Parallel and Distributed Compressed Indexes

Luís M. S. Russo Gonzalo Navarro Arlindo L. Oliveira

CITI, Departamento de Informática, Faculdade de Ciências e Tecnologia, FCT,
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal.
lsr@di.fct.unl.pt

Dept. of Computer Science, University of Chile
gnavarro@dcc.uchile.cl

INESC-ID/IST
aml@kdbio.inesc-id.pt

21st Annual Symposium on Combinatorial Pattern Matching

Outline

Applications of Suffix Trees

22 min

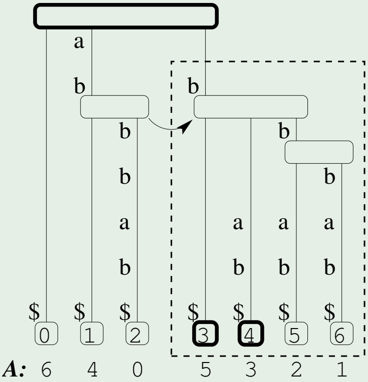
Suffix trees are important for several string problems:

- pattern matching
- longest common substring
- super maximal repeats
- bioinformatics applications
- etc

Basic concepts

21 min

Example (Suffix Tree for *abbbab*)



Compressed Indexes

19 min

Problem (Indexes need too much space)

Pointer based representations require $O(n \log n)$ bits.

Hence we use compressed indexes that use only $n \log \sigma + o(n \log \sigma)$ bits.

- `Succinct structures`, based on RANK and SELECT.
- `Data compression`, that represent T in $O(nH_k)$ bits.

Examples

FM-index, Compressed Suffix Arrays, LZ-index, etc.

We use a compressed index that supports ψ and LF.
For example the Alphabet-Friendly FM-Index.

Parallel and Distributed Indexes

18 min

Problem (Indexes are sequential)

How to adapt compress indexes to shared-memory parallel machines ? How to distribute compressed indexes across several machines?

History

17 min

- Various data layouts have been considered for distributing classical suffix trees and arrays [1, 2, 3], with optimal speedups.
- Mäkinen et al. [2] achieved optimal speedups for a batch of queries with CSAs.
- We proposed near-optimal speed ups for single queries over CSAs and FCSTs.

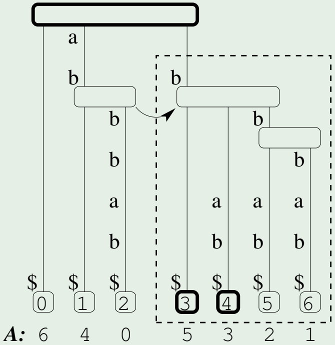
Node Representation

16 min

Intermediate search status is either a node/point in the suffix tree or an interval of leafs.

Example

Interval [3, 6] represents node *b*.



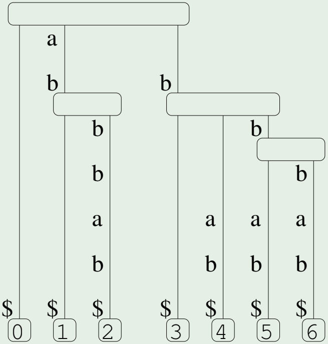
generalized branching

14 min

Observation

We use generalized branching.

Example



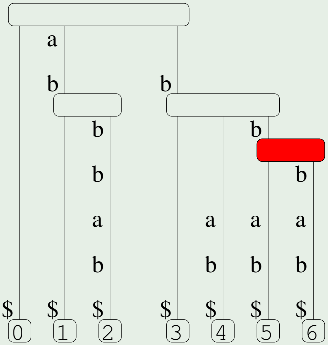
$O((\log\log n)^2 \log_\sigma n)$
time with the FCST.

generalized branching

Observation

We use generalized branching*.*

Example



$O((\log\log n)^2 \log_\sigma n)$
time with the FCST.

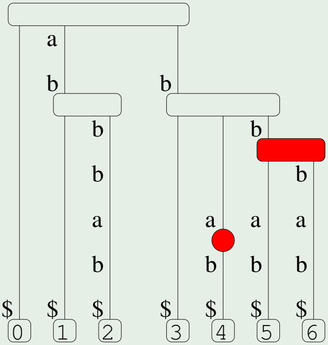
generalized branching

14 min

Observation

We use generalized branching.

Example



$O((\log\log n)^2 \log_\sigma n)$ time with the FCST.

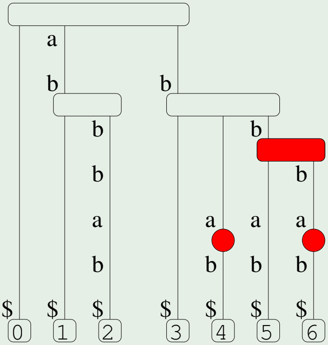
generalized branching

14 min

Observation

We use generalized branching.

Example



$O((\log\log n)^2 \log_\sigma n)$
time with the FCST.

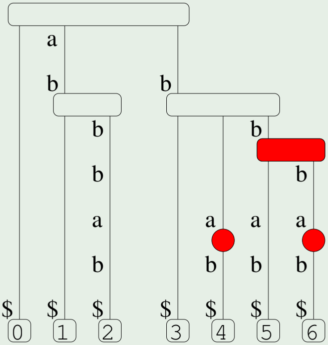
generalized branching

14 min

Observation

We use generalized branching.

Example



$O((\log\log n)^2 \log_\sigma n)$ time with the FCST.

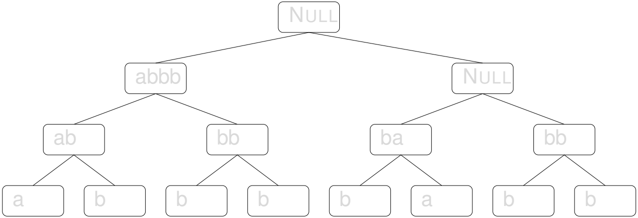
Pattern Matching

12 min

Problem

How to search for a pattern $P = abbbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n(\log p + \log\log n \log\log p))$.



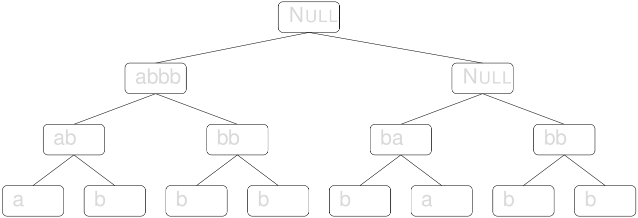
Pattern Matching

12 min

Problem

How to search for a pattern $P = abbbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n (\log p + \log\log n \log\log p))$.



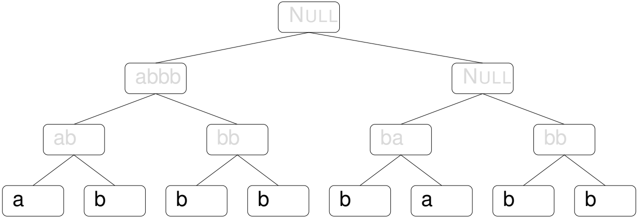
Pattern Matching

12 min

Problem

How to search for a pattern $P = abbbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n (\log p + \log\log n \log\log p))$.



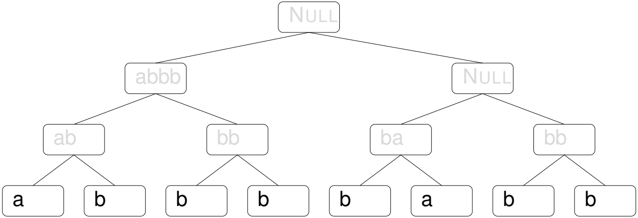
Pattern Matching

12 min

Problem

How to search for a pattern P = abbbbabb in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n(\log p + \log\log n \log\log p))$.



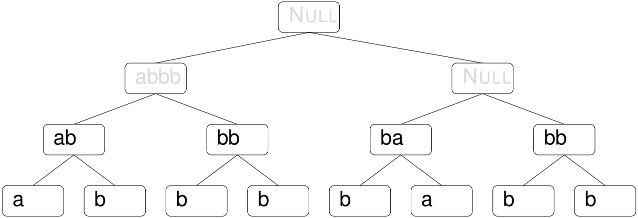
Pattern Matching

12 min

Problem

How to search for a pattern $P =$ abbbbabb in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n(\log p + \log\log n \log\log p))$.



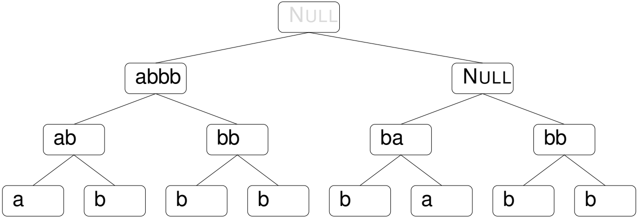
Pattern Matching

12 min

Problem

How to search for a pattern $P = abbbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n(\log p + \log\log n \log\log p))$.



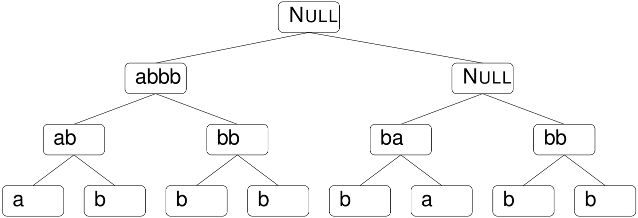
Pattern Matching

12 min

Problem

How to search for a pattern $P = abbbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n(\log p + \log\log n \log\log p))$.



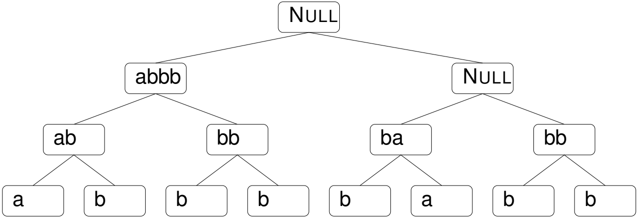
Pattern Matching

12 min

Problem

How to search for a pattern $P = abbbbabb$ in the index, using $p = 4$ processors in parallel?

- Split the pattern and search in parallel.
- Merge the resulting points.
- $O(m/p + \log n \log\log n(\log p + \log\log n \log\log p))$.



Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

4, 3, 5, 4, 3, 3, 2, 1

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

4, 3, 5, 4, 3, 3, 2, 1

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$
$P = abbbbabb$
$4, 3, 5, 4, 3, 3, 2, 1$

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

$4, 3, 5, 4, 3, 3, 2, 1$

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

$4, 3, 5, 4, 3, 3, 2, 1$

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abb\textcolor{red}{bab}$

$P = abbb\textcolor{red}{bab}b$

$4, 3, 5, 4, 3, 3, 2, 1$

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

$4, 3, 5, 4, 3, 3, 2, 1$

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

$4, 3, 5, 4, 3, 3, 2, 1$

Matching statistics

10 min

Problem

How to compute the matching statistics of a pattern P, in parallel?

Example

$T = abbbab$

$P = abbbbabb$

$4, 3, 5, 4, 3, 3, 2, 1$

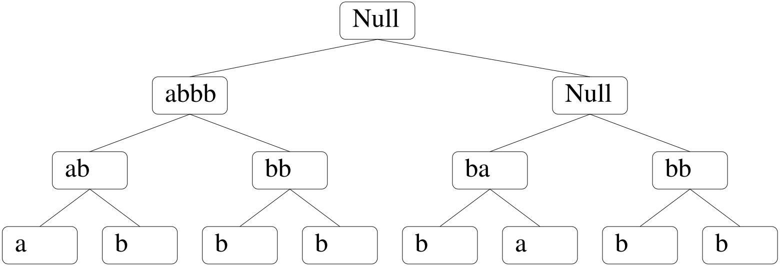
Matching statistics

9 min

Problem

How to compute $ms(2)$ for $P = ab\textcolor{red}{bbbabb}$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



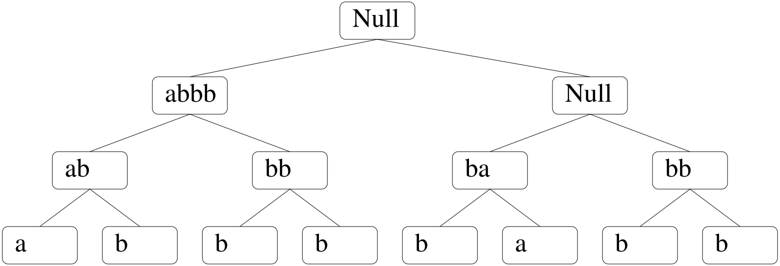
Matching statistics

9 min

Problem

How to compute $ms(2)$ for $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



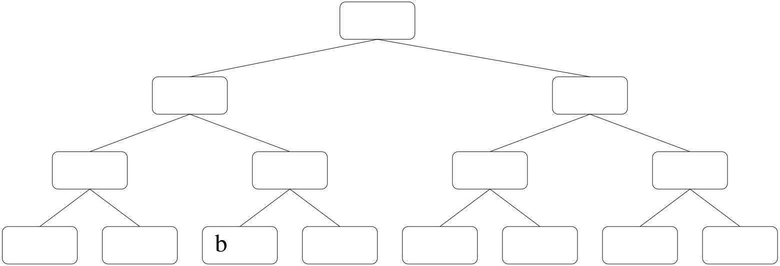
Matching statistics

9 min

Problem

How to compute $ms(2)$ *for* $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p)\log m \log n(\log\log n)^2)$.



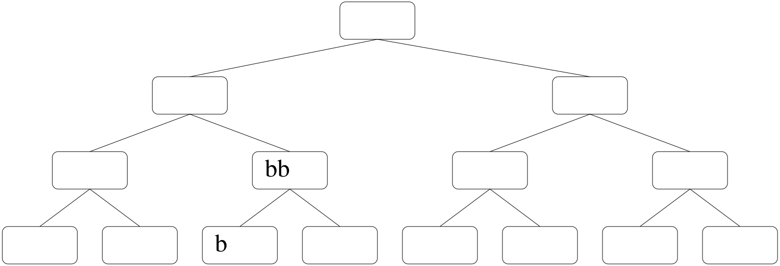
Matching statistics

9 min

Problem

How to compute $ms(2)$ *for* $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



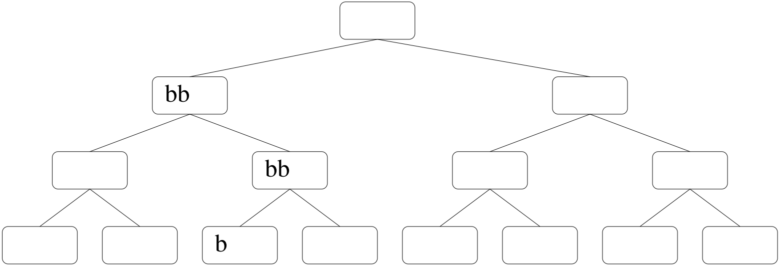
Matching statistics

9 min

Problem

How to compute $ms(2)$ *for* $P = ab$bbbabb *?*

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p)\log m \log n (\log\log n)^2)$.



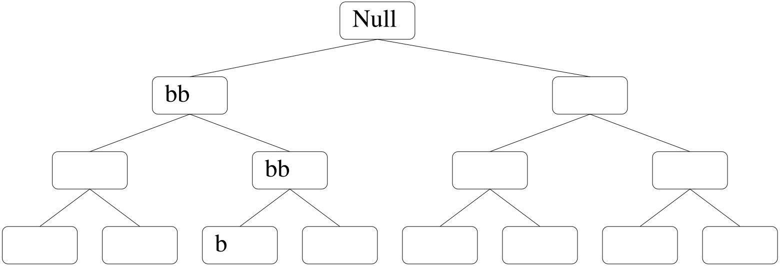
Matching statistics

9 min

Problem

How to compute ms(2) for P = abbbbabb ?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



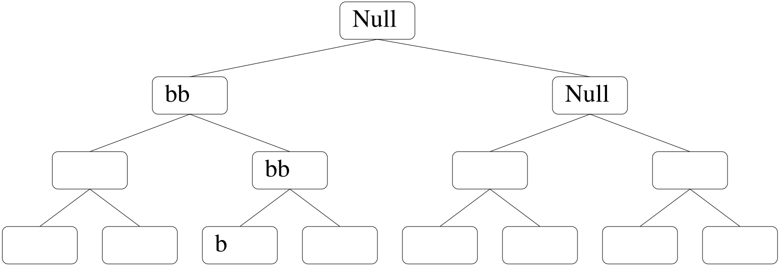
Matching statistics

9 min

Problem

How to compute $ms(2)$ for $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



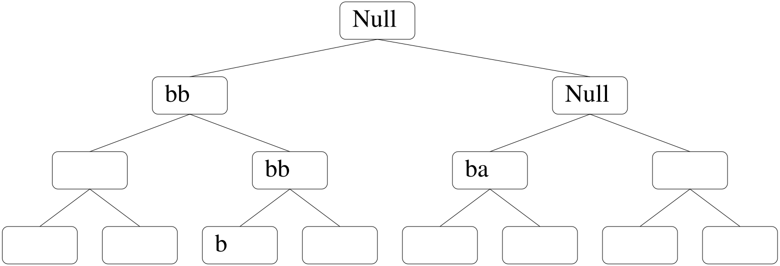
Matching statistics

9 min

Problem

How to compute $ms(2)$ *for* $P = abbbbabb$ *?*

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p)\log m \log n (\log\log n)^2)$.



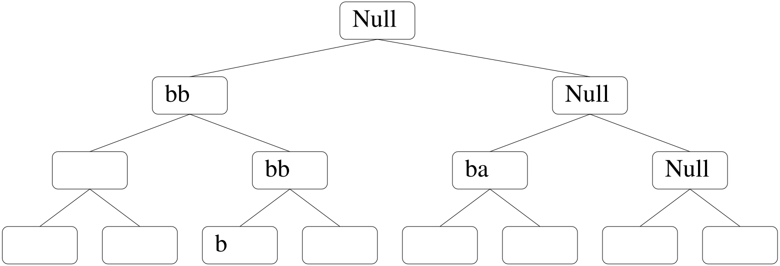
Matching statistics

9 min

Problem

How to compute ms(2) for $P = abbbbabb$?

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



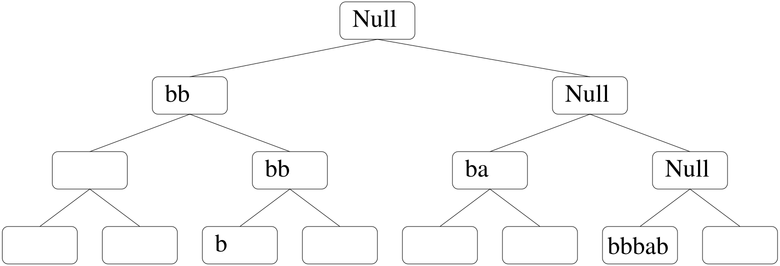
Matching statistics

9 min

Problem

How to compute $ms(2)$ *for* $P = ab\color{red}{bbbabb}$ *?*

- Build a generalized branch tree.
- Move on the tree merging points.
- $O((m/p) \log m \log n (\log \log n)^2)$.



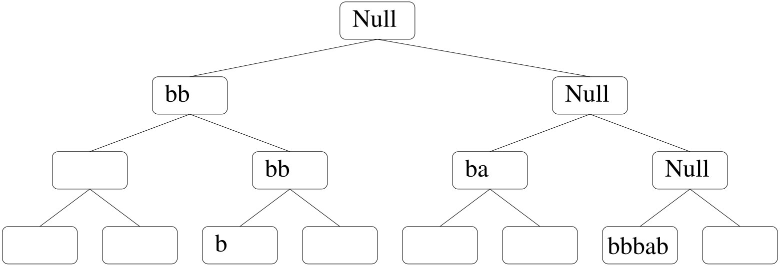
Matching statistics

9 min

Problem

How to compute $ms(2)$ *for* $P = ab\textcolor{red}{bbbabb}$ *?*

- Build a generalized branch tree.
- Move on the tree merging points.
- $O(\textcolor{red}{(m/p)} \log m \log n (\log\log n)^2)$.



Matching statistics

7 min

Problem

How to compute the longest common substring between P and T, in parallel ?

- It is a side effect of matching statistics.

Matching statistics

7 min

Problem

How to compute the longest common substring between P and T, in parallel ?

- It is a side effect of matching statistics.

Maximal Repeats

6 min

Problem

How to determine the maximal repeated subtrings of T, in parallel ?

Example

$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\text{COUNT}(\text{LF}(\text{LETTER}(v_l, -1), v)) \neq \text{COUNT}(v)$.
- Hence each node can be verified in parallel.

Maximal Repeats

6 min

Problem

How to determine the maximal repeated subtrings of T, in parallel ?

Example

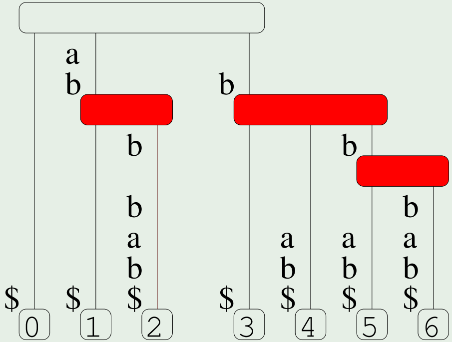
$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\text{COUNT}(\text{LF}(\text{LETTER}(v_l, -1), v)) \neq \text{COUNT}(v)$.
- Hence each node can be verified in parallel.

Maximal Repeats

4 min

Example



Maximal Repeats

6 min

Problem

How to determine the maximal repeated subtrings of T, in parallel ?

Example

$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\textsc{Count}(\mathrm{LF}(\textsc{Letter}(v_l, -1), v)) \neq \textsc{Count}(v)$.
- Hence each node can be verified in parallel.

Maximal Repeats

6 min

Problem

How to determine the maximal repeated subtrings of T, in parallel ?

Example

$T = abbbab$.

- Classical solution is the left-diverse internal nodes.
- Notice that left-diverse is equivalent to $\text{COUNT}(\text{LF}(\text{LETTER}(v_l, -1), v)) \neq \text{COUNT}(v)$.
- Hence each node can be verified in parallel.

Distributed Compressed Suffix Arrays

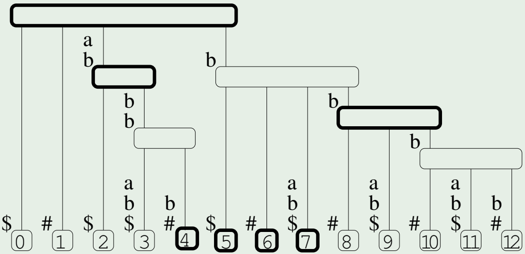
3 min

Problem

Simulate the access to a big CSA ?

- Store a bitmap $Id = 0100101010101$.
- $\mathrm{LF}_{\mathcal{C}}(X, [v_l, v_r]) = [\min_{j=0}^{q-1}\{\textsc{Select}_j(xv_{j,l}+1)\}, \max_{j=0}^{q-1}\{\textsc{Select}_j(xv_{j,r}+1)\}]$

Example



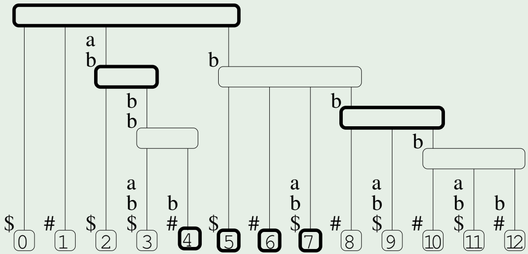
Distributed Compressed Suffix Arrays 3 min

Problem

Simulate the access to a big CSA ?

- Store a bitmap $Id = 0100101010101$.
- $\mathrm{LF}_{\mathcal{C}}(X, [v_l, v_r]) = [\min_{j=0}^{q-1}\{\mathrm{SELECT}_j(xv_{j,l}+1)\}, \max_{j=0}^{q-1}\{\mathrm{SELECT}_j(xv_{j,r}+1)\}]$

Example



Distributed Compressed Suffix Arrays

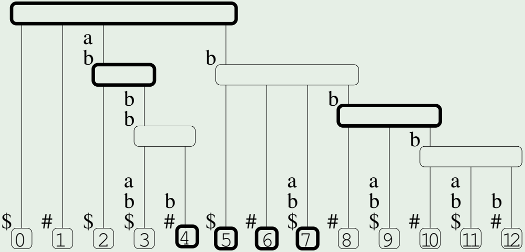
3 min

Problem

Simulate the access to a big CSA ?

- Store a bitmap $Id = 0100101010101$.
- $\mathrm{LF}_{\mathcal{C}}(X, [v_l, v_r]) =$ $[\min_{j=0}^{q-1}\{\textsc{Select}_j(xv_{j,l}+1)\}, \max_{j=0}^{q-1}\{\textsc{Select}_j(xv_{j,r}+1)\}]$

Example



Distributed Fully-Compressed Suffix Trees

1 min

Problem

How to simulate the acess to a big FCST by using several distributed FCSTs ?

- Store sampled node identifying bitmaps.
- Merge diferent LSA as in CSAs.

((0) (1) ((2) ((3) (4))) ((5) (6) (7) ((8) (9) (10) (11) (12)))))

	(	0	1	(	2	(	3	(4)))	(5) (6) (7) (	8	9	10	11	12	)	)
B :	1	0	0	1	0	1	0	10111	1011011011	0	0	0	0	0	1	1
B0:	1	0		1	0	1	0	1 111	1011 11011		0		0		1	1
B1:	1		0	1		1		10111	1 11011 11	0		0		0	1	1

Classical FCST operations

How to compute operations over FCSTs:

- $\textsc{SDep}(v) = \textsc{SDep}(\text{LCA}(v, v)) = \max_{0 \le i < d}\{i + \textsc{SDep}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r)))\}$.
- $\text{LCA}(v, v') =$
  $\text{LF}(v[0..i-1],$
  $\text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\}))),$
  for the i in sdep.

Classical FCST operations

How to compute operations over FCSTs:

- $\text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \leq i < d}\{i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r)))\}$.
- $\text{LCA}(v, v') =$
  $\text{LF}(v[0..i-1],$
  $\text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\}))),$
  for the i in sdep.

Classical FCST operations

How to compute operations over FCSTs:

- $\text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \le i < d}\{i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r)))\}$.
- $\text{LCA}(v, v') =$
  $\text{LF}(v[0..i-1],$
  $\text{LCSA}(\psi^i(\min\{v_l, v_l'\}), \psi^i(\max\{v_r, v_r'\})))$,
  for the i in sdep.

Distributed Fully-Compressed Suffix Trees 1 min

Problem

How to simulate the acess to a big FCST by using several distributed FCSTs ?

- Store sampled node identifying bitmaps.
- Merge diferent LSA as in CSAs.

((0) (1) ((2) ((3) (4))) ((5) (6) (7) ((8) (9) (10) (11) (12)))))

	(	0	1	(	2	(	3	(4)))	(5) (6) (7) (	8	9	10	11	12	)	)	
B :	1	0	0	1	0	1	0	10111	1011011011	0	0	0	0	0	1	1	
B0:	1	0		1	0	1	0	1 111	1011 11011		0		0		1	1	
B1:	1		0	1		1		10111	1 11011 11	0		0		0	1	1	

Distributed Fully-Compressed Suffix Trees

1 min

Problem

How to simulate the acess to a big FCST by using several distributed FCSTs ?

- Store sampled node identifying bitmaps.
- Merge diferent LSA as in CSAs.

((0) (1) ((2) ((3) (4))) ((5) (6) (7) ((8) (9) (10) (11) (12)))))

	(	0	1	(	2	(	3	(4)))	(5) (6) (7) (	8	9	10	11	12	)	)
B :	1	0	0	1	0	1	0	10111	1011011011	0	0	0	0	0	1	1
B0:	1	0		1	0	1	0	1 111	1011 11011		0		0		1	1
B1:	1		0	1		1		10111	1 11011 11	0		0		0	1	1

Summary

0 min

We presented parallel and distributed compressed indexes.
Our solutions obtain:

- Fast operations in compressed space.
- Support for very large indexes.

Acknowledgments

0 min

Thanks for listening.
Questions ?

Marín, M., Navarro, G.:
Distributed query processing using suffix arrays.
In: Proc. 10th SPIRE. LNCS 2857 (2003) 311–325

Mäkinen, V., Navarro, G., Sadakane, K.:
Advantages of backward searching — efficient secondary memory and distributed implementation of compressed suffix arrays.
In: Proc. 15th ISAAC. LNCS 3341 (2004) 681–692

Clifford, R.:
Distributed suffix trees.
J. Discrete Algorithms **3**(2-4) (2005) 176–197