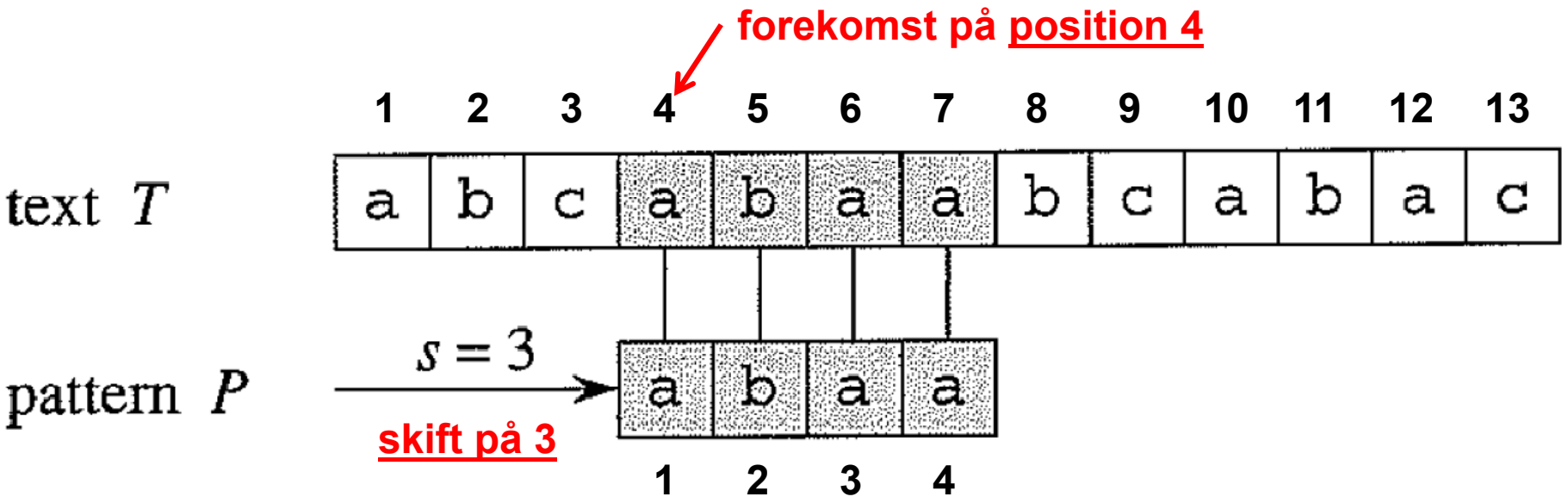
Algoritmer og Datastrukturer 2

Gerth Stølting Brodal

Mønstergenkendelse [CLRS, kapitel 32.1-32.2, 32.4]

AARHUS UNIVERSITET

Mønster genkendelse

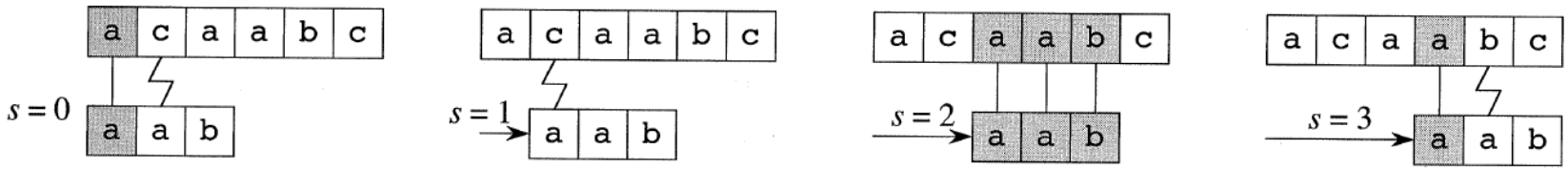


Input: Tekst T af længde n og mønster P af længde m

Output: Alle positioner i T hvor P forekommer

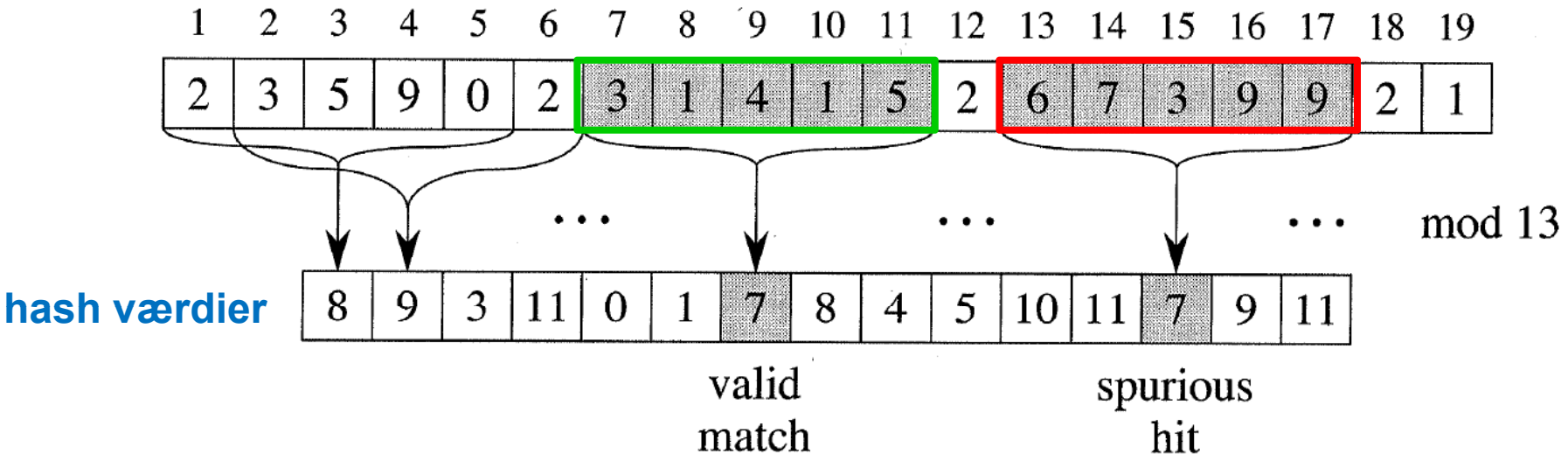
Naive Algoritme

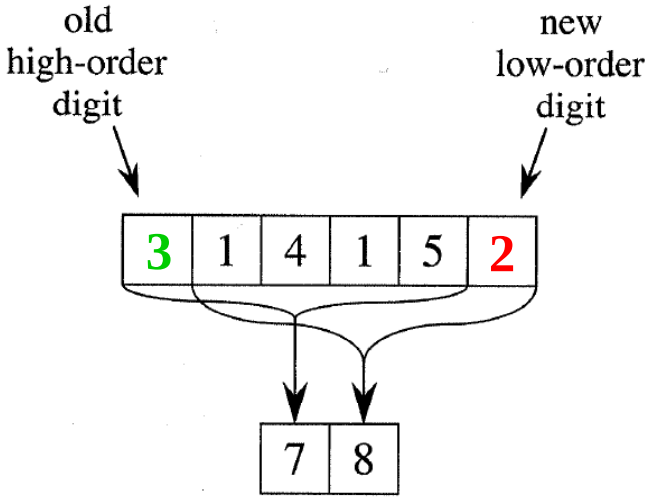
```
NAIVE-STRING-MATCHER(T, P)
1  n = T.length
2  m = P.length
3  for s = 0 to n − m
4       if P[1..m] == T[s + 1..s + m]
5            print "Pattern occurs with shift" s
```



O(n·m)

Rabin-Karp: Eksempel





$$14152 \equiv (31415 - 3 \cdot 10000) \cdot 10 + 2 \pmod{13}$$
$$\equiv (7 - 3 \cdot 3) \cdot 10 + 2 \pmod{13}$$
$$\equiv 8 \pmod{13}$$

old high-order digit — shift — new low-order digit

1981

Rabin-Karp

```
RABIN-KARP-MATCHER(T, P, d, q)
 1  n = T.length
 2  m = P.length
 3  h = d^(m−1) mod q
 4  p = 0
 5  t_0 = 0
 6  for i = 1 to m                          // preprocessing
 7       p = (dp + P[i]) mod q
 8       t_0 = (dt_0 + T[i]) mod q
 9  for s = 0 to n − m                      // matching
10       if p == t_s
11            if P[1..m] == T[s + 1..s + m]
12                 print "Pattern occurs with shift" s
13       if s < n − m
14            t_{s+1} = (d(t_s − T[s + 1]h) + T[s + m + 1]) mod q
```

$p = P[1]d^{m-1} + P[2]d^{m-2} + \cdots + P[m-1]d^{1} + P[m]d^{0} \bmod q$

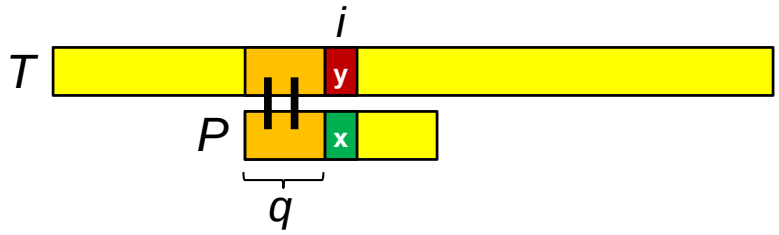
O(*n*·*m*)

1977

Knuth-Morris-Pratt

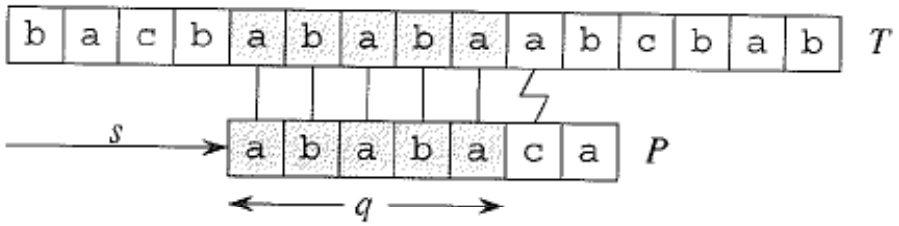
```
KMP-MATCHER(T, P)
 1   n = T.length
 2   m = P.length
 3   π = COMPUTE-PREFIX-FUNCTION(P)
 4   q = 0                                  // number of characters matched
 5   for i = 1 to n                         // scan the text from left to right
 6       while q > 0 and P[q + 1] ≠ T[i]
 7           q = π[q]                       // next character does not match
 8       if P[q + 1] == T[i]
 9           q = q + 1                      // next character matches
10       if q == m                          // is all of P matched?
11           print "Pattern occurs with shift" i − m
12           q = π[q]                       // look for the next match
```

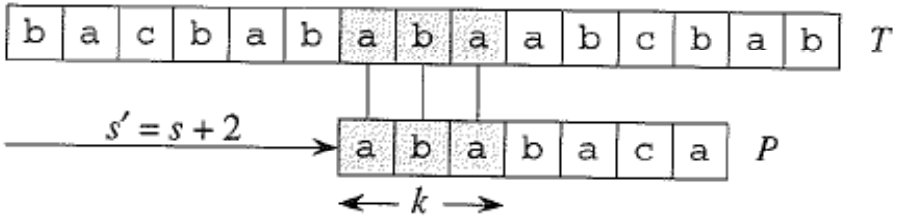
π(0) = 0
π (q) = max { i | i<q og *P*[1..i] er et suffix af *P*[1..q] }

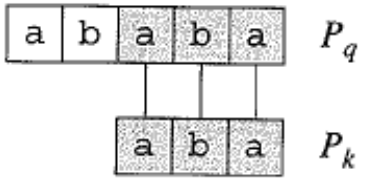


O(*n*)

Knuth-Morris-Pratt: Eksempel



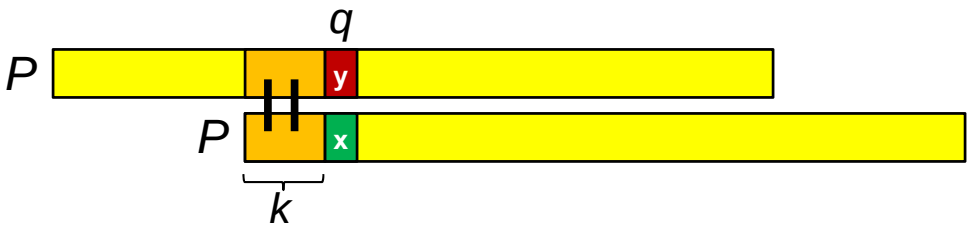




π(0) = 0
π (q) = max { i | i<q og *P*[1..i] er et suffix af *P*[1..q] }

Knuth-Morris-Pratt: Beregning af prefix funktionen

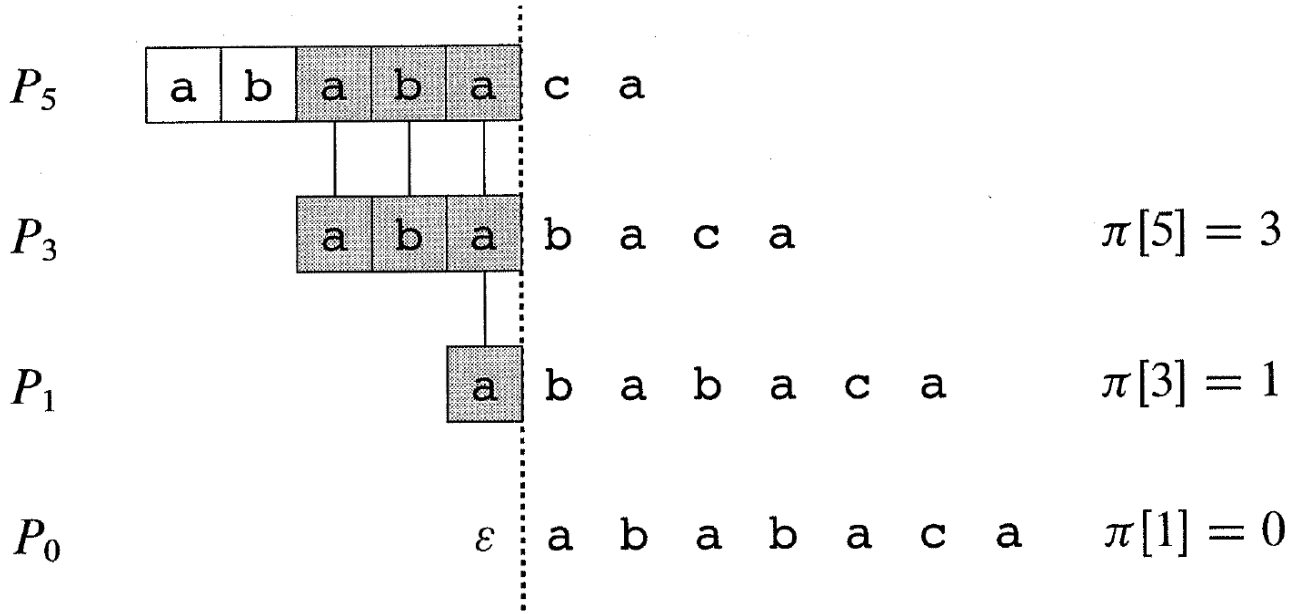
```
COMPUTE-PREFIX-FUNCTION(P)
 1  m = P.length
 2  let π[1..m] be a new array
 3  π[1] = 0
 4  k = 0
 5  for q = 2 to m
 6      while k > 0 and P[k + 1] ≠ P[q]
 7          k = π[k]
 8      if P[k + 1] == P[q]
 9          k = k + 1
10      π[q] = k
11  return π
```



O(*m*)

Knuth-Morris-Pratt: Beregning af prefix funktionen

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1



Worst-case tider

Algorithm	Preprocessing time	Matching time	[CLRS]
Naive	0	$O((n-m+1)m)$	32.1
Rabin-Karp	$\Theta(m)$	$O((n-m+1)m)$	32.2
Finite automaton	$O(m\lvert\Sigma\rvert)$	$\Theta(n)$	(32.3)
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$	32.4