

Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

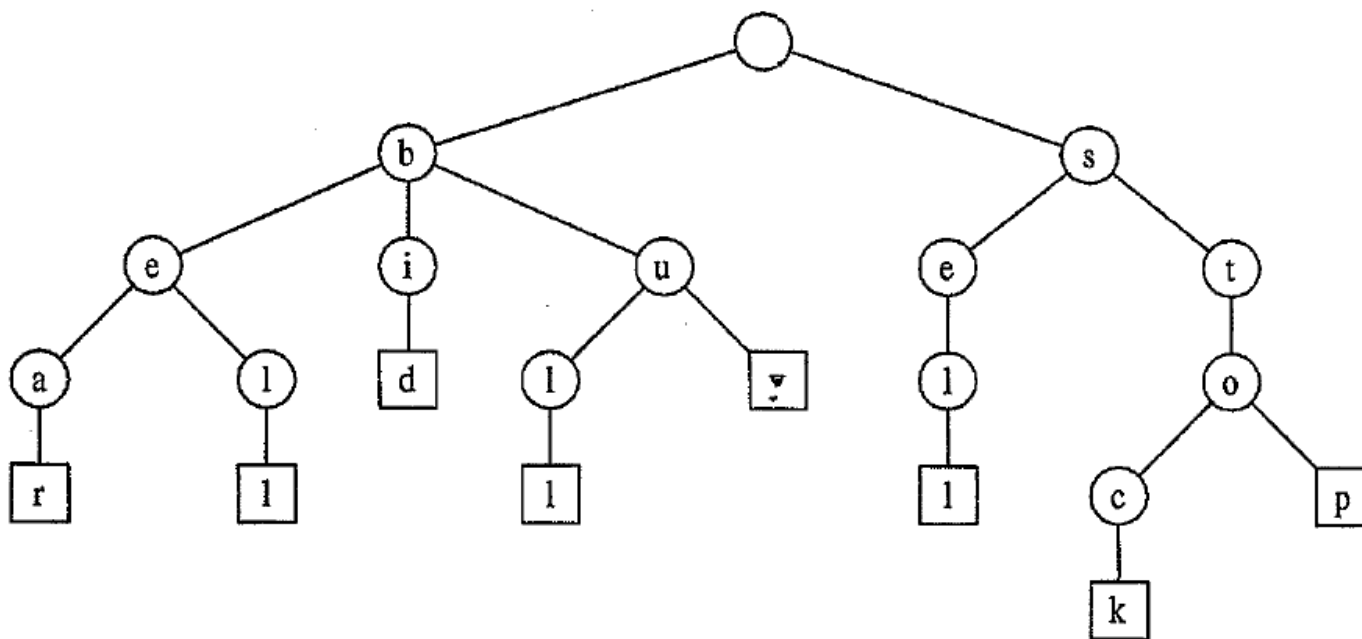
Suffiks træer [GT, kapitel 9.2], Suffiks arrays [Smyth, kapitel 5.3.2]



Trier

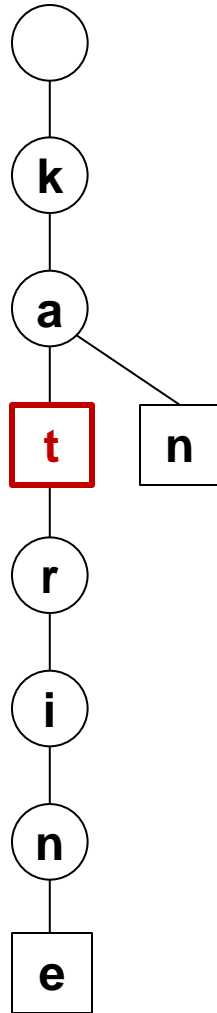
(text retrieval)

bear
bell
bid
bull
buy
sell
stock
stop

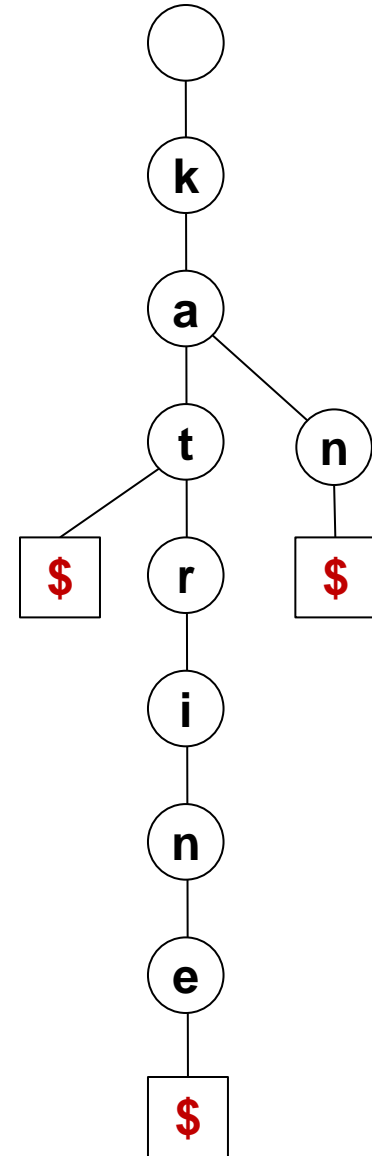


Præfiks-fri ?

kan
kat
katrine



kan\$
kat\$
katrine\$

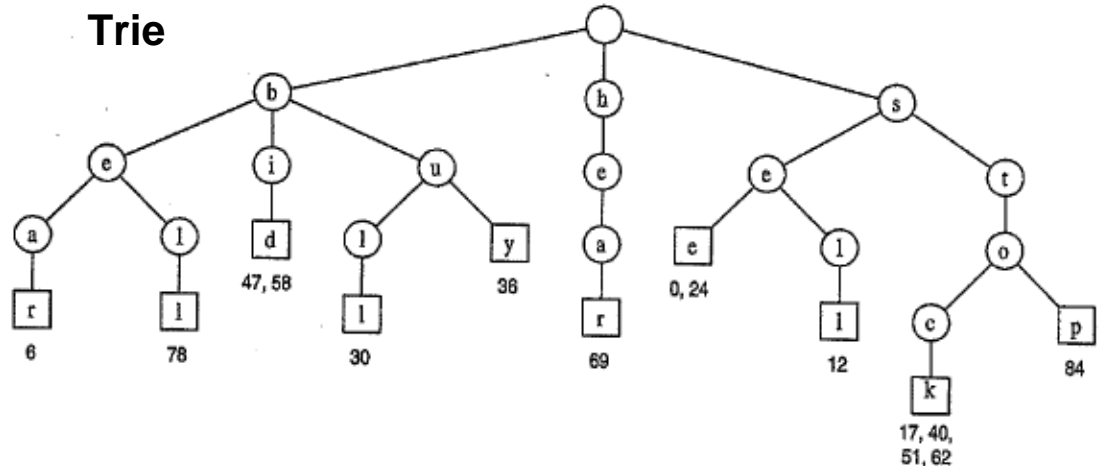


Søgning i Streng

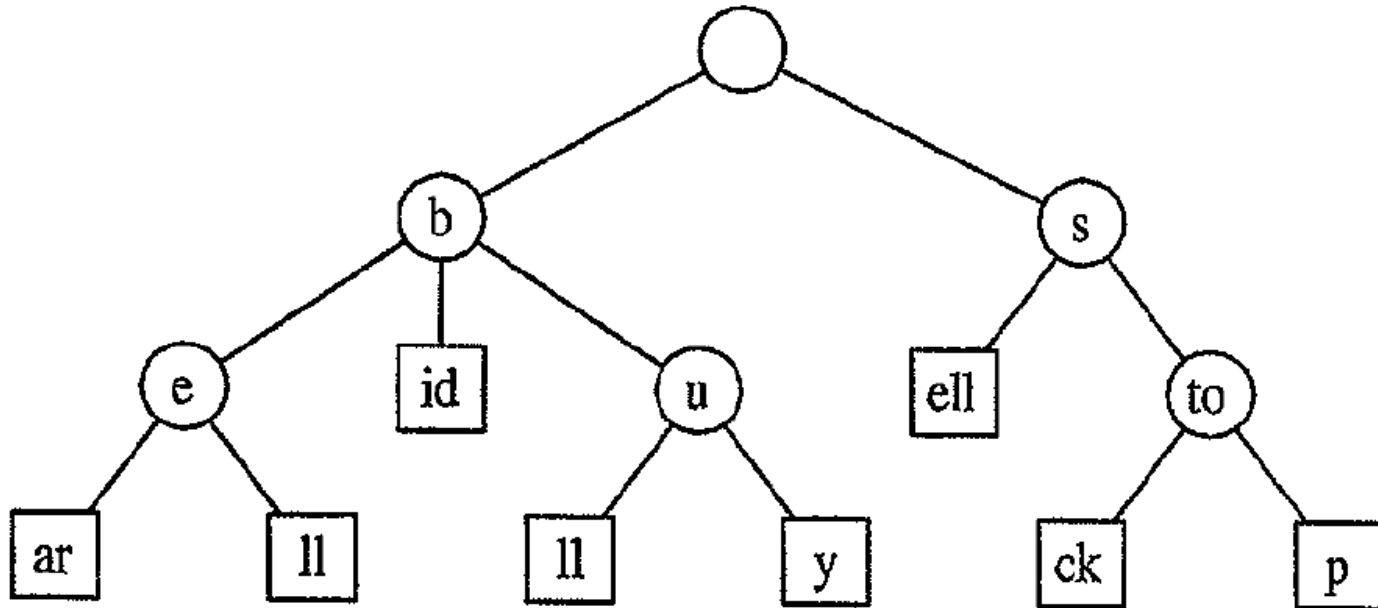
s	e	e		a		b	e	a	r	?		s	e	l	l		s	t	o	c	k	!	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
s	e	e		a		b	u	l	l	?		b	u	y		s	t	o	c	k	!		
24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	
b	i	d		s	t	o	c	k	!		b	i	d		s	t	o	c	k	!			
47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68		
h	e	a	r		t	h	e		b	e	l	l	?		s	t	o	p	!				
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88				

Inverteret fil

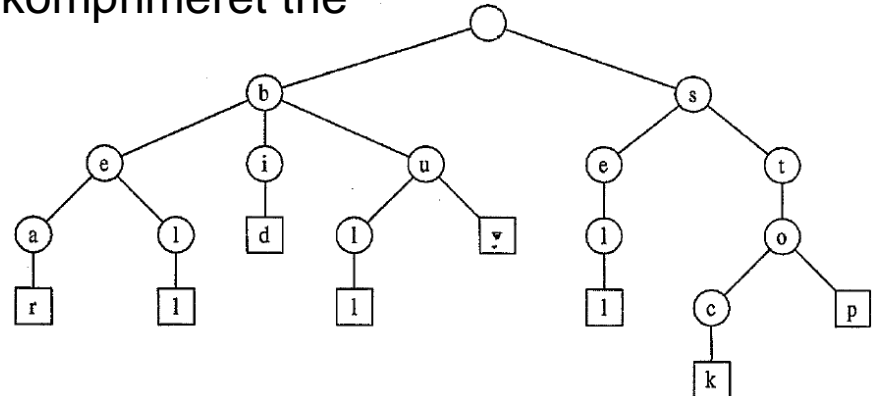
Ord	Position(er)
a	4,28
bear	6
bell	78
bid	47,58
bull	30
buy	36
hear	69
see	0,24
sell	12
stock	17,40,51,62
stop	84
the	74



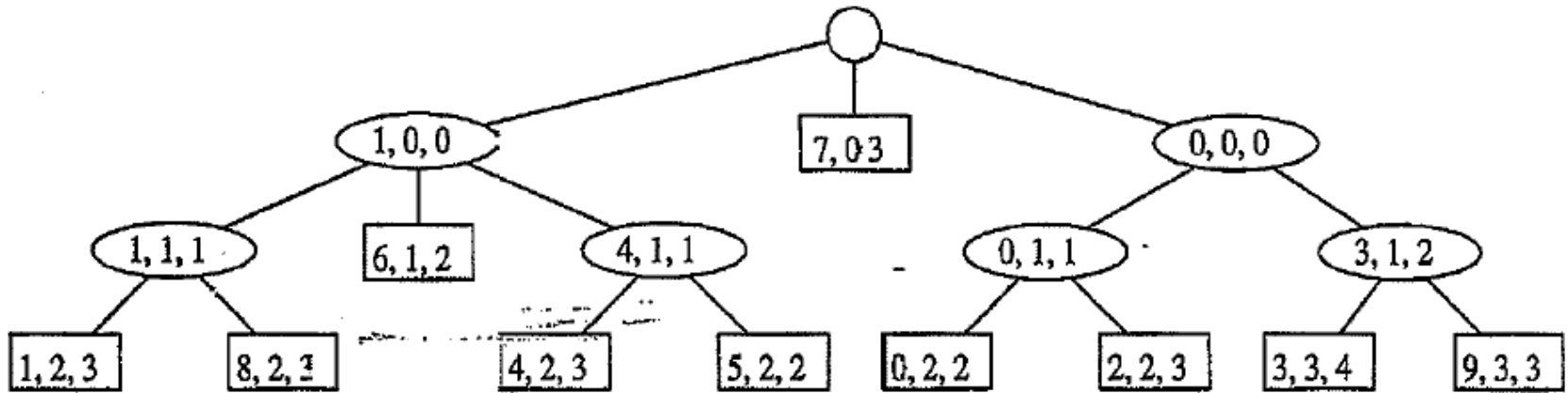
Komprimeret Trie



Ukomprimeret trie



Komprimeret Trie over Ordbog



$S[0] =$

c	1	2	3	4
s	e	e		

$S[1] =$

t	e	a	r
---	---	---	---

$S[2] =$

s	e	l	l
---	---	---	---

$S[3] =$

s	t	o	c	k
---	---	---	---	---

$S[4] =$

0	1	2	3
b	u	l	l

$S[5] =$

b	u	y
---	---	---

$S[6] =$

b	i	d
---	---	---

$S[7] =$

0	1	2	3
h	e	a	r

$S[8] =$

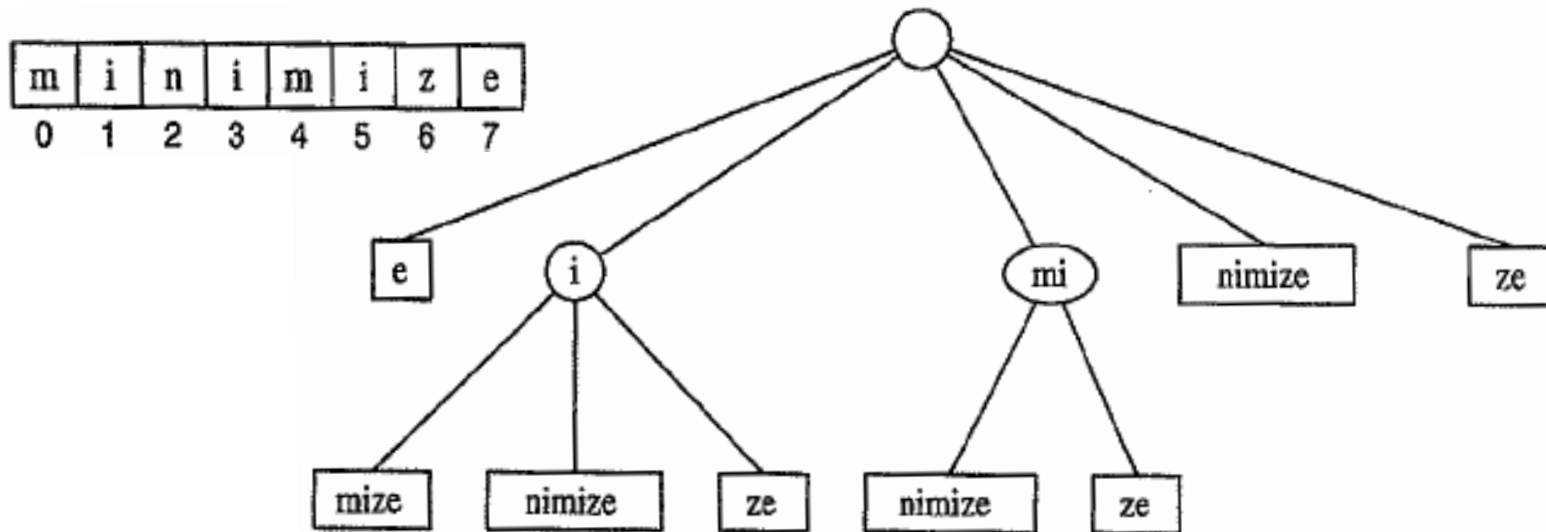
b	e	l	l
---	---	---	---

$S[9] =$

s	t	o	p
---	---	---	---

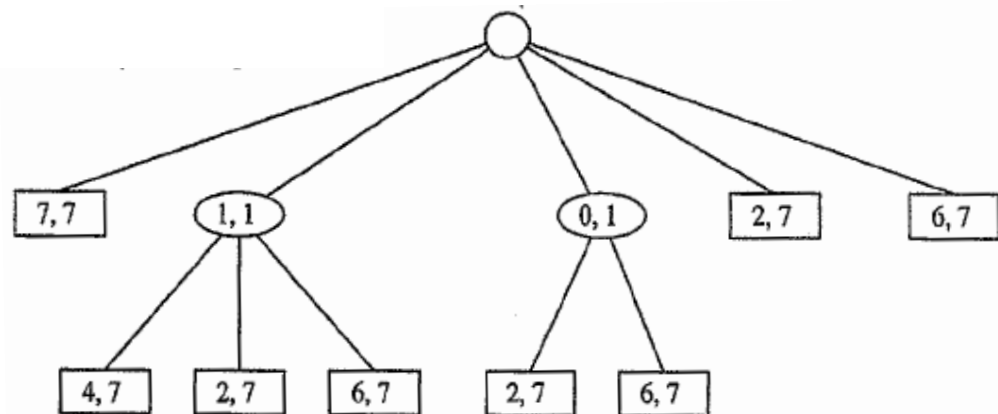
(i,j,k) er delstrengen $S[i][j..k]$

Suffiks Træer



Suffix træ =
komprimeret trie
over suffixer

Plads $O(n)$



Kan konstrueres i $O(n)$ tid (for konstant størrelse alfabet)

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

$p \leftarrow P.length()$ { length of suffix of the pattern to be matched }
 $j \leftarrow 0$ { start of suffix of the pattern to be matched }
 $v \leftarrow T.root()$

repeat

$f \leftarrow \mathbf{true}$ { flag indicating that no child was successfully processed }

for each child w of v do

$i \leftarrow \text{start}(w)$

if $P[j] = X[i]$ then

{ process child w }

$x \leftarrow \text{end}(w) - i + 1$

if $p \leq x$ 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

$p \leftarrow p - x$ { update suffix length }

$j \leftarrow j + x$ { update suffix start index }

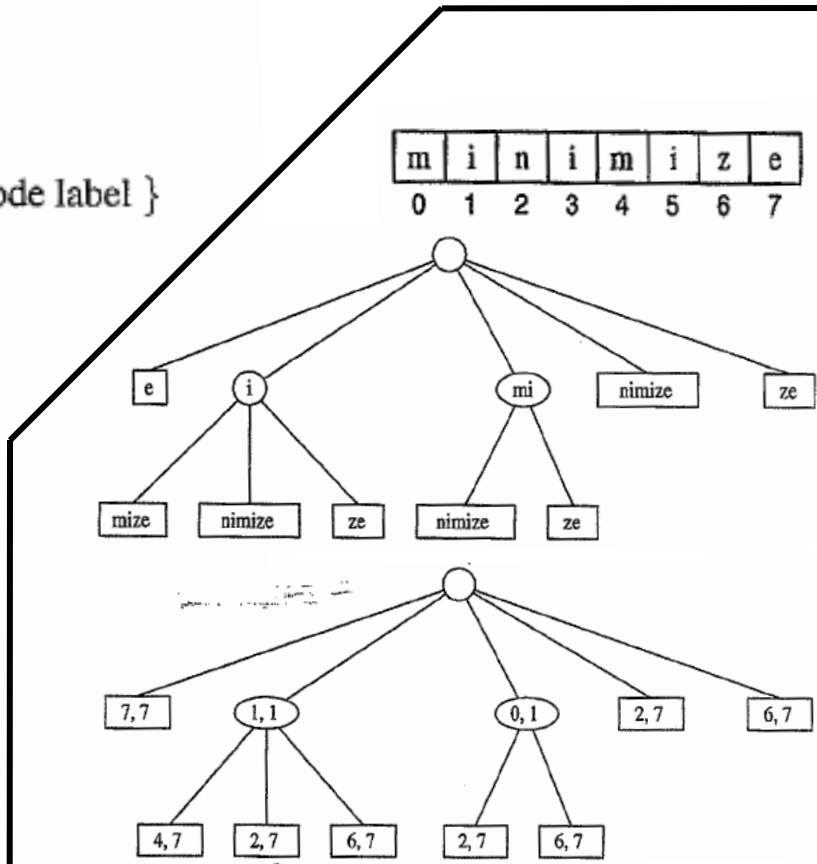
$v \leftarrow w$

$f \leftarrow \mathbf{false}$

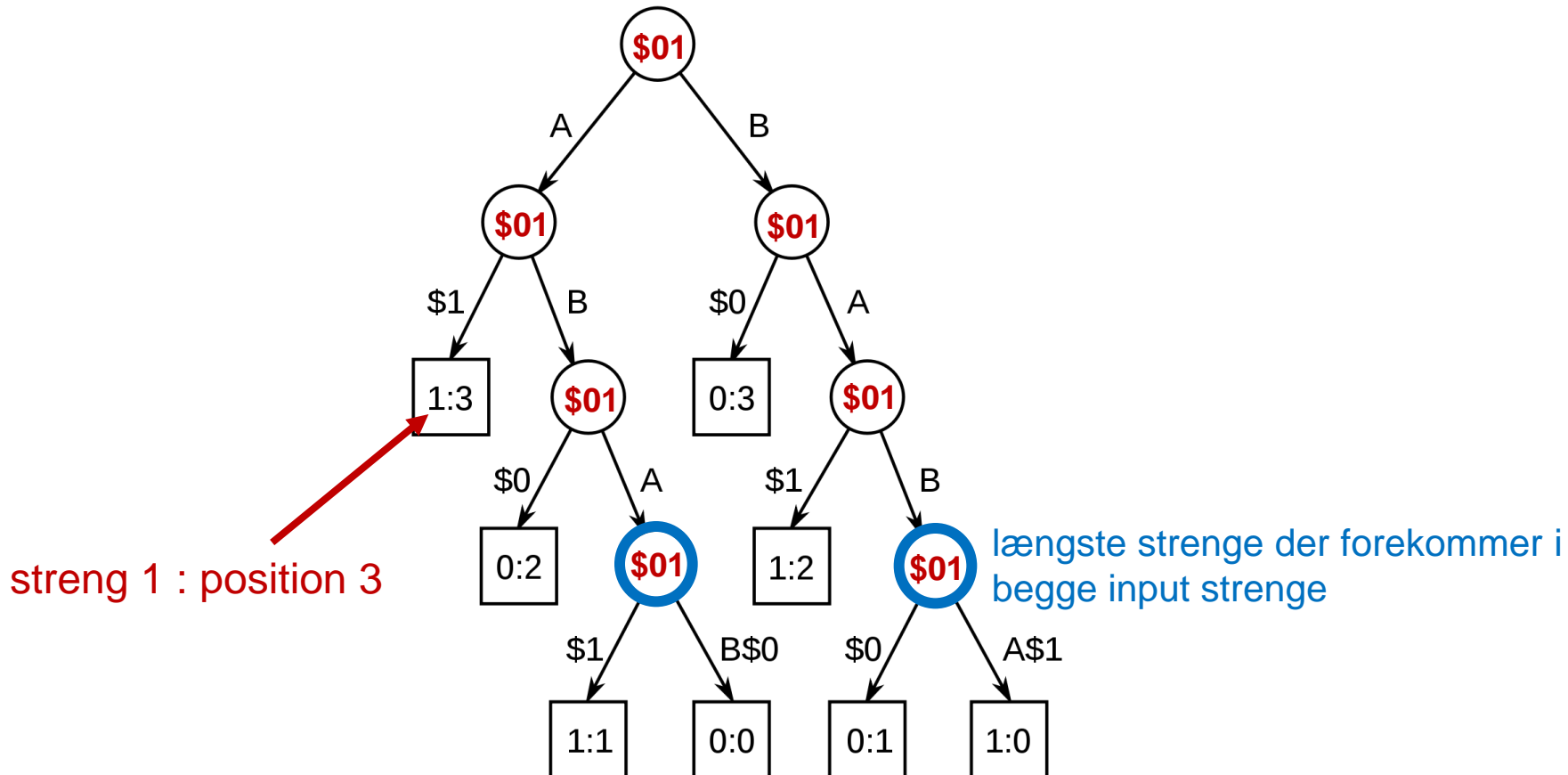
break out of the for loop

until f or $T.isExternal(v)$

return "P is not a substring of X"



Suffiks Træ over to Streng **ABAB\$₀BABA\$₁**



Suffiks Array

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

Suffikser

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 suffikser

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**

$j \leftarrow \sigma[M]$

elseif $u > x[\sigma[M].. \sigma[M] + m - 1]$ **then**

$L \leftarrow M$

else

$R \leftarrow M$

until $L = R - 1$ 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 $\text{lcp}(u, \sigma[M])$*

$P_M \leftarrow P + \text{lcp}(u[P + 1..m], x[\sigma[M] + P..n])$

if $P_M = m$ **then**

$j \leftarrow \sigma[M]$

elseif $u[P_M + 1] > x[\sigma[M] + P_M]$ **then**

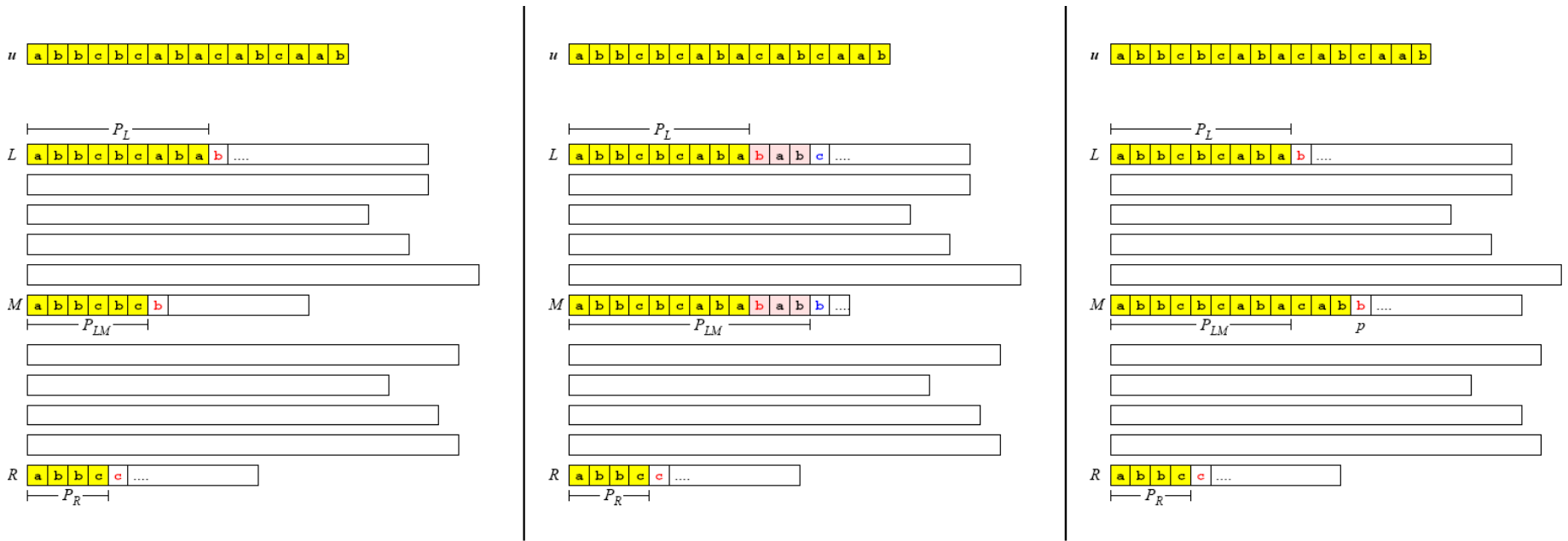
$L \leftarrow M$; $P_L \leftarrow P_M$

else

$R \leftarrow M$; $P_R \leftarrow P_M$

until $L = R-1$ or $j \neq 0$

SAComplex ($P_L \geq P_R$)



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

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

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

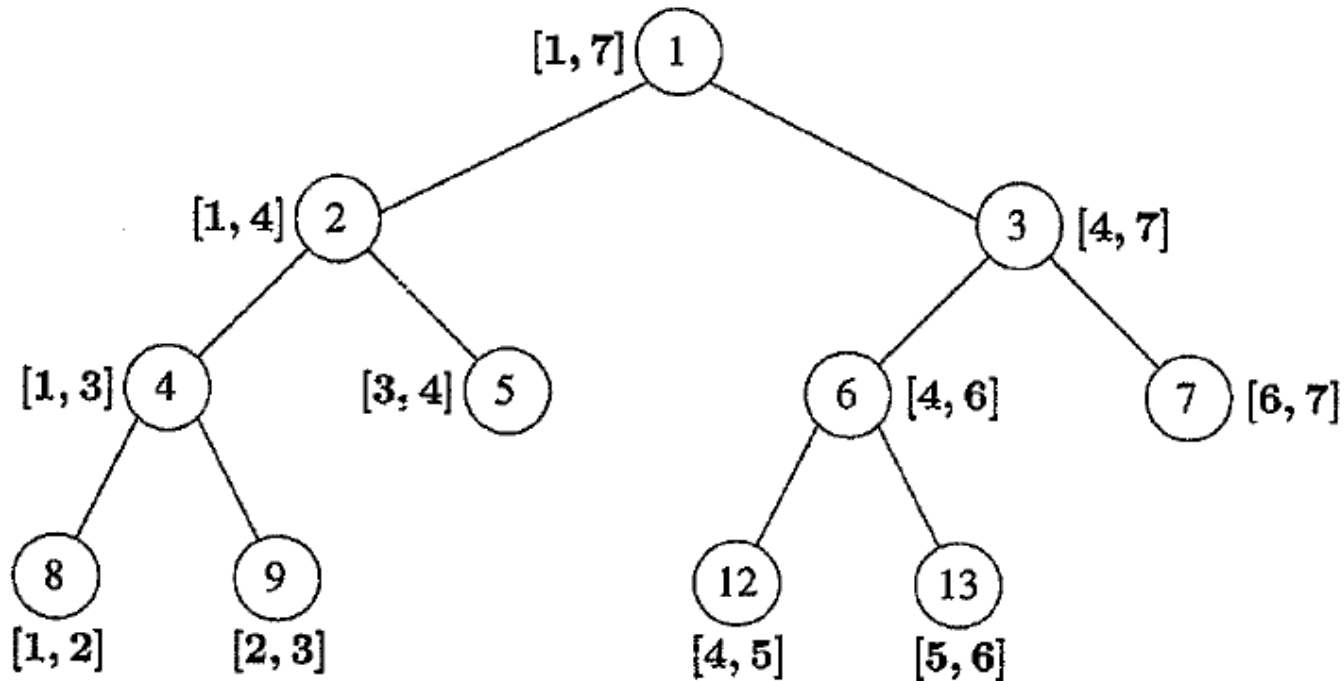
Lad p være første forskellige position:

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

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

præberegnet

Binært træ over intervaller



Mihai Pătrașcu

Søgninger i et Suffiks Array

Algorithm	Additional storage (bytes)	Theoretical time bound
SANaive	$n \log n / 8$	$O(m(\log n + k))$
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