Übungen zur Algorithmischen Bioinformatik I

Blatt 10

Xiheng He

Juni 2021

3. Aufgabe: String-Matching (10 Punkte)

(a) Zeigen Sie die Arbeitsschritte der Algorithmen an folgendem Beispiel:

$$P = \text{ACACBDA}, T = \text{ACACEDAACACBDAA}$$

- des naiven Algorithmus,

  Sehen Abbildung 1.

- des Knuth-Morris-Pratt-Algorithms (KMP)

  Sehen Abbildung 2.

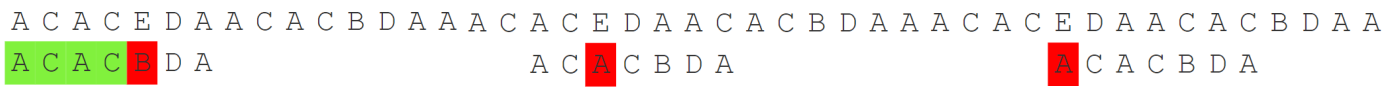
j	0	1	2	3	4	5	6
s[j]	A	C	A	C	B	D	A
border[j]	0	0	1	2	0	0	1

(a) Schritt 1 (b) Schritt 2 (c) Schritt 3

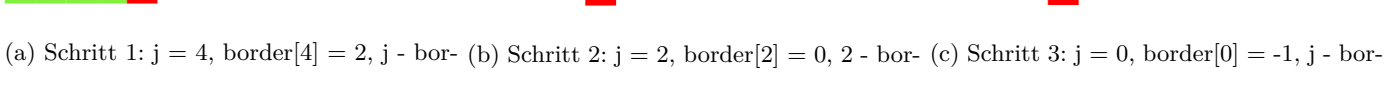
(d) Schritt 4 (e) Schritt 5 (f) Schritt 6

(g) Schritt 7 (h) Schritt 8: Match gefunden

Abbildung 1: Naive Algorithmus

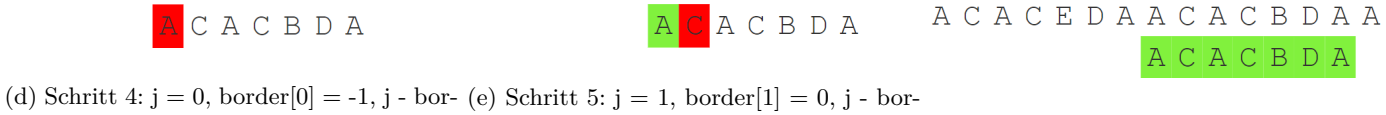


(a) Schritt 1: j = 4, border[4] = 2, j - border[4] = 2

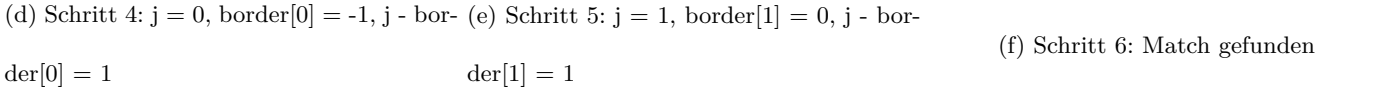


(b) Schritt 2: j = 2, border[2] = 0, 2 - border[2] = 2

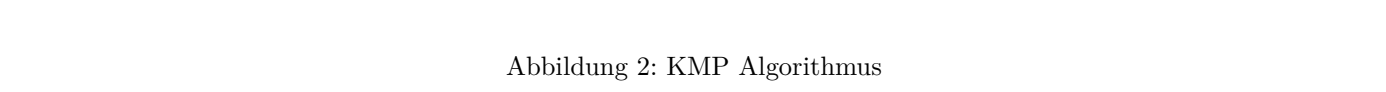
(c) Schritt 3: j = 0, border[0] = -1, j - border[0] = 1



(d) Schritt 4: j = 0, border[0] = -1, j - border[0] = 1



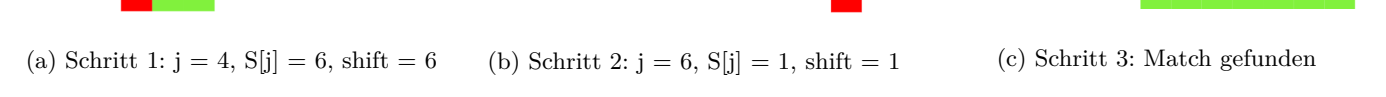
(e) Schritt 5: j = 1, border[1] = 0, j - border[1] = 1



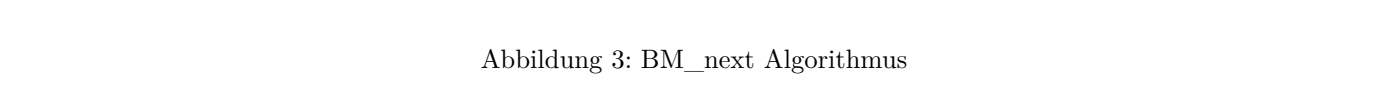
(f) Schritt 6: Match gefunden

Abbildung 2: KMP Algorithmus

(a) Schritt 1: j = 4, S[j] = 6, shift = 6



(b) Schritt 2: j = 6, S[j] = 1, shift = 1



(c) Schritt 3: Match gefunden

Abbildung 3: BM_next Algorithmus

- des Boyer-Moore-Algorithmus (BM) nur mit next-Tabelle (BM_next)

  Sehen Abbildung 3.

j	0	1	2	3	4	5	6	7
S[j]	6	6	6	6	6	4	1	7

- und Boyer-Moore nur mit bad-character-rule (BM_bad-char)

  Sehen Abbildung 4.

a	A	C	B	D
ebc[a]	6	3	4	5

Erstellen Sie dazu auch die zugehörigen *next* (improved) und *skip* Tabellen. Machen Sie deutlich, welche Verschiebungen und Vergleiche durchgeführt werden (z.B durch Zeichnungen!).

(b) Entscheidend für die "besseren" Shifts ist die Nutzung von Informationen über das Pattern, die durch die Zeichenvergleiche gewonnen wurden:

Wenn ein q-Prefix $(q]P)$ des Patterns P und der Stelle $s+1$ im Text T matcht, also

$$q]P := P[1, \ldots, q] = T[s+1, \ldots, s+q]$$

dann soll diese Information für den Shift ausgenutzt werden, so daß $P[1, \ldots k] = T[s'+1, \ldots, s'+k]$ und $s'+k = s+q$ (also ein k-Prefix $k]P$ von P matcht ein k-Suffix $(k[T)$ von T). Da aber dieses k-Suffix von T ein k-Suffix des q-Prefixes von P ist, kann des vorberechnet werden durch Vergleich von P gegen sich selbst.

Zeigen Sie, dass der Shift π im KMP Algorithmus für jedes q genau die (Länge der) maximalen Prefixe von $q]P$ berechnet, die Suffixe von $q]P$ matchten:

$$\pi[q] = max\{k < q | Q = P]q \wedge k]Q = k[Q\}$$

In KMP Algorithmus ist Shift π wie folgende definiert:

$\pi[q] := j - border[j]$ wobei j ist die Position wo Mismatch vorkommt und $border[j]$ der Länge der Prefix und auch Suffix.

$q]P := P[1, \ldots, q] = T[s+1, \ldots, s+q] \Longrightarrow$ Mismatch kommt an q + 1 vor und k sei Länge der Prefix. $\Longrightarrow \pi[q] := q + 1 - k$

Jeder Shift berechnet genau die maximalen Prefixe da Prefixe erweitert werden können somit werden die Prefixe mit maximalen Länge nicht übersehen. D.h. $\pi[q] = max\{k < q | Q = P]q\}$ Gibt's ein von Shift berechnete Prefix, dann gibt näturlich auch ein Suffix in P, so dass zwei Teilfolgen übereinstimmen. D.h. $k]Q = k[Q$ Sonst wurde kein Prefix ausgegeben.

offensichtlich, dass $k < q$, sonst wurde statt Prefix ein Suffix mit eine Teilfolge von T übereinstimen somit ein Match schon vorgekommen ist.