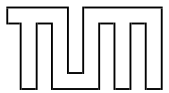
Effiziente Algorithmen 2

Dr. Hanjo Täubig

Lehrstuhl für Effiziente Algorithmen
(Prof. Dr. Ernst W. Mayr)
Institut für Informatik
Technische Universität München

Sommersemester 2009



Übersicht

Zweiter naiver Ansatz

- weiterer Ansatzpunkt zum Suchen in Texten:
  das gesuchte Wort mit einem Textstück nicht mehr von links nach rechts, sondern von rechts nach links vergleichen

- Vorteil:
  In Alphabeten mit vielen verschiedenen Symbolen kann man bei einem Mismatch an der Position $i + j$ in t, i auf $i + j + 1$ setzen, falls das im Text t vorgefundene Zeichen t_{i+j} gar nicht im gesuchten Wort s vorkommt.

- D.h. also, in so einem Fall kann man das Suchwort gleich um $j + 1$ Positionen verschieben (im günstigsten Fall m Stellen).

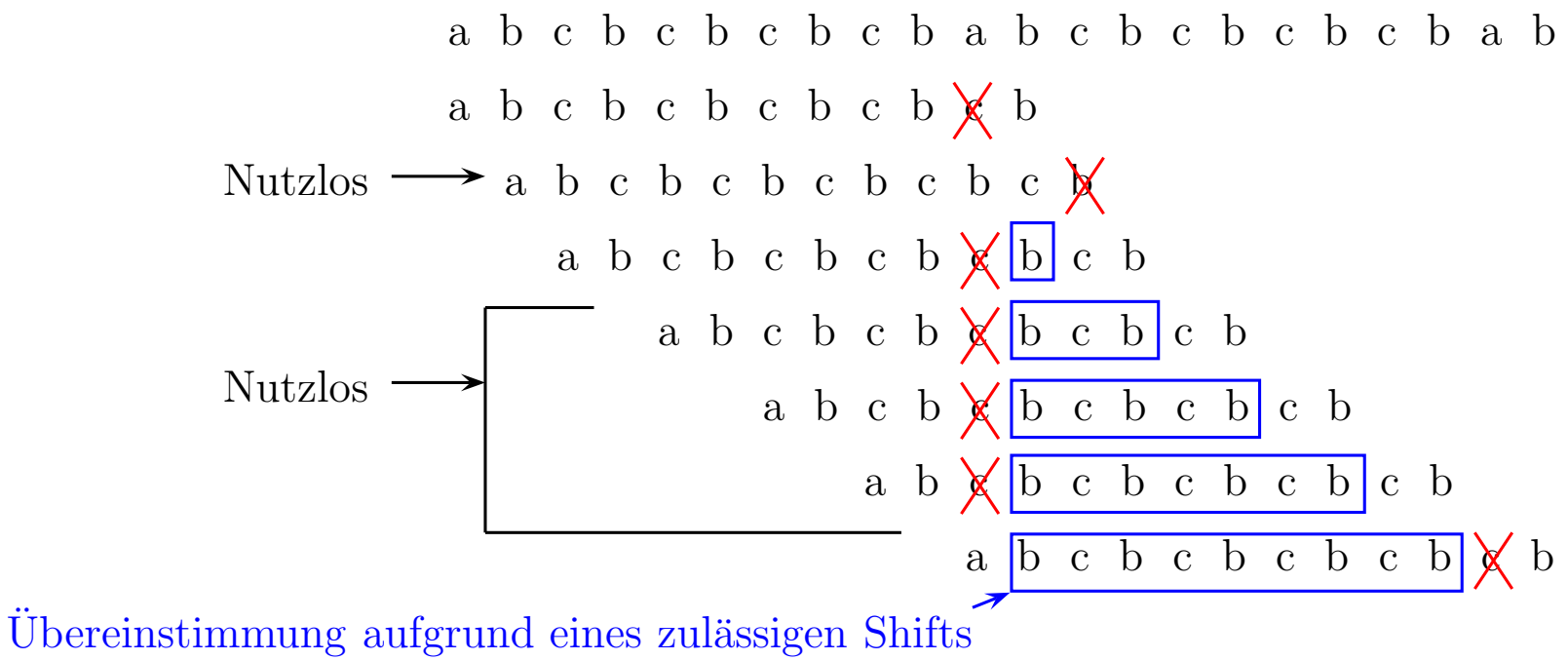
Zweiter naiver Ansatz

Algorithmus 4 : bool Naiv2(char $t[]$, int n, char $s[]$, int m)

```
int i := 0;
int j := m − 1;
while i ≤ n − m do
    while t[i + j] = s[j] do
        if j = 0 then return TRUE;
        j−−;
    i++;
    j := m − 1;
return FALSE;
```

Naive Methode mit rechts-nach-links Vergleichen
(ohne größere Shifts)

Nutzlose Verschiebungen

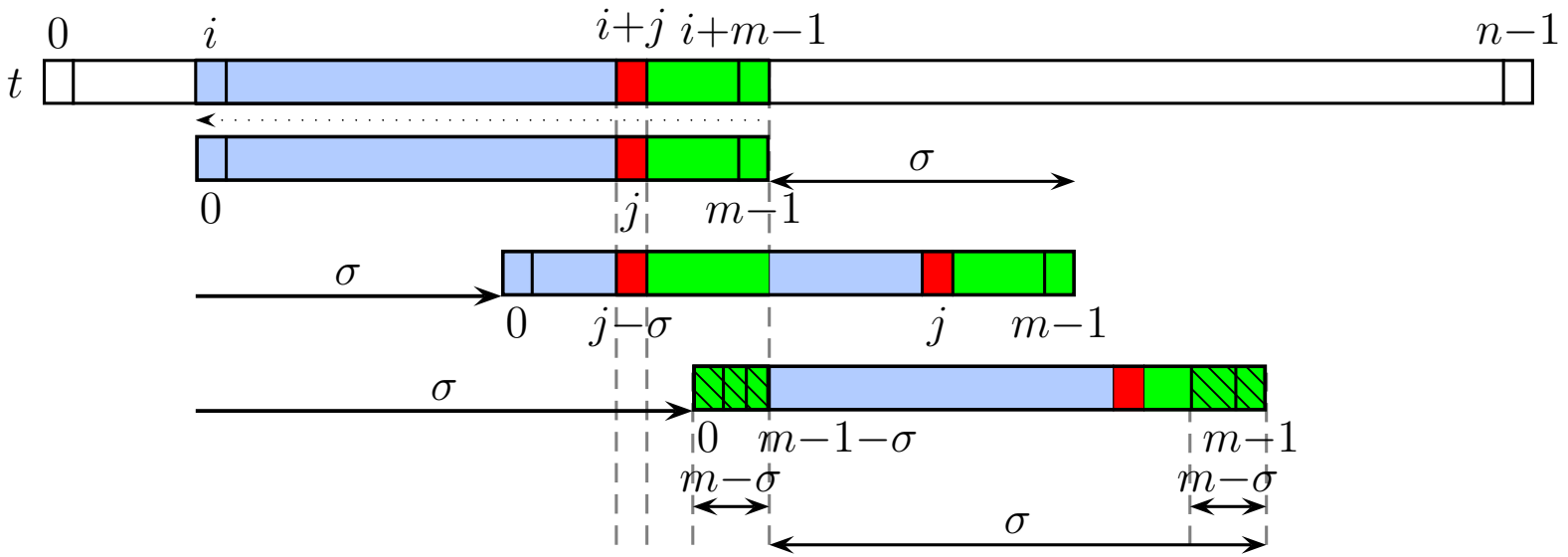


Nutzlose Verschiebungen

- Wie beim KMP-Algorithmus ist es nutzlos (siehe zweiter Versuch von oben), wenn man die Zeichenreihe so verschiebt, dass im Bereich erfolgreicher Vergleiche nach einem Shift keine Übereinstimmung mehr herrscht.

- Es macht ebenfalls keinen Sinn, das Muster so zu verschieben, dass an der Position des Mismatches in t im Muster s wiederum dasselbe Zeichen zum Liegen kommt, das schon vorher den Mismatch ausgelöst hat.
(vierter bis sechster Versuch)

Boyer-Moore-Algorithmus



Skizze: Zulässige Shifts bei Boyer-Moore (Strong-Good-Suffix-Rule)

Boyer-Moore-Algorithmus: Shifts

- Zwei mögliche Arten eines „vernünftigen“ Shifts bei der Variante von Boyer-Moore:
  - Im oberen Teil ist ein „kurzer“ Shift angegeben, bei dem im grünen Bereich die Zeichen nach dem Shift weiterhin übereinstimmen.
    Das rote Zeichen in t (das den Mismatch ausgelöst hat) soll nach dem Shift auf ein anderes Zeichen in s treffen, damit überhaupt die Chance auf Übereinstimmung besteht.
  - Im unteren Teil ist ein „langer“ Shift angegeben, bei dem die Zeichenreihe s soweit verschoben wird, dass an der Position des Mismatches in t gar kein weiterer Vergleich mehr entsteht. Aber auch hier: wieder Übereinstimmung im schraffierten grünen Bereich mit den bereits verglichenen Zeichen aus t

Boyer-Moore-Algorithmus: Good-Suffix-Rule

- Kombination der beiden Regeln: Good-Suffix-Rule
  (da man darauf achtet, die Zeichenreihen so zu verschieben, dass im letzten übereinstimmenden Bereich nach dem Shift wieder Übereinstimmung herrscht)
- Achtet man noch speziell darauf, dass an der Position, in der es zum Mismatch gekommen ist, jetzt in s ein anderes Zeichen liegt als das, welches den Mismatch ausgelöst hat, so spricht man von der Strong-Good-Suffix-Rule (andernfalls Weak-Good-Suffix-Rule).
- Wir betrachten nur die Strong-Good-Suffix-Rule, da ansonsten die worst-case Laufzeit wieder quadratisch werden kann.

Boyer-Moore-Algorithmus

Algorithmus 5 : bool Boyer-Moore(char $t[]$, int n, char $s[]$, int m)

```
int S[m + 1];
compute_shift_table(S, m, s);
int i := 0, j := m − 1;
while i ≤ n − m do
    while t[i + j] = s[j] do
        if j = 0 then return TRUE;
        j−−;
    i := i + S[j];
    j := m − 1;
return FALSE;
```

Wiederholte Vergleiche in übereinstimmenden Bereichen

- Nach dem Shift gibt es einen Bereich, in dem Übereinstimmung von s und t vorliegt.

- Allerdings werden auch in diesem Bereich wieder Vergleiche ausgeführt, da es letztendlich doch zu aufwendig ist, sich diesen Bereich explizit zu merken und bei folgenden Vergleichen von s in t zu überspringen.

Ausgangssituation direkt nach Mismatch

- Der erste Mismatch soll im zu durchsuchenden Text t an der Stelle $i + j$ (also im Suchwort s an Stelle j) auftreten.
- Da der Boyer-Moore-Algorithmus das Suchwort von hinten nach vorne vergleicht, ergibt sich folgende Voraussetzung:

$$s_{j+1} \cdots s_{m-1} = t_{i+j+1} \cdots t_{i+m-1} \quad \wedge \quad s_j \neq t_{i+j}$$

- in Worten: Das Suffix des Suchworts ab Index $j + 1$ stimmt mit den entsprechenden Zeichen im Text überein, das Zeichen an Position j jedoch nicht.

Werte der Shift-Tabelle: kleine Shifts

- Um nun einen sinnvollen Shift um σ Positionen zu erhalten, muss gelten:

$$s_{j+1-\sigma} \cdots s_{m-1-\sigma} = t_{i+j+1} \cdots t_{i+m-1} = s_{j+1} \cdots s_{m-1} \qquad \wedge$$

$$s_j \neq s_{j-\sigma}$$

- Diese Bedingung ist nur für „kleine“ Shifts mit $\sigma \leq j$ sinnvoll, also für solche, bei denen der Anfang des Suchworts s noch (mindestens) den gesamten bisher gematchten Bereich (inklusive das Mismatch-Zeichen) vom Text t abdeckt.

- Beispiel: erster Shift in der vorigen Abbildung

Werte der Shift-Tabelle: große Shifts

- Für „große“ Shifts $\sigma > j$ muss gelten, dass das Suffix des übereinstimmenden Bereichs mit dem Präfix des Suchwortes übereinstimmt, d.h.:

$$s_0 \cdots s_{m-1-\sigma} = t_{i+\sigma} \cdots t_{i+m-1} = s_\sigma \cdots s_{m-1}$$

- Zusammengefasst ergibt sich für beide Bedingungen:

$$\sigma \leq j \quad \wedge \quad s_{j+1} \cdots s_{m-1} \in \mathcal{R}(s_{j+1-\sigma} \cdots s_{m-1}) \quad \wedge \quad s_j \neq s_{j-\sigma}$$
$$\sigma > j \quad \wedge \quad s_0 \cdots s_{m-1-\sigma} \in \mathcal{R}(s_0 \cdots s_{m-1})$$

$\mathcal{R}(s)$: Menge aller Ränder von s

Zulässige und sichere Shifts

- Erfüllt ein Shift nun eine dieser Bedingungen, so nennt man diesen Shift zulässig.
- Um einen sicheren Shift zu erhalten, wählt man das minimale σ, das eine der Bedingungen erfüllt.
- Somit gilt:

$$S[j] = \min \left\{ \sigma : \begin{array}{c} (s_{j+1} \cdots s_{m-1} \in \mathcal{R}(s_{j+1-\sigma} \cdots s_{m-1}) \wedge \\ s_j \neq s_{j-\sigma} \wedge \sigma \leq j) \vee \\ (s_0 \cdots s_{m-1-\sigma} \in \mathcal{R}(s_0 \cdots s_{m-1}) \wedge \sigma > j) \vee \\ (\sigma = m) \end{array} \right\}$$

Bestimmung der Shift-Tabelle

Algorithmus 6 : compute_shift_table(int $S[]$, char $s[]$, int m)

```
for (int j := 0; j ≤ m; j++) do
    S[j] := m ;                          // Initialisierung von S[]
// Teil 1: σ ≤ j
int border2[m + 1];
border2[0] := −1;
int i := border2[1] := 0;
for (int j' := 2; j' ≤ m; j'++) do
    // Hier gilt: i = border2[j' − 1]
    while ((i ≥ 0) && (s[m − i − 1] ≠ s[m − j'])) do
        int σ := j' − i − 1;
        S[m − i − 1] := min(S[m − i − 1], σ);
        i := border2[i];
    i++;
    border2[j'] := i;
```

Bestimmung der Shift-Tabelle

Algorithmus 7 : compute_shift_table(int $S[]$, char $s[]$, int m)

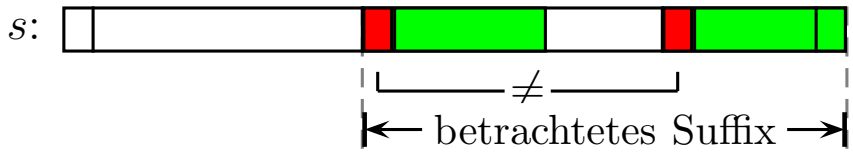
```
   ⋮

// Teil 2: σ > j
int j := 0;
for (int i := border2[m]; i ≥ 0; i := border2[i]) do
    int σ := m − i;
    while (j < σ) do
        S[j] := min(S[j], σ);
        j++;
```

Bestimmung der Shift-Tabelle

- Zu Beginn wird die Shift-Tabelle an allen Stellen mit der Länge des Suchstrings initialisiert.
- Im Wesentlichen entsprechen beide Fälle von möglichen Shifts (siehe obige Vorüberlegungen) der Bestimmung von Rändern von Teilwörtern des gesuchten Wortes.
- Dennoch unterscheiden sich die hier betrachteten Fälle vom KMP-Algorithmus, da hier, zusätzlich zu den Rändern von Präfixen des Suchwortes, auch die Ränder von Suffixen des Suchwortes gesucht sind.

Bestimmung der Shift-Tabelle: $\sigma \leq j$

- Dies gilt besonders für den ersten Fall ($\sigma \leq j$). Hier soll das Zeichen unmittelbar vor dem Rand des Suffixes ungleich dem Zeichen unmittelbar vor dem gesamten Suffix sein:



- Diese Situation entspricht dem Fall in der Berechnung eines eigentlichen Randes, bei dem der vorhandene Rand nicht zu einem längeren Rand fortgesetzt werden kann
(nur werden hier Suffixe statt Präfixe betrachtet).
- Daher sieht die erste for-Schleife genau so aus, wie in der Prozedur compute_borders des KMP-Algorithmus.
- Lässt sich ein betrachteter Rand nicht verlängern, so wird die while-Schleife ausgeführt.

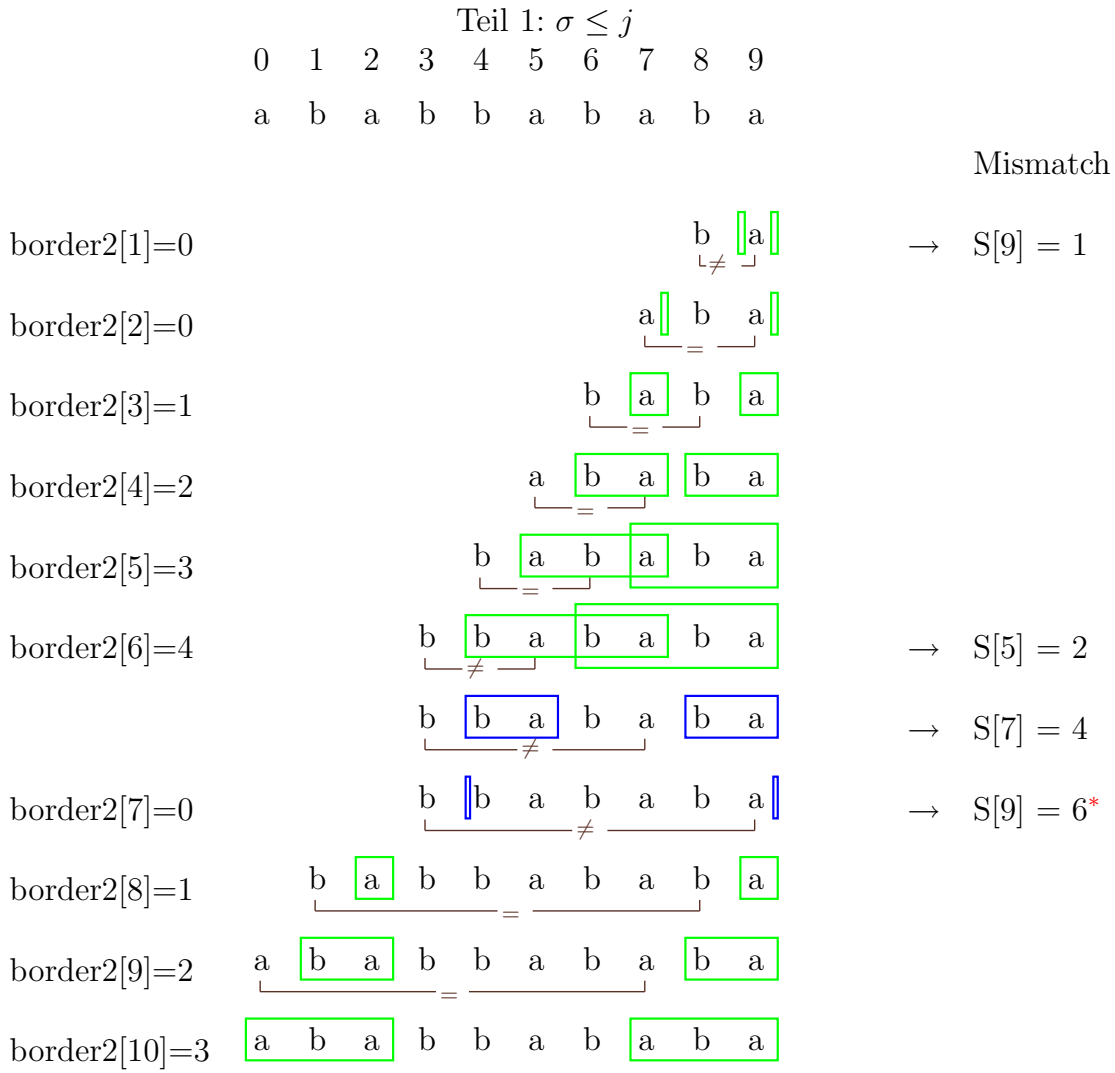
Bestimmung der Shift-Tabelle: $\sigma > j$

- Zweiter Fall ($\sigma > j$): man muss alle Ränder von s durchlaufen.
- Der längste Rand ungleich s ist der eigentliche Rand von s. Diesen erhalten wir über $border2[m]$, da ja das Suffix der Länge m von s gerade wieder s ist.
- Der nächstkürzere Rand von s muss ein Rand des eigentlichen Randes von s sein. Damit es der nächstkürzere Rand ist, muss s der eigentliche Rand des eigentlichen Randes von s sein, also das Suffix der Länge $border2[border2[s]]$.
- Somit können wir alle Ränder von s durchlaufen, indem wir immer vom aktuell betrachteten Rand der Länge ℓ das Suffix der Länge $border2[\ell]$ wählen. Dies geschieht in der for-Schleife im zweiten Teil. Solange der Shift σ größer als die Position j eines Mismatches ist, wird die Shift-Tabelle aktualisiert. Dies geschieht in der inneren while-Schleife.

Bestimmung der Shift-Tabelle

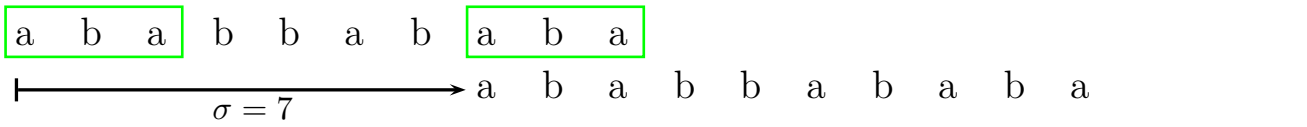
- Bei der Aktualisierung der Shift-Tabelle werden nur zulässige Werte berücksichtigt. Damit sind die Einträge der Shift-Tabelle (d.h. die Länge der Shifts) nie zu klein.
- Eigentlich müsste jetzt noch gezeigt werden, dass die Werte auch nicht zu groß sind, d.h. dass es keine Situation geben kann, in der ein kleinerer Shift möglich wäre.
- Dass dies nicht der Fall ist, kann mit einem Widerspruchsbeweis gezeigt werden (man nehme dazu an, bei einem Mismatch an einer Position j in s gäbe es einen kürzeren Shift gemäß der Strong-Good-Suffix-Rule und leite daraus einen Widerspruch her).

Bestimmung der Shift-Tabelle: Beispiel

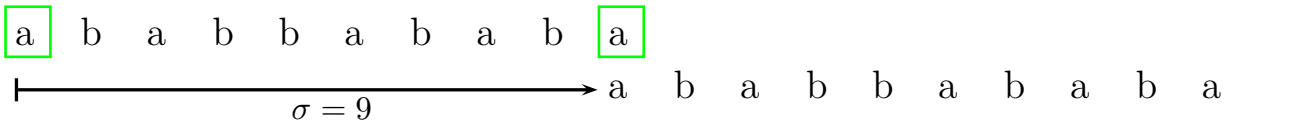


Bestimmung der Shift-Tabelle: Beispiel

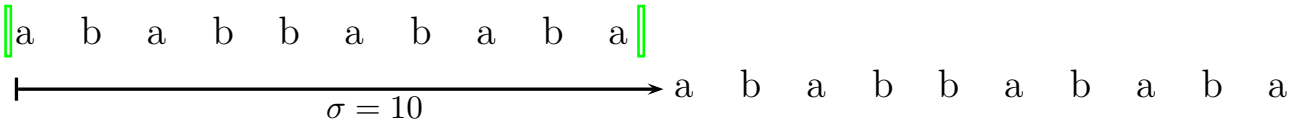
Teil 2: $\sigma > j$



bei Mismatch an Position $j \in [0:6]$ S[j] = 7



bei Mismatch an Position $j \in [7:8]$ S[j] = 9



bei Mismatch an Position $j \in [9]$ S[j] = 10

Zusammenfassung:

	1.Teil	2.Teil	Erg		1.Teil	2.Teil	Erg
S[0] =		7	7	S[5] =	2		2
S[1] =		7	7	S[6] =		7	7
S[2] =		7	7	S[7] =	4		4
S[3] =		7	7	S[8] =		9	9
S[4] =		7	7	S[9] =	1		1

Laufzeit von *compute_shift_table*

- Die Prozedur *compute_shift_table* entspricht im ersten Teil der Prozedur *compute_borders* des KMP-Algorithmus
- Anzahl (Zeichen-)Vergleiche ist also wieder $\leq 2m - 1$

- Der zweite Teil kann höchstens m-mal durchlaufen werden.
- Dabei werden gar keine (Zeichen-)Vergleiche durchgeführt.

Lemma

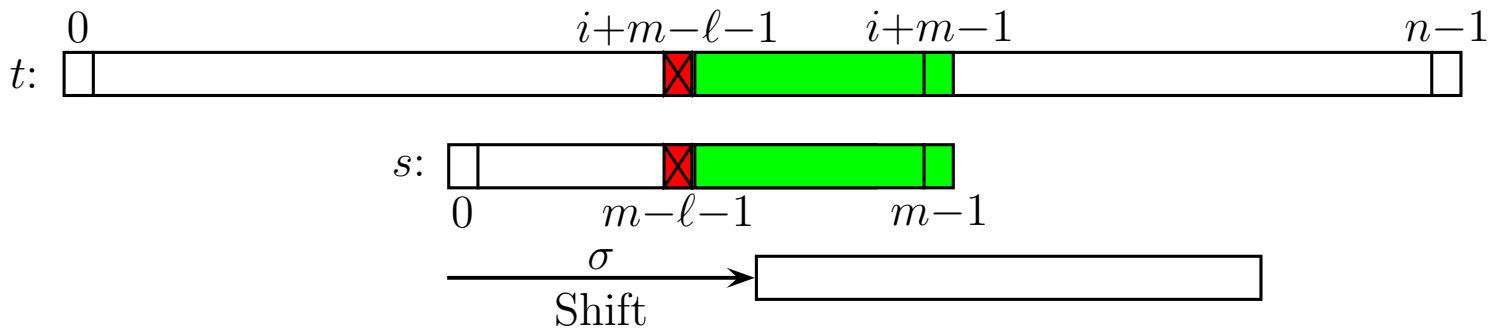
Die Shift-Tabelle des Boyer-Moore-Algorithmus lässt sich für eine Zeichenkette der Länge m mit maximal 2m Vergleichen berechnen.

Laufzeit der Hauptprozedur

- Betrachte nur die Suche nach ersten Vorkommen des Suchworts s in t (bzw. eine erfolglose Suche)
- Unterscheide initiale und wiederholte Vergleiche

- Initiale Vergleiche:
  Vergleiche von s_j mit t_{i+j}, so dass t_{i+j} zum ersten Mal beteiligt ist
- Wiederholte Vergleiche:
  Vergleiche von s_j mit t_{i+j}, so dass t_{i+j} schon früher bei einem Vergleich beteiligt war

Laufzeit der Hauptprozedur



Skizze: Shift nach ℓ erfolgreichen Vergleichen

Laufzeit der Hauptprozedur

Lemma

Für eine Textposition $i \in [0 : n - m]$ *und eine Anzahl* $\ell \in [0 : m - 1]$ *gelte*

- $s_{m-\ell} \cdots s_{m-1} = t_{i+m-\ell} \cdots t_{i+m-1}$, *sowie*
- $s_{m-\ell-1} \neq t_{i+m-\ell-1}$.

(Es gab also ℓ *erfolgreiche und einen erfolglosen Vergleich.)*

- *Dabei seien* l *initiale Vergleiche durchgeführt worden.*
- *Sei* σ *die Länge des folgenden Shifts gemäß der Strong-Good-Suffix-Rule.*

Dann gilt:

$$\ell + 1 \leq l + 4 \cdot \sigma$$

Lemma: kurze und lange Shifts

Unterscheidung des darauf folgenden Shifts in Abhängigkeit seiner Länge im Verhältnis zu ℓ:

- $\sigma \geq \lceil \ell/4 \rceil$: langer Shift,
- $\sigma < \lceil \ell/4 \rceil$: kurzer Shift.

Lemma: Fall 1 (langer Shift)

Shift σ ist lang, d.h. $\sigma \geq \lceil \ell/4 \rceil$:

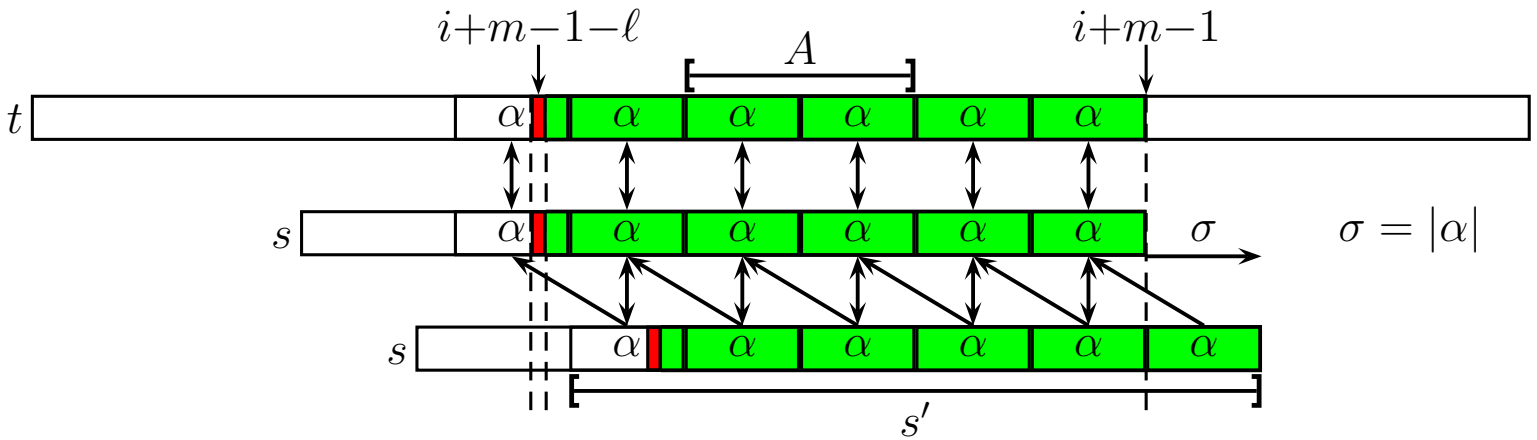
$$\begin{aligned} \text{Anzahl der ausgeführten Vergleiche} &= \ell + 1 \\ &\leq 1 + 4 \cdot \lceil \ell/4 \rceil \\ &\leq 1 + 4 \cdot \sigma \\ &\leq l + 4 \cdot \sigma \end{aligned}$$

Letzte Ungleichung: bei jedem Versuch muss es immer mindestens einen initialen Vergleich geben (zu Beginn und nach einem Shift wird das letzte Zeichen von s mit einem Zeichen aus t verglichen, das vorher noch an keinem Vergleich beteiligt war).

Lemma: Fall 2 (kurzer Shift)

Shift σ ist kurz, d.h. $\sigma \leq \lceil \ell/4 \rceil - 1 \leq \ell/4$:

Wir zeigen zuerst, dass dann das Ende von s periodisch sein muss, d.h. es gibt ein $\alpha \in \Sigma^+$, so dass s mit α^5 enden muss.



Lemma: Fall 2 (kurzer Shift)

- Im Intervall $[i+m-\ell : i+m-1]$ in t stimmen die Zeichen mit der verschobenen Zeichenreihe s überein (Good-Suffix-Rule)
- Das Suffix α von s der Länge σ muss mindestens $\lfloor \ell/\sigma \rfloor$ mal am Ende der Übereinstimmung von s und t vorkommen.
- α muss also mindestens $k := 1 + \lfloor \ell/\sigma \rfloor \geq 5$ mal am Ende von s auftreten (siehe Abbildung).
- Nur falls das Wort s kürzer als $k \cdot \sigma$ ist, ist s ein Suffix von α^k.
- Sei s' das Suffix der Länge $k \cdot \sigma$ von s, falls $|s| \geq k \cdot \sigma$ ist, und $s' = s$ sonst.
- Im Suffix s' wiederholen sich die Zeichen im Abstand von σ Positionen.

Lemma: Fall 2 (kurzer Shift)

- Definiere Intervall A:

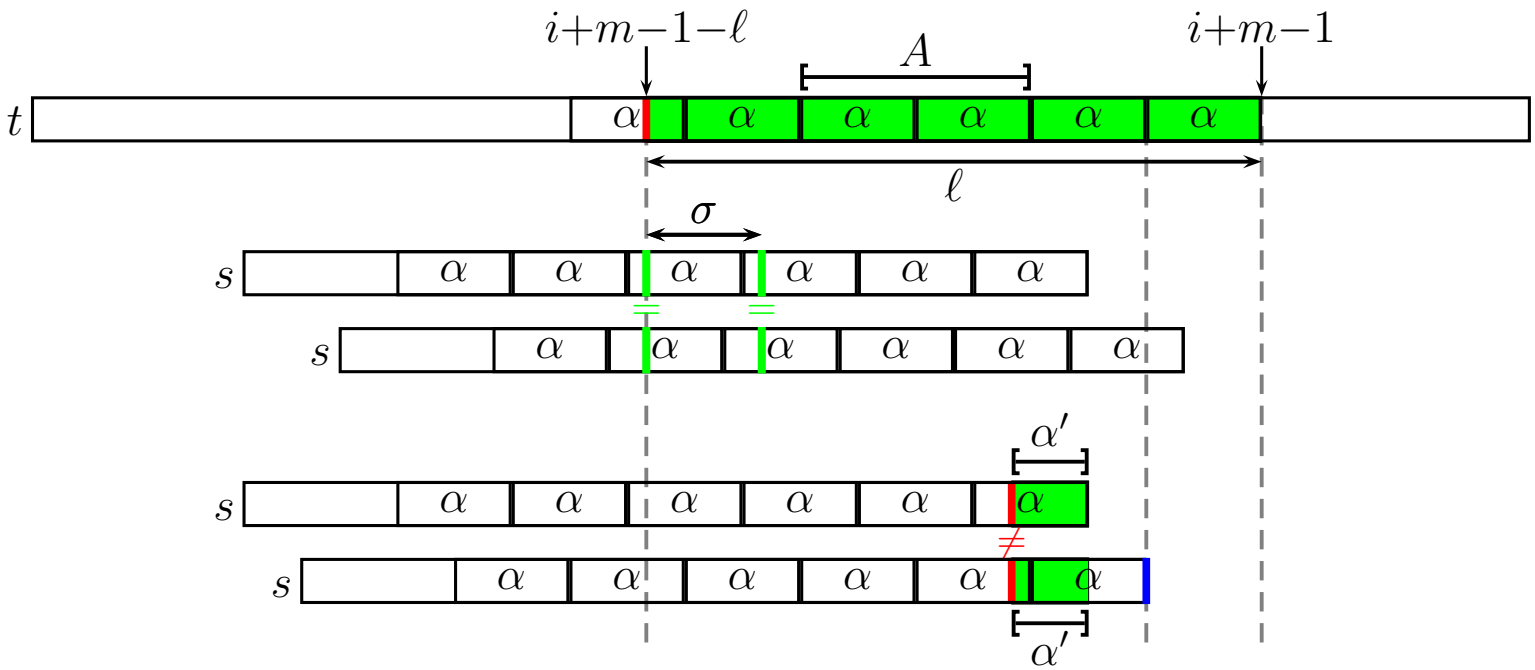
$$A := [i + m - (k-2)\sigma : i + m - 2\sigma - 1]$$

$$(k := 1 + \lfloor \ell/\sigma \rfloor \geq 5)$$

- Zu zeigen:
  Zeichen in t an den Positionen im Intervall A waren bislang noch an keinem Vergleich beteiligt
  (per Widerspruchbeweis)

- Betrachte dazu frühere Versuche, die Vergleiche im Abschnitt A hätten ausführen können.

Lemma: Fall 2 (kurzer Shift)



Lemma: Fall 2 (kurzer Shift)

Betrachte früheren Versuch, bei dem mindestens σ erfolgreiche Vergleiche ausgeführt worden sind (oberer Teil der Abbildung)

- Da mindestens σ erfolgreiche Vergleiche ausgeführt wurden, folgt aus der Periodizität von s', dass alle Vergleiche bis zur Position $i + m - \ell$ erfolgreich sein müssen.

- Der Vergleich an Position $i + m - \ell - 1$ muss hingegen erfolglos sein, da auch der jetzige Versuch, s an Position i in t zu finden, an Position $i + m - \ell - 1$ erfolglos ist.

- Behauptung: Dann hätte jeder sichere Shift gemäß der Strong-Good-Suffix-Rule die Zeichenreihe s auf eine Position größer als i verschoben.

Lemma: Fall 2 (kurzer Shift)

- Annahme: es gibt einen kürzeren sicheren Shift (wie in der Abbildung darunter dargestellt).
- Dann müsste das Zeichen an Position $i + m - \ell - 1$ in t mit einem Zeichen aus s' verglichen werden.
- Dort steht im verschobenen s' aber dasselbe Zeichen wie beim erfolglosen Vergleich des früheren Versuchs, da sich in s' die Zeichen alle σ Zeichen wiederholen.
- Damit erhalten wir einen Widerspruch zur Strong-Good-Suffix-Rule.

Lemma: Fall 2 (kurzer Shift)

Versuche, mit weniger als σ erfolgreichen Vergleichen:

- Da wir nur Versuche betrachten, die Vergleiche im Abschnitt A ausführen, muss der Versuch an einer Position im Intervall $[i-(k-2)\sigma : i-\sigma-1]$ von rechts nach links begonnen haben.
- Nach Wahl von A muss der erfolglose Vergleich auf einer Position größer als $i+m-\ell-1$ erfolgen.
- Betrachte hierzu die erste Zeile im unteren Teil der Abbildung. Sei α' das Suffix von α (und damit von s' bzw. s), in dem die erfolgreichen Vergleiche stattgefunden haben.
- Seien nun $x \neq y$ die Zeichen, die den Mismatch ausgelöst haben, wobei x unmittelbar vor dem Suffix α' in s' steht.
- Offensichtlich liegt α' völlig im Intervall $[i-m-\ell : i+m-\sigma-1]$.

Lemma: Fall 2 (kurzer Shift)

- Wir werden jetzt zeigen, dass ein Shift auf $i - \sigma$ (wie im unteren Teil der Abbildung darunter dargestellt) zulässig ist.
- Die alten erfolgreichen Vergleiche stimmen mit dem Teilwort in s' überein.
- Nach Voraussetzung steht an der Position unmittelbar vor α' in s' das Zeichen x und an der Position des Mismatches das Zeichen y.
- Damit ist ein Shift auf $i - \sigma$ zulässig.
- Da dies aber nicht notwendigerweise der Kürzeste sein muss, erfolgt ein sicherer Shift auf eine Position kleiner gleich $i - \sigma$.

Lemma: Fall 2 (kurzer Shift)

- Erfolgt ein Shift genau auf Position $i - \sigma$, dann ist der nächste Versuch bis zur Position $i + m - \ell - 1$ in t erfolgreich.
- Da wir dann also mindestens σ erfolgreiche Vergleiche ausführen, folgt, wie oben erläutert, ein Shift auf eine Position größer als i (oder der Versuch wäre erfolgreich abgeschlossen worden).
- Andernfalls haben wir einen Shift auf Position kleiner als $i - \sigma$.
- Aber auch dann gilt, dass in allen folgenden Fällen ein Shift genau auf die Position $i - \sigma$ zulässig bleibt.
- Egal, wie viele Shifts ausgeführt werden, irgendwann kommt ein Shift auf die Position $i - \sigma$.
- Also erhalten wir letztendlich immer einen Shift auf eine Position größer als i, aber nie einen Shift auf die Position i.

Lemma: Fall 2 (kurzer Shift)

- Damit ist bewiesen, dass bei einem kurzen Shift die Zeichen im Abschnitt A von t zum ersten Mal verglichen worden sind.
- Da auch der Vergleich des letzten Zeichens von s mit einem Zeichen aus t ein initialer gewesen sein musste, folgt dass mindestens $1 + |A|$ initiale Vergleiche ausgeführt wurden.
- Da $|A| \geq \ell - 4\sigma$, erhalten wir die gewünschte Abschätzung:

$$1 + \ell = (1 + |A|) + (\ell - |A|) \leq (1 + |A|) + 4\sigma.$$

- Da wie schon weiter oben angemerkt, der Vergleich des letzten Zeichens von s immer initial sein muss, und alle Vergleiche im Bereich A initial sind, folgt damit die Behauptung des Lemmas.

Laufzeit der Hauptprozedur

Lemma

Bei einer erfolglosen oder erfolgreichen Suche nach dem ersten Vorkommen von $s \in \Sigma^m$ in $t \in \Sigma^n$ treten maximal $5n + m$ Zeichenvergleiche auf.

Laufzeit der Hauptprozedur

Beweis.

Bezeichne dazu

- $V(n)$ die Anzahl aller Vergleiche, um in einem Text der Länge n zu suchen, und
- I_i bzw. V_i für $i \in [1 : k]$ die Anzahl der initialen bzw. aller Vergleiche, die beim i-ten Versuch ausgeführt wurden,
- σ_i die Länge des Shifts, der nach dem i-ten Vergleich ausgeführt wird.
  War der i-te (und somit letzte) Vergleich erfolgreich, so sei $\sigma_i := m$ (Unter anderem deswegen gilt die Analyse nur für einen erfolglosen bzw. den ersten erfolgreichen Versuch).

Laufzeit der Hauptprozedur

Beweis.

$$
\begin{aligned}
V(n) &= \sum_{i=1}^{k} V_i \quad \text{im letzten Test maximal } m \text{ Vergleiche} \\
&\leq \sum_{i=1}^{k-1} V_i + m \qquad \text{(siehe vorheriges Lemma)} \\
&\leq \sum_{i=1}^{k-1} (l_i + 4\sigma_i) + m \quad \text{(maximal } n \text{ initiale Vergleiche)} \\
&\leq n + 4 \sum_{i} \sigma_i + m \quad \text{Summe der Shifts ist maximal } n \\
&\quad \text{(Anfang von } s \text{ bleibt innerhalb von } t\text{)} \\
&\leq n + 4n + m = 5n + m
\end{aligned}
$$

Laufzeit

Satz

Der Boyer-Moore Algorithmus benötigt für das erste Auffinden des Suchmusters $s \in \Sigma^m$ in einem Text $t \in \Sigma^n$ maximal $5n + m$ Vergleiche.

Satz

Der Boyer-Moore Algorithmus benötigt maximal $3(n + m)$ Vergleiche um zu entscheiden, ob eine Zeichenreihe der Länge m in einem Text der Länge n enthalten ist.

Erweiterung auf alle Vorkommen

- Analyse ist nur für den Fall einer erfolglosen Suche oder dem ersten Auffinden von s in t gültig.
- Das Beispiel $s = a^m$ und $t = a^n$ zeigt, dass der Boyer-Moore-Algorithmus beim Auffinden aller Vorkommen von s in t wieder quadratischen Zeitbedarf bekommen kann
- Diese worst-case Laufzeit lässt sich jedoch mit einem Trick vermeiden: Nach einem erfolgreichen Test $s = t_i \cdots t_{i+m-1}$ verschieben wir das Muster gemäß der Strong-Good-Suffix-Rule
  (Man erhält denselben Shift, wie beim KMP-Algorithmus).
- Nach diesem erfolgreichen Test werden keine Vergleiche mehr mit Zeichen in t an einer Position kleiner als $i + m$ ausgeführt.
- Da wir ja wissen, dass $s = t_i \cdots t_{i+m-1}$ gilt, werden Vergleiche im Bereich kleiner als Position $i + m$ in t jetzt dadurch ersetzt, dass wir testen, ob der Bereich links von der Position $i + m$ ein Rand von s ist.

Erweiterung auf alle Vorkommen

- Mögliche Abfragen, ob $t_j \cdots t_{i+m-1}$ ein Rand ist, werden ja für steigende j gestellt.
- Somit werden die Anfragen nach Rändern nach fallender Länge gestellt. Die absteigende Aufzählung der Ränder von s lässt sich, wie schon gesehen, mit der bereits berechneten Border-Tabelle sehr leicht ausführen.
- Dabei müssen wir die Border-Tabelle im schlimmsten Falle m-Mal durchlaufen.
- Da aber jedes Durchlaufen der Border-Tabelle auch einem Shift von mindestens Eins entspricht, fällt nur ein additiver Zusatzaufwand von $O(m)$ an.
- Gesamtlaufzeit bleibt bei $O(n+m)$