Regularities in Sequences

Gerth Stølting Brodal

BRICS

University of Aarhus
Denmark

joint work with

Christian N. S. Pedersen, Rune B. Lyngsø
BRICS

Jens Stoye
German Cancer Research Center, Heidelberg, Germany

Chennai, December 1999

Regularities in Sequences

or a longer title...

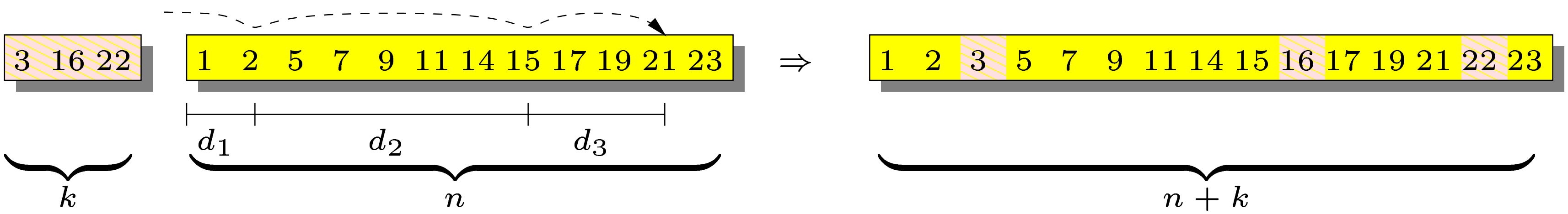
On Using Efficient Merging in $\mathcal{O}(n \log n)$ Time Algorithms for Finding Regularities in Strings

Merging $\rightarrow$ $\mathcal{O}(n \log n)$ Algorithms $\rightarrow$ Strings $\rightarrow$ Regularities

Comparison Based Merging

Problem

Merge two sorted sequences of length k and n, $k \leq n$



Solutions

(1) Sequential merging $\Rightarrow \mathcal{O}(n+k)$ comparisons

(2) Merging using exponential searching $\Rightarrow$

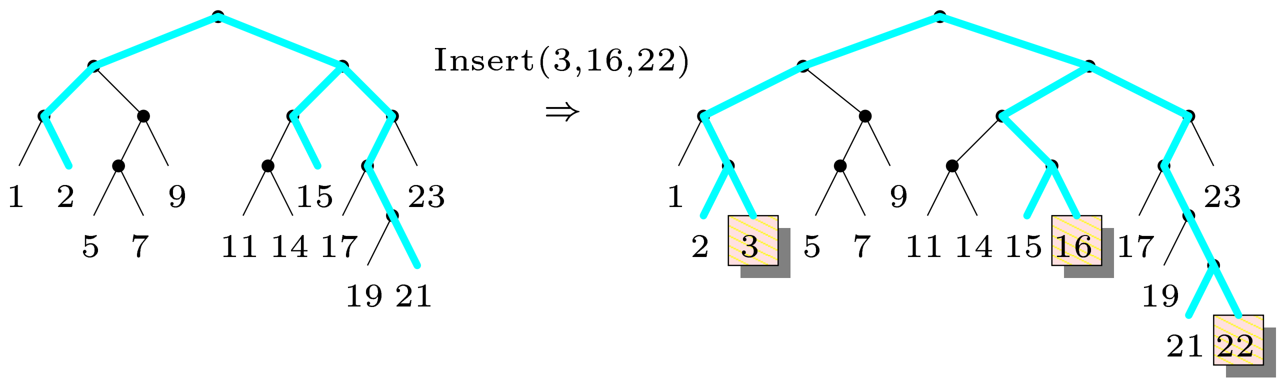
$$\Theta\left(\sum_{i=1}^{k} \log d_i\right) = \Theta\left(\log \binom{n+k}{k}\right) \text{ comparisons}$$

Hwang & Lin 1972

Time ?

Merging Search Trees

Brown & Tarjan 1979



Merging two balanced search trees of size k and n, $k \leq n$

- Size k tree ⇒ list of k elements $\mathcal{O}(k)$
- Search for the k elements in **one traversal** of the size n tree $\mathcal{O}\left(\left|\bigcup_{i=1}^{k} \text{search-path}_i\right|\right)$
- Insert the k elements $\mathcal{O}(k)$
- Rebalance typically amortized $\mathcal{O}(k)$

$\bigcup$ paths ?

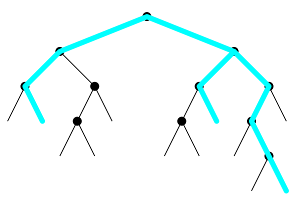
Path Theorem

Brown & Tarjan 1979

Theorem

In a **balanced** search tree with n leaves the #edges in the union of k distinct root-to-leaf paths is

$$\left| \bigcup_{i=1}^{k} \text{path}_i \right| = \mathcal{O}\left(\log \binom{n+k}{k}\right)$$



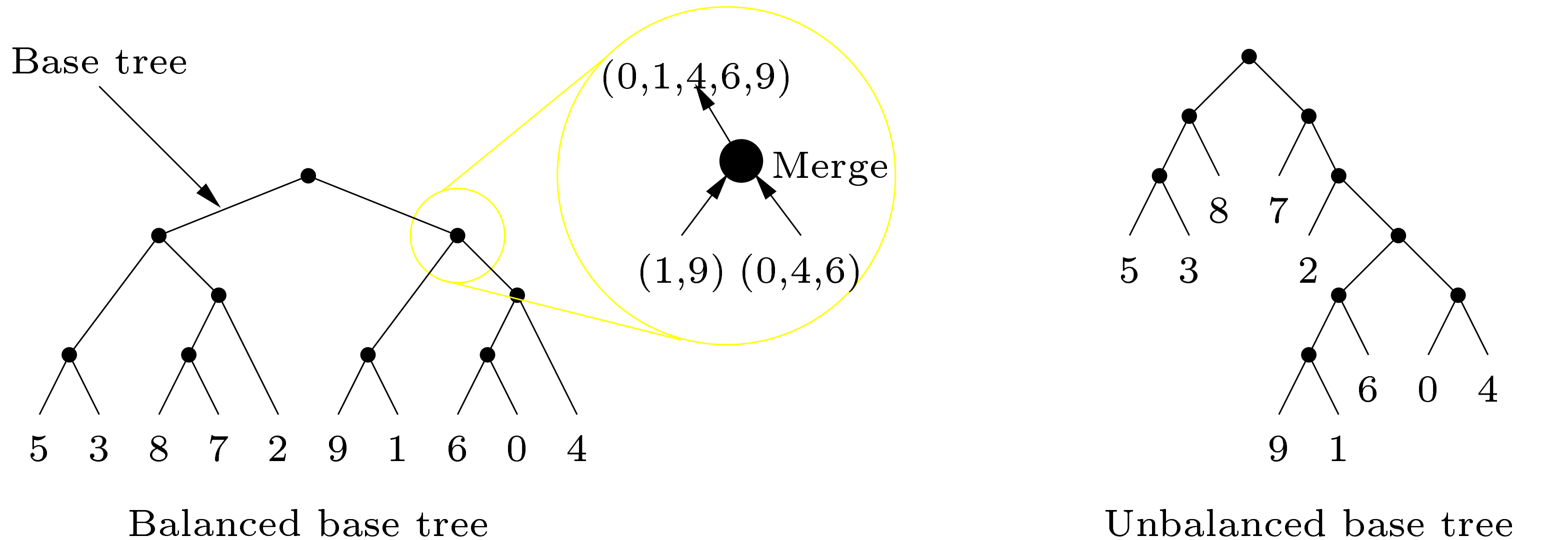
[e.g. AVL–trees, red–black trees, 2–3–trees, 2–4–trees, ...]

Corollary

Merging two trees of size k and n, $k \leq n$, takes time $\mathcal{O}\left(\log \binom{n+k}{k}\right)$ which is optimal

Merge Sort ?

Merge Sort — Revisited



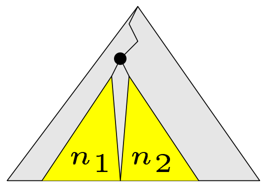
Balanced base tree	+	Sequential merging	$\Rightarrow$	$\mathcal{O}(n \log n)$
Unbalanced base tree	+	Sequential merging	$\Rightarrow$	$\mathcal{O}(n^2)$
Unbalanced base tree	+	Merging by insertion*	$\Rightarrow$	$\mathcal{O}(n \log^2 n)$
Unbalanced base tree	+	Optimal merging	$\Rightarrow$	$\mathcal{O}(n \log n)$

$n \log n$?

Merge Sort — Analysis

Theorem

If we spend time $\mathcal{O}\left(\log\binom{n_1+n_2}{n_1}\right)$ at each node with two children spanning respectively n_1 and n_2 leaves, $n_1 \leq n_2$, then we in a subtree spanning n leaves spend total time $\mathcal{O}(\log n!)$



Proof

Induction, using

$$\begin{aligned}
&\log n_1! + \log n_2! + \log\binom{n_1+n_2}{n_1} \\
&= \log n_1! + \log n_2! + \log\frac{(n_1+n_2)!}{n_1!\cdot n_2!} \\
&= \log n_1! + \log n_2! + \log(n_1+n_2)! - \log n_1! - \log n_2! \\
&= \log(n_1+n_2)!
\end{aligned}$$

□

Corollary

Merging n singleton lists using any base tree takes time $\mathcal{O}(n\log n)$

Applications of the Path Lemma

Searching

Searching for the predecessors of $e_1 \leq \cdots \leq e_k$ in a balanced search tree of size $n \geq k$ takes time $\mathcal{O}\left(\log \binom{n+k}{k}\right)$

Δ–Searching

The δ-predecessor of an x in a sorted list $L = (x_1, \ldots, x_n)$ is defined by

$$\begin{aligned} \text{max-gap}(L) &= \max\{0, x_2 - x_1, \ldots, x_n - x_{n-1}\} \\ \Delta\text{-pred}(L, \delta, x) &= \min\{y \in L \mid \text{max-gap}(L \cap [y, x]) \leq \delta\} \end{aligned}$$

Ex. $\Delta\text{-pred}(\ (1, 2, \underline{5, 7, 9, 11}, 14, 15, 17, 19, 21, 23),\ 2,\ 13\ ) = 5$

- Multi-Δ-Pred$(T, \delta, e_1, \ldots, e_k)$ for each e_i finds $\Delta\text{-pred}(T, \delta, e_i)$

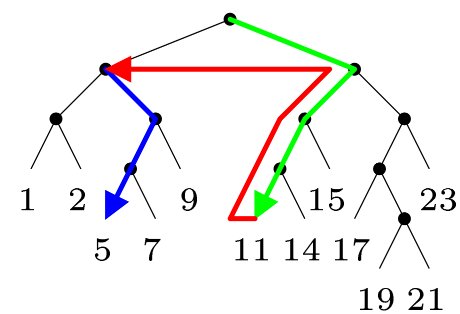
Δ-Pred ?

Δ–Searching

For each node v store $\min(T_v)$, $\max(T_v)$, and max-gap(T_v)

The computation of Δ-pred(T, δ, e) proceeds in three phases

- compute pred(T, e)
- search for Δ-pred$(T, \delta, \min(T_v))$
- search for Δ-pred$(T_v, \delta, \max(T_v))$



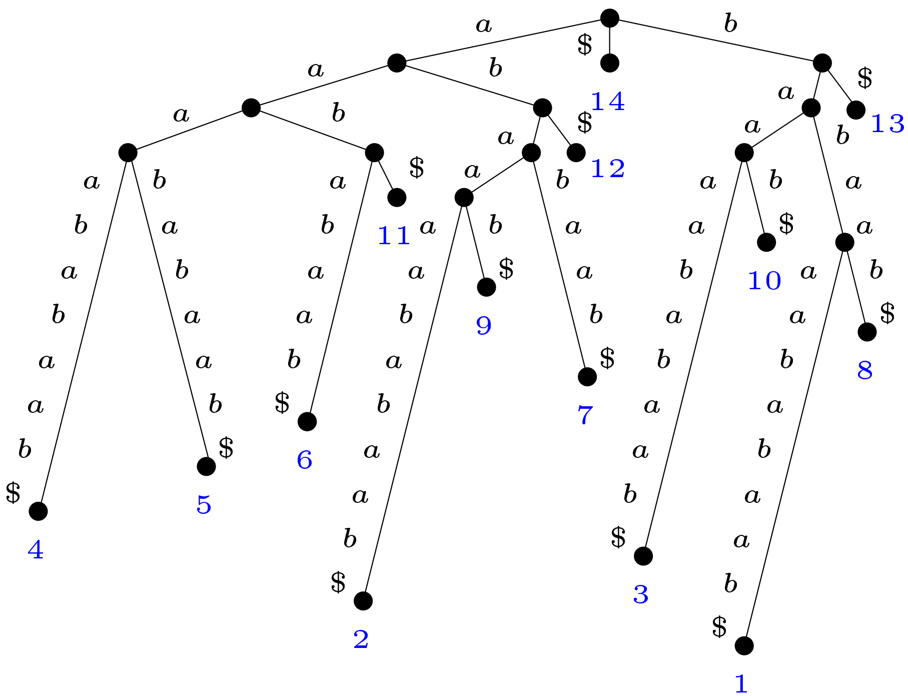
Δ-pred(T, 2, 13)

Multi-Δ-Pred$(T, \delta, e_1, \ldots, e_k)$ can be performed in time $\mathcal{O}\left(\log \binom{n+k}{k}\right)$ when $e_1 \leq \cdots \leq e_k$ and $k \leq n$, e.g. using dynamic programing to only compute Δ-pred$(T, \delta, \min(T_v))$ and Δ-pred$(T, \delta, \max(T_v))$ once per node

String Notation

Notation		Example		
Alphabet	Σ	Σ	$=$	$\{a, b\}$
String	$S \in \Sigma^*$	S	$=$	$b\,a\,b\,a\,a\,a\,a\,b\,a\,b\,a\,a\,b$
ith character	$S[i]$	$S[3]$	$=$	b
Substring	$S[i\,..\,j]$	$S[3\,..\,8]$	$=$	$b\,a\,a\,a\,a\,b$
Occurrence	(α, i)	$(\,b\,a\,b\,,\,8\,)$	$=$	$b\,a\,b\,a\,a\,a\,a\,\underline{b\,a\,b}\,a\,a\,b$
Tandem repeat	$\alpha\alpha$	$\alpha\alpha$	$=$	$\underline{b\,a}\,\underline{b\,a}$
Pair	(α, i, j)	$(\,b\,a\,b\,a\,,\,1,\,8\,)$	$=$	$\underline{b\,a\,b\,a}\,a\,a\,a\,\underline{b\,a\,b\,a}\,a\,b$

Suffix Trees



$S = b\,a\,b\,a\,a\,a\,a\,b\,a\,b\,a\,a\,b$

Suffixes

$b\,a\,b\,a\,a\,a\,a\,b\,a\,b\,a\,a\,b\,\$$
$a\,b\,a\,a\,a\,a\,b\,a\,b\,a\,a\,b\,\$$
$b\,a\,a\,a\,a\,b\,a\,b\,a\,a\,b\,\$$
$a\,a\,a\,a\,b\,a\,b\,a\,a\,b\,\$$
$a\,a\,a\,b\,a\,b\,a\,a\,b\,\$$
$a\,a\,b\,a\,b\,a\,a\,b\,\$$
$a\,b\,a\,b\,a\,a\,b\,\$$
$b\,a\,b\,a\,a\,b\,\$$
$a\,b\,a\,a\,b\,\$$
$b\,a\,a\,b\,\$$
$a\,a\,b\,\$$
$a\,b\,\$$
$b\,\$$
$\$$

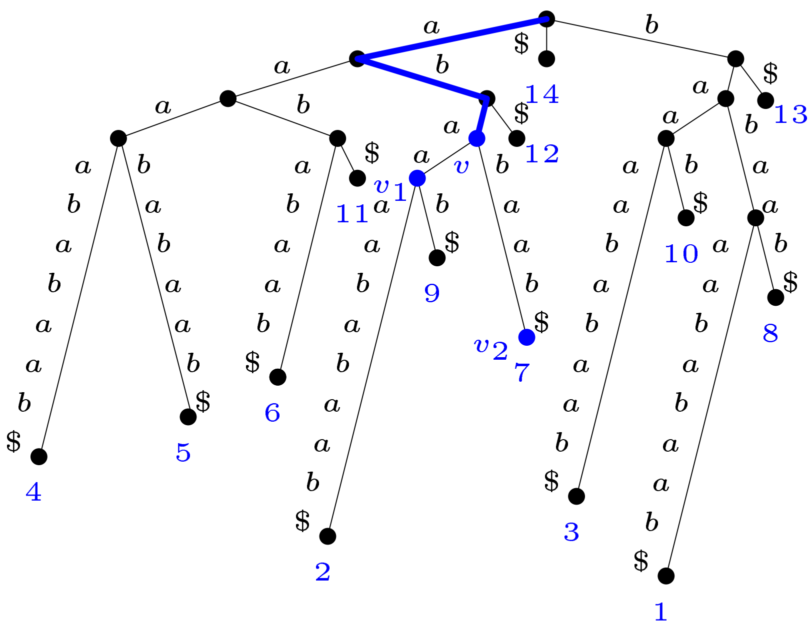
Suffix tree of S $\equiv$ trie of all suffixes of $S\$$, where $\$ \notin \Sigma$

Leaf labeled i $\equiv$ suffix $S[i\,..]\$$

Theorem

The suffix tree of S can be constructed in time $\mathcal{O}(n)$ — Weiner 1973

Path Labels and Leaf Lists



$S = b\,\underline{a\,b\,a}\,a\,a\,\underline{a\,b\,\underline{a}}\underline{\,b\,a}\,a\,b$
$L(v) = a\,b\,a$
$LL(v_1) = \{2, 9\}$
$LL(v_2) = \{7\}$
$LL(v) = LL(v_1) \cup LL(v_2) = \{2, 7, 9\}$

Let v be a node in the suffix tree of S

Path-label $L(v)$ = string spelled from the root to v

Leaf-list $LL(v)$ = set of leaf labels in the subtree rooted at v

$$LL(v) = \text{all occurrences of } L(v) = \bigcup_{c \in \text{children}(v)} LL(c)$$

Regularities

We will develop $\mathcal{O}(n \log n + |\text{output}|)$ time algorithms for

- Finding all tandem repeats in S

$$b\,a\,b\,a\,a\,a\,a\,\underline{b\,a}\,\underline{b\,a}\,a\,b$$

- Finding all maximal pairs in S with bounded gap

$$\underline{b\,a\,b\,a\,a}\,\underbrace{a\,a}_{\text{gap}}\,\underline{b\,a\,b\,a\,a}\,b$$

- Finding all maximal quasi-periodic substrings in S

$$a\,a\,a\,b\,a\,a\,b\,b\,b\,\overline{a\,b\,a\,a\,b\,a\,a\,b\,a\,a\,b\,a\,a}\,b\,b\,a\,a\,a\,b\,a$$

Tandem Repeats

Stoye & Gusfield 1998

Problem

Given S, find all substrings of S which are tandem repeats

For each tandem repeat occurrence $(\alpha\alpha, i)$ output $(|\alpha|, i)$

Ex. $S = b\,a\,b\,a\,a\,a\,a\,b\,a\,b\,a\,a\,b$

Output $= (2,1), (1,4), (1,5), (1,6), (2,7), (2,8), (1,11)$

Tandem repeat occ. $(|\alpha|, i)$ is branching iff $S[i + |\alpha|] \neq S[i + 2|\alpha|]$

Ex. $(2,7)$ is non-branching and $(2,8)$ is branching

$(|\alpha|, i)$ non-branching $\Rightarrow$ $(|\alpha|, i+1)$ tandem repeat occ.

Given all branching tandem repeat occ. we can compute all tandem repeat occ. in time $\mathcal{O}(|\text{output}|)$

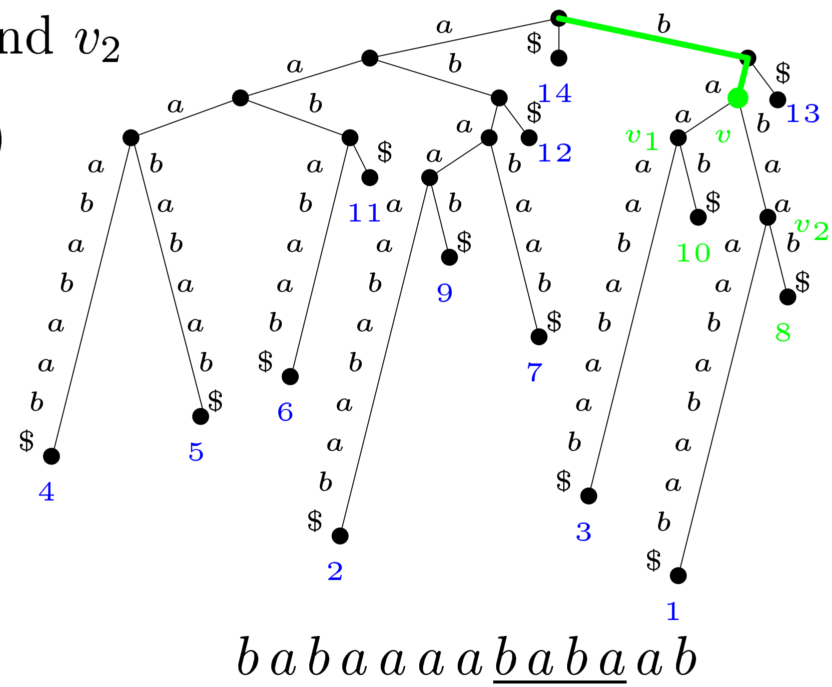
Branching ?

Branching Tandem Repeat Occurrences

Lemma

$(\alpha\alpha, i)$ is a branching tandem repeat occurrence in S iff

- $\exists$ node v in the suffix tree of S with $L(v) = \alpha$
- v has two distinct children v_1 and v_2
- $i \in LL(v_1)$ and $i + |\alpha| \in LL(v_2)$



Branching Tandem Repeat Occurrences

Algorithm

1. Compute suffix tree (w.l.o.g. all nodes $\leq$ two children) $\mathcal{O}(n)$
2. Bottom-up do for each node v with children v_1 and v_2
   $n_1 = |LL(v_1)| \leq |LL(v_2)| = n_2$

   a) For each $i \in LL(v_1)$ search for $i \pm |L(v)|$ in $LL(v_2)$
      using optimal multi-search $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$
      If $i + |L(v)| \in LL(v_2)$ then output $(|L(v)|, i)$
      If $i - |L(v)| \in LL(v_2)$ then output $(|L(v)|, i - |L(v)|)$

   b) $LL(v) = LL(v_1) \cup LL(v_2)$
      using optimal tree merging $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$

Total time $\mathcal{O}(n \log n)$, i.e. $\mathcal{O}(n \log n)$ branching tandem repeat occ.

Maximal Pairs with Bounded Gap

Brodal et al. 1999

A pair (α, i, j) is

- right-maximal iff $S[i+|\alpha|] \neq S[j+|\alpha|]$
- left-maximal iff $S[i-1] \neq S[j-1]$
- maximal iff both right- and left-maximal

The gap of $(\alpha, i, j) = \max(i,j) - \min(i,j) - |\alpha|$

Ex. $b\, \underline{\underline{a\, b}\, a}\, a\, a\, \underline{\underline{a\, b}\, a}\, b\, a\, a\, b$

$(a\, b, 2, 7)$ is left-maximal but not right-maximal

$(a\, b\, a, 2, 7)$ is maximal and gap $= 2$

Problem

Given S, find all maximal pairs in S with gap in an interval $[\Delta_L, \Delta_H]$

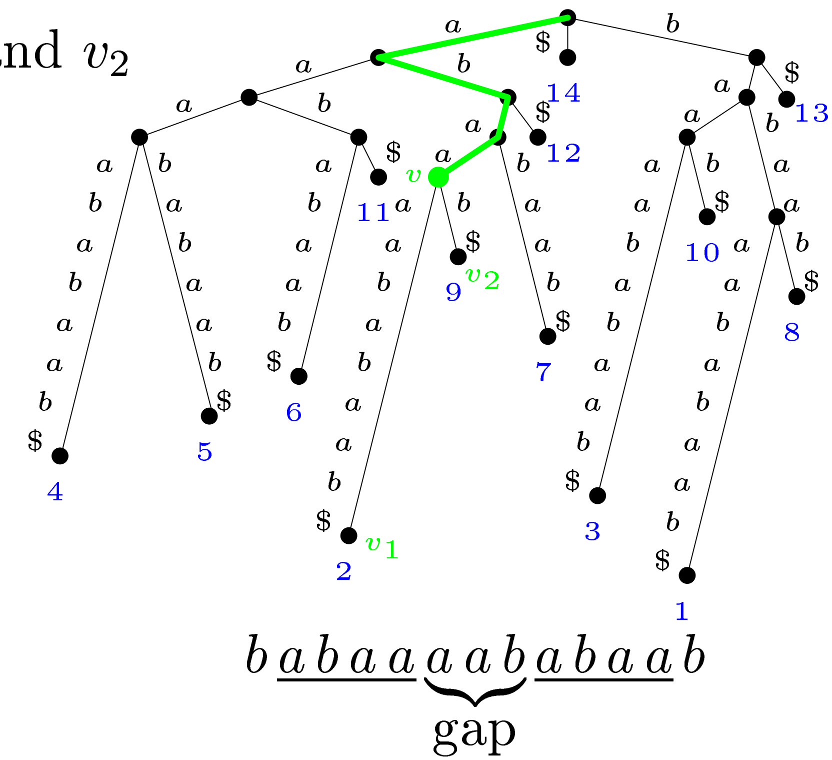
For each such pair (α, i, j) output $(|\alpha|, i, j)$

Right-Maximal Pairs

Lemma

(α, i, j) is a right-maximal pair in S iff

- $\exists$ node v in the suffix tree of S with $L(v) = \alpha$
- v has two distinct children v_1 and v_2
- $i \in LL(v_1)$ and $j \in LL(v_2)$



Right-Maximal Pairs with Bounded Gap

Algorithm

1. Compute suffix tree (w.l.o.g. all nodes $\leq$ two children) $\mathcal{O}(n)$
2. Bottom-up do for each node v with children v_1 and v_2
   $n_1 = |LL(v_1)| \leq |LL(v_2)| = n_2$

   a) For each $i \in LL(v_1)$ search for $i \pm (|L(v)| + \Delta_H)$ in $LL(v_2)$ $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$

   $j \in [i{+}|L(v)|{+}\Delta_L \,..\, i+|L(v)|{+}\Delta_H] \cap LL(v_2)$ output $(|L(v)|, i, j)$

   $j \in [i{-}|L(v)|{-}\Delta_H \,..\, i-|L(v)|{-}\Delta_L] \cap LL(v_2)$ output $(|L(v)|, j, i)$

   $\mathcal{O}(|\text{output}|)$

   b) $LL(v) = LL(v_1) \cup LL(v_2)$ $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$

Total time $\mathcal{O}(n \log n + |\text{output}|)$ Maximal ?

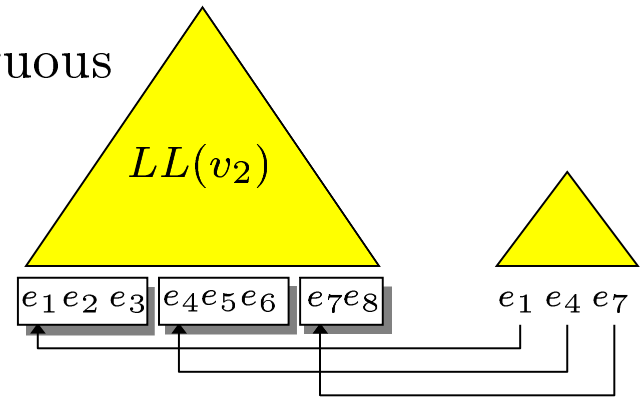
Maximal Pairs with Bounded Gap

Simple approach

- Compute all right-maximal pairs $(|\alpha|, i, j)$ with bounded gap
- Output all $(|\alpha|, i, j)$ where $S[i-1] \neq S[j-1]$

... but requires too much time

Efficient approach



Divide the leaf-lists into blocks of contiguous indices with same left-character

For each $LL(v)$ keep a tree with the starting points of each block

In the computation of

$$\{j \in [i+|L(v)|+\Delta_L .. i+|L(v)|+\Delta_H] \cap LL(v_2) \ : \ S[j-1] \neq S[i-1]\}$$

a block of j's where $S[j-1] = S[i-1]$ can be skipped in time $\mathcal{O}(1)$

Total time $\mathcal{O}(n \log n + |\text{output}|)$

Maximal Quasi-Periodic Substrings

A string is quasi-periodic if it can be covered by a shorter substring

$$a\,b\,a\,a\,b\,a\,a\,b\,a\,a\,b\,a\,a\,b$$

$S[i\,..\,j]$ is a maximal quasi-periodic substring with quasi-period α iff

- α is not quasi-periodic and α covers $S[i\,..\,j]$
- no extension of $S[i\,..\,j]$ is covered by α
- $\alpha\, S[j+1]$ does not cover $S[i\,..\,j+1]$

$$S[i\,..\,j]$$

$$a\,a\,a\,b\,a\,a\,b\,b\,b\,a\,b\,a\,a\,b\,a\,a\,b\,a\,a\,b\,a\,a\,b\,b\,a\,a\,a\,b\,a$$

$$\alpha$$

Problem

Find all $(|\alpha|, i, j)$ where $S[i\,..\,j]$ is a maximal quasi-periodic substring with quasi-period α

Apostolico & Ehrenfeucht 1993

$\mathcal{O}(n \log^2 n)$

Maximal Quasi-Periodic Substrings

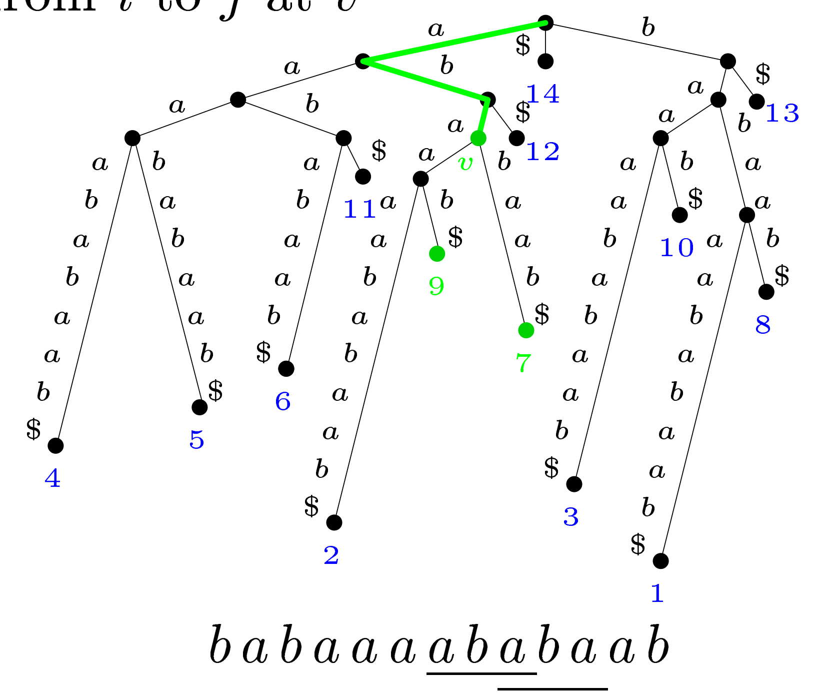
Theorem

$(|\alpha|, i, j)$ is a maximal quasi-periodic substring of S iff

1. There exists a node v in the suffix tree of S with $L(v) = \alpha$
2. The $L(v)$ is not quasi-periodic
3. There exists a coalescing run from i to j at v

A run is a maximal subsequence of adjacent or overlapping occurrences of $L(v)$ in $LL(v)$

A coalescing run is a run containing indices from leaf-lists of two children of v



Finding Maximal Quasi-Periodic Substrings

Algorithm

1. Mark all v where $L(v)$ not quasi-periodic $\mathcal{O}(n \log n)$

   The most complicated step — applies multi-searches and merging

2. Bottom-up do for each node v with children v_1 and v_2

   $n_1 = |LL(v_1)| \leq |LL(v_2)| = n_2$

   a) Compute $LL(v) = LL(v_1) \cup LL(v_2)$ $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$

   b) For each coalescing run in $LL(v)$ identify at least one $i \in LL(v_1)$ in the run $\mathcal{O}(n_1)$

   c) Apply $\textsf{Multi-}\Delta\textsf{-Pred}(LL(v), |L(v)|, \ldots)$ to the result of b) to find the start and end of each coalescing run $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$

   d) If v is marked then report all the coalescing runs found in c) as maximal quasi-periodic substrings

Total time $\mathcal{O}(n \log n)$, i.e. $\mathcal{O}(n \log n)$ maximal quasi-periodic substrings

Theorems Applied

Theorem

In a **balanced** search tree with n leaves the #edges in the union of k distinct root-to-leaf paths is

$$\left|\bigcup_{i=1}^{k} \text{path}_i\right| = \mathcal{O}\left(\log \binom{n+k}{k}\right)$$

Theorem

If we spend time $\mathcal{O}\left(\log \binom{n_1+n_2}{n_1}\right)$ at each node with two children spanning respectively n_1 and n_2 leaves, $n_1 \leq n_2$, then we in a subtree spanning n leaves spend total time $\mathcal{O}(\log n!)$

Conclusion

- We have presented $\mathcal{O}(n \log n + |\text{output}|)$ time algorithms for finding all tandem repeats, maximal pairs with bounded gap, and maximal quasi-periodic substrings of a string
- Based on efficient merging and two related theorems
- The data structuring approach allowed us to concentrate on a few crucial underlying string properties

Open problem

- Can the time be improved to $o(n \log n)$ for finding all pairs with bounded gap or all maximal quasi-periodic substrings ?

Without an upper bound on the gaps all pairs can be found in time $\mathcal{O}(n + |\text{output}|)$ Brodal et al. 1999

A compact representation of all tandem repeats can be found in time $\mathcal{O}(n)$ Gusfield & Stoye 1998