Introduce to Bioinformatics: Distributed Suffix Trees

Instructor: Dr. M. Sakalli
Student: Piotr Jan Dendek

Marmara University, 13.12.2010

Agenda

- A Suffix Tree – what is it?
- What is bootleneck of a suffix tree
- What are workarounds
- The Distributted Sufix Tree (DST)
- Conceptual construction demonstration
- Suffix Tree vs DST
- Drawbacks
- Advanteges
- Conclusions

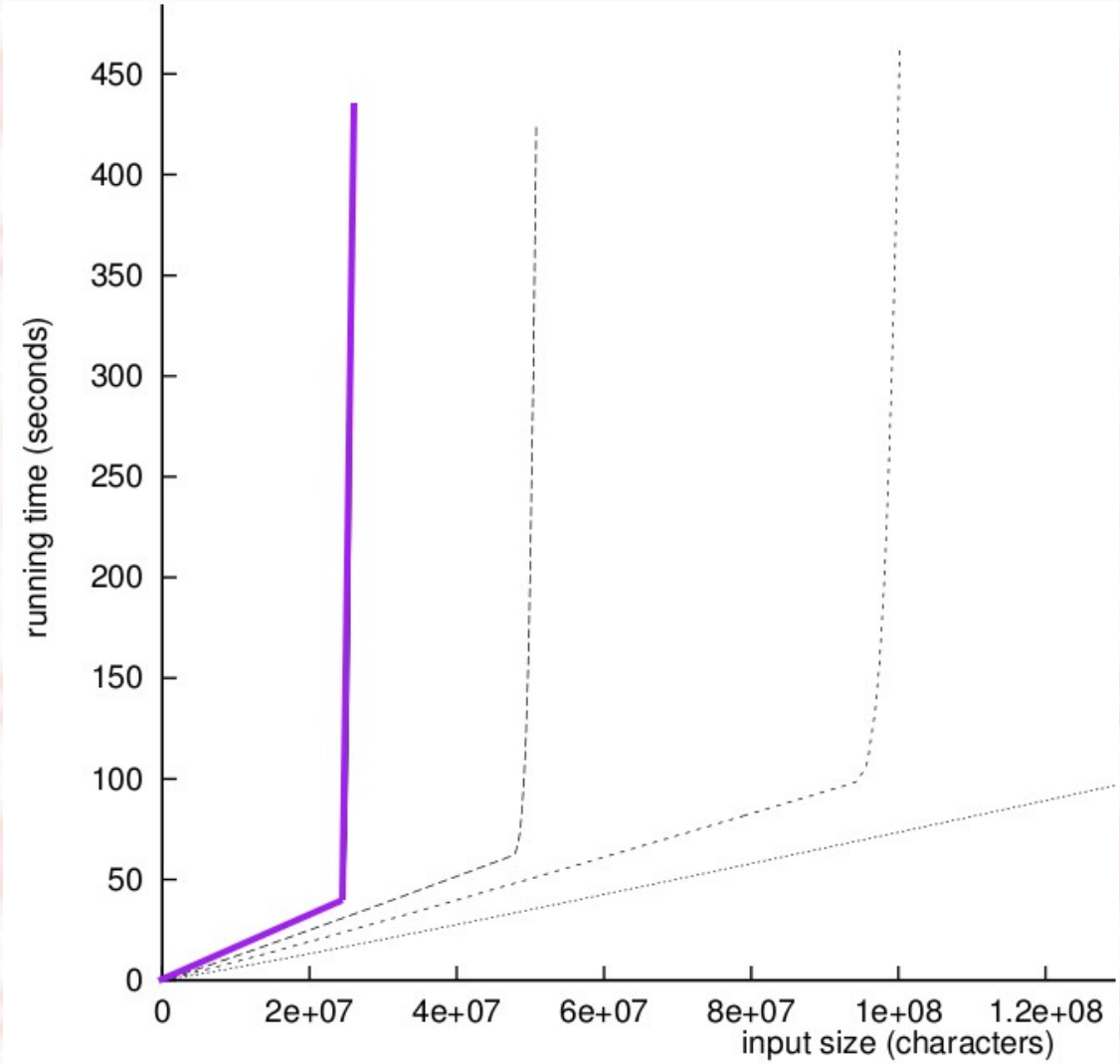
Suffix tree

- Evolution of Trie
- Construction time O(n) (where n is length of input string)
- Good in many application
  - Check if given word is in a text
  - Check # of occurences
  - Check possition of strings occurences

Bootleneck of a Suffix Tree

In fact construction time is O(n)
up to RAM exhaustage.

Bootleneck of a Suffix Tree



Workarounds

- Do we need workaround?
  - Yes, e.g. Human Genom Project produced great amount of data on which pople would like to navigate. Those data can not be stored in singular RAM unit
- Name some workarounds
  - On-disk Suffix Trees (rather slow)
  - Distributed Suffix Trees

Tips of a day

- Suffix Tree gives us ...
- paths from its root to leafs ...
- which provides in t (or less then t) hops ...
- leaf representing t-long word.
- Especially for
  - input text,
  - input text without first letter (suffix with index 1),
  - input text without first two letter (suffix with index 2),
  - ... and so on
- So suffixes of all indexes from 0 to t, where t is input text length, are covered

Distributed Suffix Tree

- Let's go back to Distributed Suffix Tree!
- Having a set <V>_{<z>} and input text <t>
  - Where <z> is substring of original input string
  - Where items in <V>_{<z>} are indexes of concatenations of <z> and text after <z> in text <t>
  - e.g. <t>=”aaabbaabab”, <z>=”aa”
  - <t>_<z>=”12abb6abab”
  - <V>_{aa}={1,2,6}

Distributed Suffix Tree

- e.g. <t>=”aaabbaabab”, <z>=”aa”
- <t>_<z>=”12abb6abab”
- <V>_{aa}={1,2,6}
- There are still unused suffixes: 3,4,5,7,8,9,10,11
- 11 is for empty string better known as „$”

- Let's follow the exhaustive example from the beginning
- Starting letters of used suffixes are uppercase

Distributed Suffix Tree

- There are still unused suffixes: 1,2,3,4,5,6,7,8,9,10,11
- <t>=”aaabbaabab”, <z>=”aa”, <V>_{aa}={1,2,6}
- There are still unused suffixes: 3,4,5,7,8,9,10,11
- <t>=”AAabbAabab”, <z>=”ab”, <V>_{ab}={3,7,9}
- There are still unused suffixes: 4,5,8,10,11
- <t>=”AAAbbAAbAb”, <z>=”ba”, <V>_{ba}={5,8}

Distributed Suffix Tree

- There are still unused suffixes: 4,5,8,10,11
- <t>=”AAAbbAAbAb”, <z>=”ba”, <V>_{ba}={5,8}
- There are still unused suffixes: 4,10,11
- <t>=”AAAbBAABAb”, <z>=”bb”, <V>_{bb}={4}
- There are still unused suffixes: 10,11
- <t>=”AAABBAABAB”, <z>=”b$”, <V>_{b$}={10}
- There are still unused suffixes: 11
- <t>=”AAABBAABAB”, <z>=”$”, <V>_{$}={11}

Distributed Suffix Tree

- All suffixes of input text are covered!
- We can divide in similar way Suffix Tree using each <z> as subroot
- Sadly – we must drop suffix tree links between nodes in separate subtrees
- Luckily – this is not equal with losing of O(n) construction time

Distributed Suffix Tree

- >>Sadly – we must drop suffix tree links between nodes in separate subtrees<<...
- ... but how we'll construct a tree then?!
- With „proper” suffix
- E.g. „acacc”
  - Normally there would be suffix tree link between „acac” and „cac” and from „acacc” to „cacc”
  - But because strings under „ac”-root starts with „ac”, the proper suffix of „acac” is „ac” (insted of using „cac” we are using next suffix)
  - Similarly, proper suffix of „acacc” is „acc”

Time for a demonstration

- Let's take input string:
  - aacacccacacaccacaaa$
- Let's point its lower mers (letters and 2-mers)
  - 1-mer(aacacccacacaccacaaa$)={a,c,**$**}
  - 2-mer(aacacccacacaccacaaa$)={**aa**,**ac**,**ca**,**cc**,**a$**}
- Now point out mers which are not covered by others
  - a_{1-mer} IN **aa_{2-mer}** or **ac_{2-mer}** or **a$_{2-mer}**
  - c_{1-mer} IN **cc_{2-mer}** or **ca_{2-mer}** [[[or c$_{2-mer}]]]
  - **$_{1-mer}** NOT IN 2-mer

Time for a demonstration

- Now point out mers which are not covered by others
  - **aa_{2-mer}** **ac_{2-mer}** **a$_{2-mer}**
  - **cc_{2-mer}** **ca_{2-mer}** *c$_{2-mer}*
  - **$_{1-mer}**
- *c$_{2-mer}* have not occured but should be mentioned to make division general for alphabet {a,c,$}

Time for a demonstration

- Now point out mers which are not covered by others
  - **aa**
  - **ac**
  - **a$**
  - **cc**
  - **ca**
  - **c$**
  - **$**

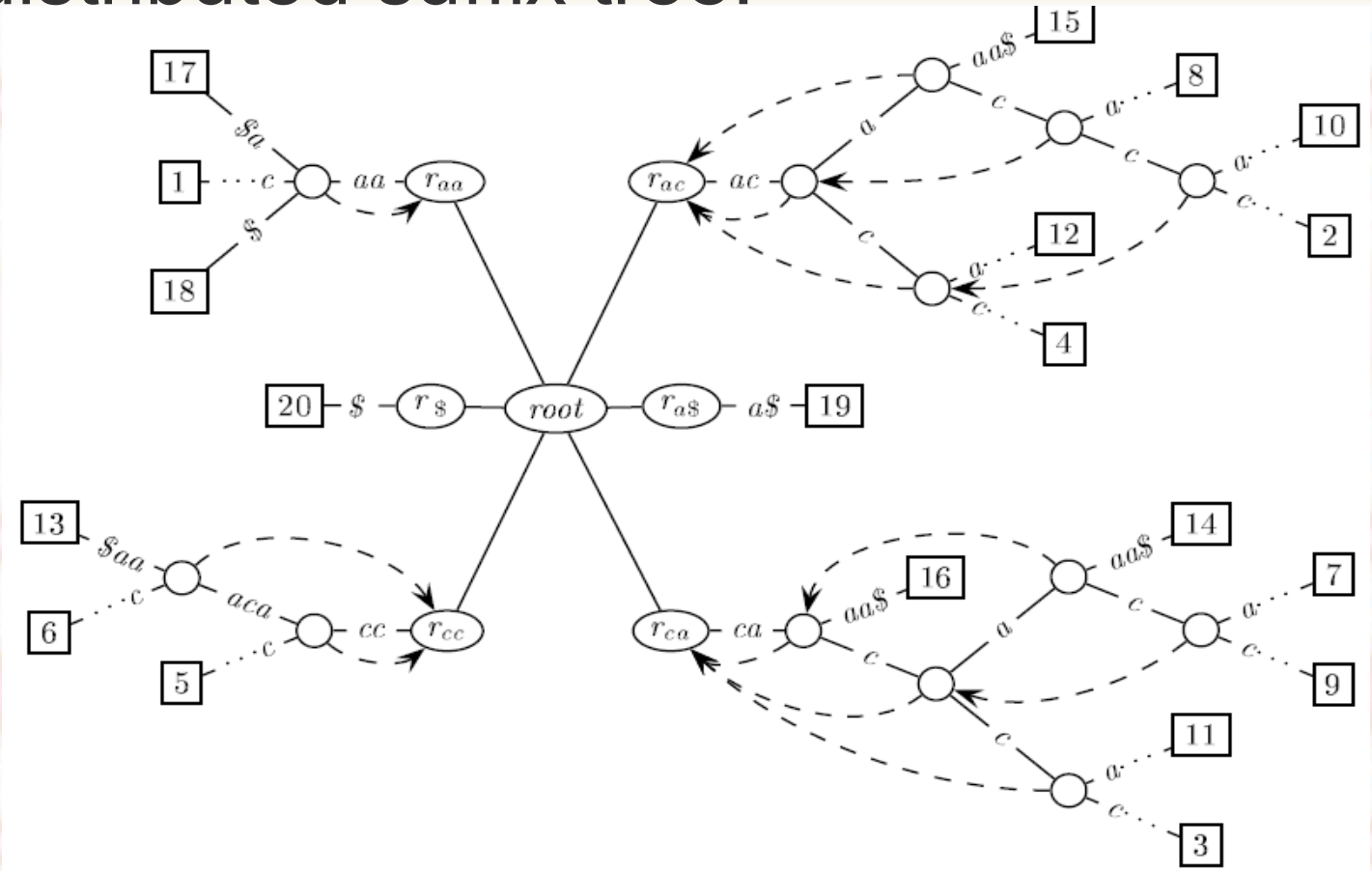
Time for a demonstration

- Now point out mers which are not covered by others (aa, ac, a$, cc, ca, c$, $)
- And make them sub-roots in Distributed Suffix Tree

- If max # of digits in subroot is <N> and # of alphabet letters without $ is <L> then # of subroots <S> is
- <S> = <N>^<L-1>*<N+1> | regular roots
  +<N>^<L-2>*<1>+... | $-ended roots
  +<N>^<1> *<1> | till <digit>$
  +1 | and pure $

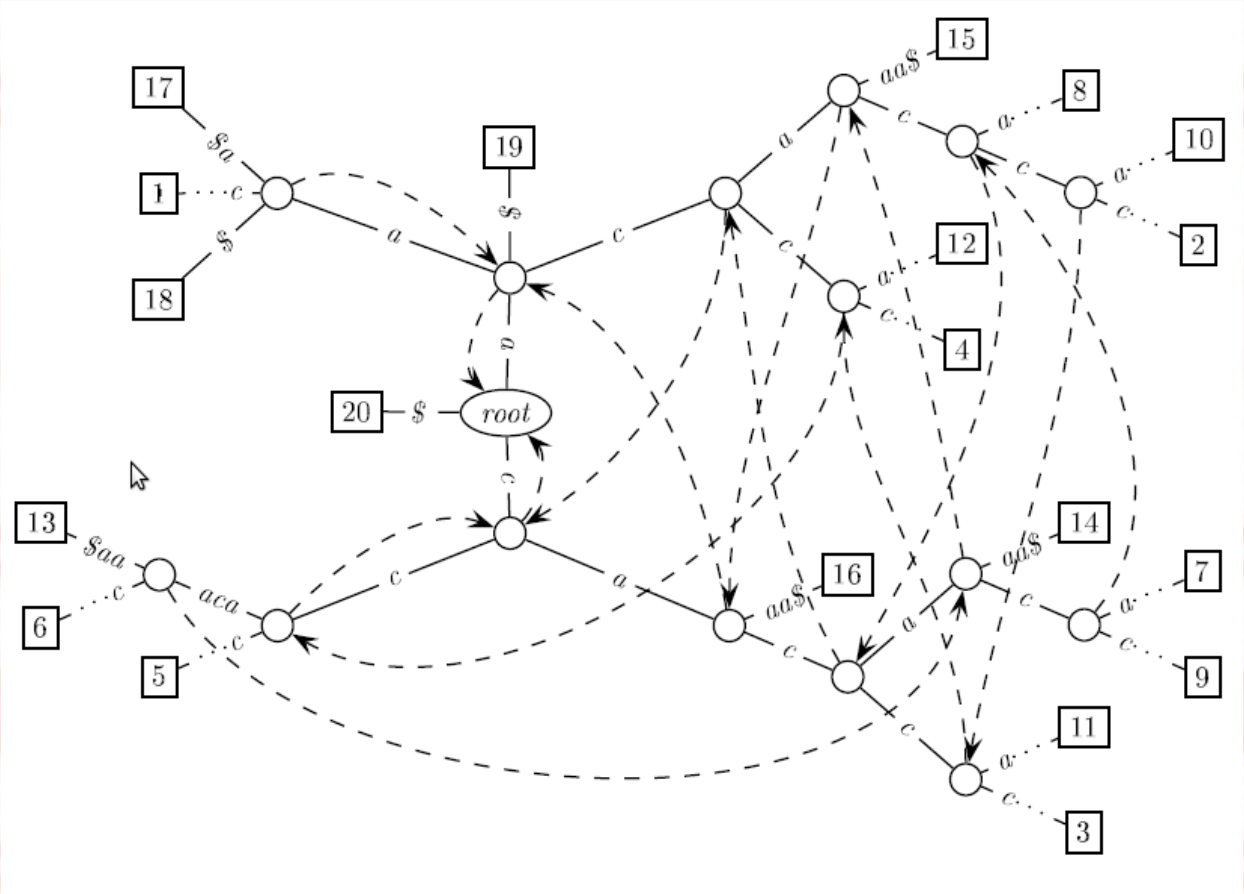
Time for a demonstration

- the distributed suffix tree:



Time for a demonstration

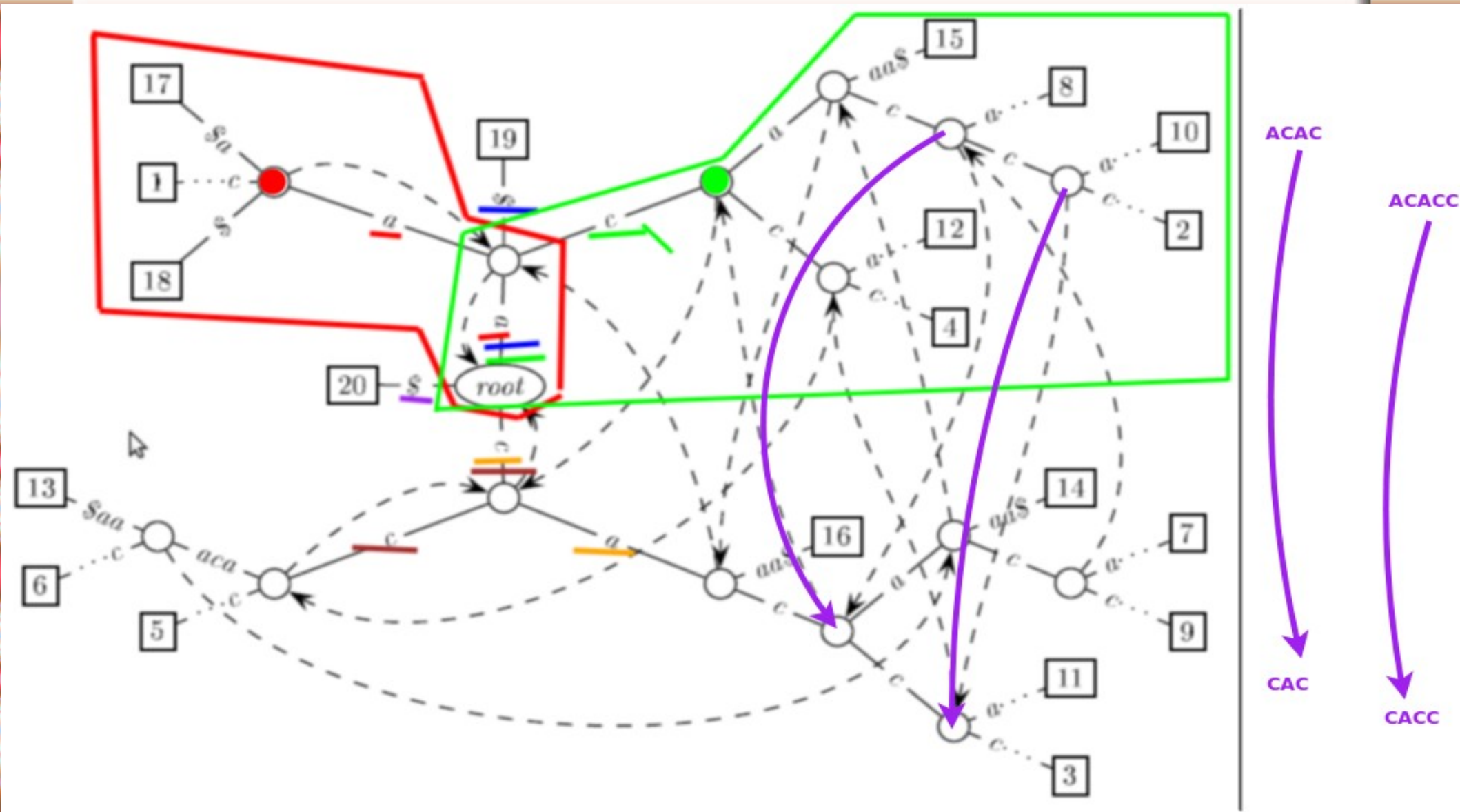
- the suffix tree:



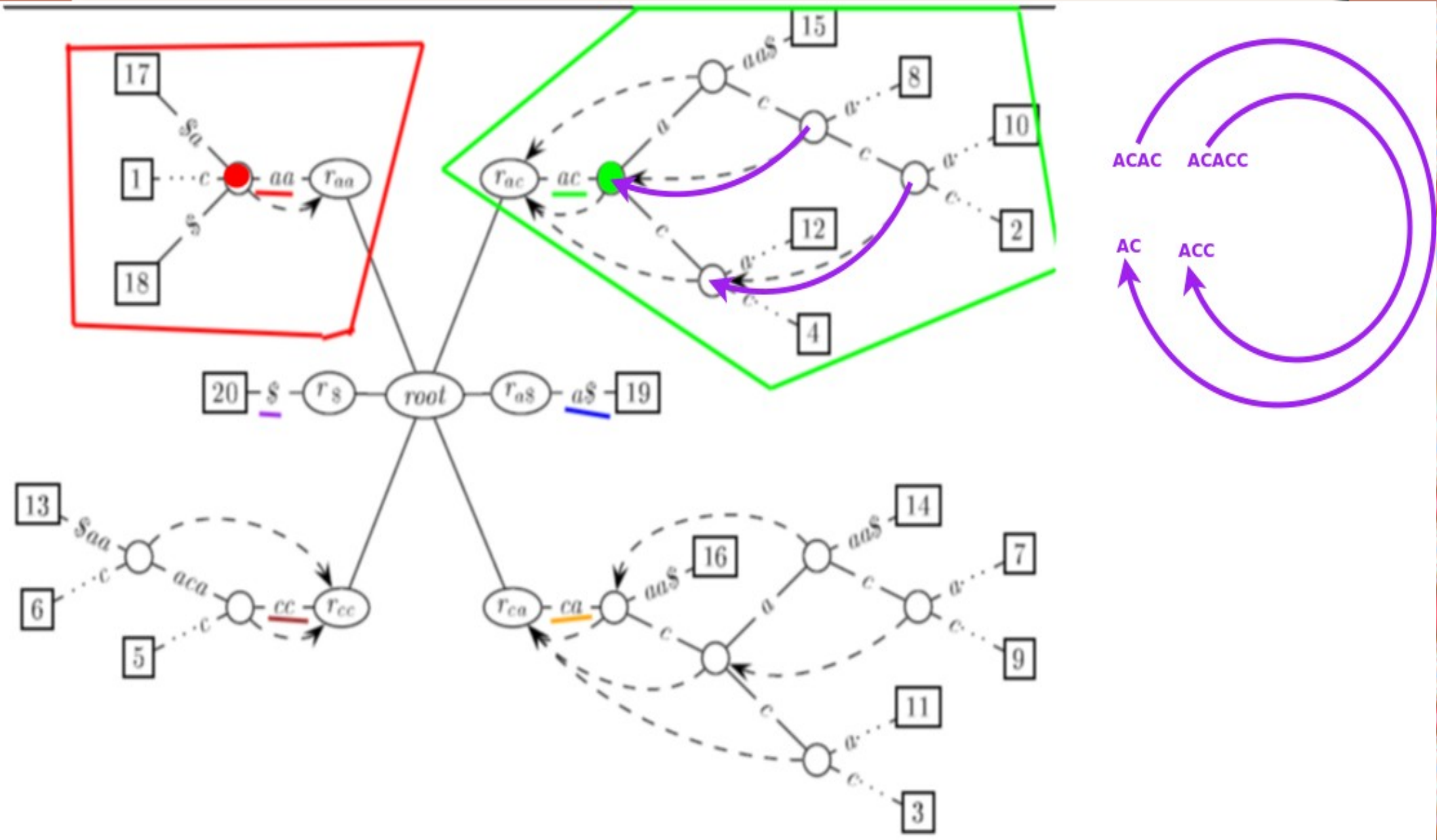
Normal vs Distributed

- Clones of nodes in Distributed Suffix Tree (DSF)
- Deletions of not proper arcs in DSF
- Look closer!

Look closer: Suffix Tree



Look closer: Distributed Suffix Tree



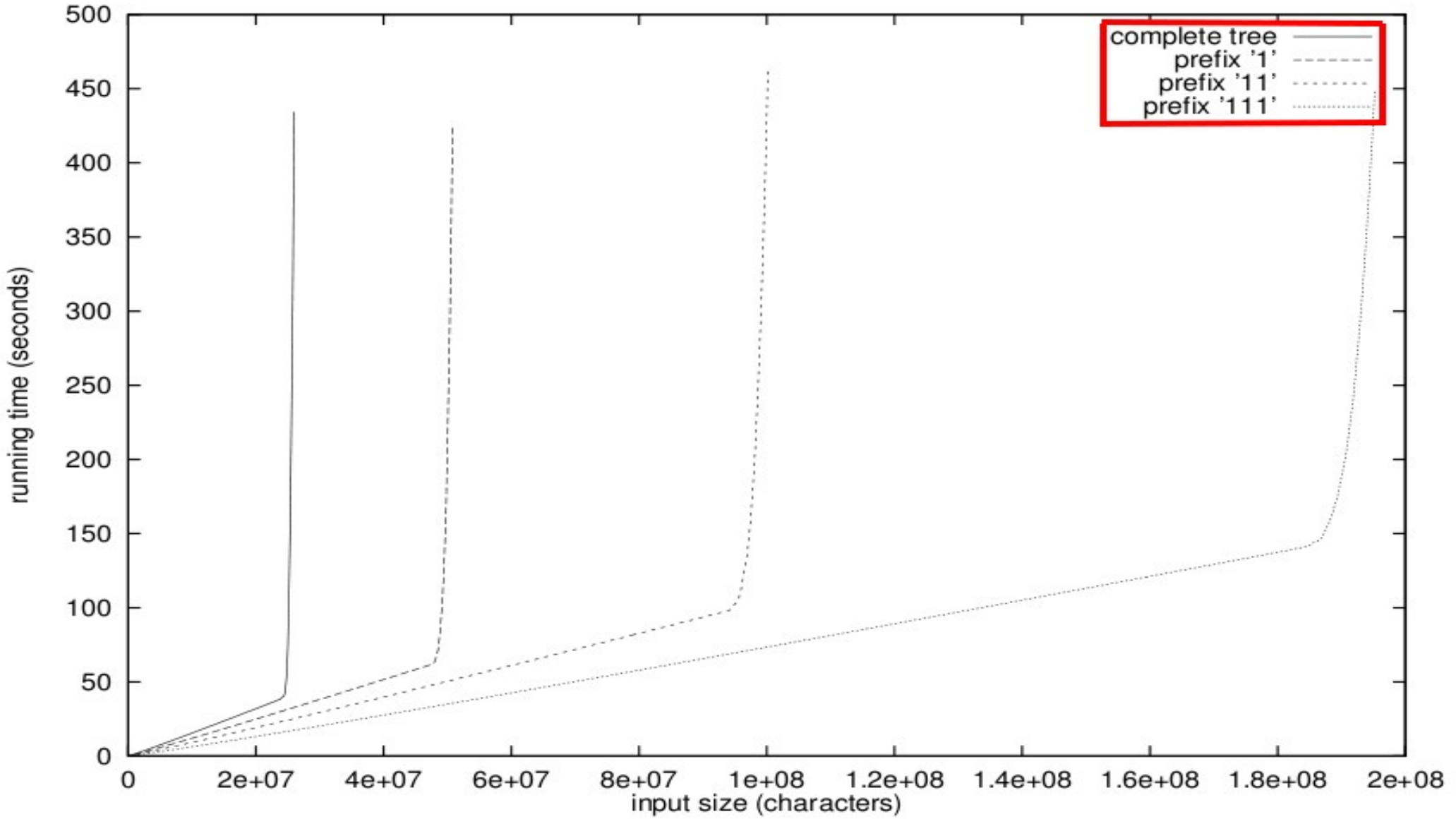
Drawbacks

- Dependency on hardware connection
- Wired structure
- Some statistical operations perform time is much worse

Advanteges

- Realtime sublinear construction
- Chance to introduce grid calculations
- And again the plot from beginning of presentation...

Advanteges


complete tree
prefix '1'
prefix '11'
prefix '111'
running time (seconds)
input size (characters)
0
50
100
150
200
250
300
350
400
450
500
0
2e+07
4e+07
6e+07
8e+07
1e+08
1.2e+08
1.4e+08
1.6e+08
1.8e+08
2e+08

Advanteges

- ... which means that Distributed Suffix Tree allow us to ...
- ... use more RAM (sum of RAM across all involved computers) ...
- ... which preserve O(n) for longer time
- With 3 digit root it is extended to 780% of base input text length
- You must say „WOW!”

Conclusions

- Till now topic of Distributed Suffix Trees has not been enough good covered
  (2 articles by the same authors from Imperial College, UK)
- An idea of providing multicore implementation of DST may be interesting
- Using arcs from DST in Suffix Trees concatenated with de Brujin graph sounds interesting

References

- „Distributed Suffix Trees”, Raphaël Clifford, King’s College, London, UK
- „Distributed and Paged Suffix Trees for Large Genetic Databases”, Raphaël Clifford and Marek Sergot, Imperial College London, UK
- „Distributed Suffix Tree Overlay for Peer-to-Peer Search”, Hai Zhuge and Liang Feng (IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 2, February 2008)
- „Distributed Suffix Tree for Peer-to-Peer Search”, Hai Zhuge and Liang Feng, Chinese Academy of Sciences, Beijing, China

Thank you for your attention!