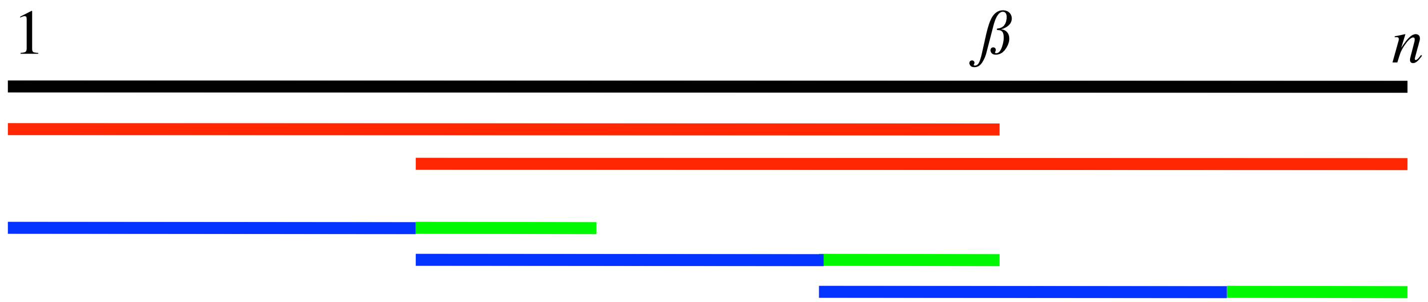
Tandem repeats

Recall

Any string $\boldsymbol{S}[1..n]$ can be written in its normal form:



Let $p = n - \beta$, then $\boldsymbol{S} = \boldsymbol{w}^{n/p} = \boldsymbol{w}^{\lfloor n/p \rfloor}\boldsymbol{w'}$, where $\boldsymbol{w'} = \boldsymbol{S}[1..\, n - \lfloor n/p \rfloor \cdot p]$

The ***normal form*** of $\boldsymbol{S}[1..n]$ is $\boldsymbol{w}^{n/p^*}$, where p^* is the minimum period

If a string $\boldsymbol{S}[1..n]$ is not periodic, i.e. $p^*=n$, then it is ***primitive***

Tandem repeats

A string ***w*** is a tandem repeat (or square) if:

$$w = \boxed{\quad \alpha \quad}\boxed{\quad \alpha \quad} \qquad \alpha \in \Sigma^+$$

A tandem repeat $\alpha\alpha$ is primitive if and only if α is primitive

A string $\boldsymbol{w} = \alpha^{\boldsymbol{k}}$ for $k = 3,4,\ldots$ is (by some) called a tandem array

Tandem repeats

A string ***w*** is a tandem repeat (or square) if:

$$w = \boxed{\quad\alpha\quad}\boxed{\quad\alpha\quad} \qquad \alpha \in \Sigma^+$$

A tandem repeat $\alpha\alpha$ is primitive if and only if α is primitive

A string $\boldsymbol{w} = \alpha^k$ for $k = 3,4,\ldots$ is (by some) called a tandem array

Note: "tandem repeat" (and "tandem array") is a string property, i.e. a string can be a "tandem repeat" ...

How difficult is it to decide if $\boldsymbol{S}[1..n]$ is a tandem repeat?

Tandem repeats

A string w is a tandem repeat (or square) if:

$$w = \boxed{\quad \alpha \quad}\boxed{\quad \alpha \quad} \qquad \alpha \in \Sigma^+$$

A tandem repeat $\alpha\alpha$ is primitive if and only if α is primitive

A string $w = \alpha^k$ for $k = 3,4,\ldots$ is (by some) called a tandem array

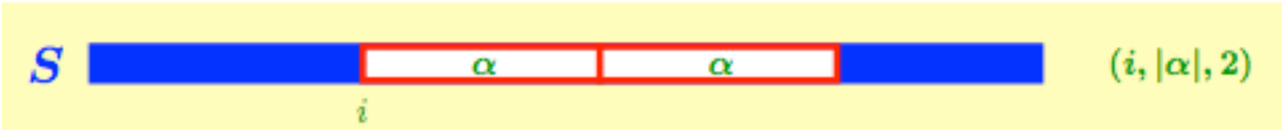
Note: "tandem repeat" (and "tandem array") is a string property,

Find minimum period p^* in time $O(n)$, if n/p^* is 2,4,6..., then "yes"

How difficult is it to decide if $S[1..n]$ is a tandem repeat?

Occurrences of tandem repeats

An occurrence of a tandem repeat in ***S*:**

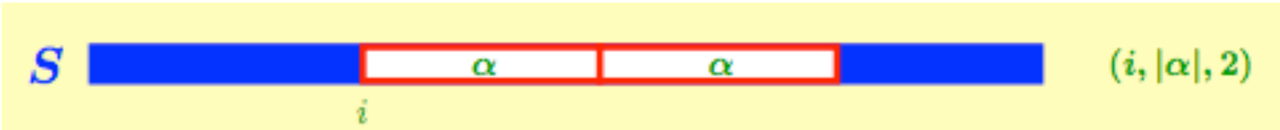


Tandem repeat αα is in ***S***, if it occurs one or more times in ***S***

Note that an occurrence can be encoded in space O(1) as (i, |α|, 2)

Occurrences of tandem repeats

An occurrence of a tandem repeat in ***S*:**



Tandem repeat $\alpha\alpha$ is in ***S***, if it occurs one or more times in ***S***

Note that an occurrence can be encoded in space O(1) as (i, $|\alpha|$, 2)

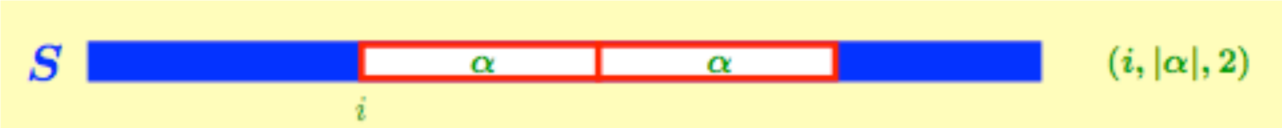
Computational problems:

(1) Find all occurrences of tandem repeats in $S[1..n]$

(2) Find all (primitive) tandem repeats in $S[1..n]$

Occurrences of tandem repeats

An occurrence of a tandem repeat in ***S*:**

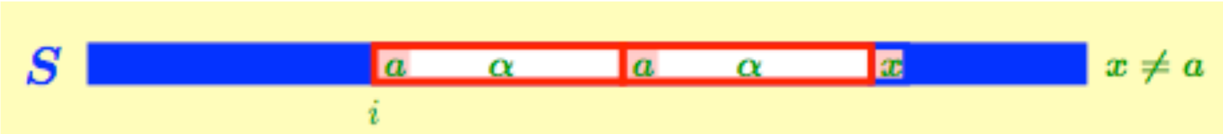


Tandem repeat $\alpha\alpha$ is in ***S***, if it occurs one or more times in ***S***
Note that an occurrence can be encoded in space O(1) as $(i, |\alpha|, 2)$

Computational problems:
(1) Find all occurrences of tandem repeats in $S[1..n]$

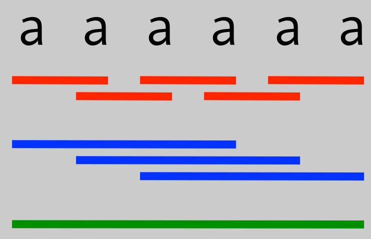
(2) Find all (primitive) tandem repeats in $S[1..n]$

A branching occurrence of a tandem repeats in ***S***:



A simple example

The strings *aaaaaa* contains:

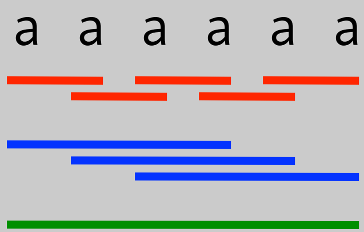


5 occurrences of *aa*
3 occurrences of *aaaa*
1 occurrence of *aaaaaa*

In general the a^n contains $\lfloor n^2/4 \rfloor$ occurrences of $\lfloor n/2 \rfloor$ tandem repeats, but there is only one primitive tandem repeat *aa* ...

A simple example

The strings *aaaaaa* contains:



5 occurrences of *aa*
3 occurrences of *aaaa*
1 occurrence of *aaaaaa*

In general the a^n contains $\lfloor n^2/4 \rfloor$ occurrences of $\lfloor n/2 \rfloor$ tandem repeats, but there is only one primitive tandem repeat *aa* ...

... or equivalently one maximal repetition $(1, 1, n)$, which we also call a primitive tandem array ...

Observation: there can be no more than $O(n^2)$ occurrences of tandem repeats in $S[1..n]$, but how many are e.g primitive?

Fibonacci strings

Lemma 3.4.1: The Fibonacci string f_n is defined recursively as:

$$\boldsymbol{f}_0 = b, \boldsymbol{f}_1 = a, \text{ and } \boldsymbol{f}_n = \boldsymbol{f}_{n\text{-}1}\boldsymbol{f}_{n\text{-}2}$$

$\boldsymbol{f}_0 = b$
$\boldsymbol{f}_1 = a$
$\boldsymbol{f}_2 = ab$
$\boldsymbol{f}_3 = aba$
$\boldsymbol{f}_4 = abaab$
$\boldsymbol{f}_5 = abaababa$

The length of $\boldsymbol{f}_n$ is the nth Fibonacci number f_n, i.e. f_0=1, f_1=1, and $f_n = f_{n\text{-}1}+f_{n\text{-}2}$ ...

Fibonacci strings are highly repetitive ...

Fibonacci strings

Lemma 3.4.1: The Fibonacci string f_n is defined recursively as:

$$f_0 = b, f_1 = a, \text{ and } f_n = f_{n-1} f_{n-2}$$

$f_0 = b$
$f_1 = a$
$f_2 = ab$
$f_3 = aba$
$f_4 = abaab$
$f_5 = abaababa$

The length of f_n is the nth Fibonacci number f_n, i.e. $f_0=1$, $f_1=1$, and $f_n = f_{n-1}+f_{n-2}$ ...

Fibonacci strings are highly repetitive ...

Theorem 3.4.8: A Fibonacci string of length n contains $\Omega(n \log n)$ occurrences of primitive tandem repeats ...

... actually f_n contains $0.7962 f_n \log f_n + O(f_n)$ occurrences of primitive tandem repeats [Fraenkel & Simpson 1999] ...

Fibonacci strings

Lemma 3.4.1: The Fibonacci string f_n is defined recursively as:

$$f_0 = b, f_1 = a, \text{ and } f_n = f_{n-1} f_{n-2}$$

$f_0 = b$

... since Crochemore's algorithm (sect. 12.1) finds all occurrences of primitive tandem repeats in $S[1..n]$ in time $O(n \log n)$, no string of length n contains more than $O(n \log n)$ occurrences of primitive tandem repeats ....

$f_4 = abaab$

$f_5 = abaababa$

Theorem 3.4.8: A Fibonacci string of length n contains $\Omega(n \log n)$ occurrences of primitive tandem repeats ...

... actually f_n contains $0.7962 f_n \log f_n + O(f_n)$ occurrences of primitive tandem repeats [Fraenkel & Simpson 1999] ...

Algorithms

Crochemore 1981 (Smyth 12.1.1): Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n).

Algorithms

Crochemore 1981 (Smyth 12.1.1): Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n).

Apostolico and Preparata 1983: Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n). Suffix tree based.

Algorithms

Crochemore 1981 (Smyth 12.1.1): Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n).

Apostolico and Preparata 1983: Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n). Suffix tree based.

Main and Lorentz 1984 (Smyth 12.1.2) : Find all occurrences of tandem repeats in time O(n log n + |output|) and space O(n).

Algorithms

Crochemore 1981 (Smyth 12.1.1): Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n).

Apostolico and Preparata 1983: Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n). Suffix tree based.

Main and Lorentz 1984 (Smyth 12.1.2) : Find all occurrences of tandem repeats in time O(n log n + |output|) and space O(n).

Stoye and Gusfield 1998: Find all occurrences of tandem repeats in time O(n log n + |output|) and space O(n), can also be adapted to find all occurrences of primitive tandem repeats in time O(n log n). Suffix tree based.

Algorithms

Crochemore 1981 (Smyth 12.1.1): Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n).

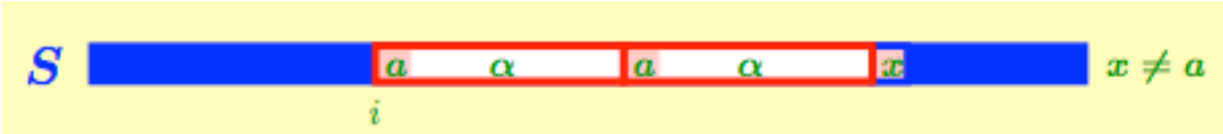
Apostolico and Preparata 1983: Find all occurrences of primitive tandem repeats in time O(n log n) and space O(n). Suffix tree based.

Main and Lorentz 1984 (Smyth 12.1.2) : Find all occurrences of tandem repeats in time O(n log n + |output|) and space O(n).

Stoye and Gusfield 1998: Find all occurrences of tandem repeats in time O(n log n + |output|) and space O(n), can also be adapted to find all occurrences of primitive tandem repeats in time O(n log n). Suffix tree based.

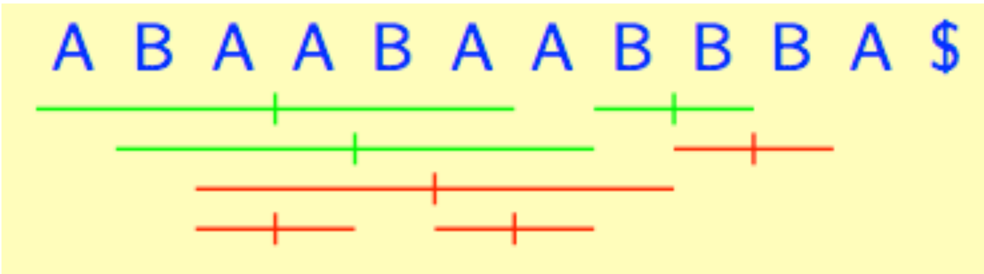
Basic observation

A branching occurrence of a tandem repeats in ***S***:



Lemma: any non-branching occurrence $(i, l, 2)$ of a tandem repeat is the left-rotation of another tandem repeat $(i+1, l, 2)$, starting one position to its right.

Example:



Suffix trees and tandem repeats

Lemma (folklore): Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a tandem repeat

b. i and j occur in the same leaf-list of some node v in $T(S)$ with depth $D(v) \geq l$

Suffix trees and tandem repeats

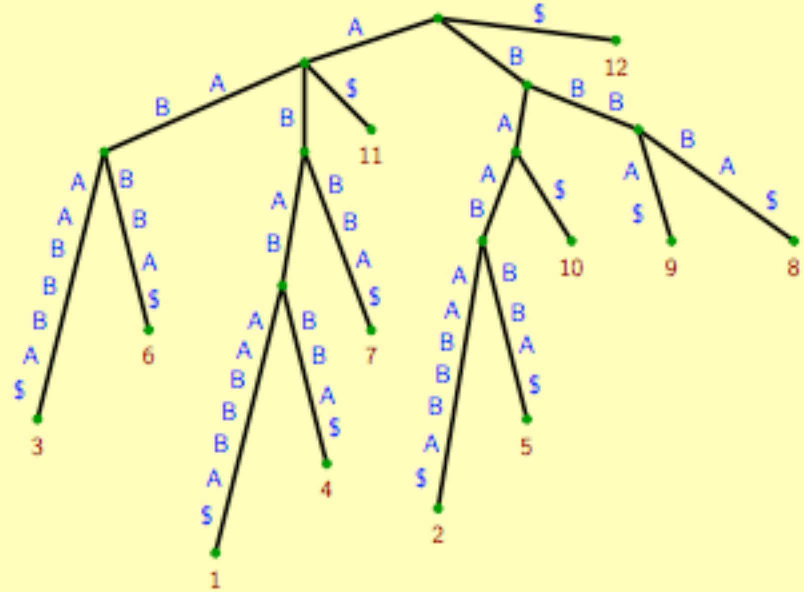
Lemma (folklore): Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a tandem repeat

b. i and j occur in the same leaf-list of some node v in $T(S)$ with depth $D(v) \geq l$

Example: A B A A B A A B B B A $

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

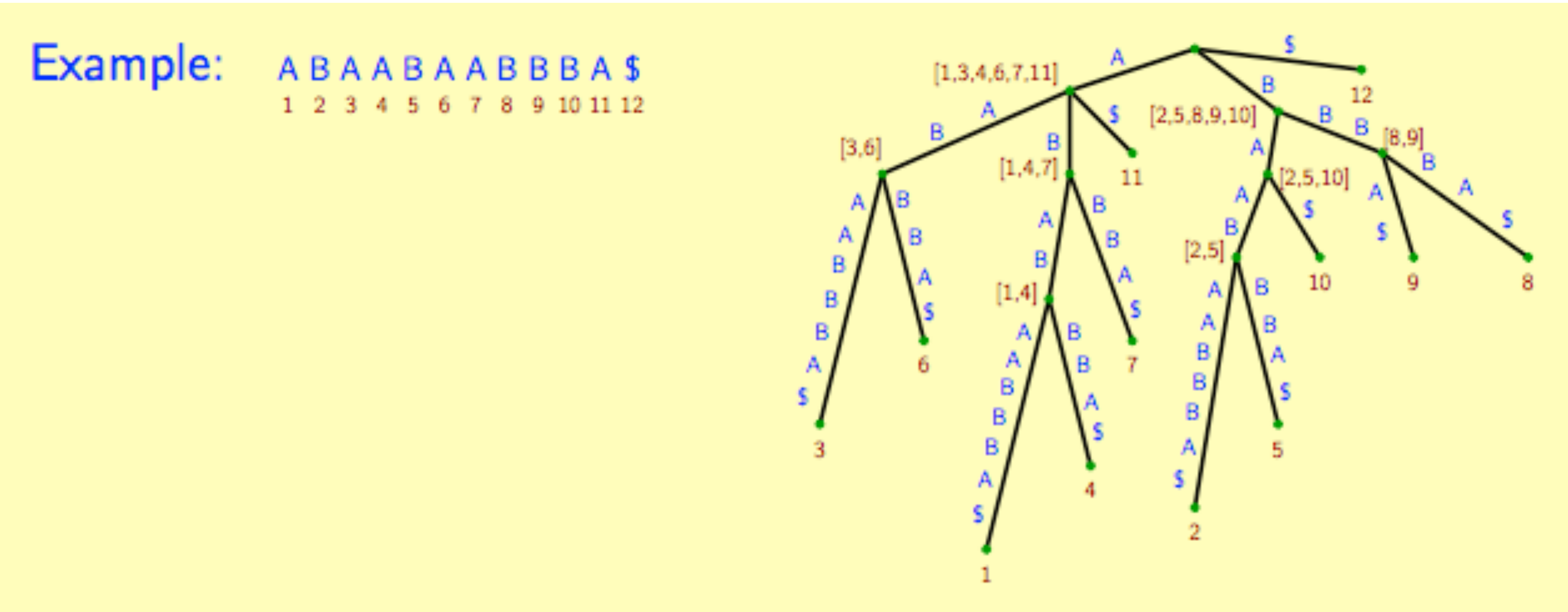


Suffix trees and tandem repeats

Lemma (folklore): Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a tandem repeat

b. i and j occur in the same leaf-list of some node v in $T(S)$ with depth $D(v) \geq l$

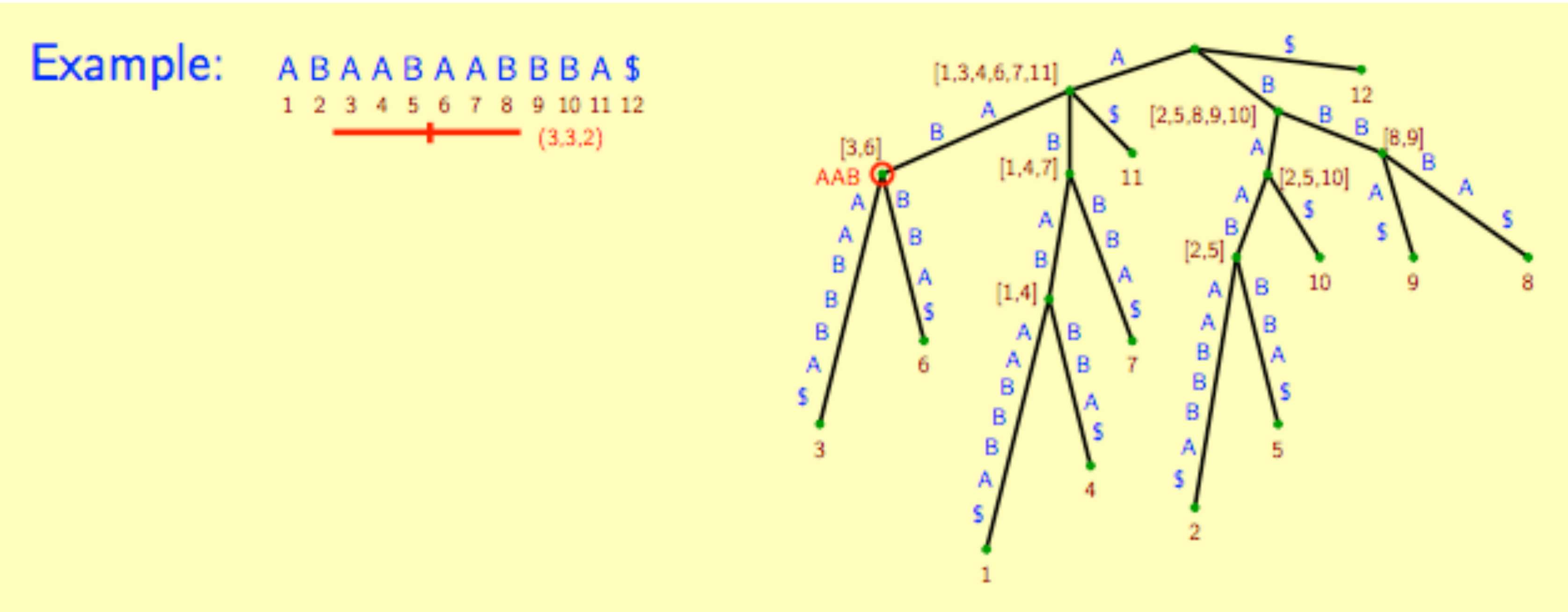


Suffix trees and tandem repeats

Lemma (folklore): Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a tandem repeat

b. i and j occur in the same leaf-list of some node v in $T(S)$ with depth $D(v) \geq l$

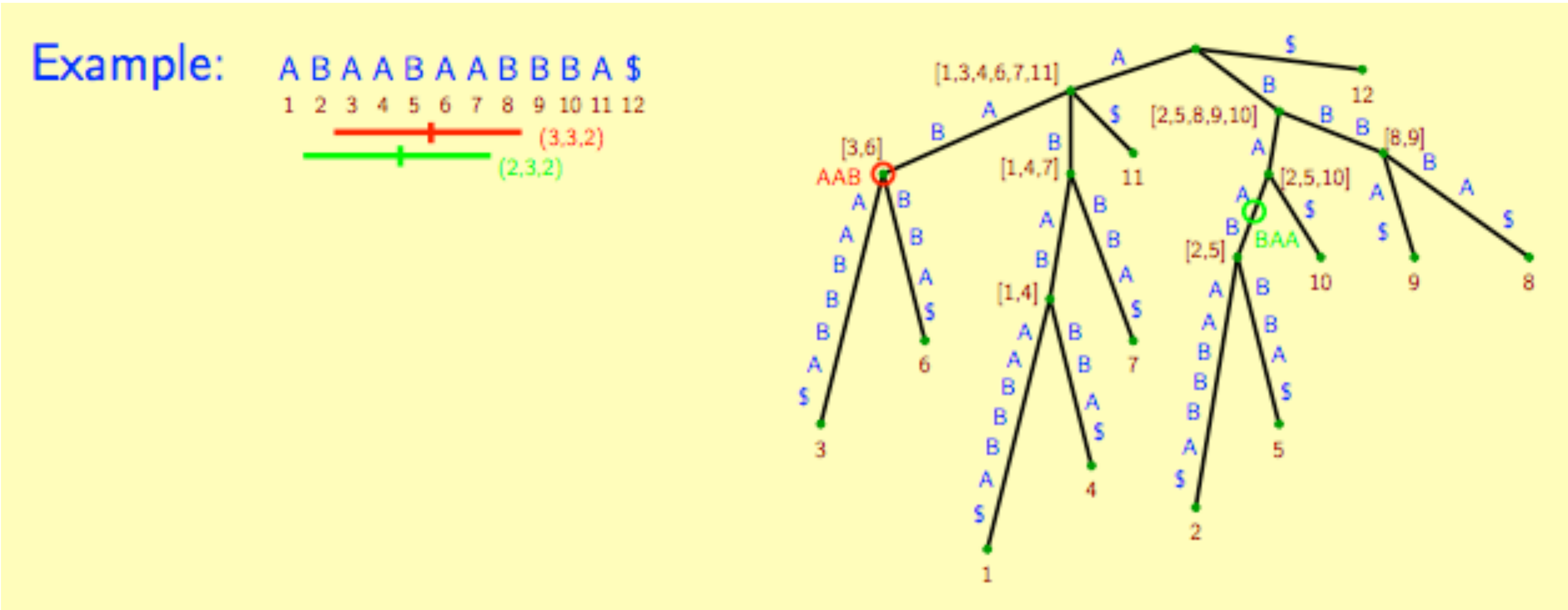


Suffix trees and tandem repeats

Lemma (folklore): Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a tandem repeat

b. i and j occur in the same leaf-list of some node v in $T(S)$ with depth $D(v) \geq l$



... and branching tandem repeats

Lemma: Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a branching tandem repeat

b. i and j occur in the same leaf-list of a node v in $T(S)$ with depth $D(v) = l$ and not in the same leaf-list of any node with depth $> l$

... and branching tandem repeats

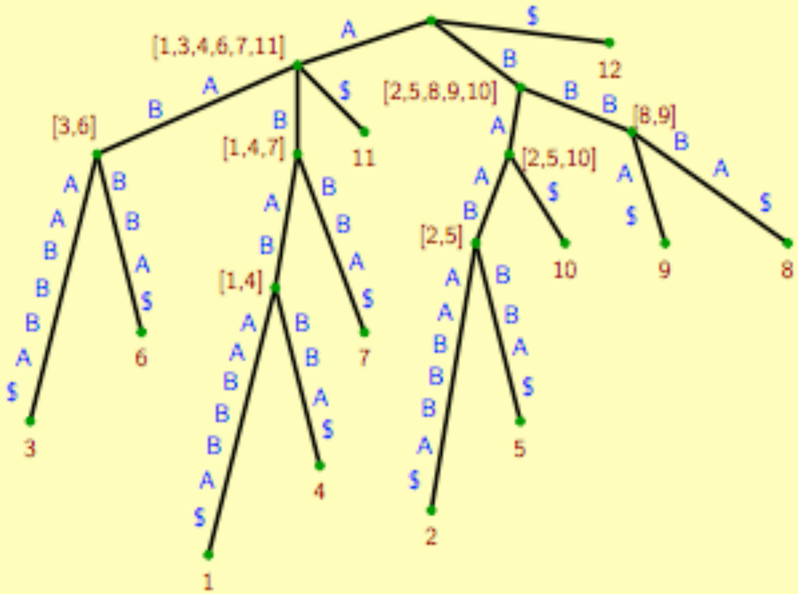
Lemma: Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a branching tandem repeat

b. i and j occur in the same leaf-list of a node v in $T(S)$ with depth $D(v) = l$ and not in the same leaf-list of any node with depth $> l$

Example:

A	B	A	A	B	A	A	B	B	B	A	$
1	2	3	4	5	6	7	8	9	10	11	12



... and branching tandem repeats

Lemma: Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

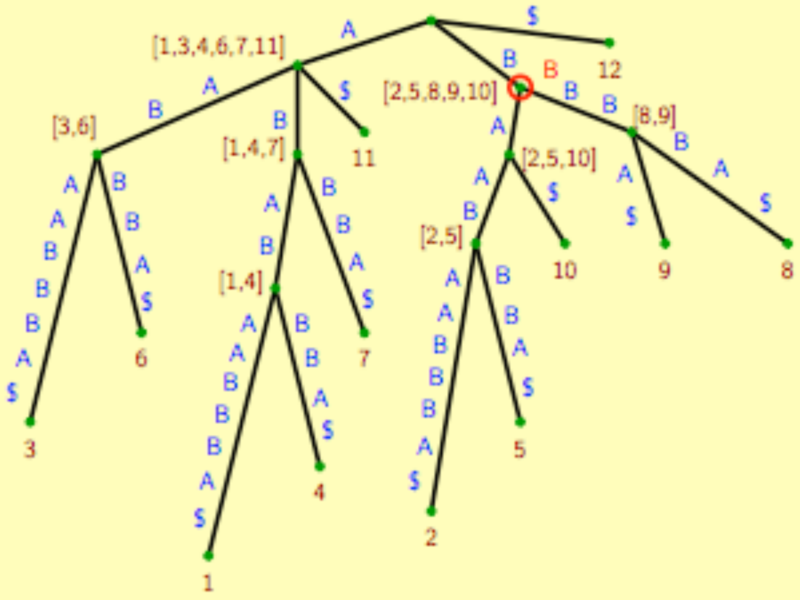
a. $(i, l, 2)$ is an occurrence of a branching tandem repeat

b. i and j occur in the same leaf-list of a node v in $T(S)$ with depth $D(v) = l$ and not in the same leaf-list of any node with depth $> l$

Example: A B A A B A A B B B A $

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

(9,1,2)



... and branching tandem repeats

Lemma: Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

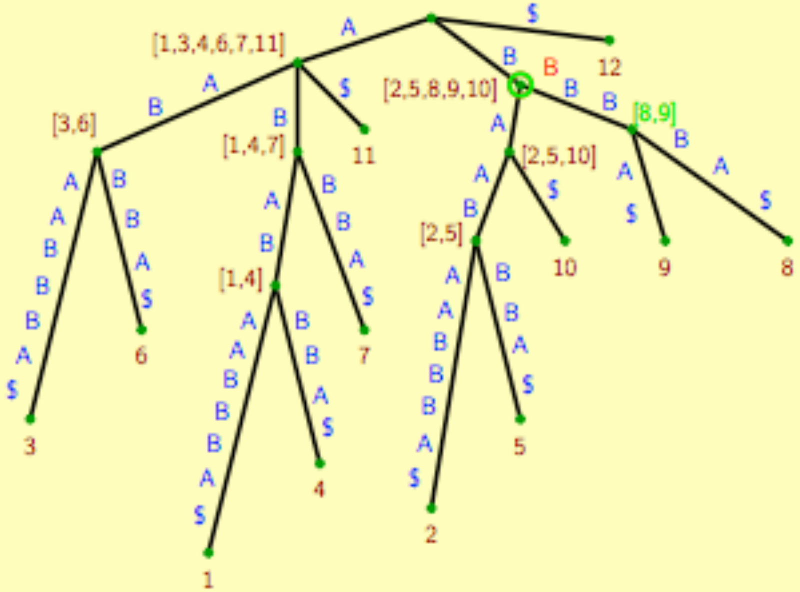
a. $(i, l, 2)$ is an occurrence of a branching tandem repeat

b. i and j occur in the same leaf-list of a node v in $T(S)$ with depth $D(v) = l$ and not in the same leaf-list of any node with depth $> l$

Example:

A	B	A	A	B	A	A	B	B	B	A	$
1	2	3	4	5	6	7	8	9	10	11	12

(9,1,2)
(8,1,2)



Basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $\alpha\alpha = L(v)L(v)$ occurs as a branching tandem repeat $(i, D(v), 2)$ ...

Algorithm:

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

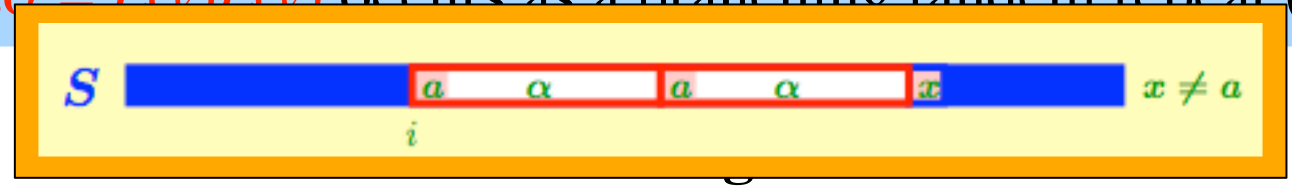
1. Select an unmarked internal node v.
   Mark v and execute steps 2a and 2b for node v.

2a. Collect the leaf-list $LL(v)$.

2b. For each leaf i in $LL(v)$, test whether the leaf $j = i + D(v)$ is in $LL(v)$.
If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

Basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $\alpha\alpha = L(v)L(v)$ occurs as a branching tandem repeat $(i, D(v), 2)$ ...



All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marl

1. Select an unmarked internal node v.
   Mark v and execute steps 2a and 2b fc

2a. Collect the leaf-list $LL(v)$.

2b. For each leaf i in $LL(v)$, test whether the leaf $j = i + D(v)$ is in $LL(v)$.
   If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

i and j are in leaf-lists of distinct children of v, i.e they are not in the same leaf-list for any node with depth > $D(v)$ ...

Basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $\alpha\alpha = L(v)L(v)$ occurs as a branching tandem repeat $(i, D(v), 2)$ ...

Algorithm:

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a and 2b for node v.

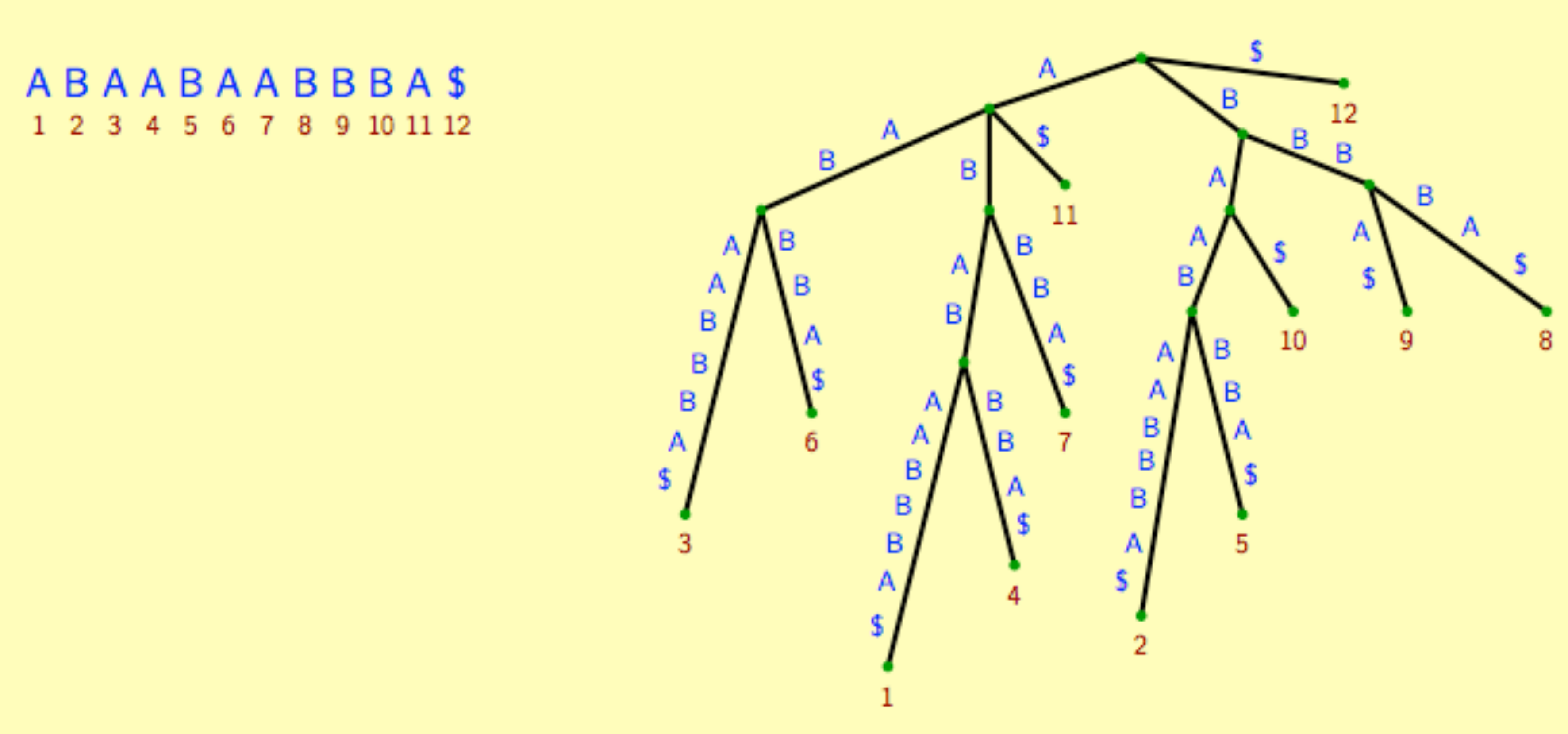
2a. Collect the leaf-list $LL(v)$.

2b. For each leaf i in $LL(v)$, test whether the leaf $j = i + D(v)$ is in $LL(v)$.
If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

Analysis: $O(n^2)$ time, and $O(n)$ space, if we can test in time $O(1)$ ....

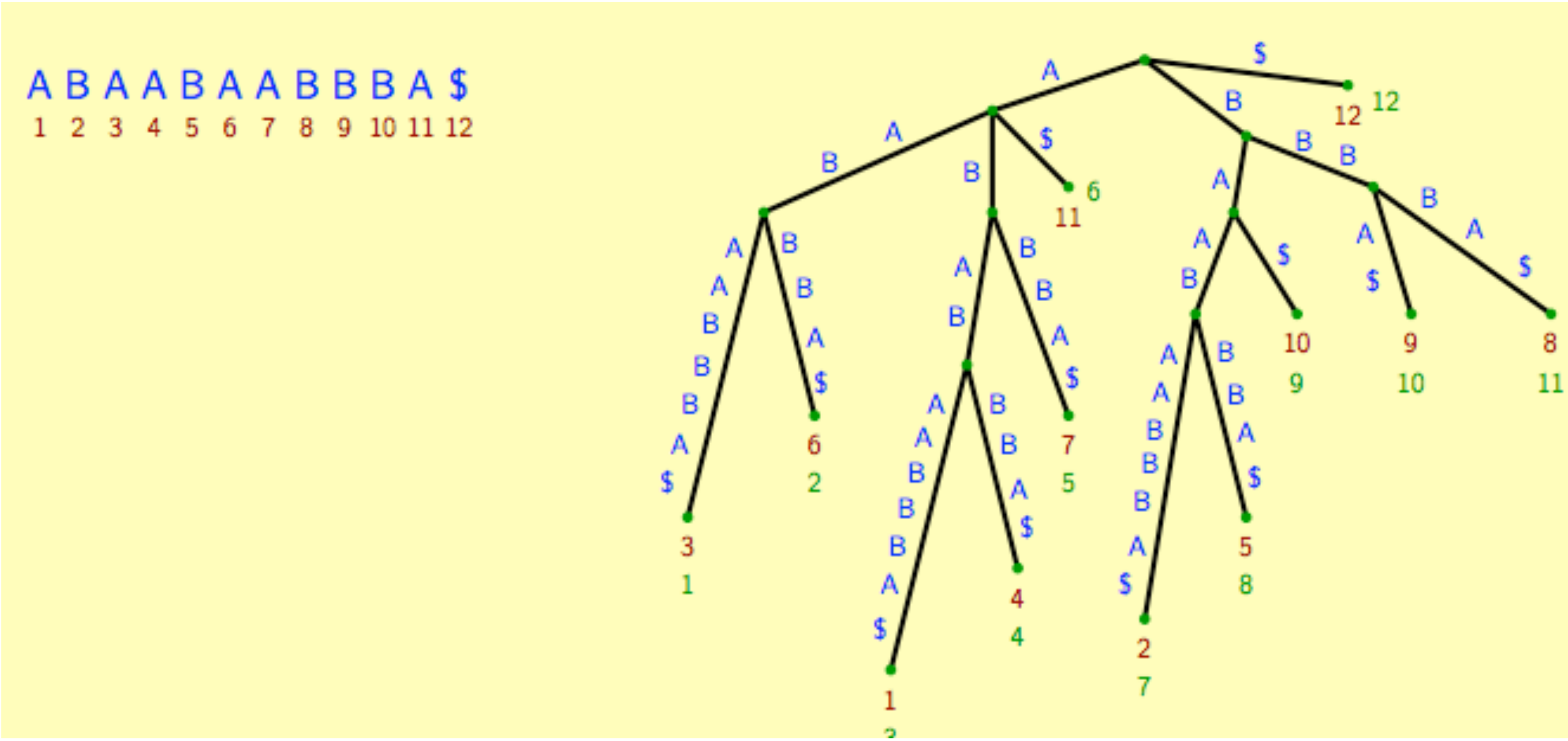
Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...



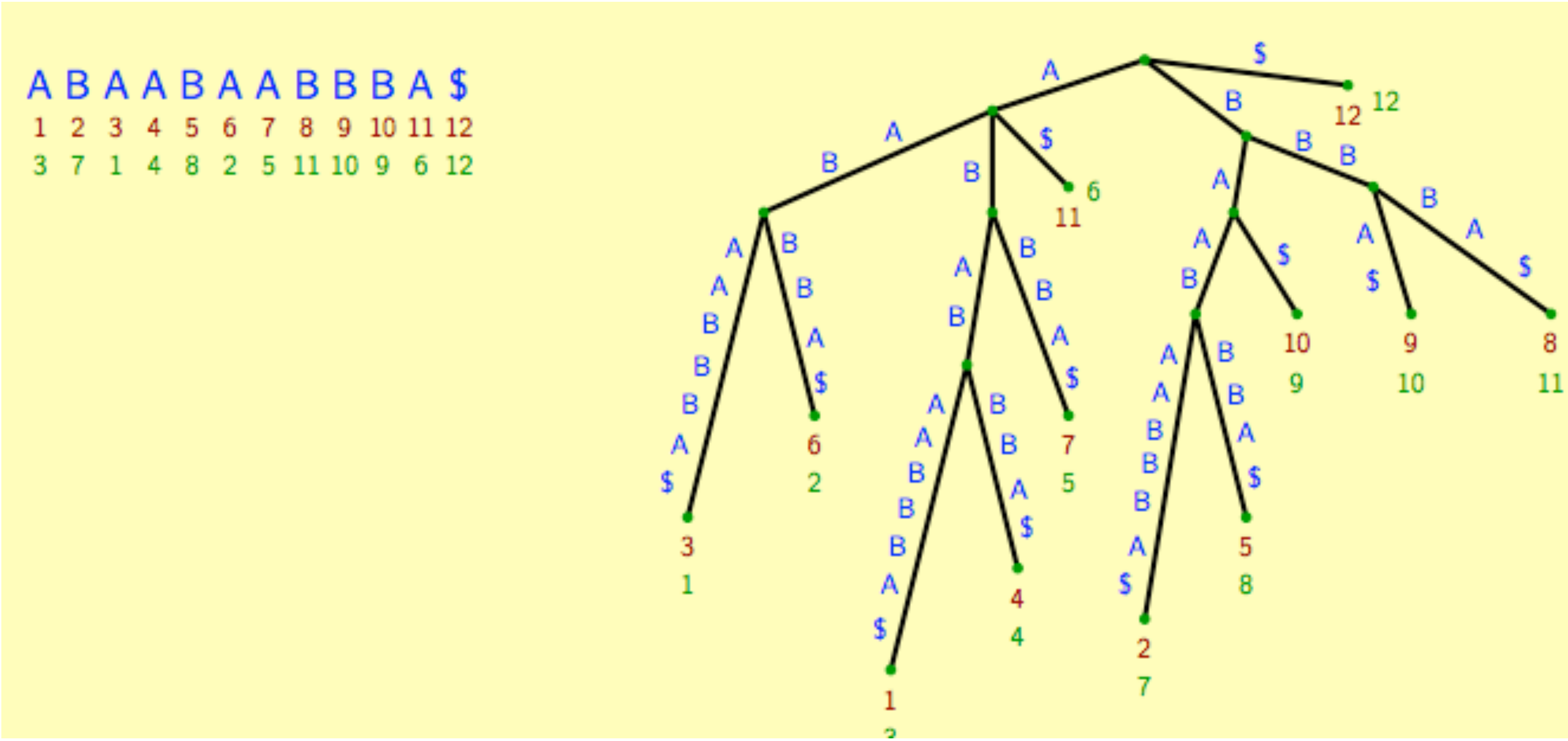
Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...



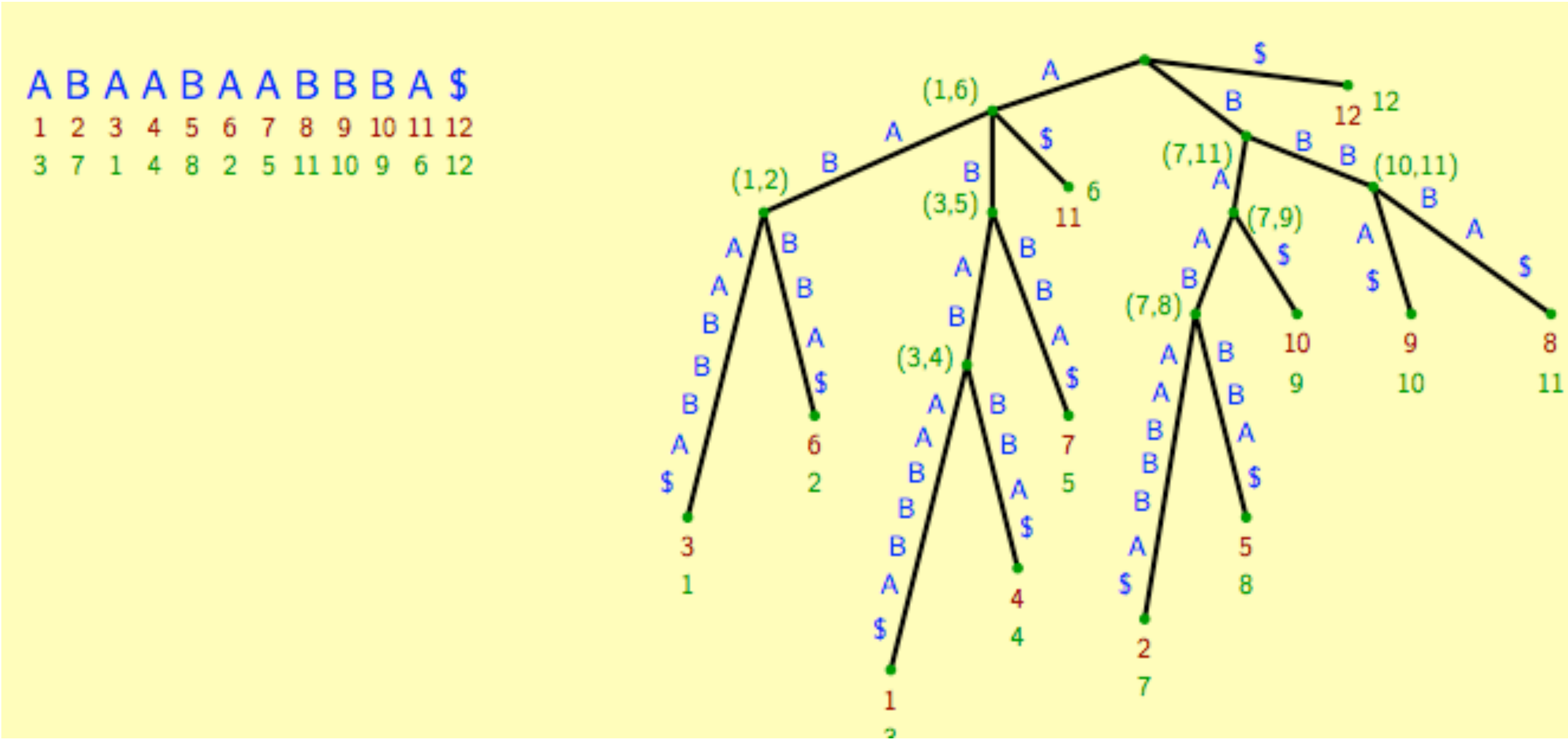
Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...



Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...



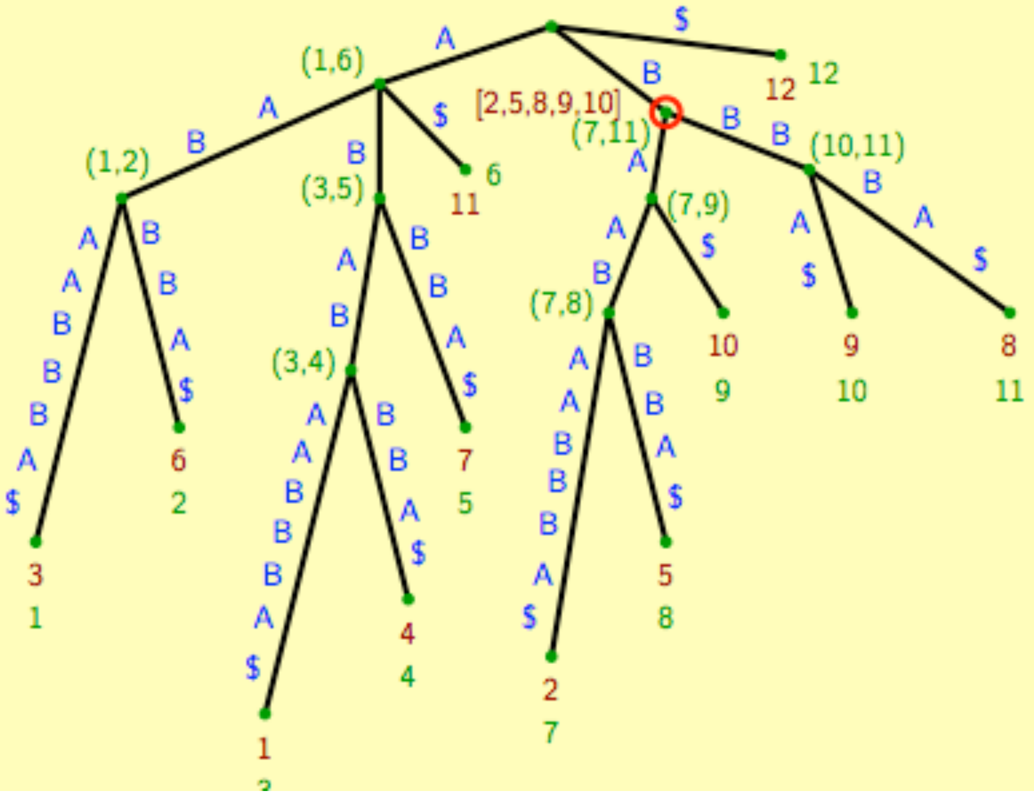
Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...

A	B	A	A	B	A	A	B	B	B	A	\$
1	2	3	4	5	6	7	8	9	10	11	12
3	7	1	4	8	2	5	11	10	9	6	12

B: $8 \xrightarrow{D=1} 9? \xrightarrow{table} 10 \in (7,11)$

but: $(8,9)$ not branching



Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...

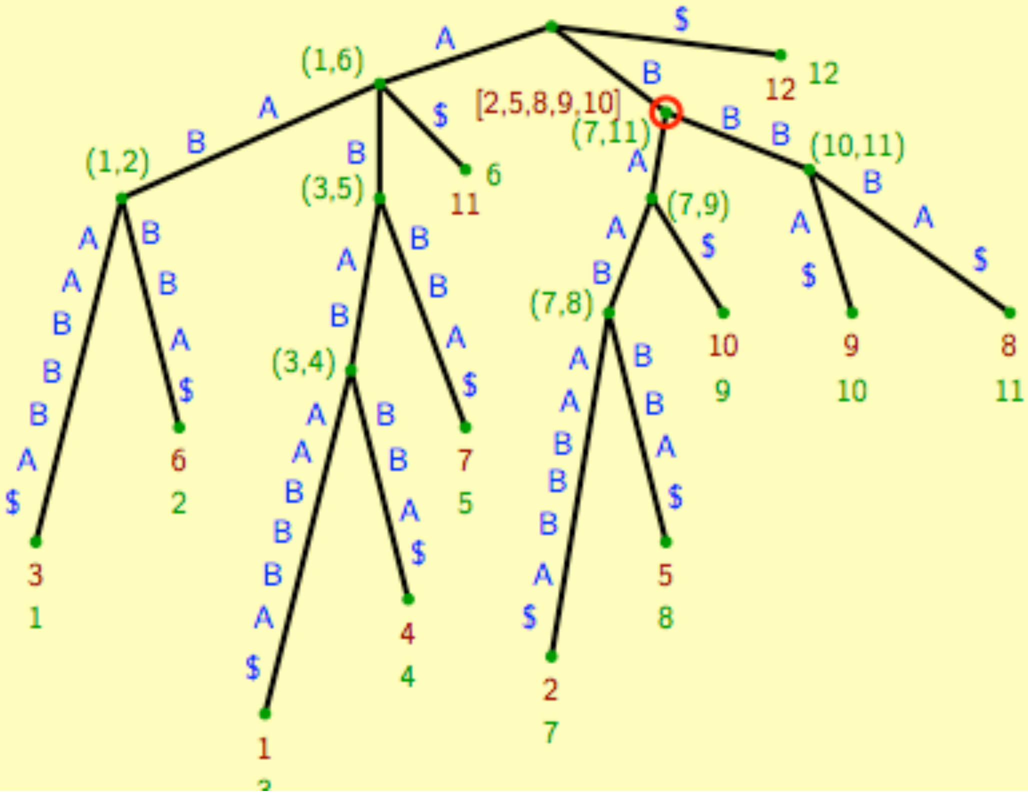
A	B	A	A	B	A	A	B	B	B	A	$
1	2	3	4	5	6	7	8	9	10	11	12
3	7	1	4	8	2	5	11	10	9	6	12

B: $8 \xrightarrow{D=1} 9? \xrightarrow{table} 10 \in (7,11)$

but: $(8,9)$ not branching

$9 \xrightarrow{D=1} 10? \xrightarrow{table} 9 \in (7,11)$

and: $(9,10)$ is branching



Testing in constant time

Idea: make a depth-first numbering of the leaves in $T(S)$ and a corresponding lookup-table ...

A	B	A	A	B	A	A	B	B	B	A	$
1	2	3	4	5	6	7	8	9	10	11	12
3	7	1	4	8	2	5	11	10	9	6	12

B: $8 \xrightarrow{D=1} 9? \xrightarrow{table} 10 \in (7,11)$

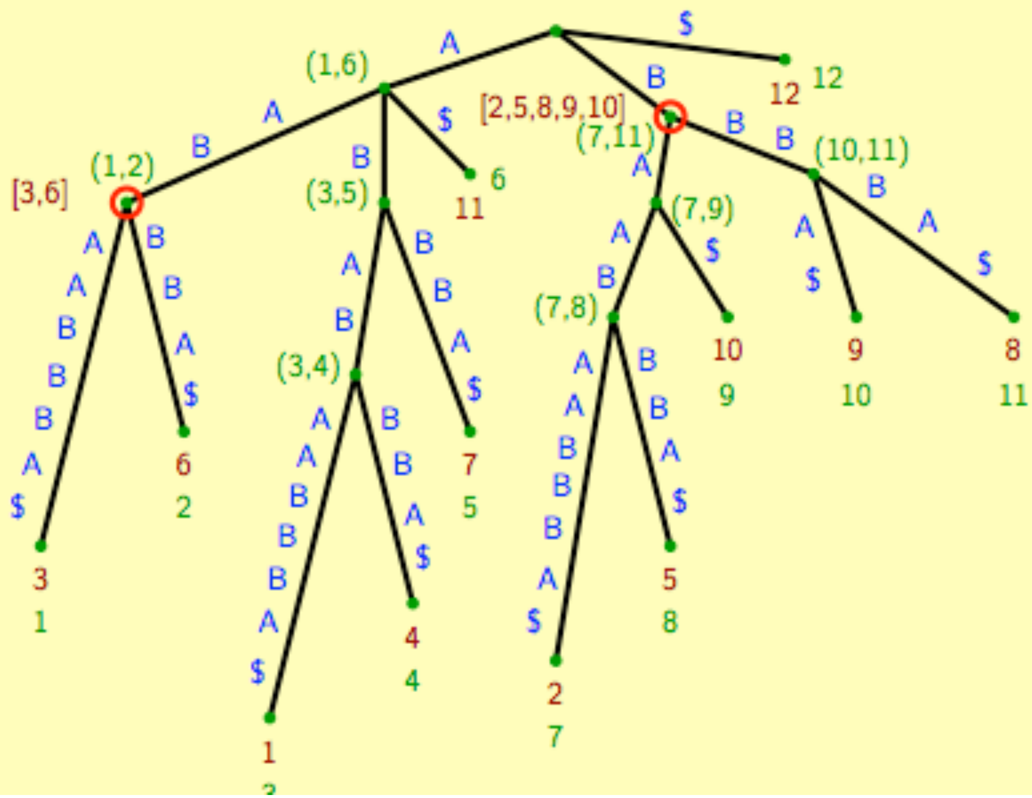
but: $(8,9)$ not branching

$9 \xrightarrow{D=1} 10? \xrightarrow{table} 9 \in (7,11)$

and: $(9,10)$ is branching

B: $3 \xrightarrow{D=3} 6? \xrightarrow{table} 2 \in (1,2)$

and: $(3,6)$ is branching



Speeding up the basic algorithm

We cannot afford to spend time O(|$LL(v)$|) at each node, ...

For each leaf i in $LL(v)$, test whether the leaf $j = i + D(v)$ is in $LL(v)$.
If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

Speeding up the basic algorithm

We cannot afford to spend time O(|$LL(v)$|) at each node, ...

For each leaf i in $LL(v)$, test whether the leaf $j = i + D(v)$ is in $LL(v)$.
If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

Recall:

Lemma: Consider two positions i and j of S, $1 \le i < j \le n$, let $l = j - i$.
Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a branching tandem repeat

b. i and j occur in the same leaf-list of a node v in $T(S)$ with depth $D(v) = l$ and not in the same leaf-list of any node with depth $> l$

Speeding up the basic algorithm

We cannot afford to spend time $O(|LL(v)|)$ at each node, ...

For each leaf i in $LL(v)$, test whether the leaf $j = i + D(v)$ is in $LL(v)$. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

Recall:

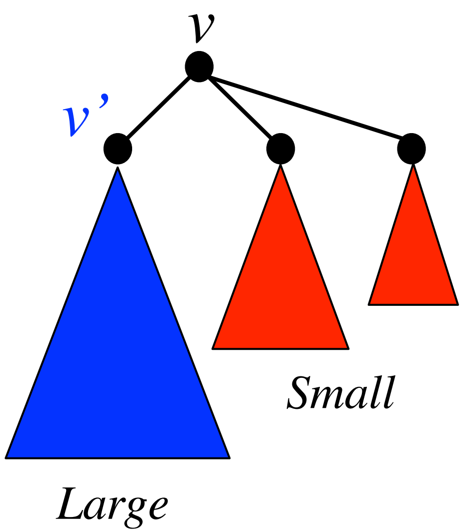
Lemma: Consider two positions i and j of S, $1 \leq i < j \leq n$, let $l = j - i$. Then the following assertions are equivalent:

a. $(i, l, 2)$ is an occurrence of a branching tandem repeat

b. i and j occur in the same leaf-list of a node v in $T(S)$ with depth $D(v) = l$ and not in the same leaf-list of any node with depth $> l$

... i and j must be in leaf-lists of different children of v ...

Speeding up the basic algorithm

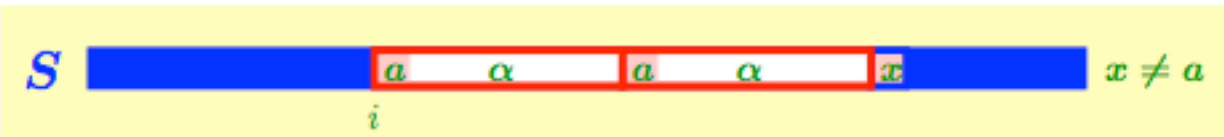


For node v let v' be the child of v with the largest leaf-list ...

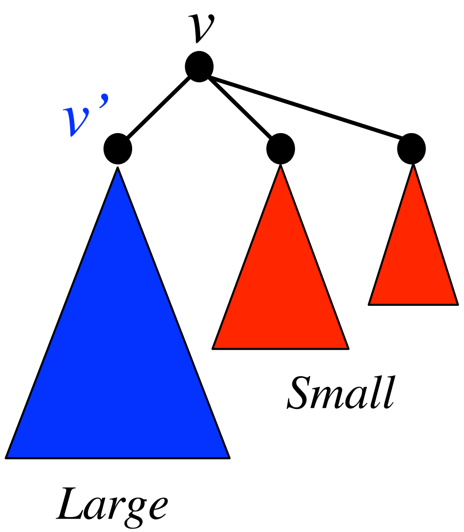
$Large(v) = LL(v')$

$Small(v) = LL'(v) = LL(v) - LL(v')$

Observation: There is no branching occurrence $(i, D(v), 2)$ where i and $j = i+D(v)$ are both in $Large(v)$, i.e. either i or $i+D(v)$ are in $Small(v)$ ...



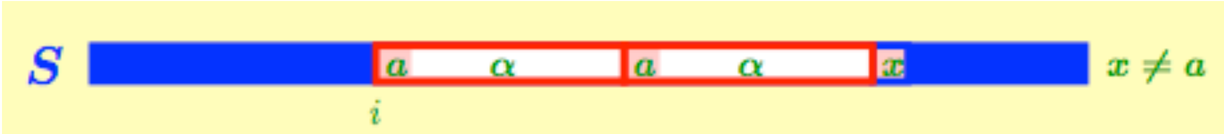
Speeding up the basic algorithm



For node v let v' be the child of v with the largest leaf-list ...

$Large(v) = LL(v')$

$Small(v) = LL'(v) = LL(v) - LL(v')$



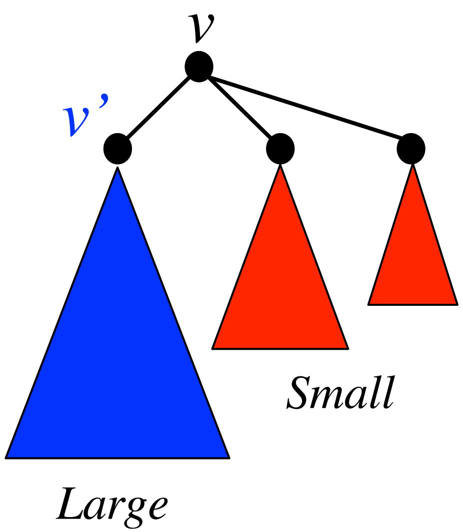
Observation: There is no branching occurrence $(i, D(v), 2)$ where i and $j = i+D(v)$ are both in $Large(v)$, i.e. either i or $i+D(v)$ are in $Small(v)$ ...

Idea: For every i in $Small(v)$ report:

$(i, D(v), 2)$ as branching iff $i+D(v) \in LL(v)$ and $\mathbf{S}[i] \neq \mathbf{S}[i+2D(v)]$

$(i\text{-}D(v), D(v), 2)$ as branching iff $i\text{-}D(v) \in LL(v')$ and $\mathbf{S}[i\text{-}D(v)] \neq \mathrm{S}[i+D(v)]$

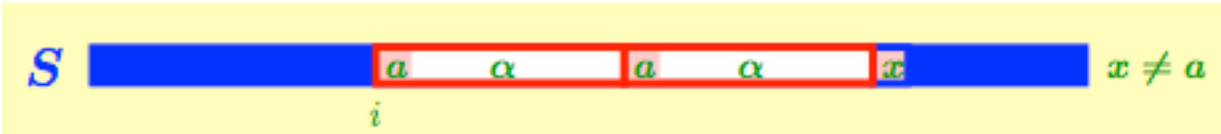
Speeding up the basic algorithm



For node v let v' be the child of v with the largest leaf-list ...

$Large(v) = LL(v')$

$Small(v) = LL'(v) = LL(v) - LL(v')$



Observation: There is no branching occurrence $(i, D(v), 2)$ where i and $j = i+D(v)$ are both in $Large(v)$, i.e. either i or $i+D(v)$ are in $Small(v)$

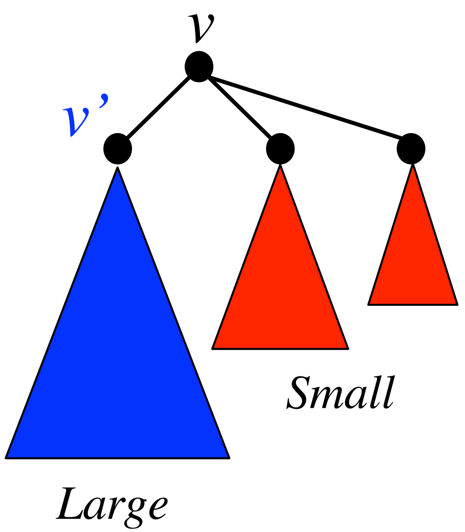
This finds all branching occurrences of $L(v)L(v)$ at positions in $Small(v)$

Idea: For every i in $Small(v)$ report:

$(i, D(v), 2)$ as branching iff $i+D(v) \in LL(v)$ and $\mathbf{S}[i] \neq \mathbf{S}[i+2D(v)]$

$(i-D(v), D(v), 2)$ as branching iff $i-D(v) \in LL(v')$ and $\mathbf{S}[i-D(v)] \neq S[i+D(v)]$

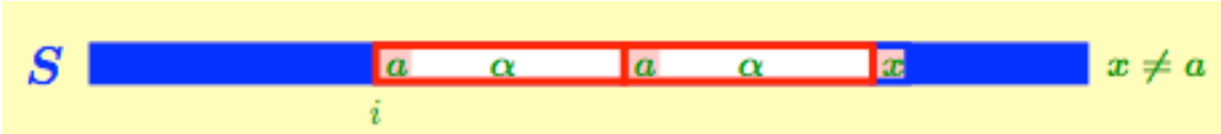
Speeding up the basic algorithm



For node v let v' be the child of v with the largest leaf-list ...

$Large(v) = LL(v')$

$Small(v) = LL'(v) = LL(v) - LL(v')$



Observation: There is no branching occurrence $(i, D(v), 2)$ where i and $j = i+D(v)$ are both in $Large(v)$, i.e. either i or $i+D(v)$ are in $Small(v)$ ...

Idea: For every i in $Small(v)$ report:
$(i, D(v), 2)$ as branching iff $i+D(v) \in$

This finds all branching occurrences of $L(v)L(v)$ at positions in $Large(v)$

$(i\text{-}D(v), D(v), 2)$ as branching iff $i\text{-}D(v) \in LL(v')$ and $\mathbf{S}[i\text{-}D(v)] \neq \mathrm{S}[i+D(v)]$

Optimized basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $(i, D(v), 2)$ is a branching occurrence by examining *Small*(v) ...

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a, 2b and 2c for node v.

2a. Collect the list $LL'(v)$ for v.

2b. For each leaf i in $LL'(v)$, test whether leaf $j = i + D(v)$ is in $LL(v)$, the leaf-list of v. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

2c. Do the same test for each leaf j in $LL'(v)$, and $i = j - D(v)$.

Optimized basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $(i, D(v), 2)$ is a branching occurrence by examining *Small*(v) ...

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a, 2b and 2c for node v.

2a. Collect the list $LL'(v)$ for v.

2b. For each leaf i in $LL'(v)$, test whether leaf $j = i + D(v)$ is in $LL(v)$, the leaf-list of v. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

2c. Do the same test for each leaf j in $LL'(v)$, and $i = j - D(v)$.

Optimized basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $(i, D(v), 2)$ is a branching occurrence by examining *Small*(v) ...

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a, 2b and 2c for node v.

2a. Collect the list $LL'(v)$ for v.

2b. For each leaf i in $LL'(v)$, test whether leaf $j = i + D(v)$ is in $LL(v)$, the leaf-list of v. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

2c. Do the same test for each leaf j in $LL'(v)$, and $i = j - D(v)$.

i.e test if i is in $LL(v')$, i.e. *Large*(v), and if $S[i] \neq S[i+2D(v)]$ ...

Optimized basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $(i, D(v), 2)$ is a branching occurrence by examining *Small*(v) ...

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a, 2b and 2c for node v.

2a. Collect the list $LL'(v)$ for v.

2b. For each leaf i in $LL'(v)$, test whether leaf $j = i + D(v)$ is in $LL(v)$, the leaf-list of v. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

2c. Do the same test for each leaf j in $LL'(v)$, and $i = j - D(v)$.

Running time?

Optimized basic algorithm

Idea: for each node v of $T(S)$, find the positions i in $LL(v)$ where $(i, D(v), 2)$ is a branching occurrence by examining *Small*(v) ...

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a, 2b and 2c for node v.

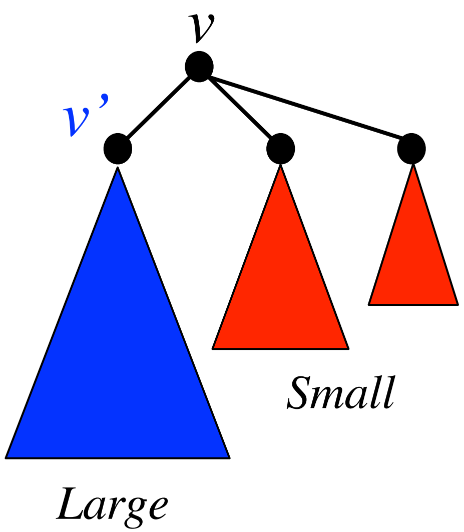
2a. Collect the list $LL'(v)$ for v.

2b. For each leaf i in $LL'(v)$, test whether leaf $j = i + D(v)$ is in $LL(v)$, the leaf-list of v. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

2c. Do the same test for each leaf j in $LL'(v)$, and $i = j - D(v)$.

Analysis: O(|*Small*(v)|) time at each node, and O(n) space

"Smaller half" trick



For node v let v' be the child of v with the largest leaf-list ...

$Large(v) = LL(v')$

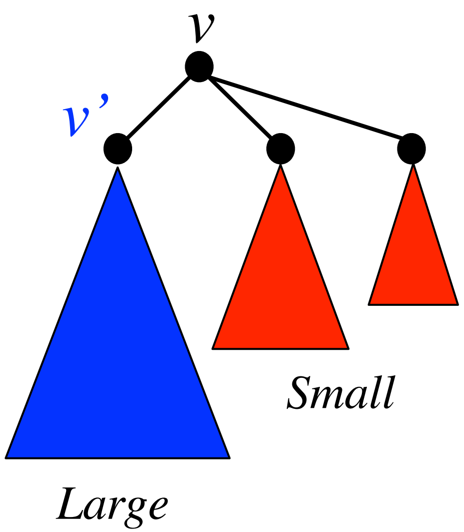
$Small(v) = LL'(v) = LL(v) - LL(v')$

"Smaller half" trick:

$$\sum_v |Small(v)| = \mathrm{O}(n \log n)$$

Using time $\mathrm{O}(|Small(v)|)$ at each node, implies time $\mathrm{O}(n \log n)$ in total

“Smaller half” trick



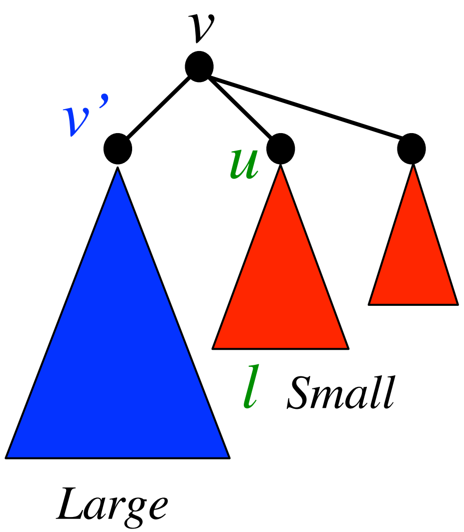
Proof: count how many times each leaf can be in *Small*(v) for any v ...

“Smaller half” trick:

$$\sum_{v} \left| Small(v) \right| = \mathrm{O}(n \log n)$$

Using time O(|*Small*(v)|) at each node, implies time O(n log n) in total

“Smaller half” trick



Proof: count how many times each leaf can be in *Small*(v) for any v ...

If leaf l is in *Small*(v), then $|T(u)| \leq |T(v)|/2$, otherwise $T(v')$ wouldn’t be large …

I.e. leaf l can be in a *Small*(v) at most log(n) times a long the path from l to the root.

“Smaller half” trick:

$$\sum_v |Small(v)| = \mathrm{O}(n \log n)$$

Using time O(|*Small*(v)|) at each node, implies time O(n log n) in total

Optimized basic algorithm

Idea: for each node v of $T(S)$, determine if $\alpha\alpha = L(v)L(v)$ is a branching tandem repeat by examining $Small(v)$ ...

All nodes of $T(S)$ begin unmarked.
Step 1 is repeated until all nodes are marked.

1. Select an unmarked internal node v.
   Mark v and execute steps 2a, 2b and 2c for node v.

2a. Collect the list $LL'(v)$ for v.

2b. For each leaf i in $LL'(v)$, test whether leaf $j = i + D(v)$ is in $LL(v)$, the leaf-list of v. If so, test whether $S[i] \neq S[i + 2D(v)]$. There is a branching tandem repeat of length $2D(v)$ starting at position i if and only if both tests return true.

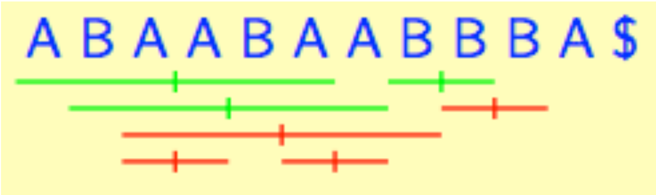
2c. Do the same test for each leaf j in $LL'(v)$, and $i = j - D(v)$.

Analysis: O(n log n) time, and O(n) space

Putting it all together

Algorithm: Start at each occurrence of a branching tandem repeats, and do a series of consecutive left-rotations to find all occurrences of tandem repeats ...

Example:

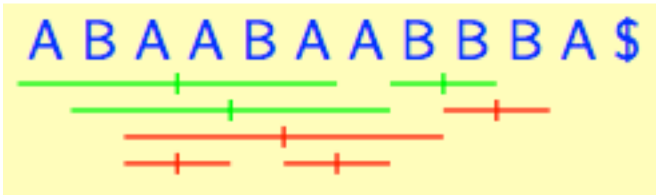


Analysis: $O(n \log n + |\text{output}|)$ time, and $O(n)$ space

Putting it all together

Algorithm: Start at each occurrence of a branching tandem repeats, and do a series of consecutive left-rotations to find all occurrences of tandem repeats ...

Example:



Analysis: O(n log n + |output|) time, and O(n) space

The algorithm can be extended to find all occurrences of primitive tandem repeats in time O(n log n) ...

Implementation details

Make "depth first"-numbering of leaves, lookup-table, and annotation of internal with interval in an initial depth-first traversal of $T(S)$ ...

Report (or count) occurrences of tandem repeats in a depth-first traversal of $T(S)$, report from v when all children have been reported from ...

Note that the annotation of nodes with intervals makes it easy to determine $|LL(v)|$ and v'...

Keep track of leaf-lists in e.g. linked lists which can be concatenated in time O(1) ... or do you need explicitly to keep track of the leaf-lists at all? (Hint: You don't)