# Übungen zur Algorithmischen Bioinformatik I

# Blatt 10

Xiheng He

Juni 2021

**4. Aufgabe: KMP und BM Implementierung (10 Punkte)**

Implementieren Sie die Algorithmen (Siehe Aufgabe 1): KMP, BM_next und BM_bad-char.

(a) Begründen Sie jeweils die Korrektheit des Algorithmus bzw. Ihrer Implementierung.

- Korrektheit des KMP Algorithmus:

  **Lemma 1.** *Gilt $s_k = t_{i+k}$ für alle $k \in [0 : j-1]$ und $s_j \neq t_{i+j}$, dann ist der Shift $i \to i + j - border[j]$ sicher.*

  Nach Lemma 1 (Lemma 2.9 auf Skript S.112) ist jede Verschiebung um $j - border[j]$ nicht nur sicher sondern auch zulässig. D.h. Wenn $s$ am Ende nicht mehr verschoben werden kann, steht $s$ nicht in $t$. Sonst kann $s$ nach mehrmals Verschiebungen gefunden werden und $s$ kann nicht wegen Verschiebung in $t$ übersprungen werden da jede Verschiebung um $j - border[j]$ sicher und zulässig.

- Korrektheit des BM_next

  Die Shift Tabelle für Good-Suffix-Rule des Boyer-Moore Algorithmus berechnet wesentlich auch die Ränder von Teilwörtern des gesuchten Wortes. Daher ist jede Verschiebungen nach Lemma 1 (Lemma 2.9 auf Skript S.112) auch sicher wenn $s_j \neq t_{i+j}$ und $s_k = t_{i+k}$ für alle $k \in [0 : j-1]$ somit ignoriert der Algorithmus nicht wenn ein Match vorkommt.

- Korrektheit des BM_bad-char

  Analog ist die jede Verschiebung nach "bad-character-rules" auch sicher. In Boyer-Moore Algorithmus ist
  Bei einem Mismatch von $s_j$ mit $t_{i+j}$ wird $s$ so verschoben, dass das rechteste Vorkommen von $t_{i+j}$ in $s$ genau unter $t_{i+j}$ zu liegen kommt damit bedeutet dies, dass es keine übereinstimmende Zeichenfolge $s$ übersprungen werden kann da es nur Mismatch von rechtesten Vorkommen bis Position $j$ steht. Ferner, ist die Verschiebung auch sicher wenn $s$ um $m$ verschoben wird da das Zeichen des Mismatch $t_{i+j}$ in $s$ gar nicht gibt. Der Algorithmus

ist damit korrekt wenn jede Verschiebung sicher ist. Das gesuchte $s$ wird entweder nach mehrmals Verschiebungen gefunden oder existiert in $t$ nicht.

(b) Analysieren Sie die Laufzeit Ihrer Verfahren.

- Laufzeitanalyse des KMP Algorithmus

  **Theorem 1.** *Für die Berechnung der Tabelle border sind maximal* $2m1$ *Zeichenvergleiche notwendig.*

  **Theorem 2.** *Der Algorithmus von Knuth, Morris und Pratt benötigt maximal* $2n + m$ *Zeichenvergleiche, um festzustellen, ob ein Muster* $s$ *der Länge* $m$ *in einem Text* $t$ *der Länge* $n$ *enthalten ist.*

  Nach Theorem 1 (Theorem 2.13 auf Skript S.118) und Theorem 2 (Theorem 2.13 auf Skript S.118) beträgt die Laufzeit des KMP Algorithmus in $O(n + m)$.

- Laufzeitanalyse des BM_next

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

  Nach Theorem 3 (Theorem 2.22 auf Skript S.141) ist die Laufzeit des BM_next in $O(n + m)$.

- Laufzeitanalyse des BM_bad-char

  Da eine Konstruktion der Tabelle für bad-character eine Laufzeit mit $O(|\Sigma|)$ benötigt (für Extended-Bad-Character-Rule wird $O(m \cdot |\Sigma|)$) ist die Laufzeit in $O(n + |\Sigma|)$.

(c) Messen Sie die Laufzeit Ihrer Implementierungen und Vergleichen Sie mit dem naiven Algorithmus!

| input size | $n = 10^4, m = 10^2$ | $n = 10^6, m = 10^2$ | $n = 10^7, m = 10^2$ | $n = 10^8, m = 10^2$ | $n = 10^9, m = 10^3$ |
|---|---|---|---|---|---|
| **Naive** | 0ms | 5ms | 22ms | 113ms | 1688ms |
| **KMP** | 0ms | 5ms | 11ms | 37ms | 482ms |
| **BM_next** | 0ms | 0ms | 4ms | 14ms | 214ms |
| **BM_bad-char** | 0ms | 4ms | 11ms | 46ms | 698ms |

(d) Zählen Sie wieder die Zeichen-Vergleiche um die theoretischen Analysen zu validieren.

Gegeben sei $t$ = ACACEDAACACBDAA, $s$ = ACACBDA, $n = 15, m = 7$

| | praktisch | theoritsch (maximal) |
|---|---|---|
| **Naive** | 14 | 63 |
| **KMP** | 16 | 37 |
| **BM_next** | 14 | 66 |
| **BM_bad-char** | 9 | $O(nm)$ |