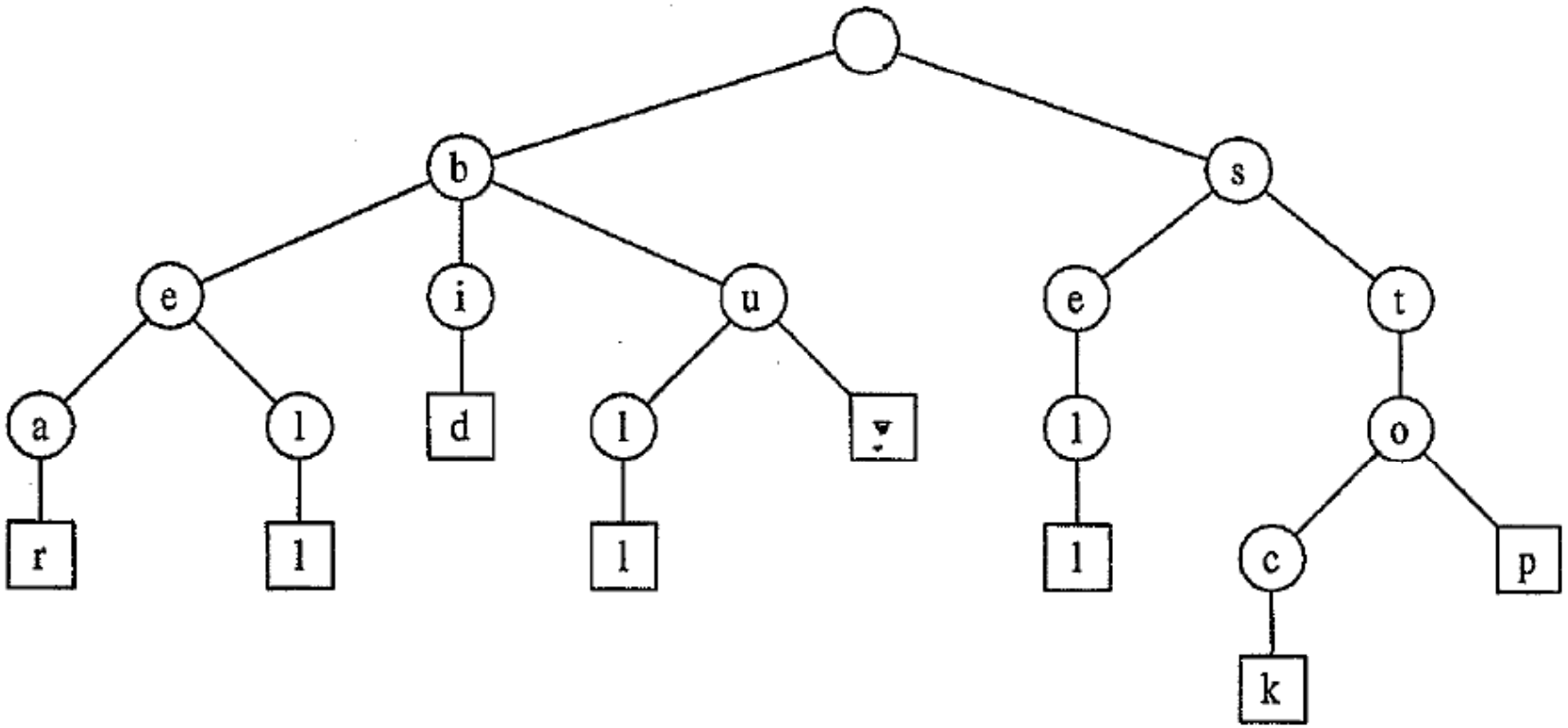
Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

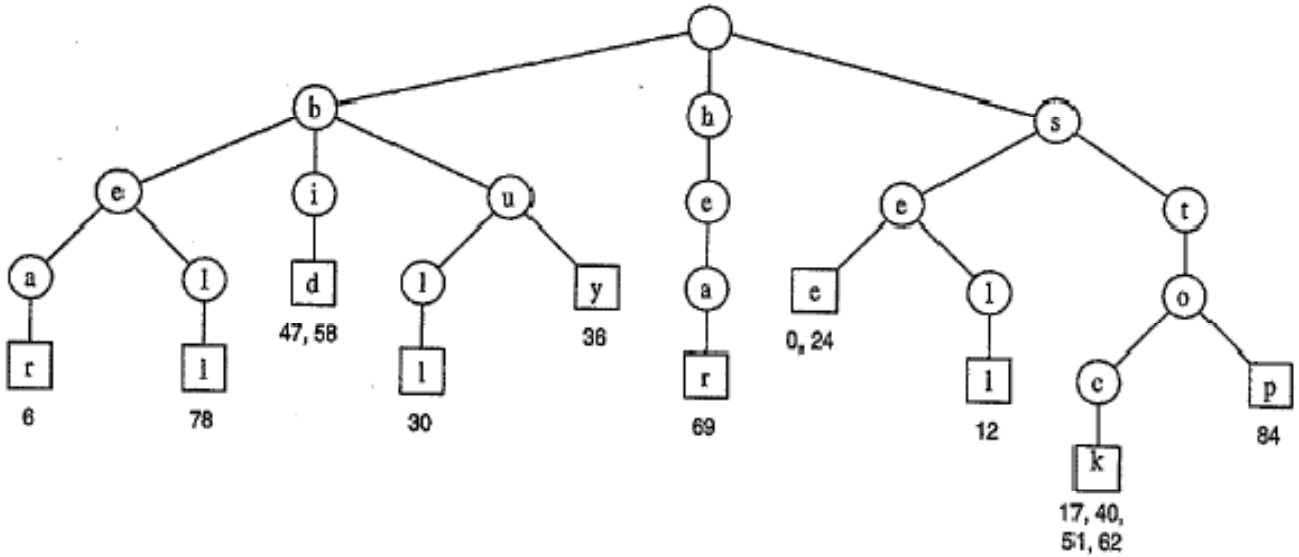
Suffix træer [GT, kapitel 9.2], Suffix arrays [Smyth, kapitel 5.3.2]

AARHUS UNIVERSITET

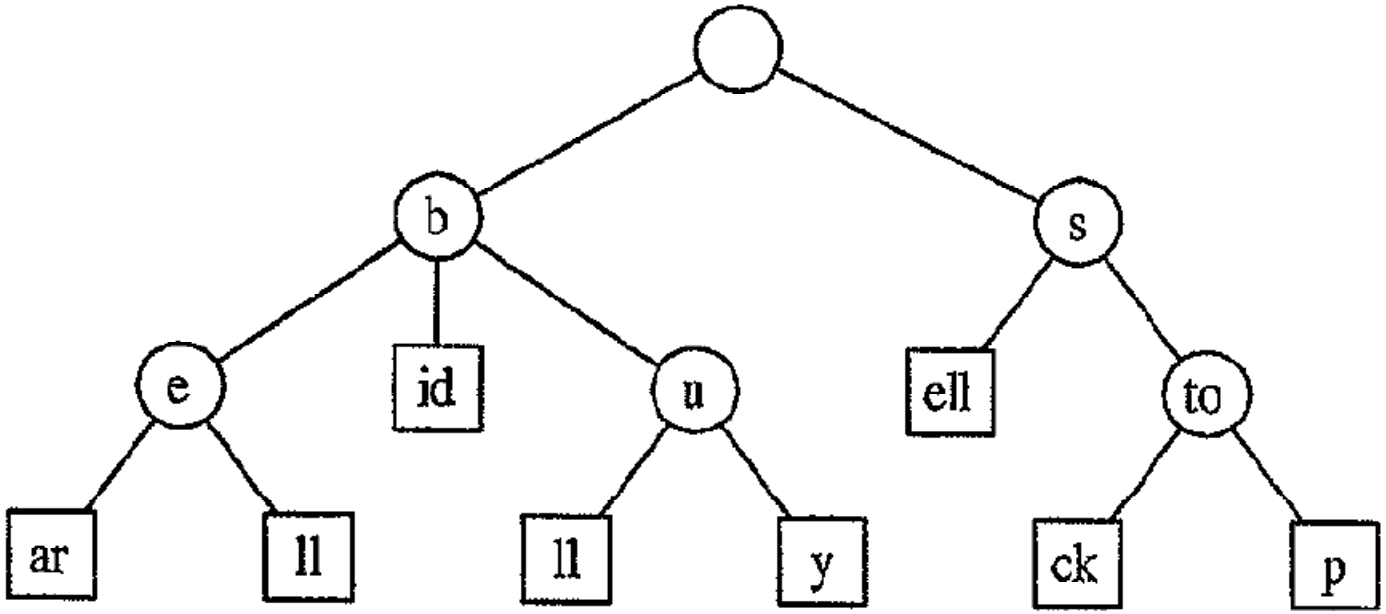
Trier



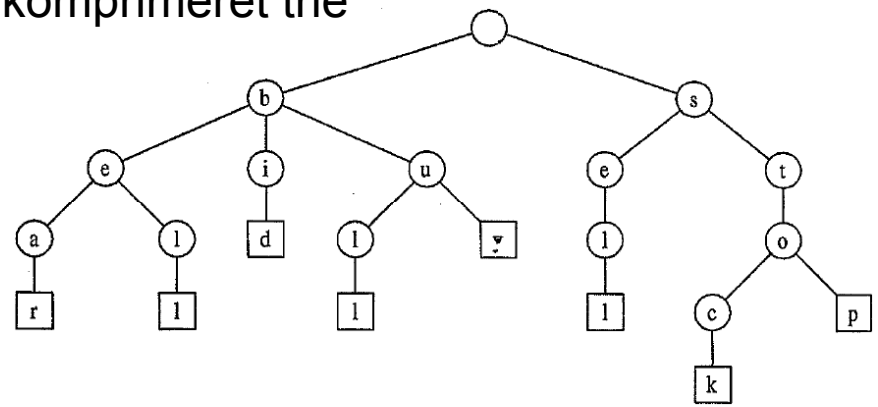
Søgning i Streng v.h.a.Trie



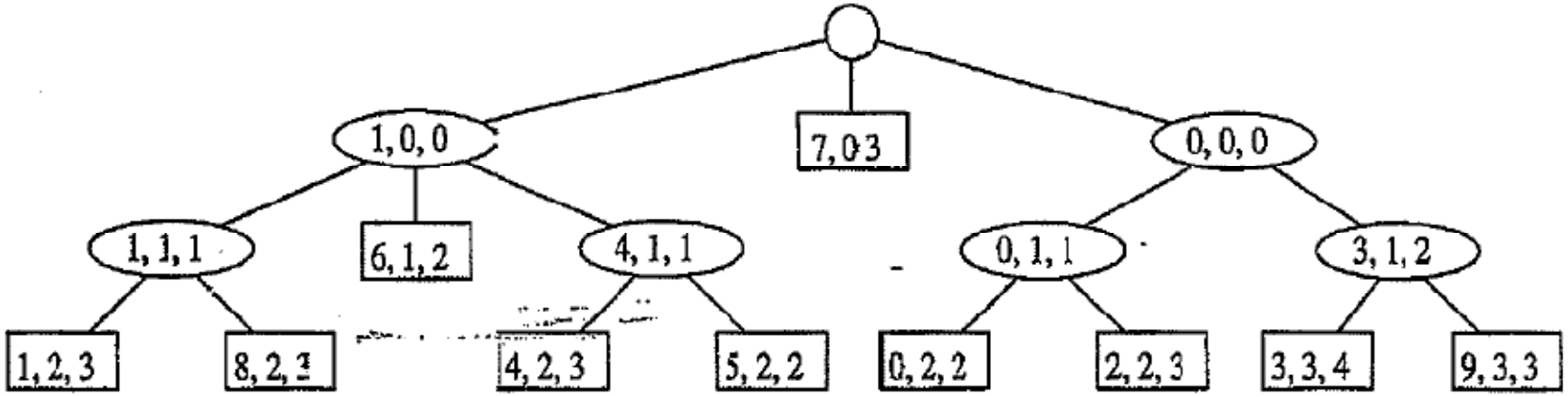
Komprimeret Trie



Ukomprimeret trie



Komprimeret Trie over Ordbog



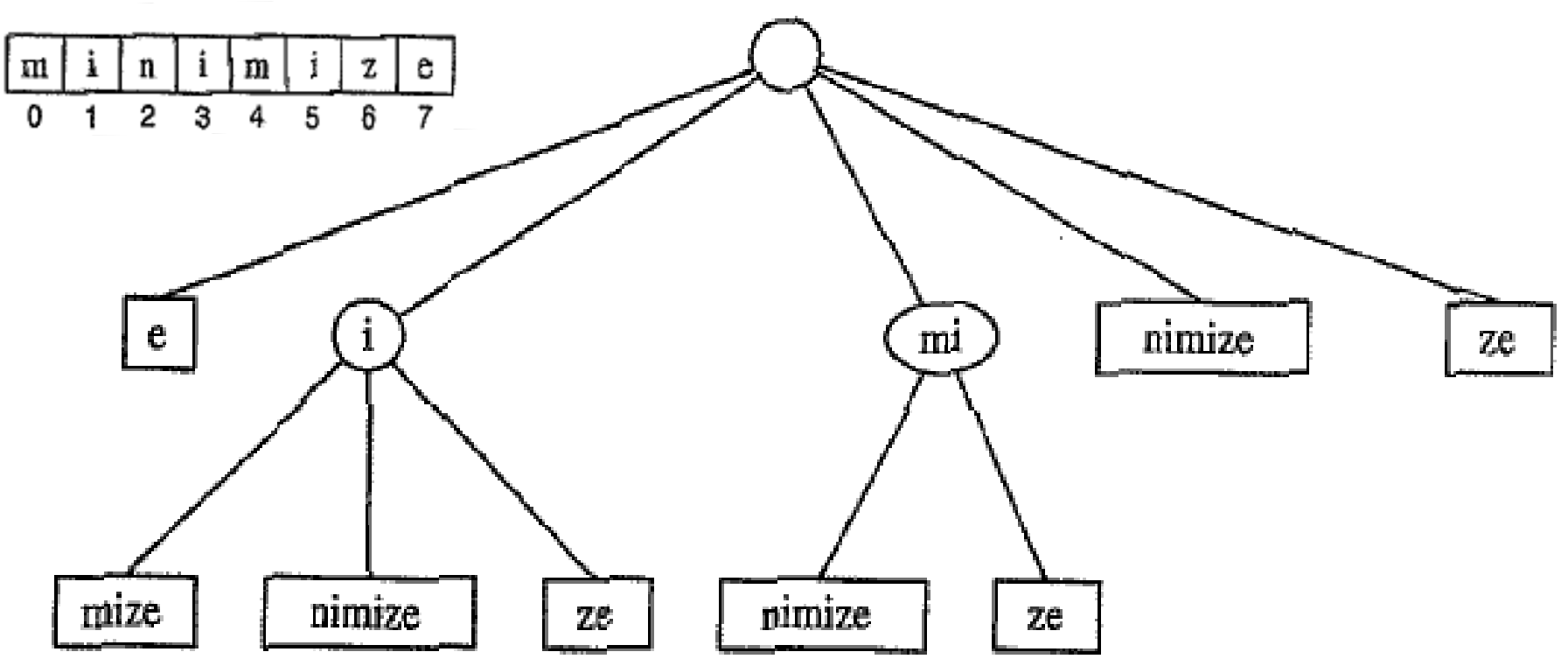
	0	1	2	3	4
$S[0]$ =	s	e	e		
$S[1]$ =	b	e	a	r	
$S[2]$ =	s	e	l	l	
$S[3]$ =	s	t	o	c	k

	0	1	2	3
$S[4]$ =	b	u	l	l
$S[5]$ =	b	u	y	
$S[6]$ =	b	i	d	

	0	1	2	3
$S[7]$ =	h	e	a	r
$S[8]$ =	b	e	l	l
$S[9]$ =	s	t	o	p

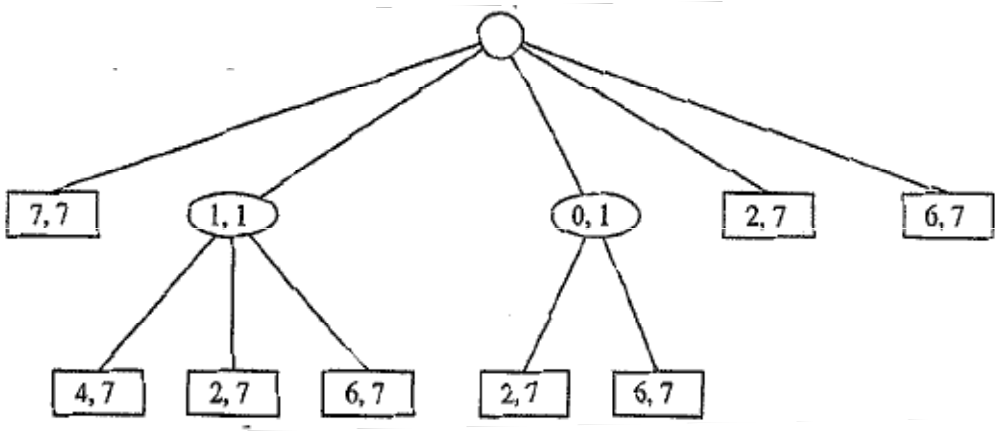
(i,j,k) er delstrengen $S[i][j..k]$

Suffix Træer



Suffix træ = komprimeret trie over suffixer

Plads $O(n)$

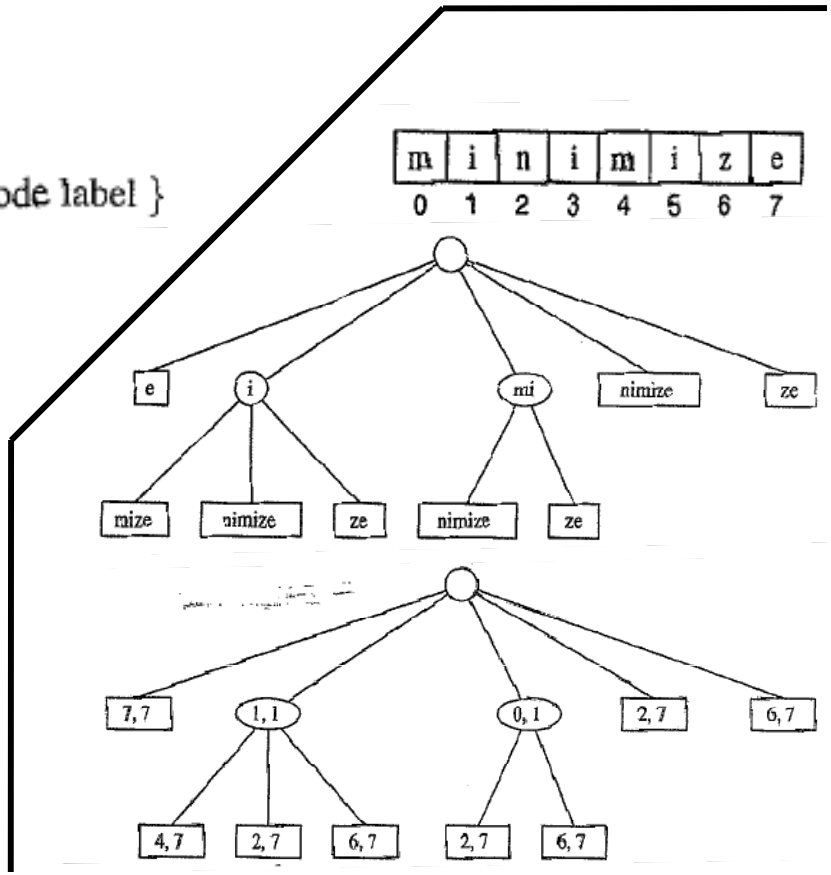


Algorithm suffixTrieMatch(T, P):

Input: Compact suffix trie T for a text X and pattern P

Output: Starting index of a substring of X matching P or an indication that P is not a substring of X

```
p ← P.length() { length of suffix of the pattern to be matched }
j ← 0 { start of suffix of the pattern to be matched }
v ← T.root()
repeat
    f ← true { flag indicating that no child was successfully processed }
    for each child w of v do
        i ← start(w)
        if P[j] = X[i] then
            { process child w }
            x ← end(w) − i + 1
            if p ≤ x then
                { suffix is shorter than or of the same length of the node label }
                if P[j..j+p−1] = X[i..i+p−1] then
                    return i − j { match }
                else
                    return "P is not a substring of X"
            else
                { suffix is longer than the node label }
                if P[j..j+x−1] = X[i..i+x−1] then
                    p ← p − x { update suffix length }
                    j ← j + x { update suffix start index }
                    v ← w
                    f ← false
                    break out of the for loop
until f or T.isExternal(v)
return "P is not a substring of X"
```



Suffix Array

1 2 3 4 5 6 7 8
tekst = *a b a a b a a b*

Suffixer

1	*a b a a b a a b*
2	*b a a b a a b*
3	*a a b a a b*
4	*a b a a b*
5	*b a a b*
6	*a a b*
7	*a b*
8	*b*

Sorterede suffixer

6	*a a b*
3	*a a b a a b*
7	*a b*
4	*a b a a b*
1	*a b a a b a a b*
8	*b*
5	*b a a b*
2	*b a a b a a b*

Suffix array

σ	6	3	7	4	1	8	5	2

Algorithm SANaïve [Smyth, s.151]

– *Naïvely use a suffix array to locate u in x*

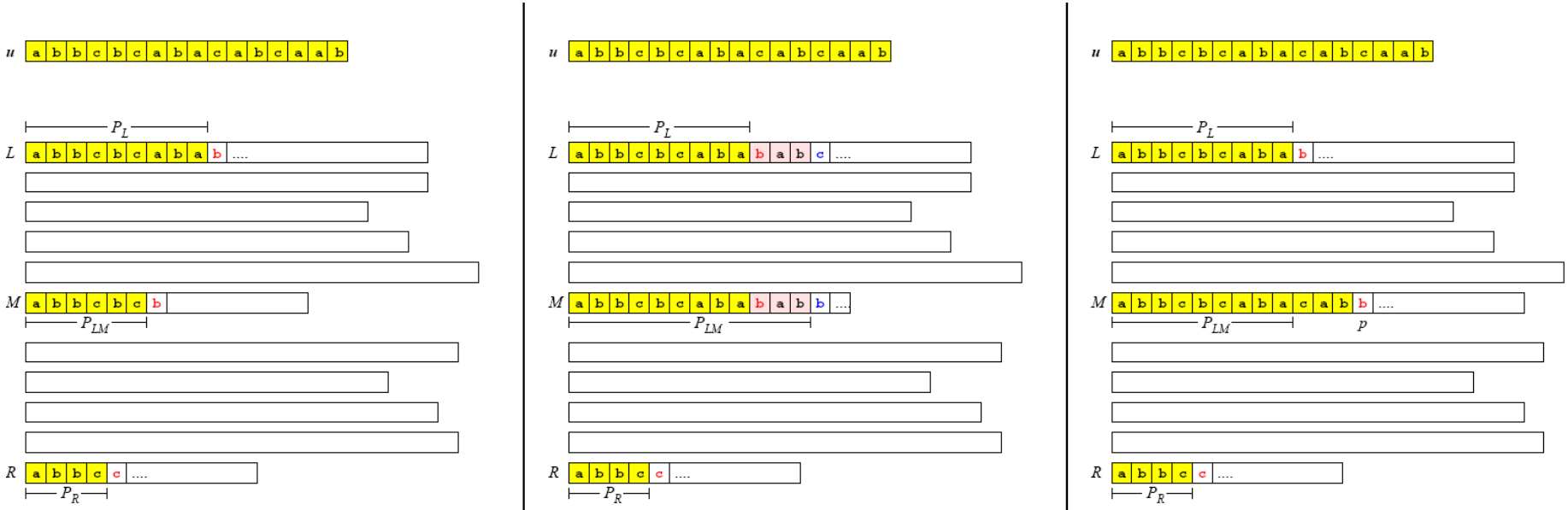
$j \leftarrow 0$; $L \leftarrow 0$; $R \leftarrow n+1$
repeat
  $M \leftarrow \lceil (R+L)/2 \rceil$
  if $u = x[\sigma[M]..\sigma[M]+m-1]$ **then**
    $j \leftarrow \sigma[M]$
  elsif $u > x[\sigma[M]..\sigma[M]+m-1]$ **then**
    $L \leftarrow M$
  else
    $R \leftarrow M$
until $L = R\text{-}1$ or $j \neq 0$

Algorithm SASimple [Smyth, s.151]

– *Simply use a suffix array to locate u in x*

$j \leftarrow 0;\ L \leftarrow 0\ ;\ R \leftarrow n{+}1$
$P_L \leftarrow 0;\ P_R \leftarrow 0$
repeat
  $P \leftarrow \min\{P_L, P_R\}$
  $M \leftarrow \lceil (R+L)/2 \rceil$
  – *Compute* $\mathrm{lcp}(u, \sigma[M])$
  $P_M \leftarrow P + \mathrm{lcp}\left(u[P+1..m], x[\sigma[M]+P..n]\right)$
  if $P_M = m$ **then**
    $j \leftarrow \sigma[M]$
  elsif $u[P_M+1] > x[\sigma[M]+P_M]$ **then**
    $L \leftarrow M;\ P_L \leftarrow P_M$
  else
    $R \leftarrow M;\ P_R \leftarrow P_M$
until $L = R{-}1$ or $j \neq 0$

SAComplex ($P_L \geq P_R$)



præberegnet

$P_{LM} < P_L$: $R \leftarrow M$, $P_R \leftarrow P_{LM}$

$P_{LM} > P_L$: $L \leftarrow M$

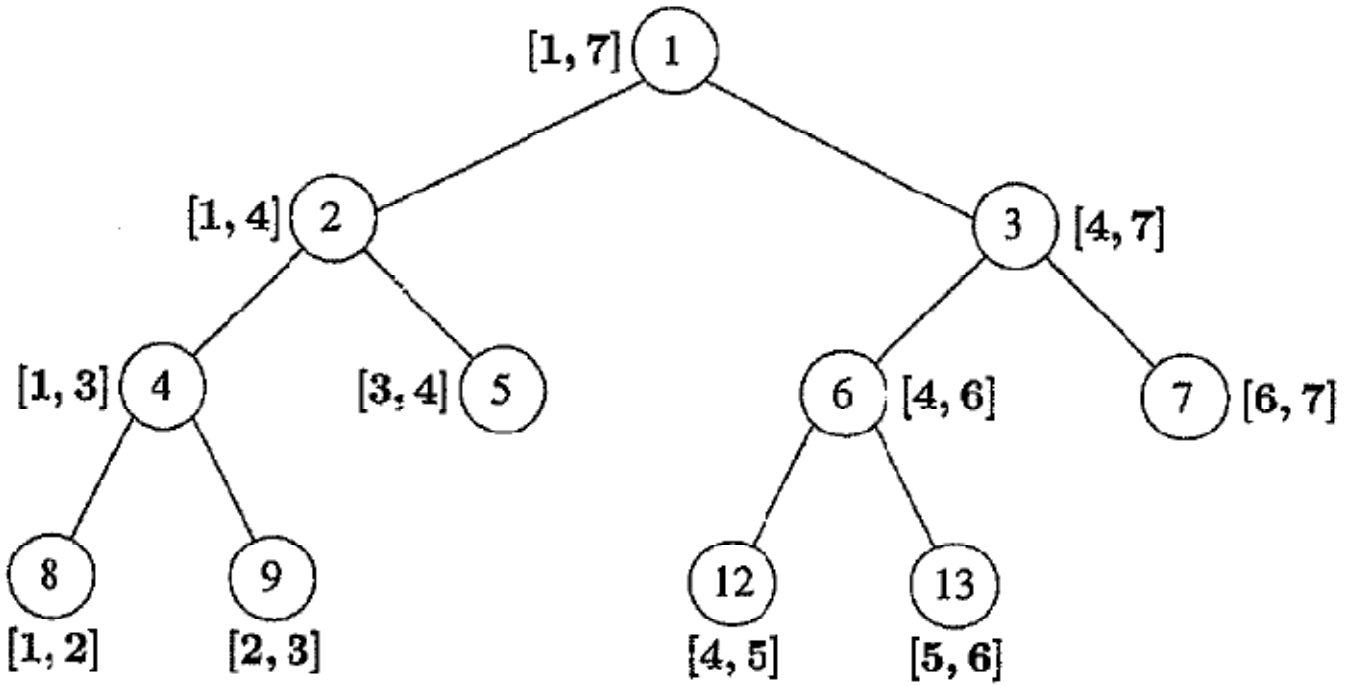
$P_{LM} = P_L$: **Start sammenligning på position** $P_{LM}+1$

Lad p **være første forskellige position:**

$u[p] < \sigma[M][p]$: $R \leftarrow M$, $P_R \leftarrow p\text{-}1$

$u[p] > \sigma[M][p]$: $L \leftarrow M$, $P_L \leftarrow p\text{-}1$

Binært træ over intervaller



Mihai Pătraşcu

Søgninger i et Suffix Array

Algorithm	Additional storage (bytes)	Theoretical time bound
SANaïve	$n \log n/8$	$O(m(\log n + k))$
SASimple	$n \log n/8$	$O(m(\log n + k))$
SAComplex	$4n \log n/8$	$O(m + \log n + k)$

n = tekst længde, m = mønster længde, k = antal forekomster