Algorithmen und Datenstrukturen (Th. Ottmann und P. Widmayer)

Folien: Boyer-Moore

Autor: Sven Schuierer

Institut für Informatik

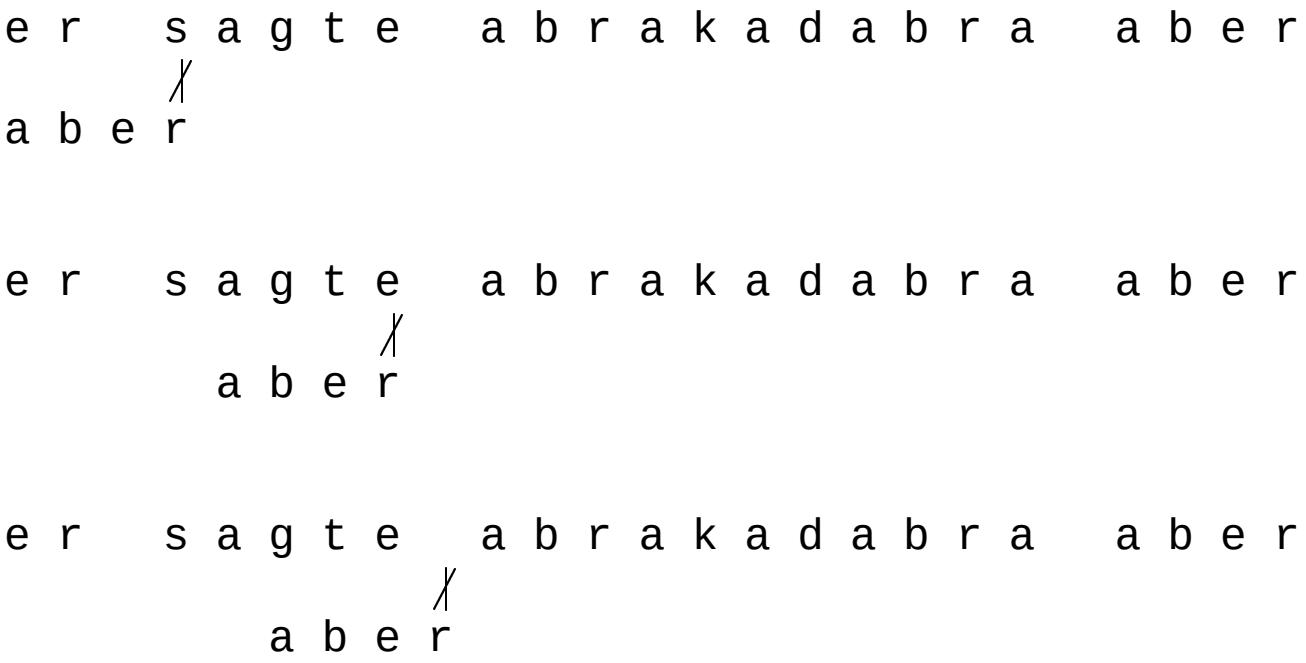
Georges-Köhler-Allee

Albert-Ludwigs-Universität Freiburg

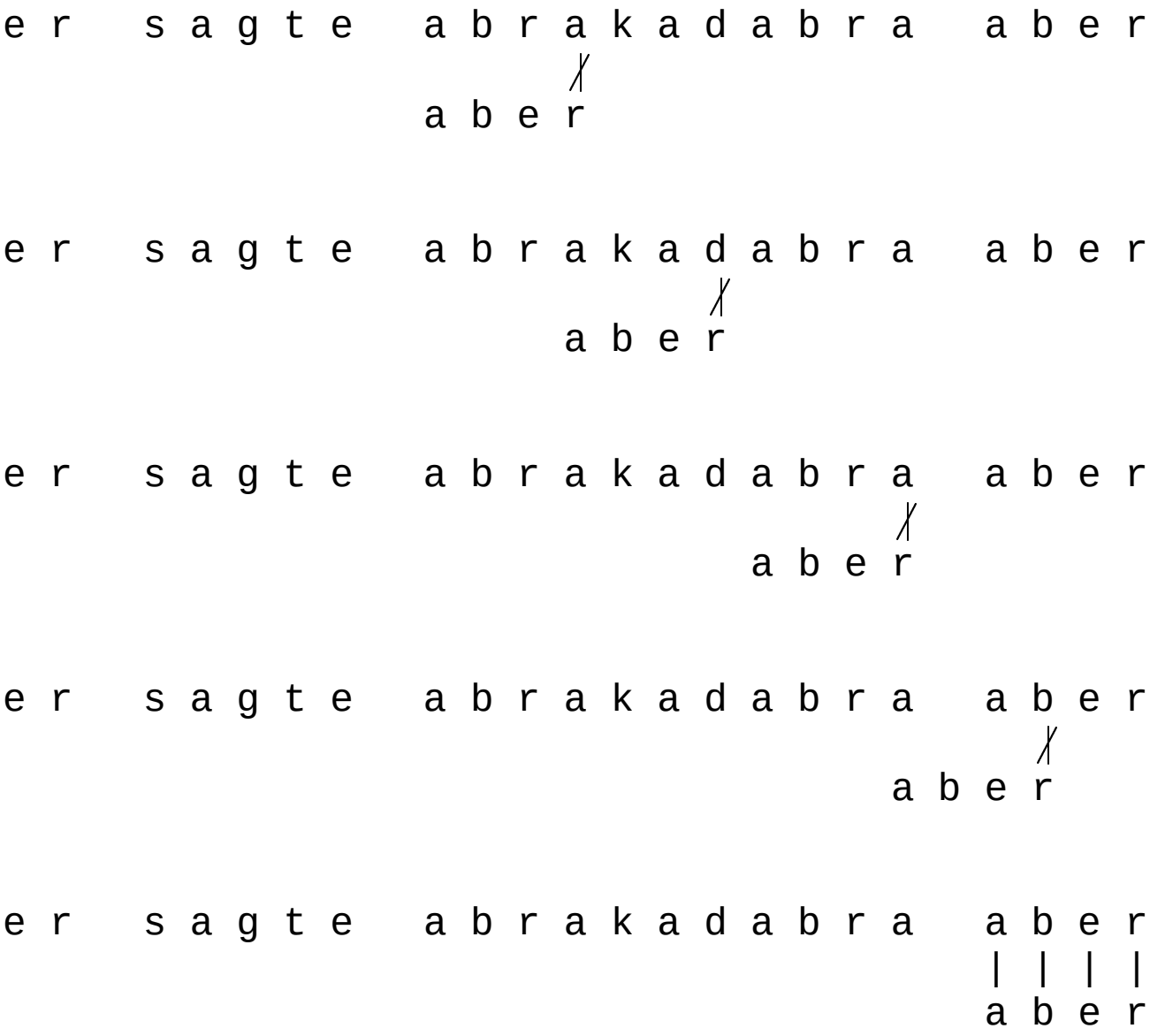
1 Boyer-Moore Textsuche

Idee: Das Muster von links nach rechts anlegen, aber zeichenweise von rechts nach links vergleichen

Beispiel:



Beispiel


e r s a g t e a b r a k a d a b r a a b e r
a b e r
e r s a g t e a b r a k a d a b r a a b e r
a b e r
e r s a g t e a b r a k a d a b r a a b e r
a b e r
e r s a g t e a b r a k a d a b r a a b e r
a b e r
e r s a g t e a b r a k a d a b r a a b e r
a b e r

Die Vorkommensheuristik

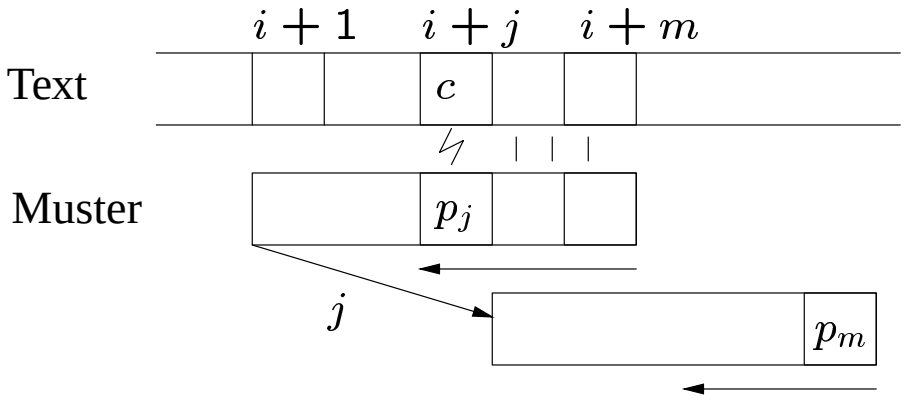
$c =$ das den Mismatch verursachende Zeichen

$j =$ Index des aktuellen Zeichens im Muster ($c \neq p_j$)

Berechnung der Musterverschiebung

Case 1 c kommt nicht im Muster P vor.

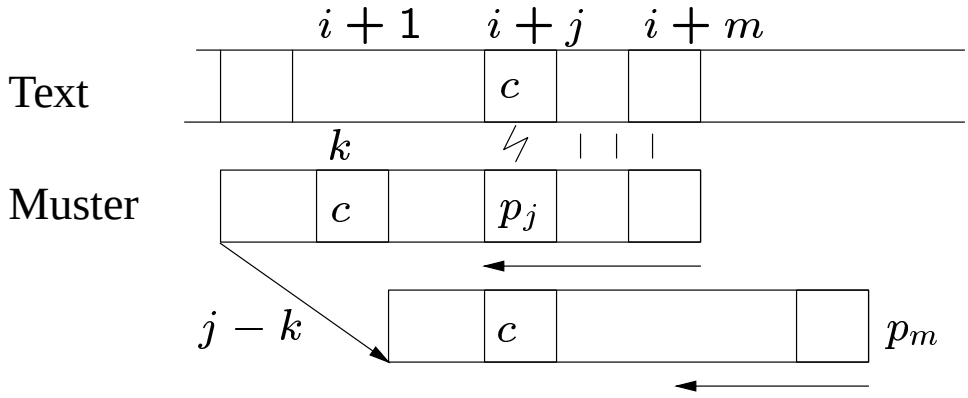
Verschiebe das Muster um j Positionen nach rechts



Die Vorkommensheuristik

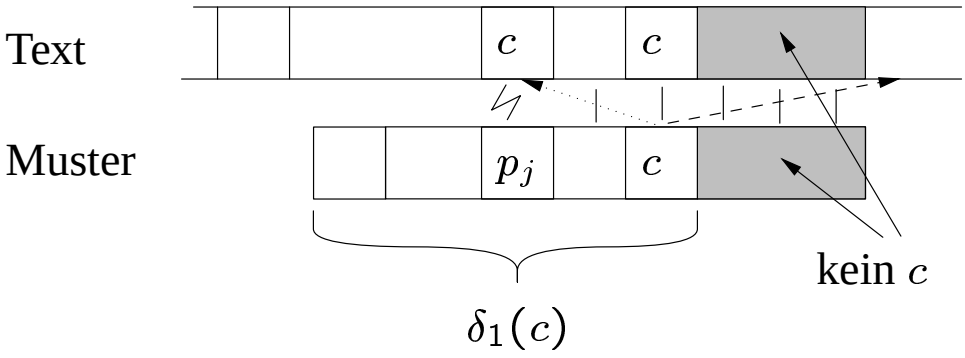
Case 2 c kommt im Muster vor.

Verschiebe das Muster soweit nach rechts, daß das rechteste c im Muster über dem c im Text liegt.



Die Vorkommensheuristik

Fall a: $\delta(c) > j$

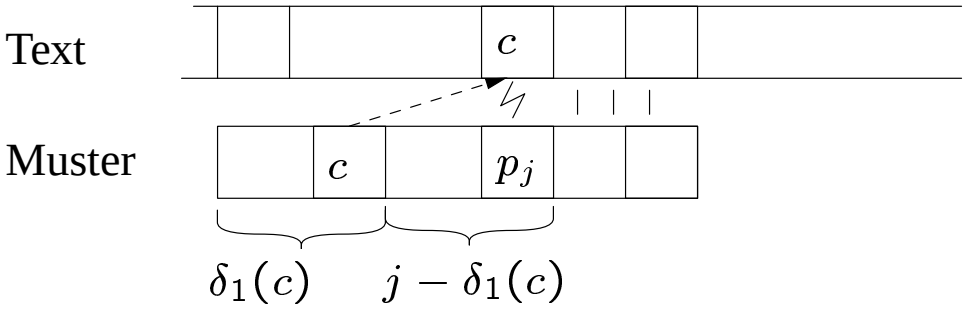


Verschiebung des rechtesten c im Muster auf c im Text:
Verschiebung nach links!

$\Rightarrow$ Verschiebung um $m - \delta(c) + 1$

Die Vorkommensheuristik

Fall b: $\delta(c) < j$



Verschiebung des rechtesten c im Muster auf c im Text:
Verschiebung um $j - \delta(c)$

Implementation

BM-search1

Input: Text T und Pattern P

Output: Verschiebungen für alle Vorkommen von P in T

1 $n := length(T)$; $m := length(P)$
2 berechne δ
3 $i := 0$
4 while $i \leq n - m$ do
5 $\quad j := m$
6 $\quad$ while $j > 0$ and $P[j] = T[i + j]$ do
7 $\quad\quad j := j - 1$
$\quad$ end while;
8 $\quad$ if $j = 0$
9 $\quad\quad$ then gebe Verschiebung i aus
10 $\quad\quad\quad i := i + 1$
11 $\quad$ else if $\delta(T[i + j]) > j$
12 $\quad\quad$ then $i := i + m + 1 - \delta[T[i + j]]$
13 $\quad\quad$ else $i := i + j - \delta[T[i + j]]$
end while;

Berechnung von wrw

$wrw[j] =$ Position, an der das von rechts her nächste Vorkommen des Suffix $P_{j+1} \cdots P_m$ endet, dem nicht das Zeichen P_j vorangeht.

Muster: banana

$wrw[j]$	betracht. Suf.	verb. Zeich.	weit. Auft.	Pos.
$wrw[5]$	a	n	banana	2
$wrw[4]$	na	a	***bana na	0
$wrw[3]$	ana	n	banana	4
$wrw[2]$	nana	a	banana	0
$wrw[1]$	anana	b	banana	0
$wrw[0]$	banana	ϵ	banana	0

aber z.B. $wrw[2] = wrw[0] = 3$ für anana

Boyer-Moore Textsuche

BM-search2

Input: Text T und Pattern P

Output: Verschiebungen für alle Vorkommen von P in T

```
1  n := length(T); m := length(P)
2  berechne δ und γ
3  i := 0
4  while i ≤ n − m do
5      j := m
6      while j > 0 and P[j] = T[i + j] do
7          j := j − 1
       end while;
8      if j = 0
9        then gebe Verschiebung i aus
10             i := i + γ[0]
11       else i := i + max(γ[j], j − δ[T[i + j]])
   end while;
```