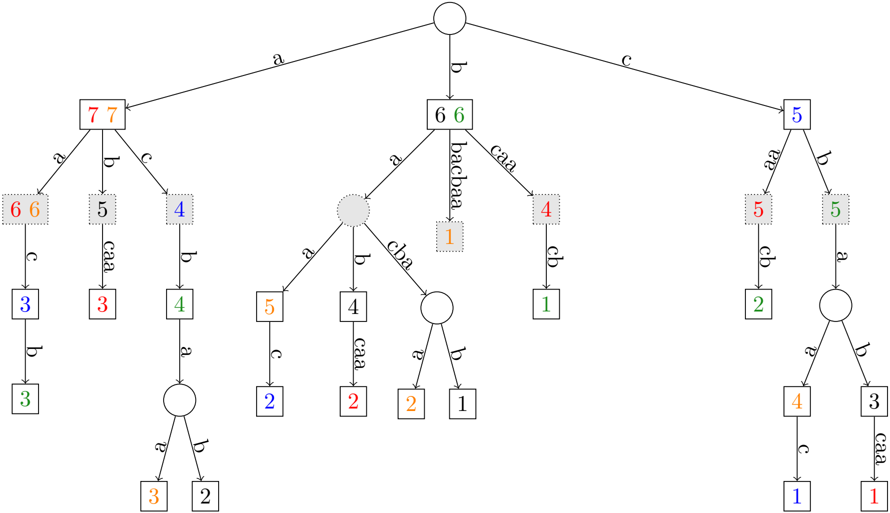
One defines 3 specific types of nodes in the GST (in V_S):



Some properties

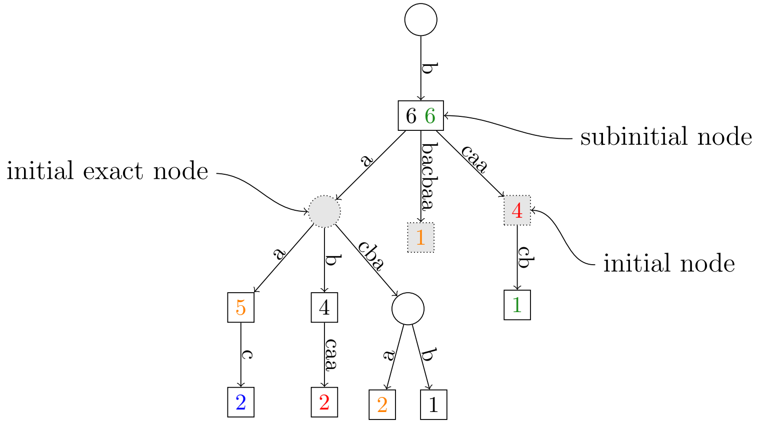


Generalized Suffix Tree (GST)



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

Generalized Suffix Tree (GST)



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

Nodes of the de Bruijn Graph

Notation: $Init_S$

Let $Init_S$ denote the set of initial nodes of the GST of S.

Property: node correspondence

The set of k-mers of DBG_k^+ of S is isomorphic to $Init_S$.

Arcs of the de Bruijn Graph

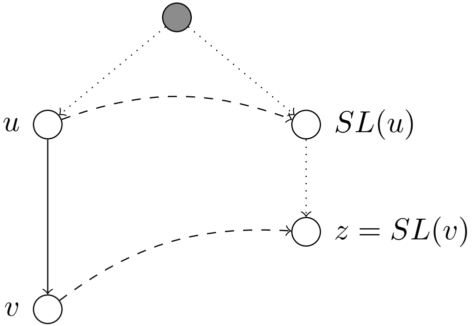
Idea

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

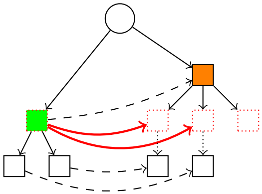
Let v be an initial node, u its father, and z the node pointed at by the suffix link of v.

Property 2

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 $par(v)$.



Arcs of DBG_k^+: case figure



v starting initial node: green

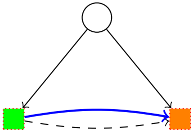
$z := \mathrm{SL}(v)$ node pointed at by its suffix link: orange

black straight arrows : arcs of ST

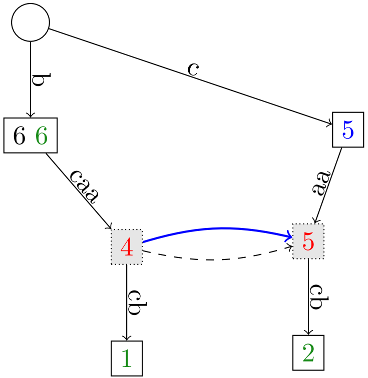
dotted arrows: suffix links

colored plain arrows: created arcs of the DBG_k^+

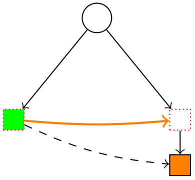
Arcs of DBG_k^+: Type 1



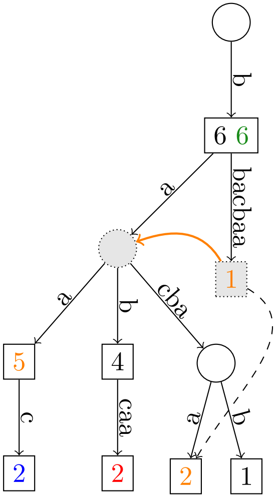
v is initial not exact &
z is initial



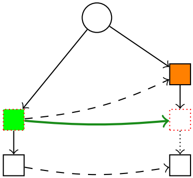
Arcs of DBG_k^+: Type 2



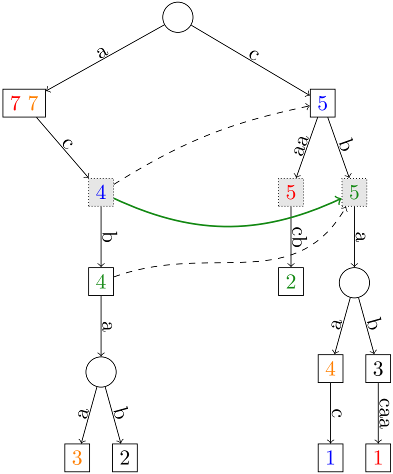
v is initial not exact &
z is not initial
i.e. z is deeper than word depth k



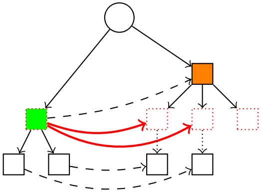
Arcs of DBG_k^+: Type 3



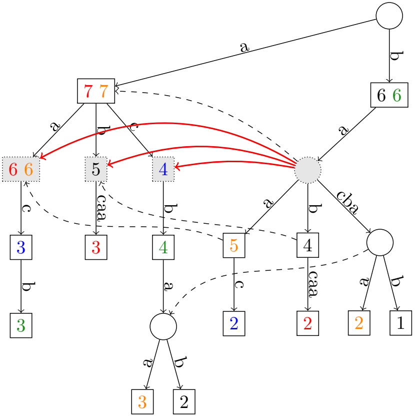
v is initial exact
with a single child



Arcs of DBG_k^+: Type 4



v is initial exact
with a several children



DBG construction

Theorem

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

DBG construction

Theorem

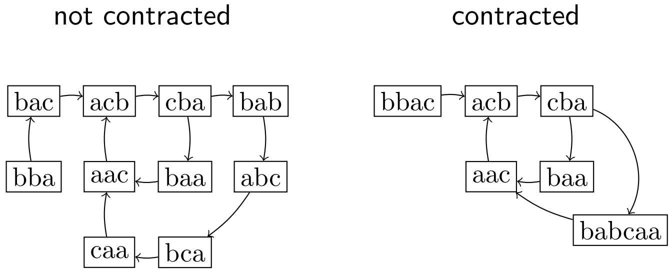
Given the GST of a set of words S.
The construction of the De Bruijn Graph takes linear time in $||S||$.

Proof

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

Contracted De Bruijn Graph

Example



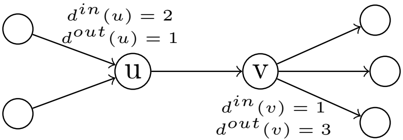
$S = \{\textit{bbacbaa}, \textit{cbaac}, \textit{bacbab}, \textit{cbabcaa}, \textit{bcaacb}\}$

Left and right extensible nodes

Left or right extensible node

Let v be a node of the De Bruijn graph; we say that

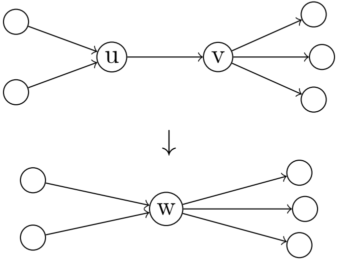
- v is left extensible if and only if $d^{in}(v) = 1$.
- v is right extensible if and only if $d^{out}(v) = 1$.



Contracted De Bruijn Graph

Contracted De Bruijn Graph

Let $G := (V_G, E_G)$ be the De Bruijn graph of order k of S. We call $K := (V_K, E_K)$ the Contracted De Bruijn Graph ($CDBG_k^+$) of order k of S, the graph obtained from G by contracting iteratively the arcs (u, v) where u is right extensible and v is left extensible.



Find right extensible nodes

Property 4: right extensible

- Initial nodes of type 1, 2 or 3 are right-extensible.
- Initial nodes of type 4 are not right-extensible.

Find right extensible nodes

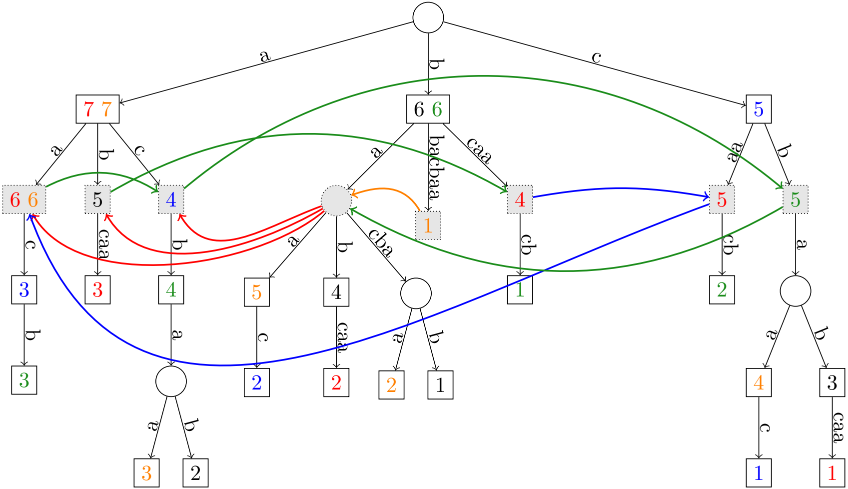
Property 4: right extensible

- Initial nodes of type 1, 2 or 3 are right-extensible.
- Initial nodes of type 4 are not right-extensible.

Proof

- Types $1, 2$ or 3: word of node v has only one extension in S.
- Types 4 nodes have several extensions in S.

Case of right extensible nodes ex. for $k = 2$

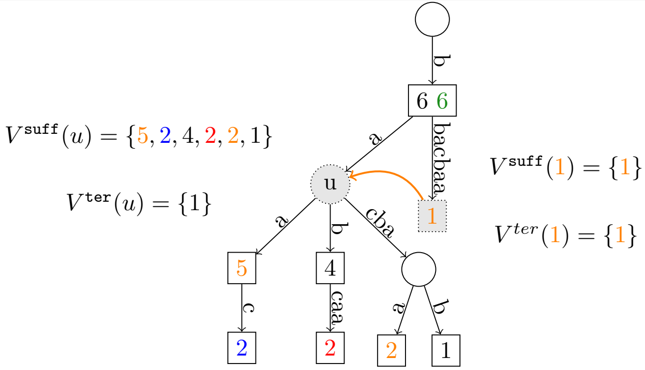


Find left extensible nodes

Let v be a right extensible k-mer/node and (v, u) an arc of DBG_k^+.
Question: is u left extensible?

Property 5: left extensible

u is left extensible if and only if $V^{\mathtt{suff}}(u) - V^{\mathtt{ter}}(u) = V^{\mathtt{suff}}(v)$



Find left extensible nodes

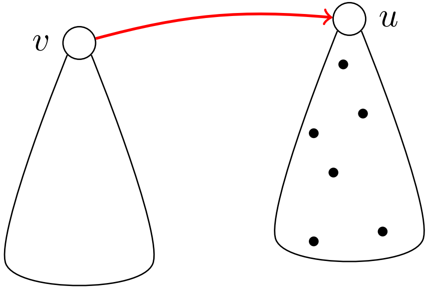
Let v be a right extensible k-mer/node and (v, u) an arc of DBG_k^+.
Question: is u left extensible?

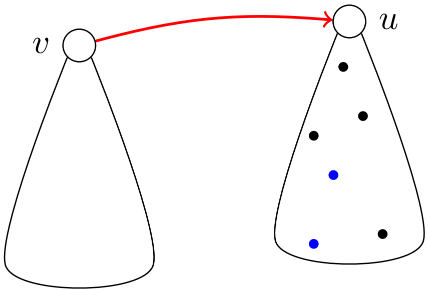
Property 5: left extensible

u is left extensible if and only if $V^{\mathtt{suff}}(u) - V^{\mathtt{ter}}(u) = V^{\mathtt{suff}}(v)$

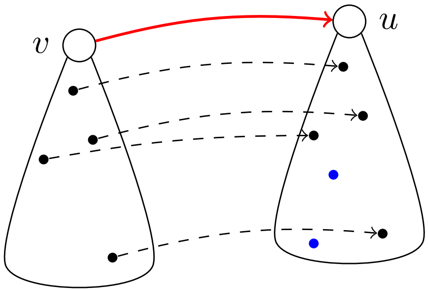
Idea

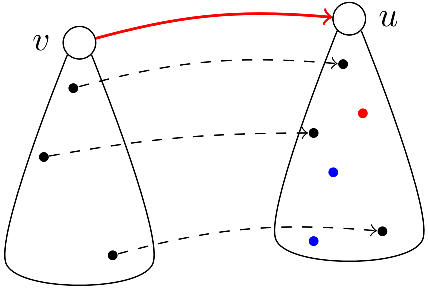
- determine how many nodes in sub-tree of u have Suffix Links pointing to them coming from the sub-tree of v
- do not account for nodes representing the first suffix of words in S which we call *terminal* nodes
- compare the cardinalities of non terminal suffixes in u sub-tree with suffixes in v sub-tree.


v
u


v
u

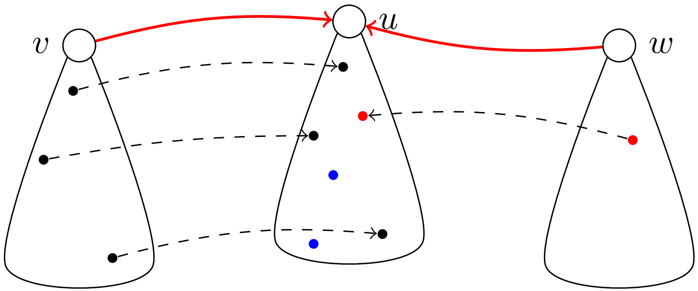
u is left extensible



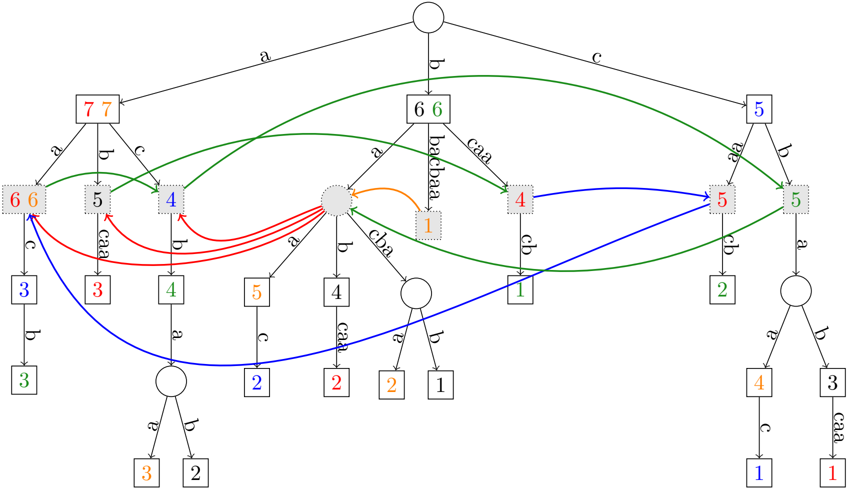


v
u

u is not left extensible



Case of left extensible nodes ex. for $k = 2$



Contracted DBG construction

Theorem

Given the GST of a set of words S.
The construction of the Contracted DBG_k^+ takes linear time in $||S||$.

Idea of proof.

Detection of right extensible nodes is obtained from the 4 types.

To get left and right extensible nodes in constant time preprocess:

- $V^{\mathtt{suff}}(u)$ and $V^{\mathtt{ter}}(u)$ for each node u of GST, and
- for each node w of ST such that $|w| \geq k$
  a pointer to its ancestral initial node.

□

Conclusion and Future works

Conclusion

A linear time algorithm that

builds the Contracted De Bruijn Graph

from a Generalized Suffix Tree or a Suffix Array

Future works

- Use only a slice of the suffix tree
- Update the order of the DBG: dynamically changing k
- Go for compressed indexes instead of a Suffix Tree

Funding and acknowledgments

cnrs dépasser les frontières

AGENCE NATIONALE DE LA RECHERCHE ANR

INVESTISSEMENTS D'AVENIR

IBC Institut de biologie computationnelle

Colib'read

Thanks for your attention

Questions?