The Gapped-Factor Tree

PSC'06

Pierre Peterlongo[1] Julien Allali[1] Marie-France Sagot[2]

[1] Institut Gaspard-Monge, Université de Marne-la-Vallée

[2] Inria Rhône-Alpes, UMR 5558 Biometrie et Biologie Évolutive, Lyon and King's College, London

August 30

Outline

Goal - Motivations

Overview

Preliminaries
- Ukkonen suffix tree construction
- k-factor tree construction (Allali - Sagot)

Construction Algorithm
- Construction
- Complexity

Conclusion

Goal - Motivations

Overview

Preliminaries

Ukkonen suffix tree construction

k-factor tree construction (Allali - Sagot)

Construction Algorithm

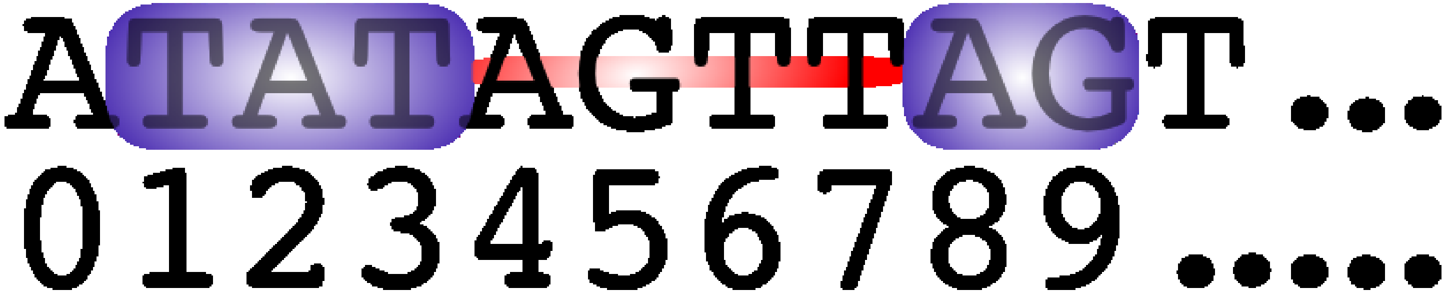
Construction

Complexity

Conclusion

Goal

- Indexation of *gapped-factors* :
  - A k-factor,a gap of length d, and a k'-factor
  - a (k, d, k')-gapped-factor



Motivations

Stringology

- Extensive use of k-factors (q-gram, k-mer)
- Gapped-factors for sets of k-factors
- Indexation structure : interesting application of the suffix tree

Bioinformatics

- Motif inference
- Binding site detection

Goal - Motivations

Overview

Preliminaries

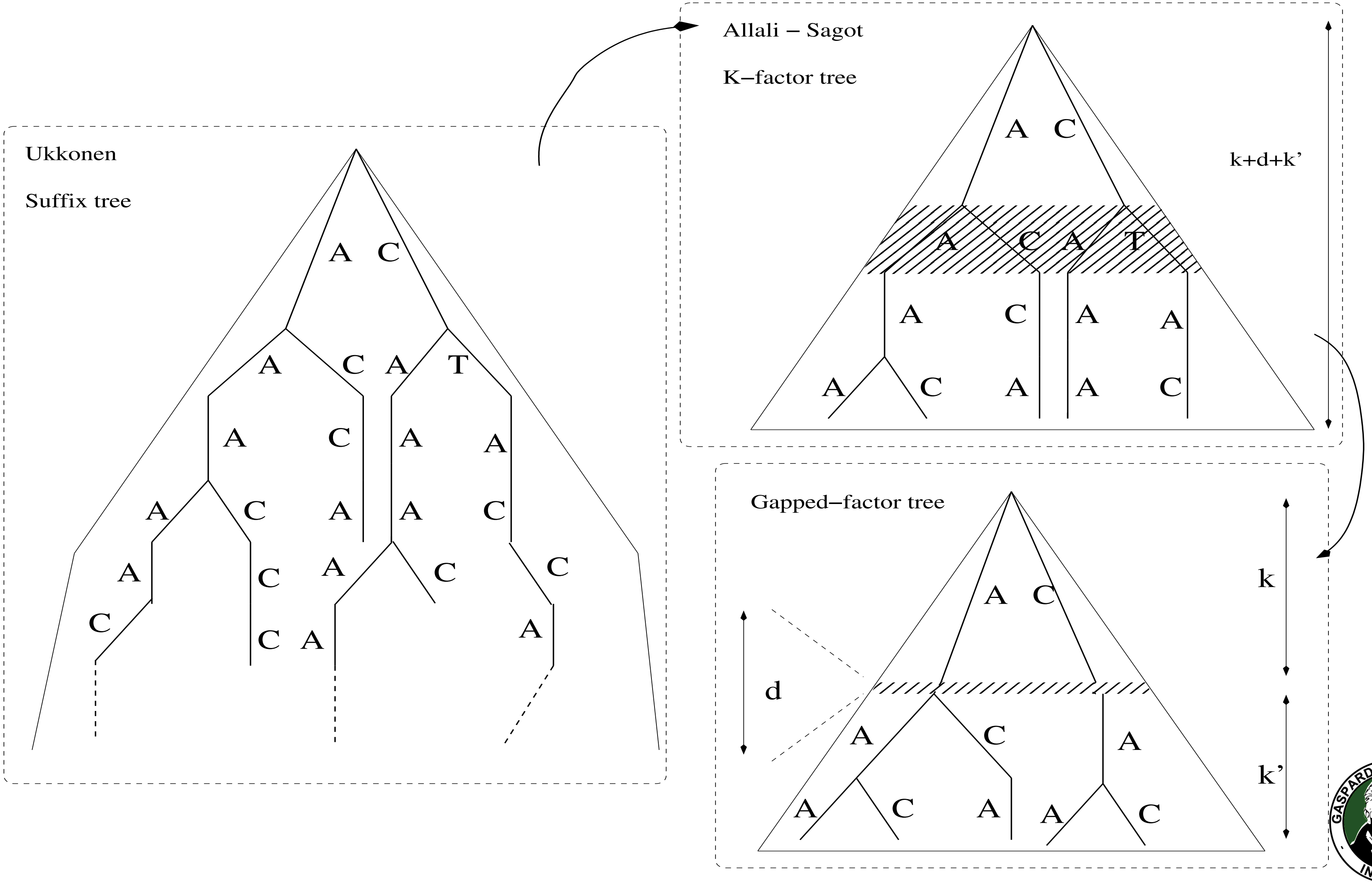
- Ukkonen suffix tree construction
- *k*-factor tree construction (Allali - Sagot)

Construction Algorithm

- Construction
- Complexity

Conclusion

Overview



Ukkonen

Suffix tree

Allali – Sagot

K–factor tree

Gapped–factor tree

Goal - Motivations

Overview

Preliminaries

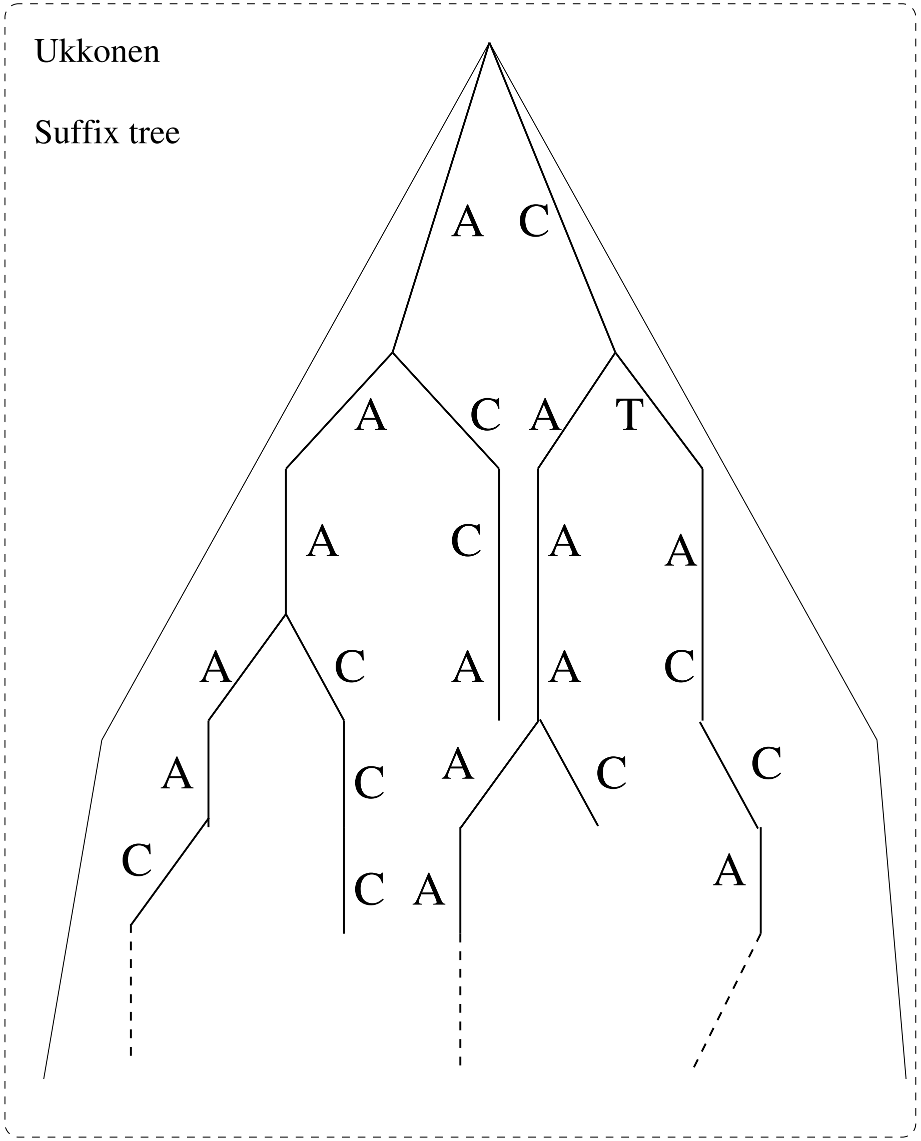
- Ukkonen suffix tree construction
- *k*-factor tree construction (Allali - Sagot)

Construction Algorithm

- Construction
- Complexity

Conclusion

Overview

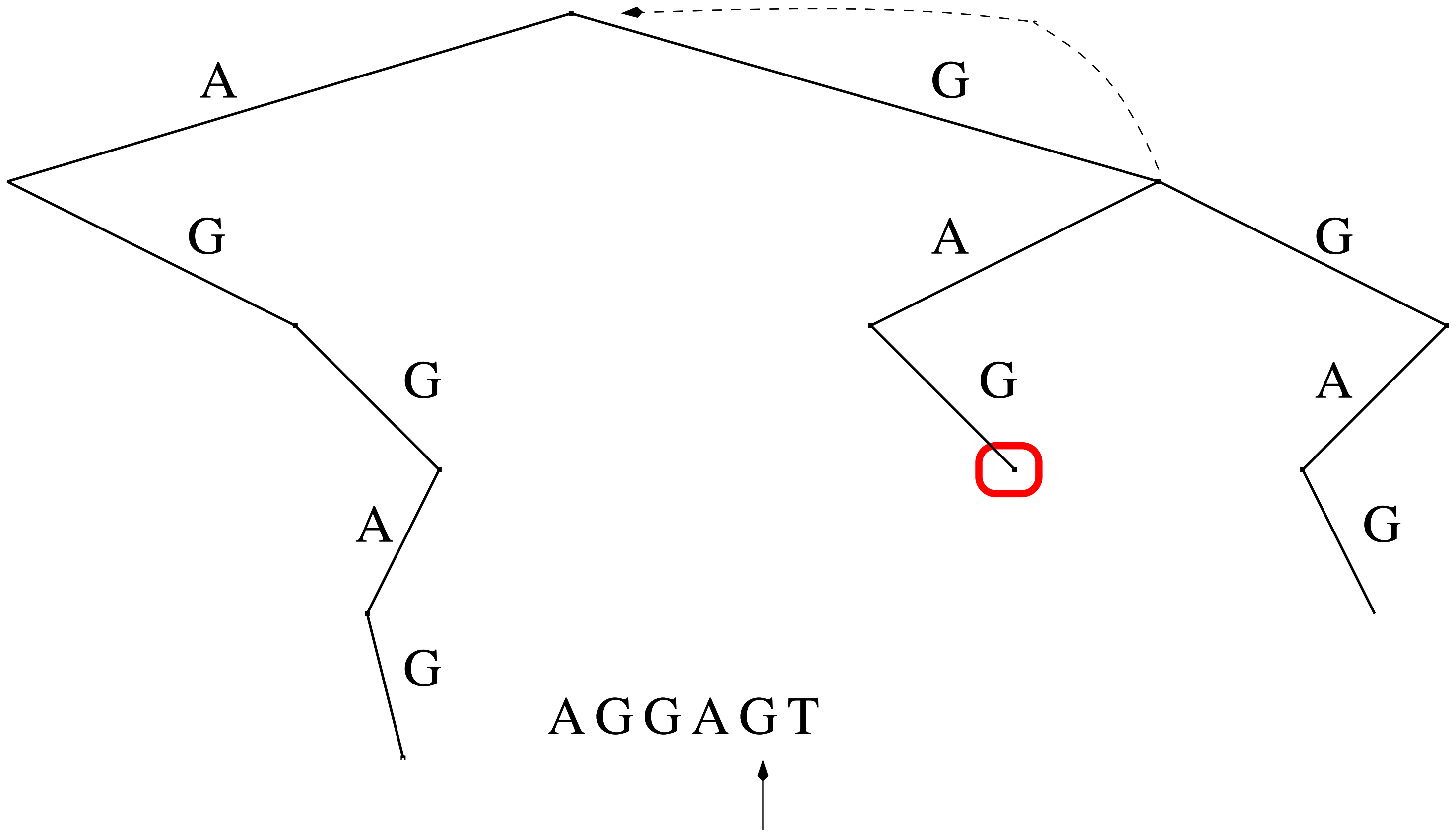


Ukkonen

Suffix tree

Suffix tree construction

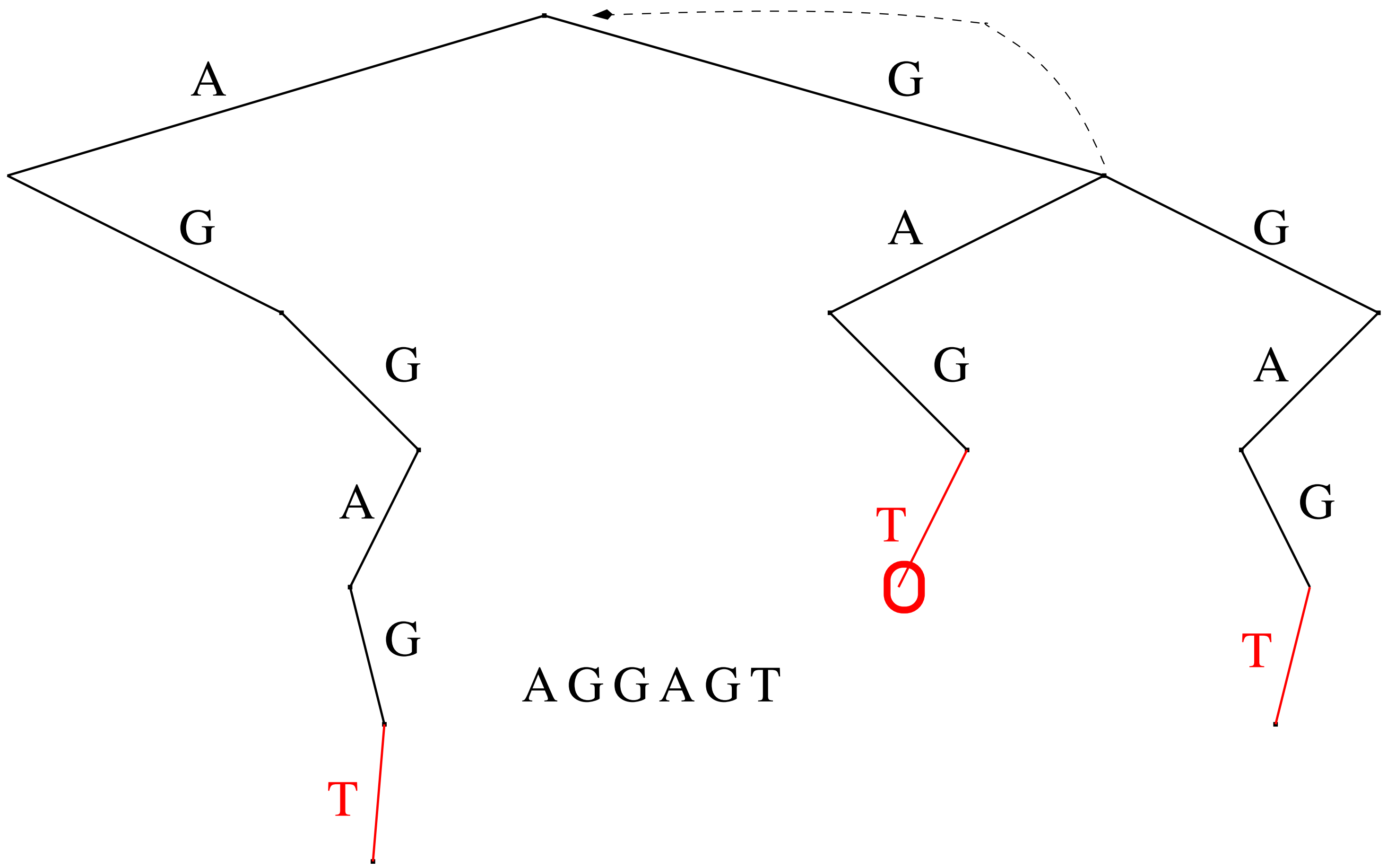
Suffix tree for *AGGAG*, *last_leaf* is marked



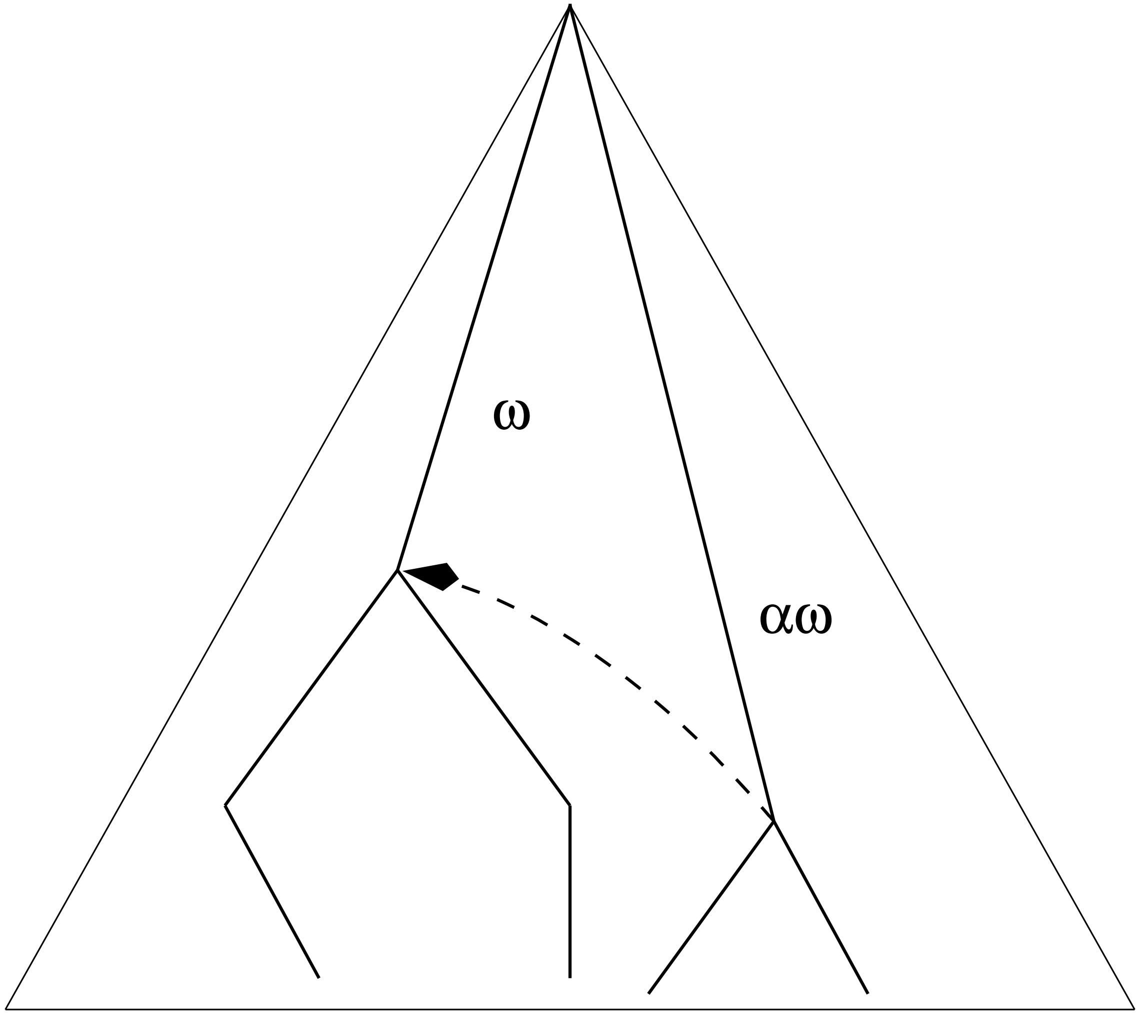
Suffix tree construction

Adding T to $AGGAG$, implicit extention of all leaves

$[Start\ \text{-}\ End] \Rightarrow end$: global variable

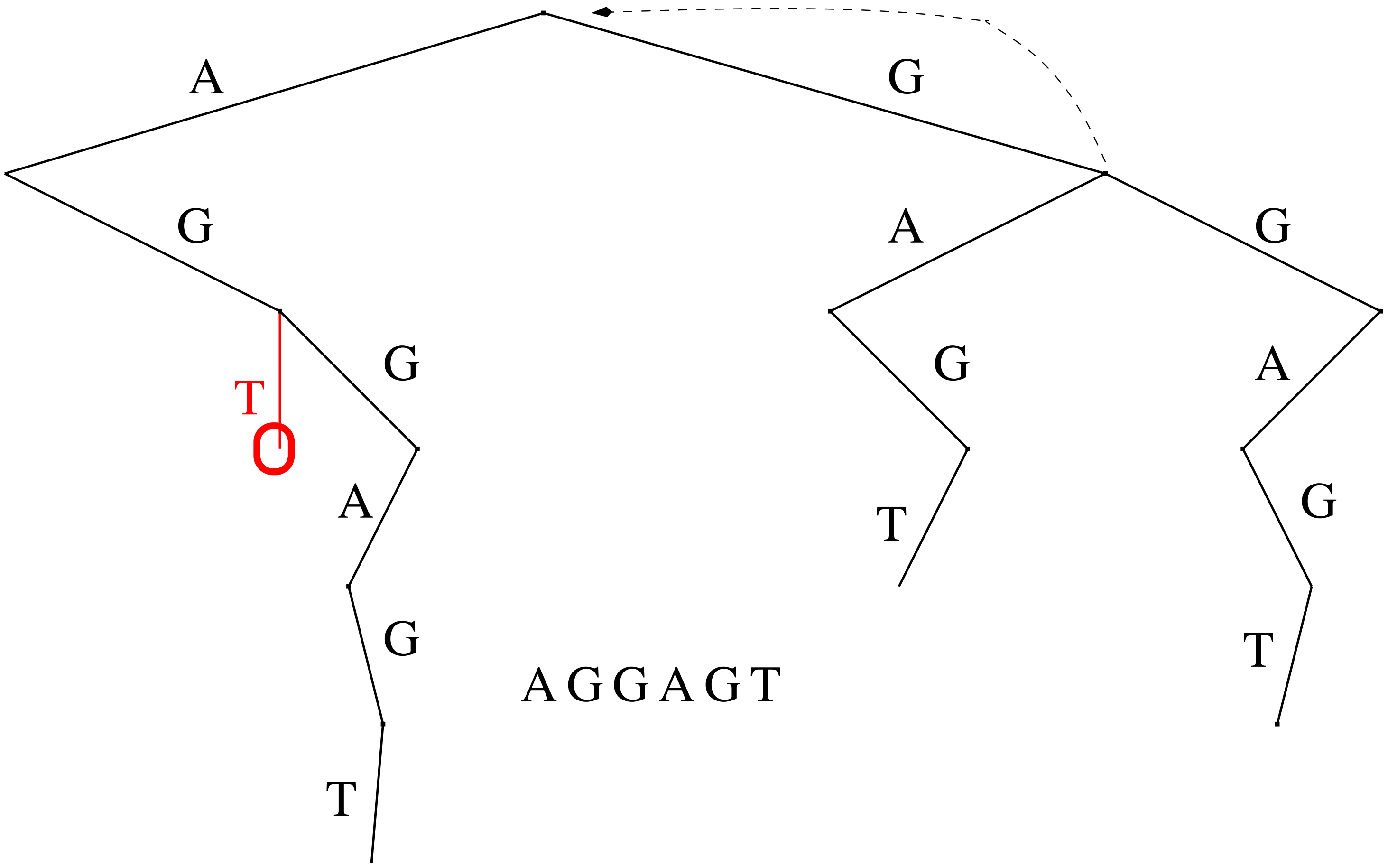


Suffix tree construction



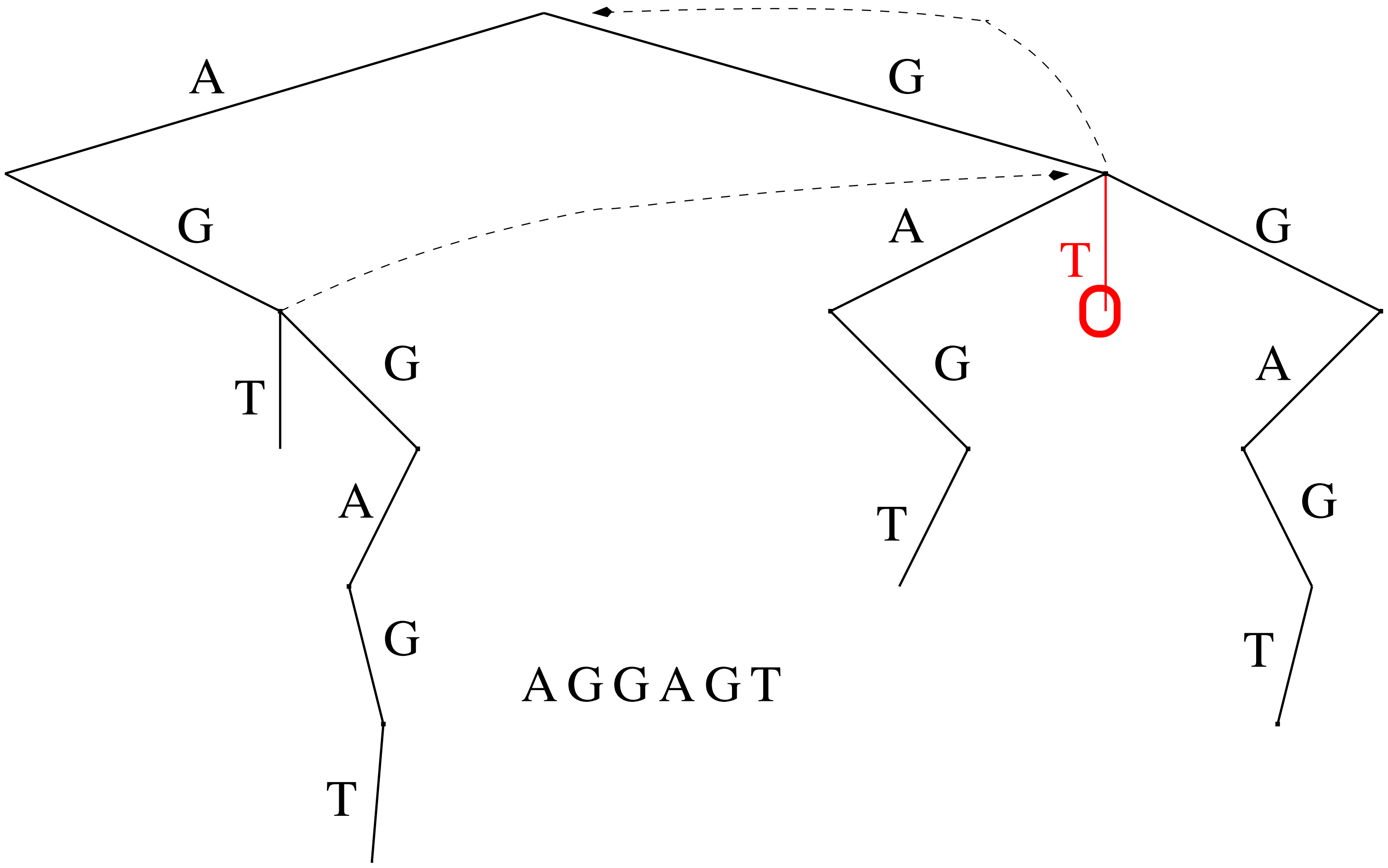
Suffix tree construction

Adding T to $AGGAG$, fast insertion of AGT from the root



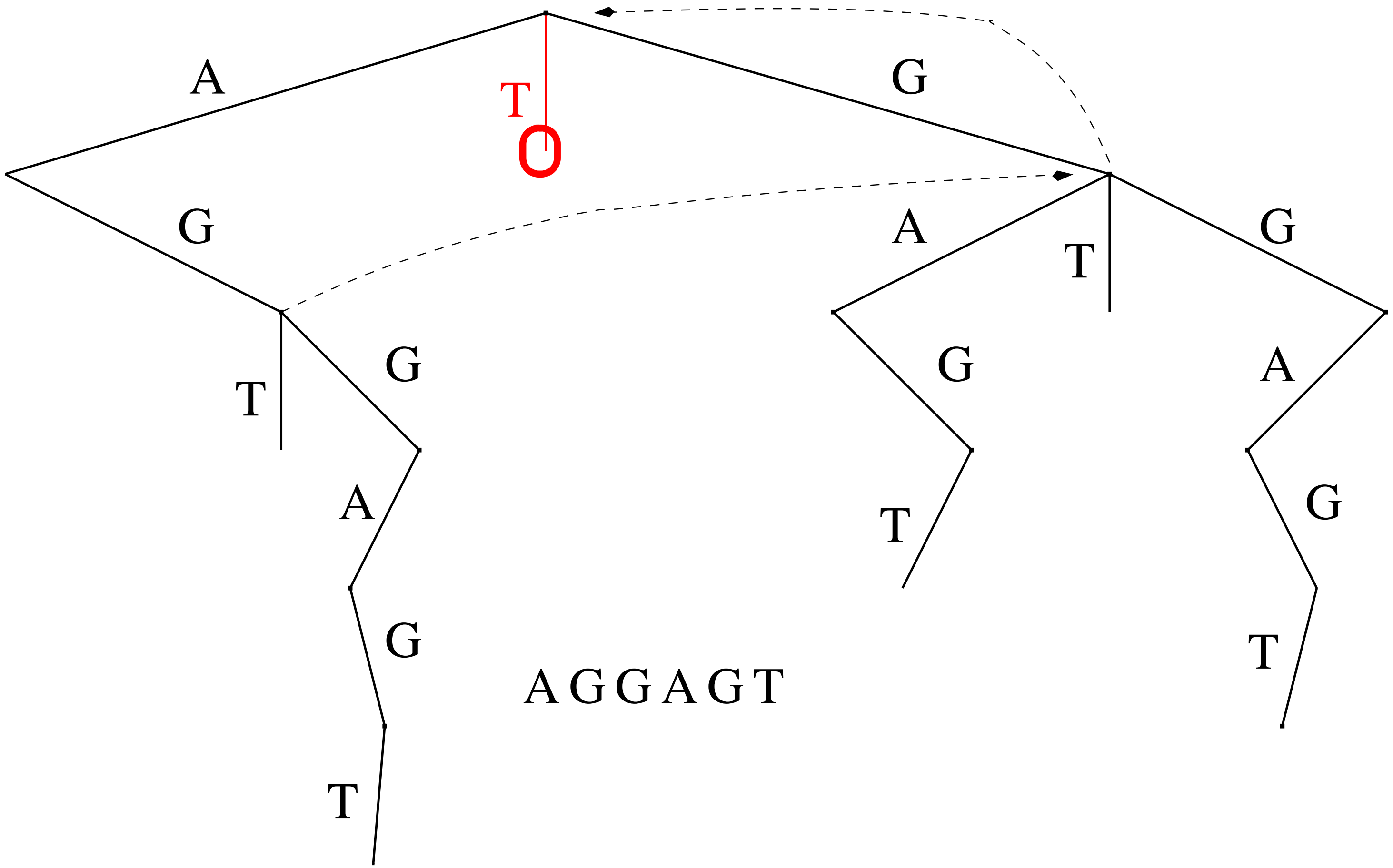
Suffix tree construction

Adding T to $AGGAG$, fast insertion of GT from the root



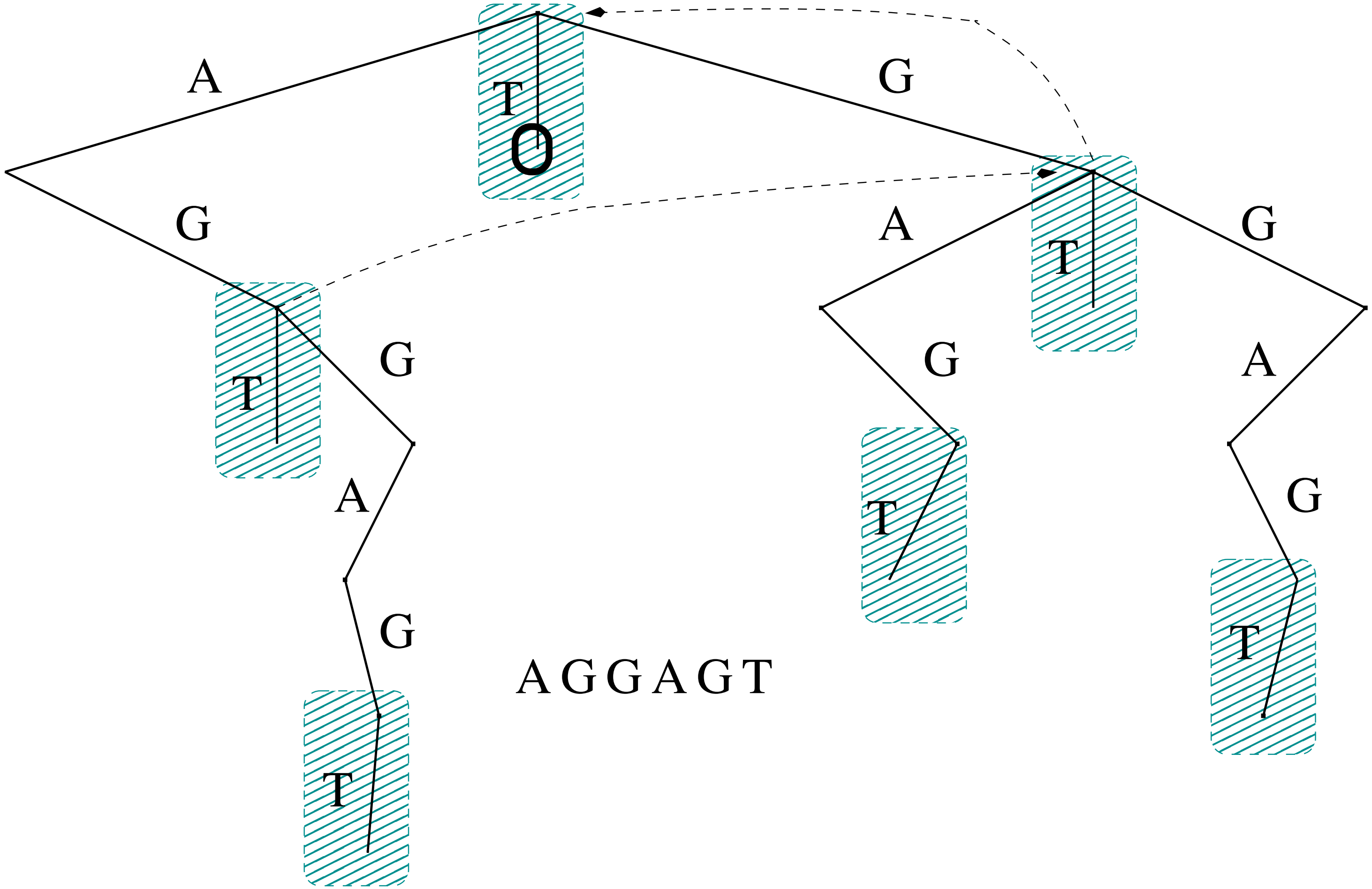
Suffix tree construction

Adding T to $AGGAG$, fast insertion of T from the root

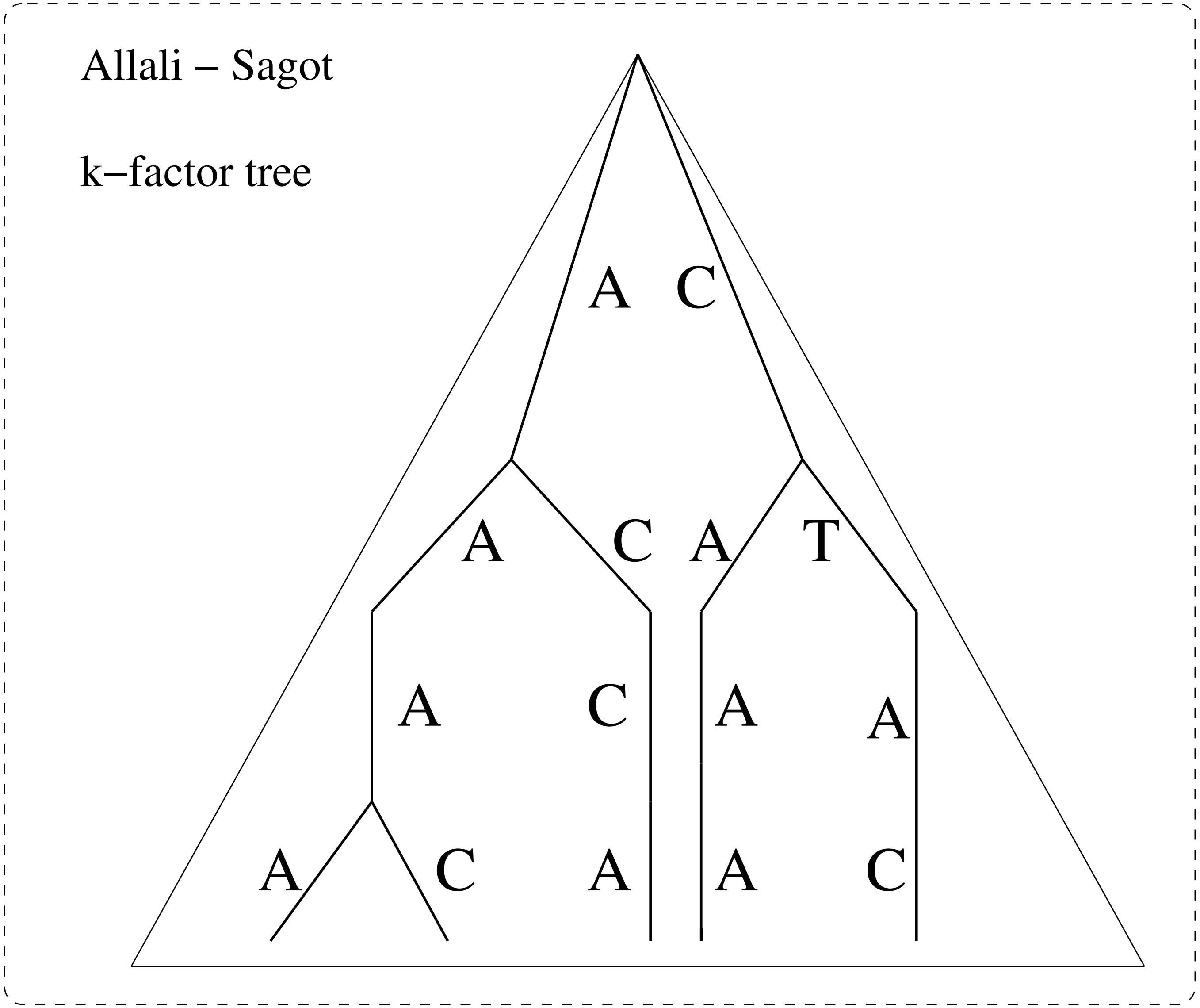


Suffix tree construction

remark : leaves are consecutively created at each level

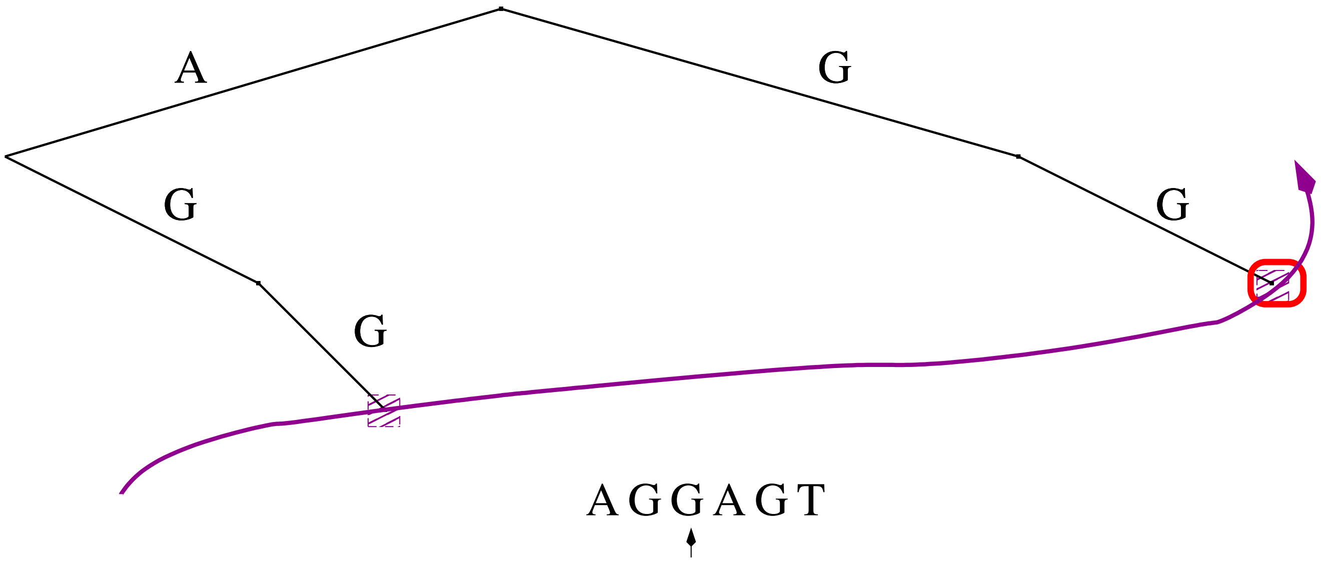


Overview



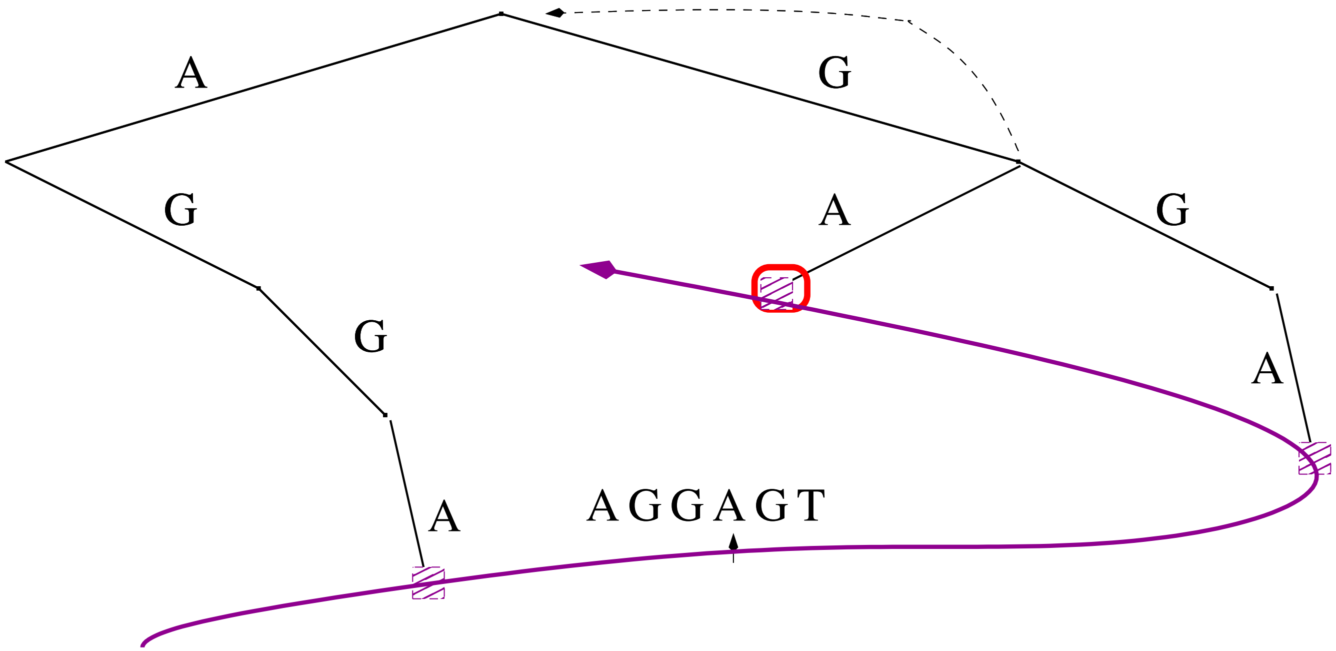
k-factor tree construction ($k = 4$)

$k - 1$ first phases : usual construction of a suffix tree, putting the leaves in a queue



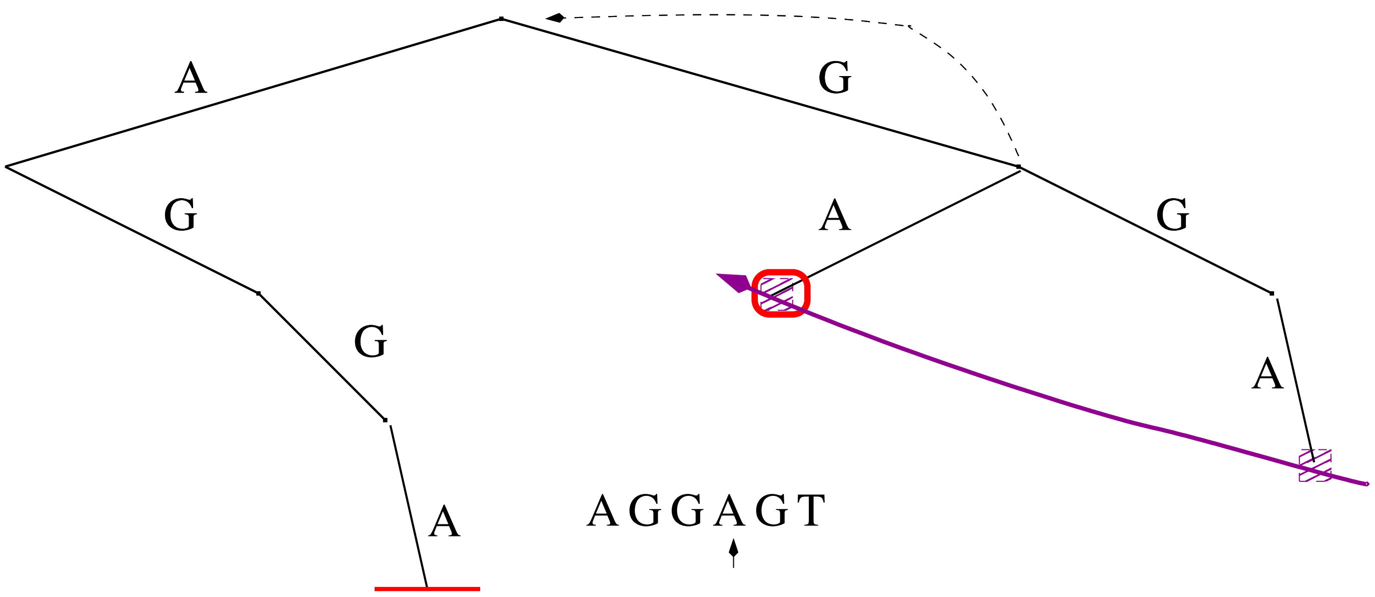
k-factor tree construction ($k = 4$)

After k^{th} phase, usual construction of a suffix tree, still putting the leaves in a queue, but...



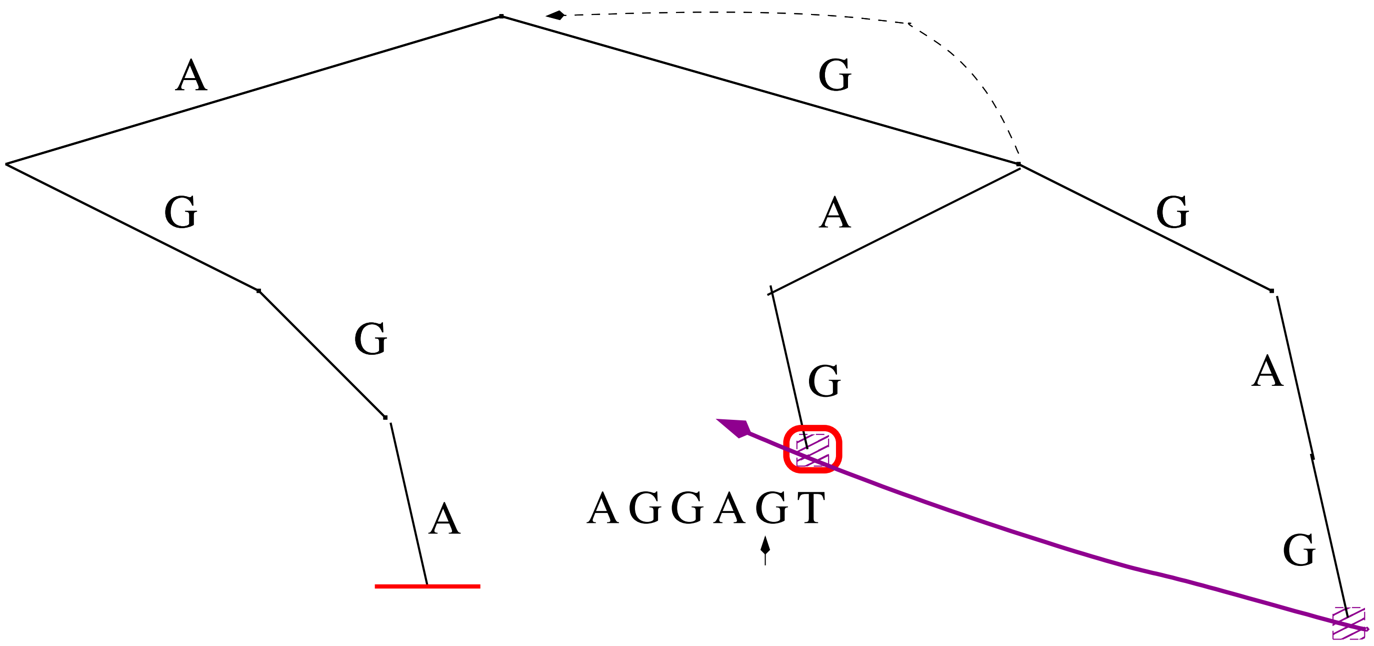
k-factor tree construction ($k = 4$)

...but removing the end of the queue at the end of the phase, stopping the automatic extension of the leaving leaf



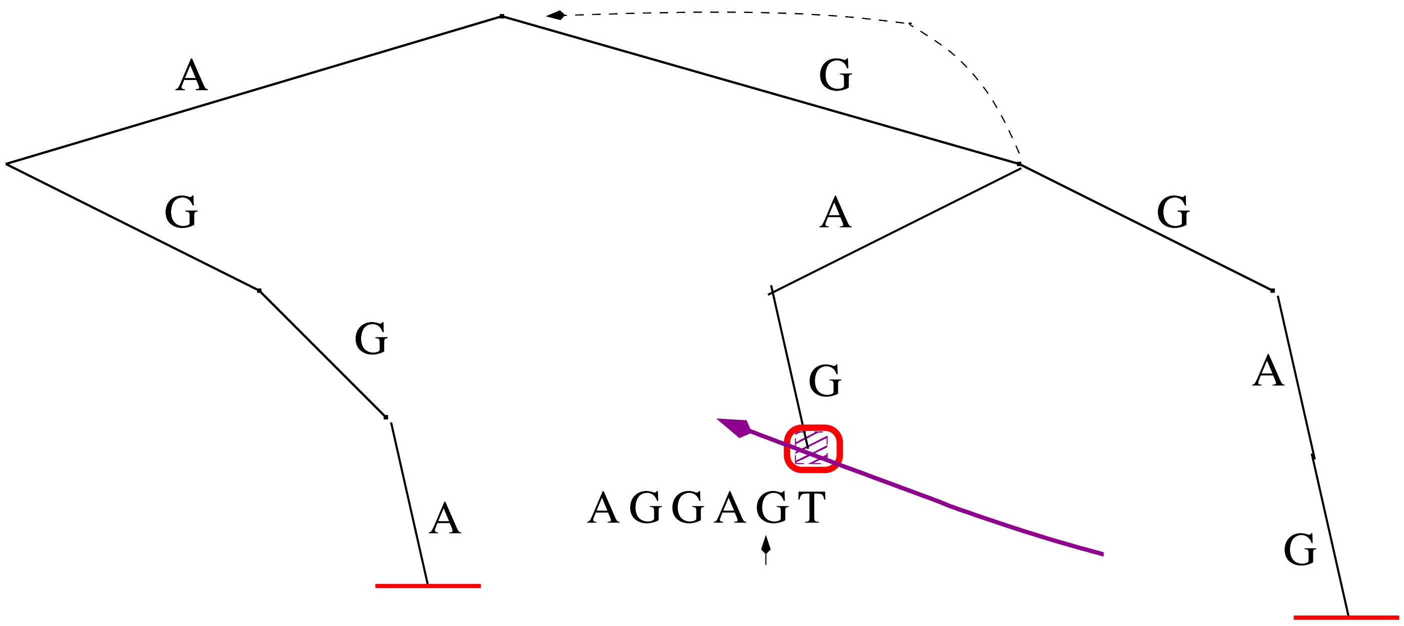
k-factor tree construction ($k = 4$)

Next step,



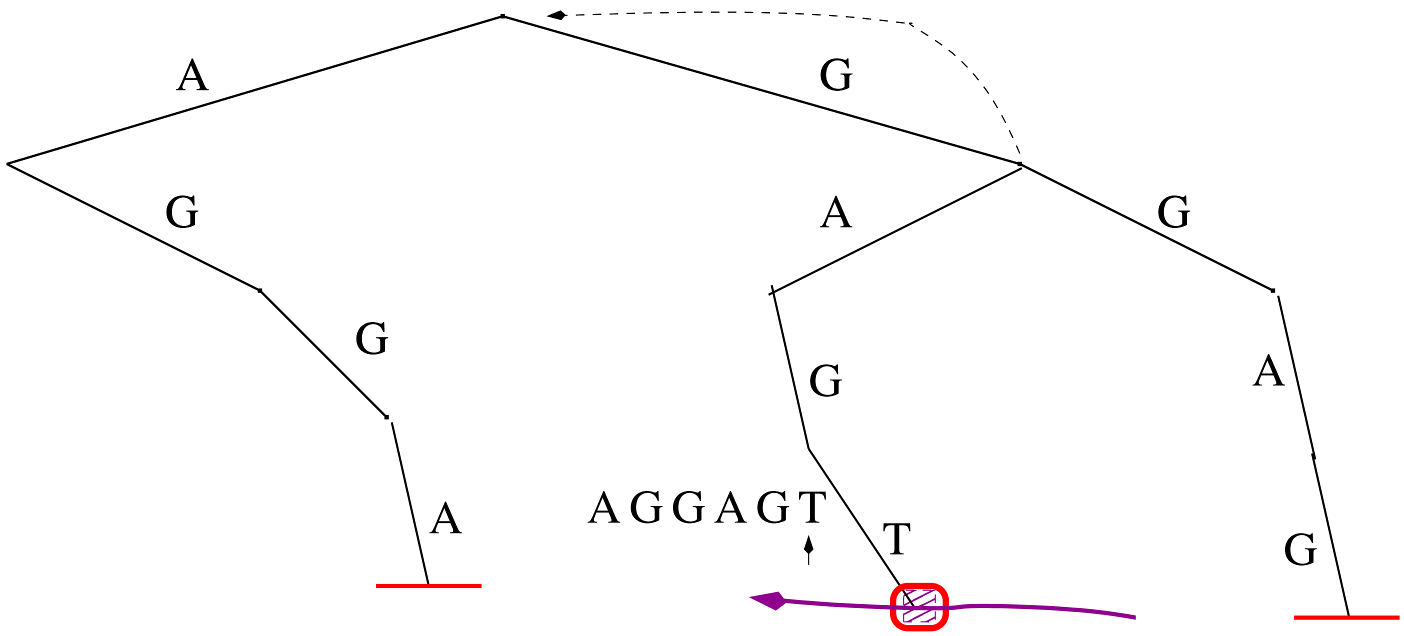
k-factor tree construction ($k = 4$)

Next step, remove head



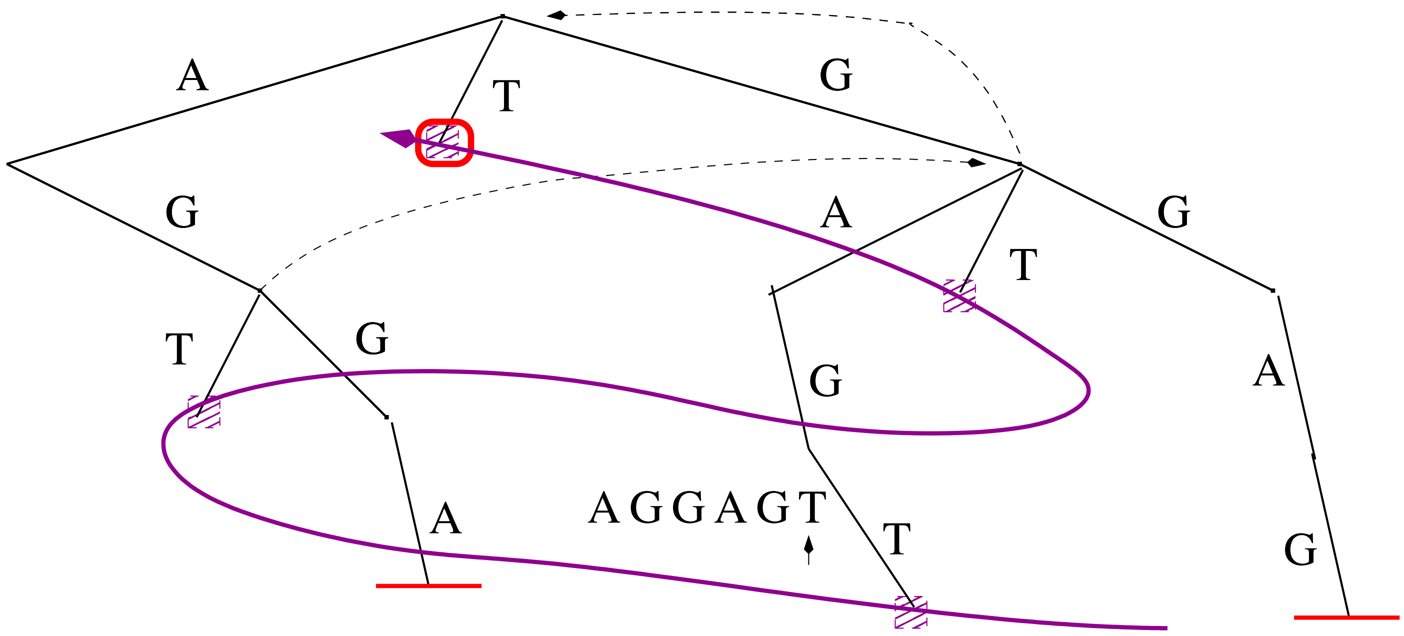
k-factor tree construction ($k = 4$)

Next step,



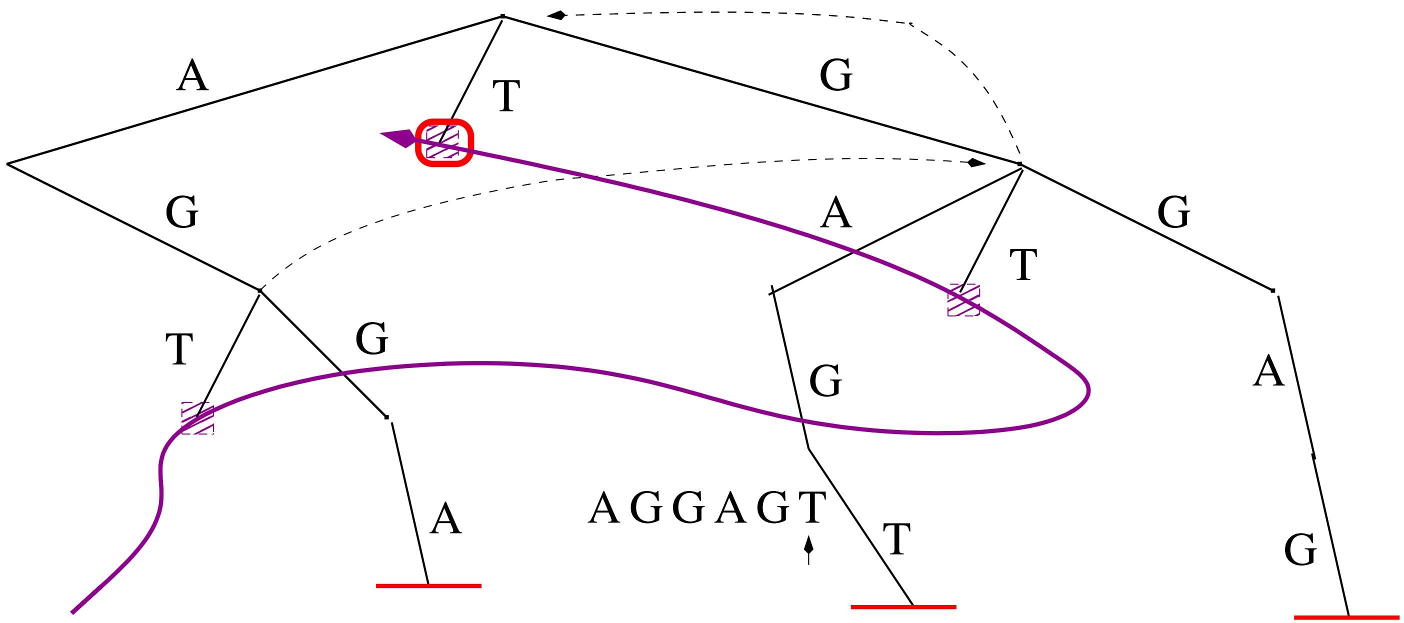
k-factor tree construction ($k = 4$)

Next step,



k-factor tree construction ($k = 4$)

Next step, remove head



Goal - Motivations

Overview

Preliminaries

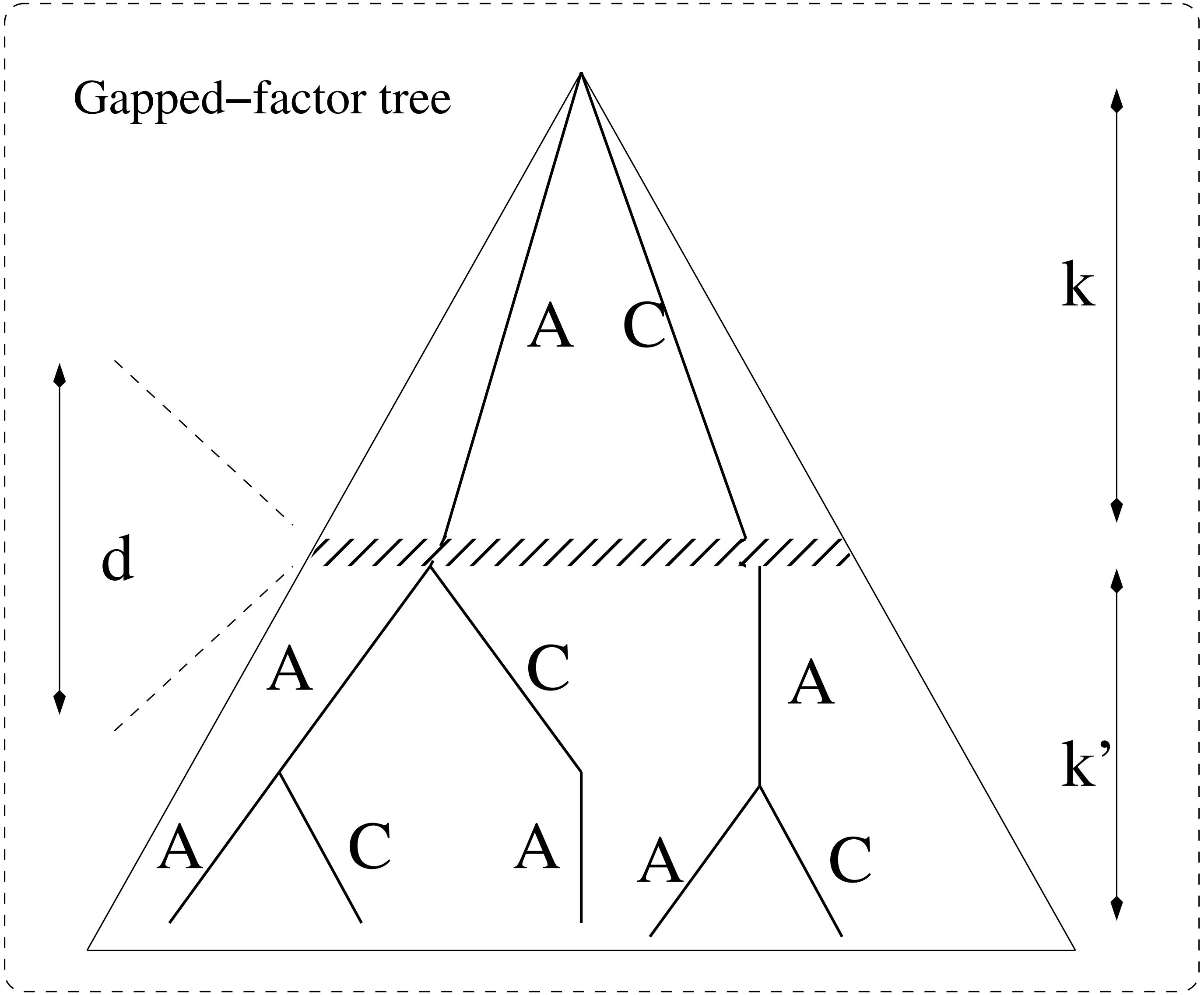
- Ukkonen suffix tree construction
- *k*-factor tree construction (Allali - Sagot)

Construction Algorithm

- Construction
- Complexity

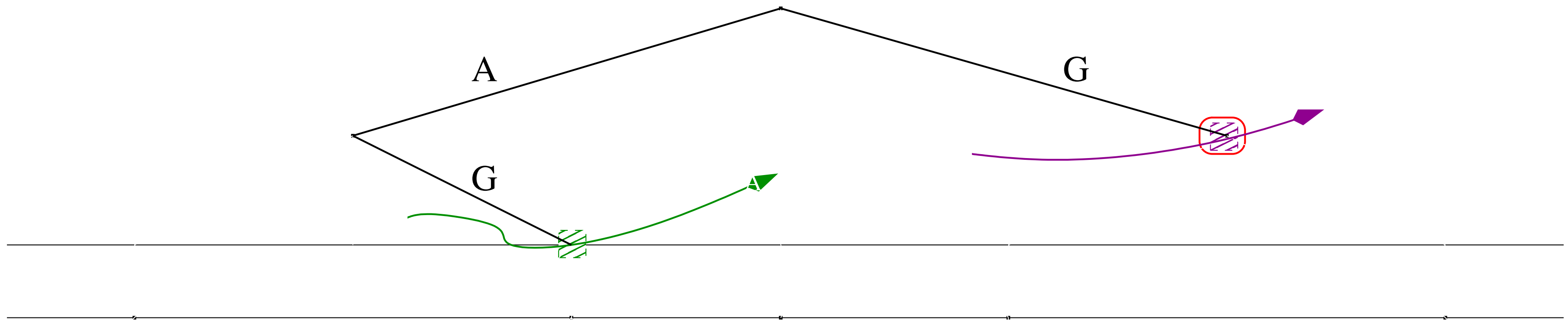
Conclusion

Overview



Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

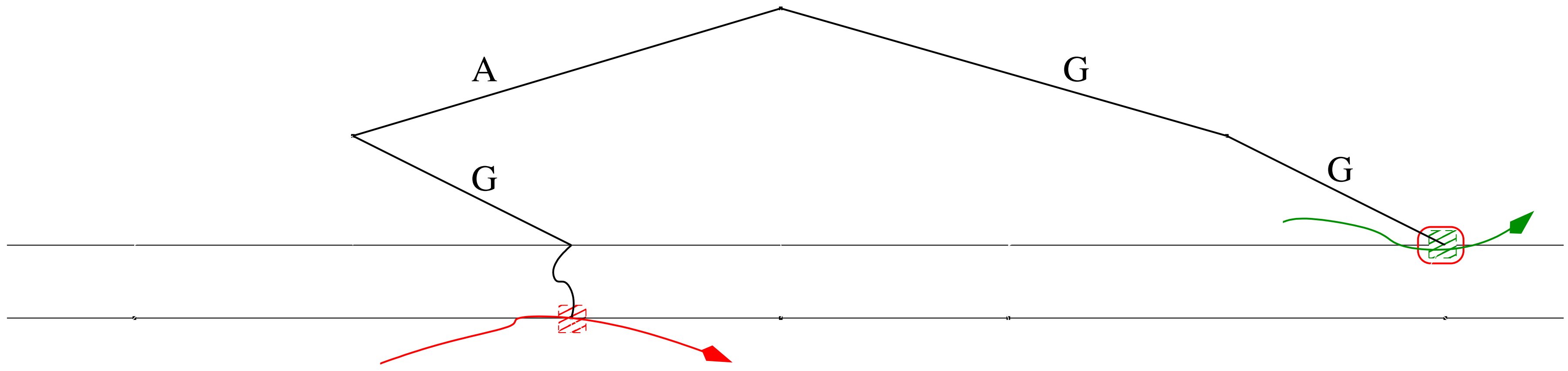
Three queues : upper extension leaves →, hidden extension leaves →,



AGGAGAACAA

Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

Three queues : upper extension leaves ⟶, hidden extension leaves ⟶, lower extension leaves ⟶

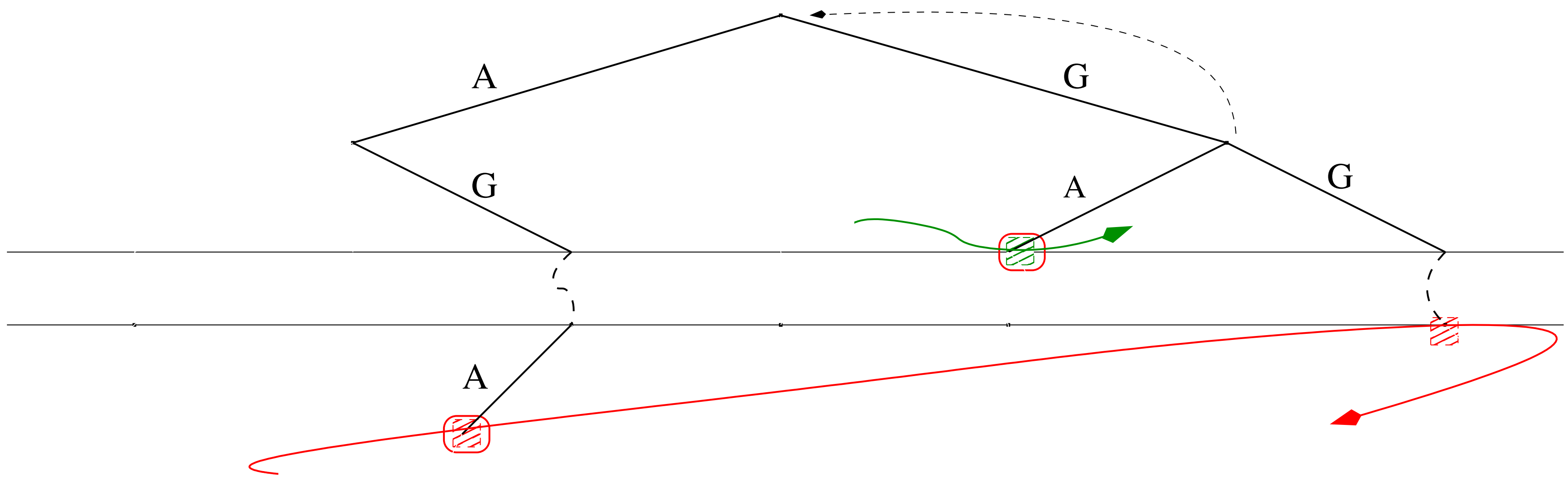


AGGAGAACAA

↑

Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

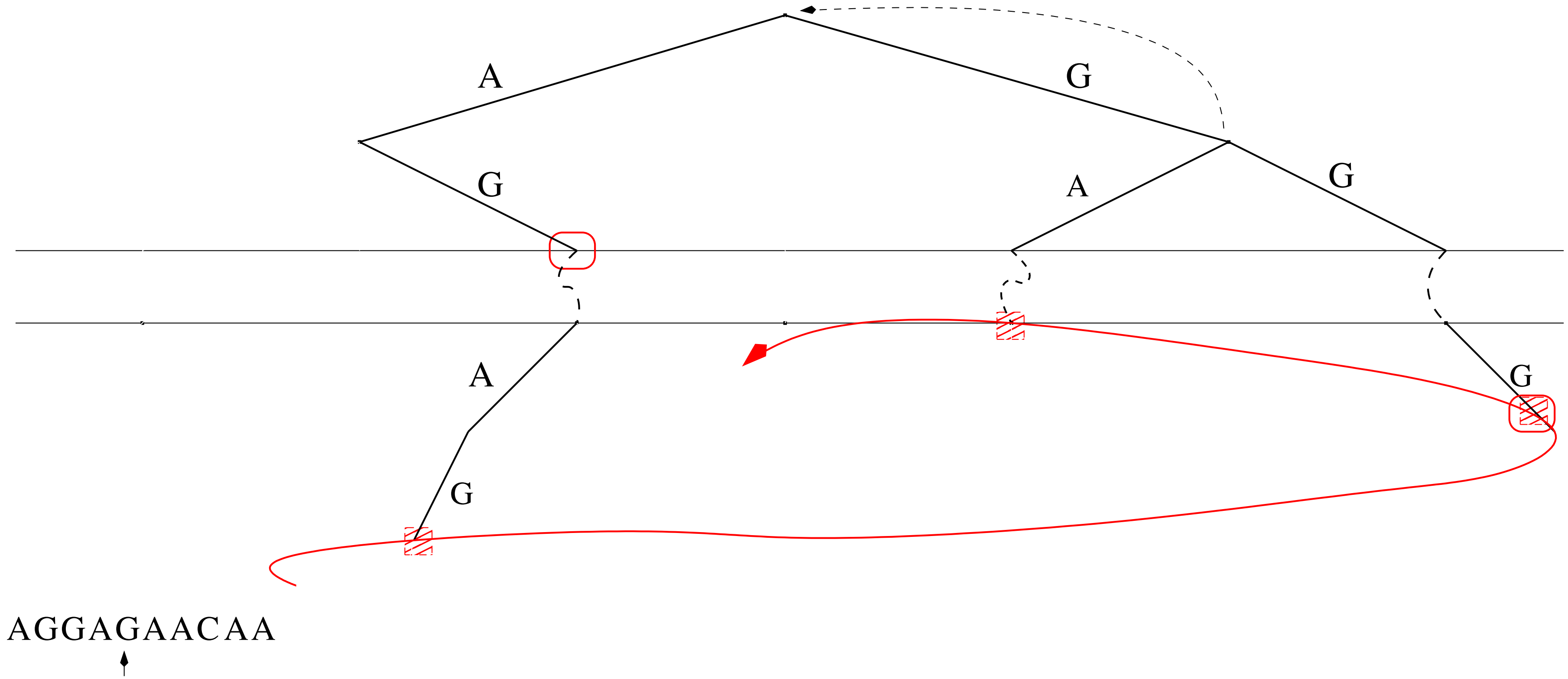
Lower part of the tree : same principles as for the upper part



AGGAGAACAA

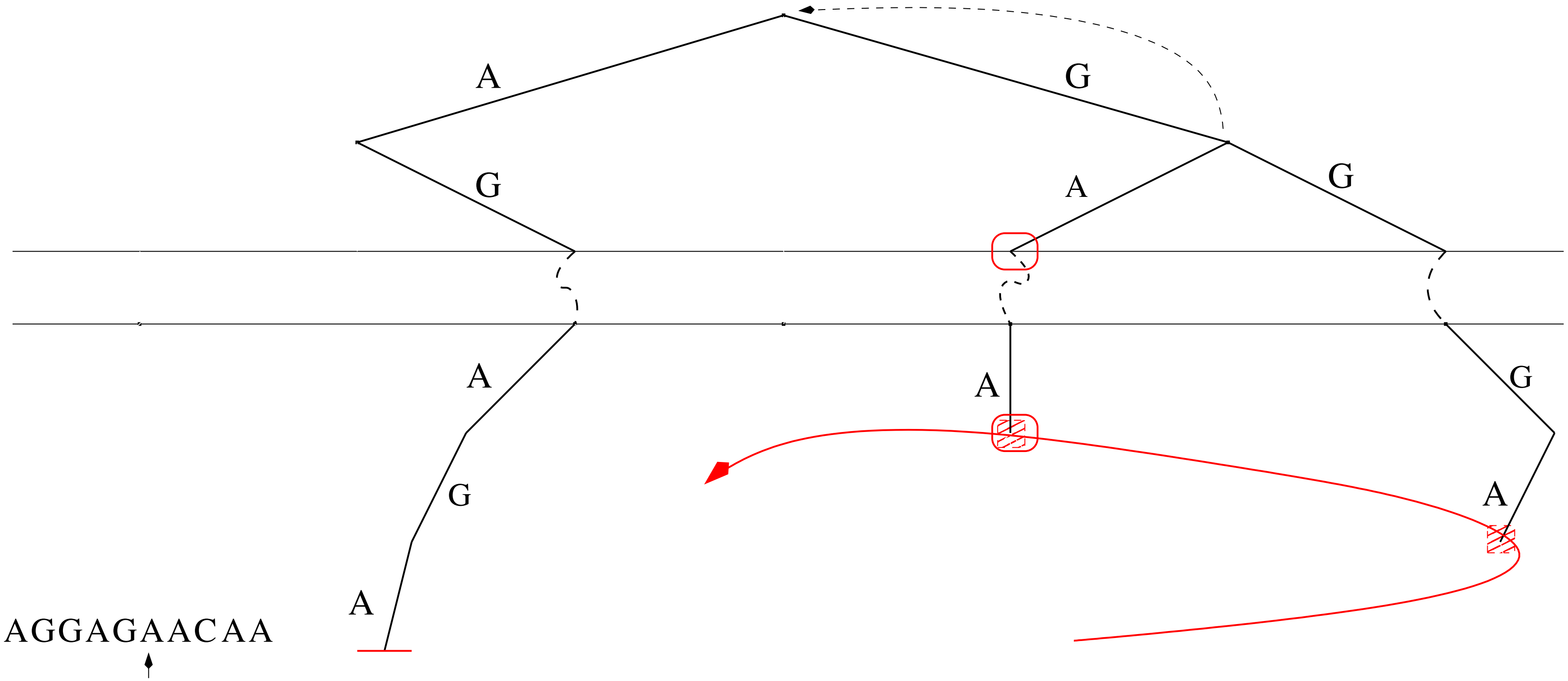
Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

Lower part of the tree : same principles as for the upper part



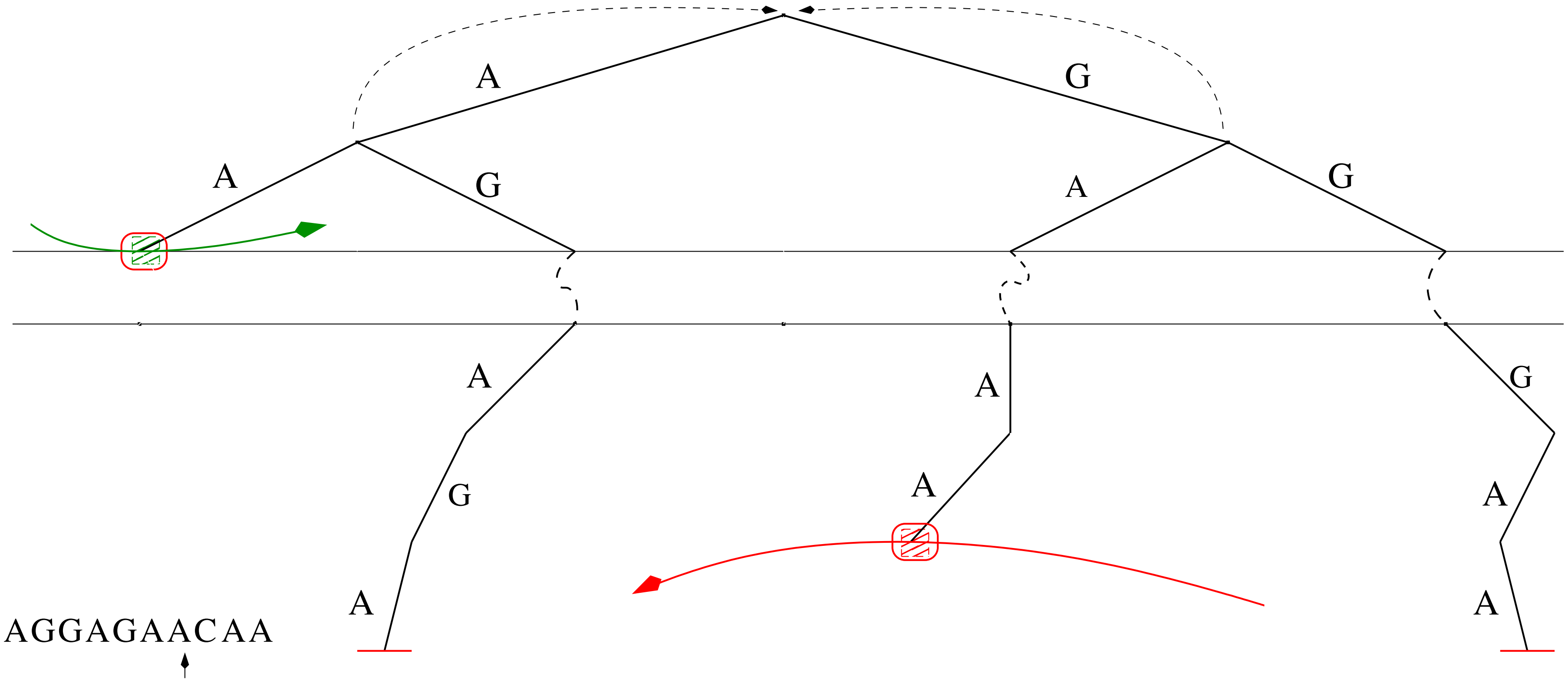
Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

Lower part of the tree : same principles as for the upper part



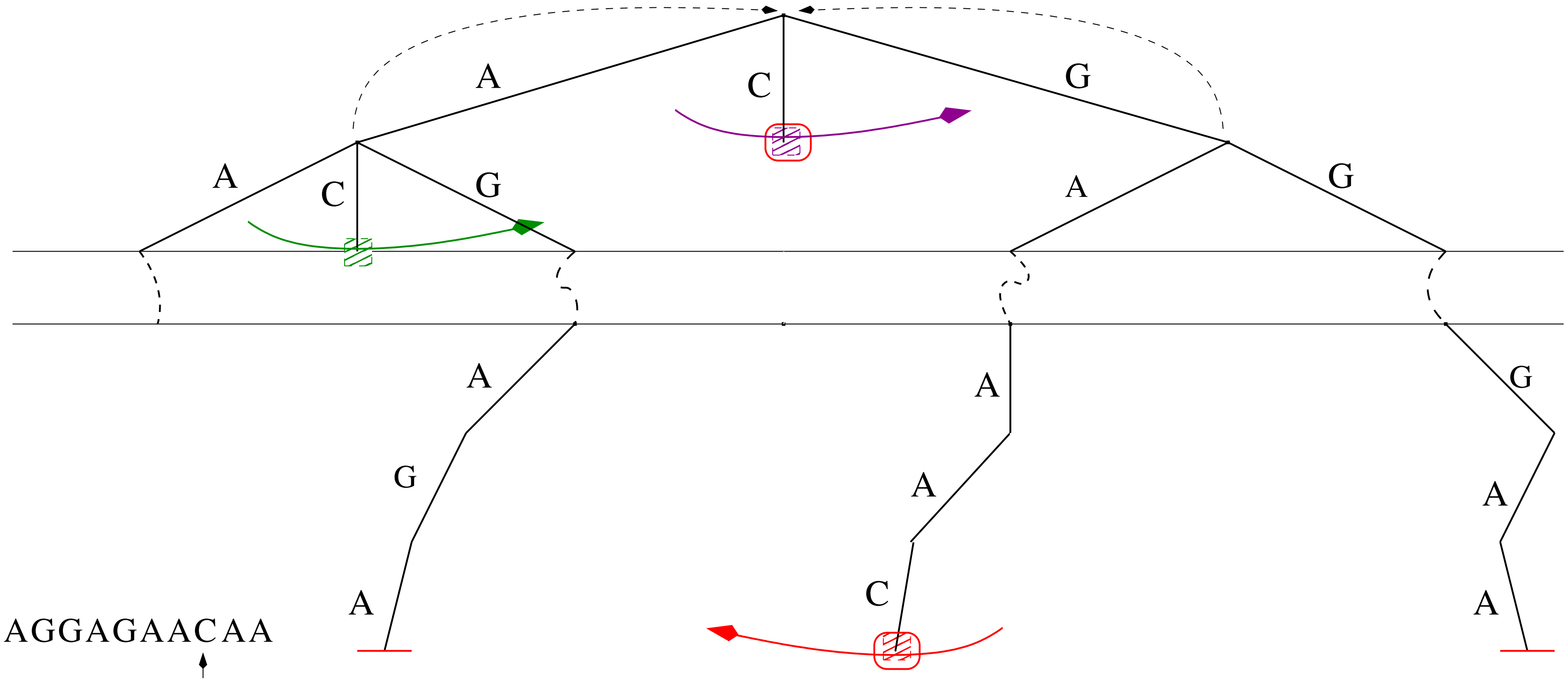
Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

Lower part of the tree : same principles as for the upper part

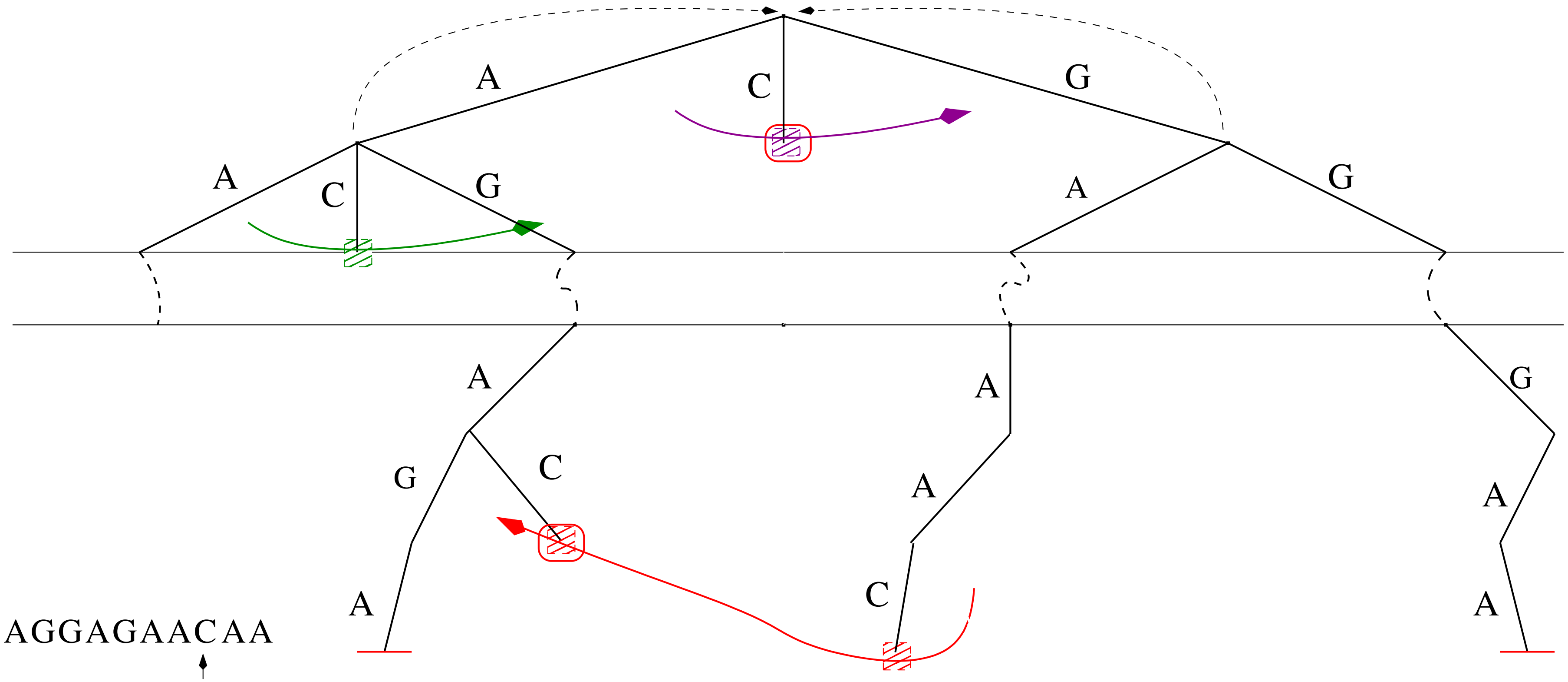


Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)

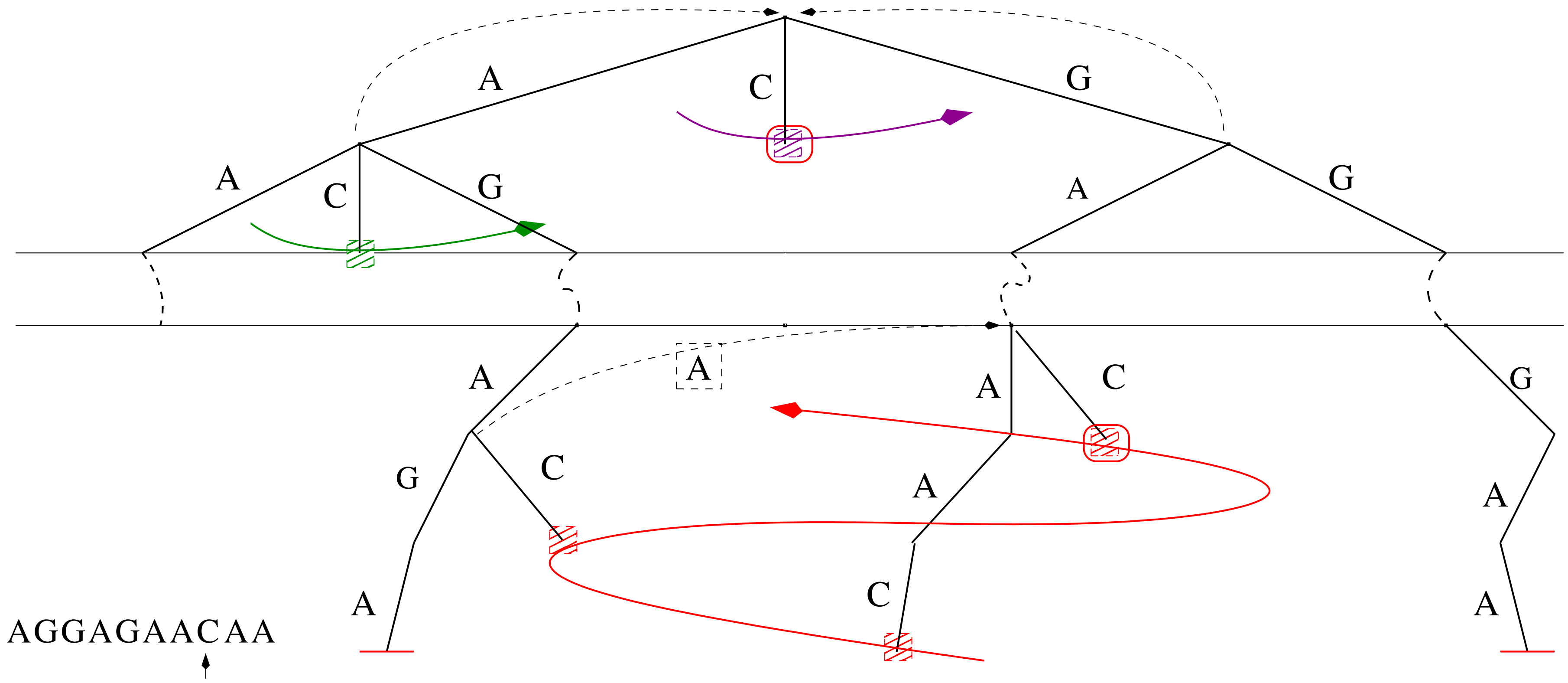
Multiple suffix link



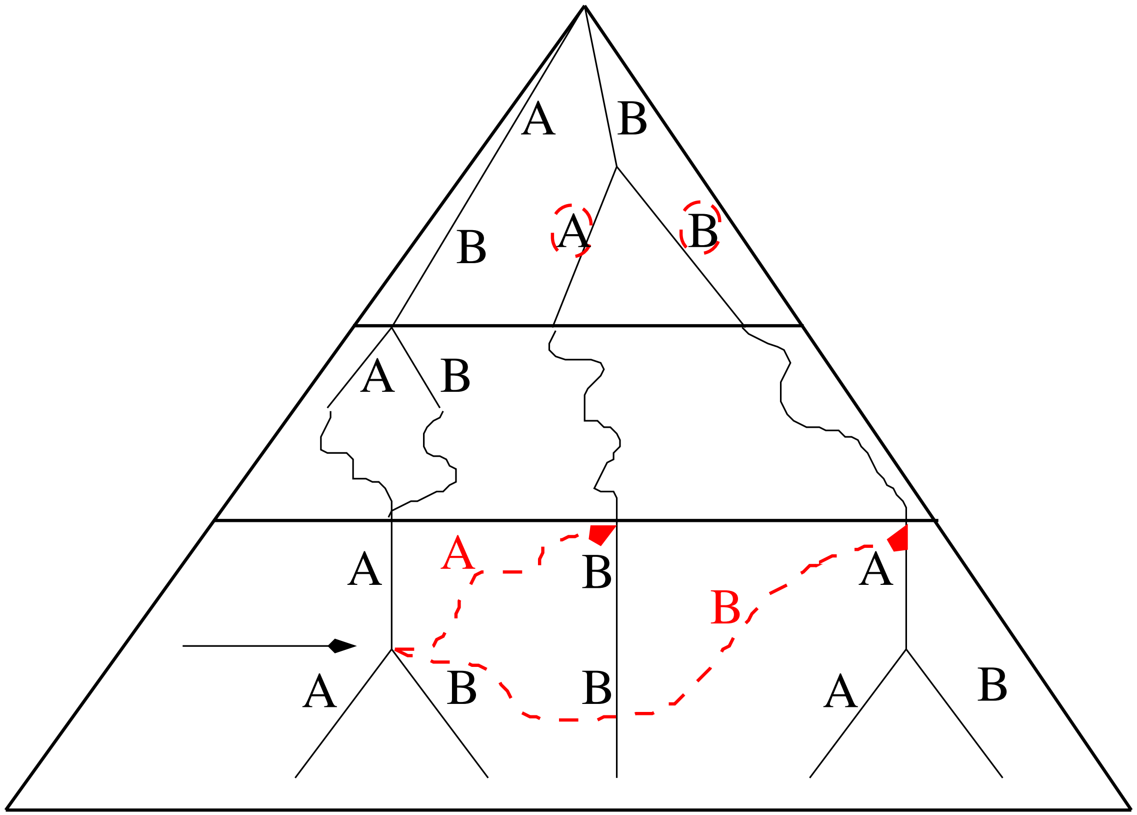
Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)



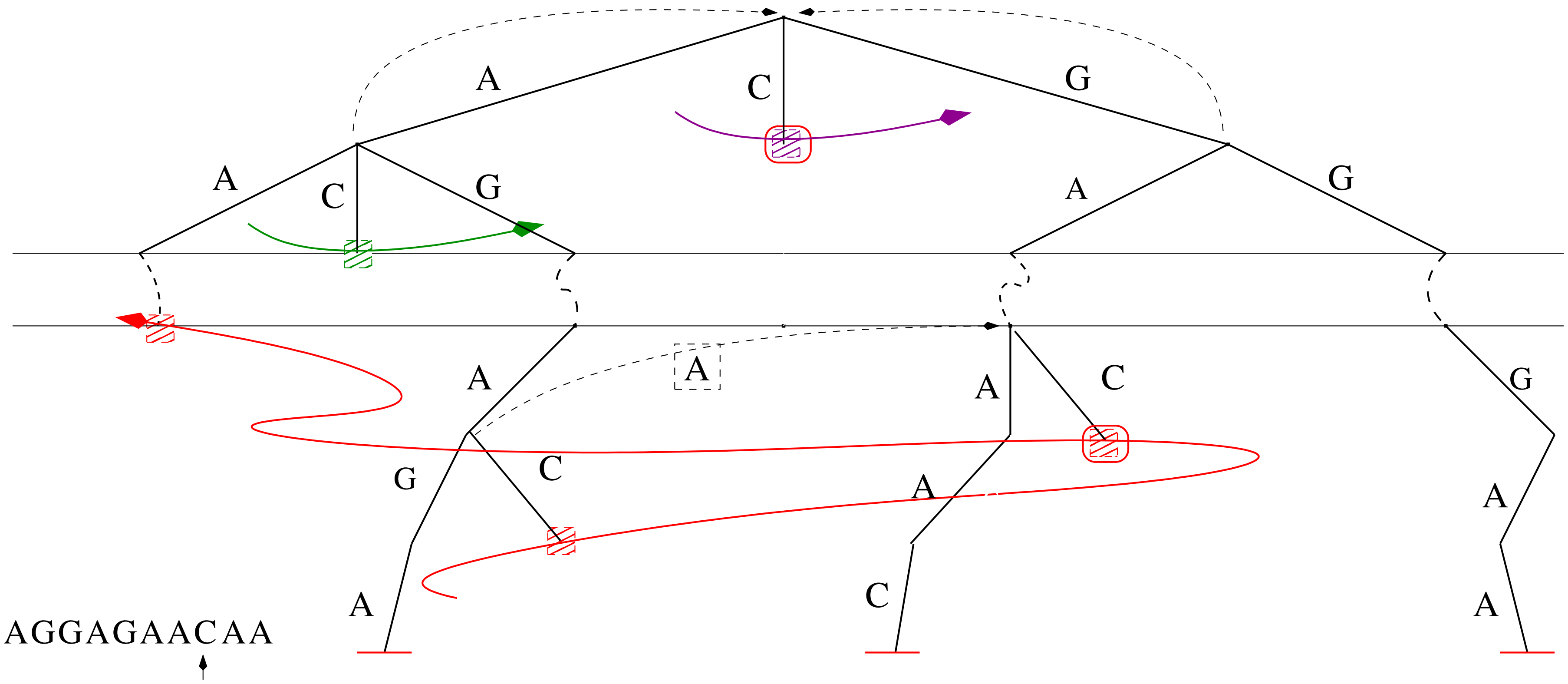
Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)



Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)



Gapped-factor tree construction ($k = 2$, $d = 1$, $k' = 3$)



Complexity analysis

Time and memory

- During the construction : $O(n \times |\Sigma|)$
- Using the index : $O(n)$

Goal - Motivations

Overview

Preliminaries

- Ukkonen suffix tree construction
- *k*-factor tree construction (Allali - Sagot)

Construction Algorithm

- Construction
- Complexity

Conclusion

Conclusion

- Interesting properties on suffix trees
- Use and development of Ukkonen’s method
- Indexing structure useful for various stringology problems

Conclusion

- Interesting properties on suffix trees
- Use and development of Ukkonen's method
- Indexing structure useful for various stringology problems

- Gapped suffix **array** $O(n)$