Textsuche

Dozent: Prof. Dr. Michael Eichberg
Kontakt: michael.eichberg@dhbw.de, Raum 149B

Version: 1.1.5 [Themed]

Quelle: Die Folien sind teilweise inspiriert von oder basierend auf Lehrmaterial von Prof. Dr. Ritterbusch

---

Folien: https://delors.github.io/theo-algo-text_suche/folien.de.rst.html
https://delors.github.io/theo-algo-text_suche/folien.de.rst.html.pdf

Fehler melden: https://github.com/Delors/delors.github.io/issues

Arrays und Textsuche

Texte können als unsortierte Arrays von Zeichen verstanden werden. Eine typische Frage ist hier das Finden von Textsequenzen im Text.

Einfache Textsuche

```
1  function NaiveTextSearch(text,needle)
2      n := length(text)
3      m := length(needle)
4      for i := 1,...,n-m + 1 do
5          j := 0
6          while text[i + j] == needle[j + 1] do
7              j := j + 1
8              if j == m then
9                  return i // Found at i
10     return nil
```

> **Bemerkung**
>
> Die Laufzeit der einfachen Textsuche kann asymptotisch durch $O(n \cdot m)$ abgeschätzt werden.

Beispiel bei einfacher Suche nach `aaab` in `aaaaaaaab`:

```
a a a a a a a a b
-----------------

a a a b
  a a a b
    a a a b
      a a a b
        a a a b
          a a a b
```

Sind so viele Vergleiche notwendig?

Knuth-Morris-Pratt Verfahren - Grundlagen

Das Verfahren von Knuth-Morris-Pratt vermeidet unnötige Vergleiche, da es zunächst die Suchwortteile auf den größten Rand, also das größte Prefix, das auch Postfix ist, untersucht.

Definition: : Präfix, Postfix und Rand

Für ein Wort $w = (w_1, \ldots, w_n)$ sind die Präfixe $p^{(k)} = (w_1, \ldots, w_k)$ und die Postfixe $q^{(k)} = (w_{n-k+1}, \ldots, w_n)$ für $0 \leq k \leq n$.

Ist $p^{(k)} = q^{(k)} = r^{(k)}$ für ein $0 \leq k < n$, so ist $r(k)$ ein Rand von w.

Für $k < n$ werden $p^{(k)}$ und $q^{(k)}$ auch echte Prä- und Postfixe genannt.

Beispiel/Idee

Text	010110101
Gesucht/Muster	010101
Übereinstimmung	✓✓✓✓✗

Beobachtungen:

1. Wir haben an Stelle 5 ein Mismatch.
2. Wenn wir im Text das Muster um eine Stelle nach rechts verschoben suchen, so haben wir garantiert wieder ein Mismatch.

? Frage

Wie weit kann man also das Muster im Allgemeinen verschoben werden ohne ein Vorkommen zu übersehen?

Beispiel/Idee

	1.	2.
Text	01101100	0102111
Gesucht/Muster	01100	010201
Übereinstimmungen	✓✓✓✓✗	✓✓✓✓✗

Beobachtungen bzgl.:

1. Beim Mismatch an Stelle 5 kann das Muster „nur“ um 3 Stellen nach rechts verschoben werden.
2. Beim Mismatch an Stelle 5 kann das Muster um 4 Stellen nach rechts verschoben werden.

Wie weit wir das Muster verschieben können, hängt also vom Rand des Teils des Musters ab, der bereits übereinstimmt.

Beispiel

Das Wort $aufkauf$ hat die *echten* Präfixe und Postfixe:

$$\{p^{(k)} : 0 \leq k < n\} = \{\varepsilon, a, au, auf, aufk, aufka, aufkau\}$$

$$\{q^{(k)} : 0 \leq k < n\} = \{\varepsilon, f, uf, auf, kauf, fkauf, ufkauf\}$$

und die Ränder:

$$\{r^{(k)} : 0 \leq k < n\} = \{\varepsilon, auf\}.$$

Das bedeutet, dass wenn $aufkauf$ erkannt wurde, die letzten drei Buchstaben schon den nächsten Treffer einleiten können, wie beispielsweise in $aufkaufkauf$.

Das KMP-Verfahren fängt nicht immer von vorne an, sondern prüft, ob ein Rand eines Präfixes $-\varepsilon$ ausgenutzt werden kann. Dazu werden die entsprechenden größten Ränder bestimmt.

Beispiel: ananas

Präfixe $\setminus\{\varepsilon\}$	Größter Rand	Länge des Randes
a	ε	0
an	ε	0
a n ā	a	1
a n ā n̄	an	2
a n ā n̄ ā	ana	3
ananas	ε	0

Beispiel: axaaxax

Präfixe $\setminus\{\varepsilon\}$	Größter Rand	Länge des Randes
a	ε	0
ax	ε	0
a x ā	a	1
a x a ā	a	1
a x a ā x̄	ax	2
a x a ā x̄ ā	axa	3
a x a a x ā x̄	ax	2

Die Idee ist also, dass wir beim Musterabgleich nach einem Mismatch, wenn der übereinstimmende Teil einen Rand hat, beim Abgleich des Musters an einer späteren Stelle - basierend auf der Größe des Randes - weitermachen können. Wir müssen also nicht immer das ganze Muster von vorne anfangen zu vergleichen.

Übung

0.1. Ränder und Randlängen bestimmen

Bestimmen Sie die Ränder und die Längen der $Präfixe - \varepsilon$ für die Worte:

1. $tultatul$
2. $eikleike$
3. $okokorok$
4. $trattrad$

Knuth-Morris-Pratt Verfahren

```
function ComputePrefixFunction(needle)
    m := length(needle)
    sei B[1...m] ein Array // Array für die Längen der Ränder der Teilworte
    B[1] := 0
    j := 0 // j ist die Länge des Randess
    for i := 2,...,m do
        j := j + 1
        while j > 0 and needle[j] ≠ needle[i] do
            if j > 1 then
                j := B[j-1] + 1
            else
                j := 0
        B[i] := j
    return B
```

Komplexität: $O(m)$

```
function KMP(text,needle)
    n := length(text), m := length(needle)
    B := ComputePrefixFunction(needle)
    q := 0                  // Anzahl der übereinstimmenden Zeichen
    R := []                 // Ergebnisliste der Indizes der Übereinstimmungen
    for i := 1,...,n do
        while q > 0 and needle[q + 1] ≠ text[i] do
            q := B[q]       // ... die nächsten Zeichen stimmen nicht überein
        if needle[q + 1] == text[i] then
            q := q + 1      // Übereinstimmung
        if q == m then
            R append (i - m + 1)
            q := B[q]       // Suche nach nächster Übereinstimmung
    return R
```

Komplexität: $O(n + m)$

Details ComputePrefixFunction

Die Funktion $ComputePrefixFunction$ berechnet die größten Werte der Präfixe für das Suchwort $needle$ der Länge m und gibt diese als Array (B) zurück. Das Array B enthält somit die größten Ränder der Präfixe $needle[1, \ldots, i]$. (Der Wert von $B[1]$ ist immer 0, da es keinen Rand gibt.)

Beispiel für eine KMP-Textsuche

Gesucht wird `ananas` in `saansanananas`

```
   s a a n s a n a n a n a s
i  ———————————————————————
1  a̶
...
3    a n̶
...
5      a n a̶
...
11           a n a n a s̶       Beim Auftreten des Mismatch (ln 7) ist
...                            q=5 und wird auf p[5]=3 (ln 8) gesetzt
13               a n a n a s
```

> **Bemerkung**
>
> Dargestellt sind die Fälle, in denen ein Mismatch auftritt. `i` ist der Index des aktuellen Zeichen im Text, das mit dem Muster verglichen wird.

Übung

0.2. KMP-Algorithmus

Bestimmen Sie die Randlängen der Muster und stellen Sie die Teilschritte bei der Durchführung des KMP-Algorithmus zur Suche des Wortes/Muster im Text dar.

Stellen Sie insbesondere die Fälle dar in denen ein Mismatch auftritt.

Muster	Text
`aaab`	`aaaaaaaab`
`barbara`	`abbabarabarbarbara`

Boyer-Moore-Algorithmus (vereinfacht)

Beobachtung

Häufig ist das Alphabet des Textes größer als das des Musters.

- Der Algorithmus vergleicht das Muster (Pattern) von rechts nach links mit dem Text.

  Viele andere Algorithmen führen die Vergleiche von links nach rechts durch.
- Der Boyer-Moore-Algorithmus nutzt dies aus, indem er die Verschiebung des Musters anhand des letzten Zeichens des Musters und des Textes vornimmt.

Wird beispielsweise das Wort `Banane` im Text `Orangen,␣Ananas␣und␣Bananen` gesucht, so wird zunächst die Sprungtabelle für das verwendete Alphabet in Bezug auf das Suchwort (`Banane` mit Länge 6) bestimmt:

Zeichen im Text	␣	,	A	B	O	a	d	e	g	n	r	s	u
Sprung	6	6	6	5	6	2	6	0	6	1	6	6	6

```
O r a n g e n , ␣ A n a n a s ␣ u n d ␣ B a n a n e n
B a n a n e
          B a n a n e
            B a n a n e
              B a n a n e
                  B a n a n e
                              B a n a n e
                                  B a n a n e
                                      B a n a n e
                                        B a n a n e
```

Bemerkung

Unterschrichen sind die durchgeführten Vergleiche. Die Verschiebung des Musters erfolgt anhand des letzten Zeichens des Musters und des Textes, dass nicht übereinstimmt. Dabei ist die Verschiebung durch das Zeichen des Textes gegeben, das nicht mit dem Muster übereinstimmt.

Komplexität

Im guten und häufigen Fall erreicht das Verfahren $O(\frac{n}{m})$, aber in speziellen Fällen ist auch $O(n \cdot m)$ möglich.

Bei der Sprungtabelle handelt es sich um eine Tabelle, die *für jedes Zeichen des Alphabets des Textes* die Verschiebung des Musters angibt, wenn das Zeichen im Text mit dem Muster nicht übereinstimmt. Die Zeichen `A`, `O`, `d`, `g`, `r`, `s`, `u`, `,` und das Leerzeichen haben die größte Verschiebung, da sie nicht im Muster vorkommen. Das Zeichen `e` hat die kleinste Verschiebung, da es das letzte Zeichen des Musters ist. Das Zeichen `n` hat eine Verschiebung von 1, da es im Muster als vorletztes Zeichen vorkommt, das Zeichen `a` hat eine Verschiebung von 2, da das späteste Vorkommen an drittletzer Stelle ist, und das Zeichen `B` hat eine Verschiebung von 5, da es nur einmal vorkommt und das erste Zeichen des Musters ist.

Übung - Boyer-Moore-Algorithmus

0.3. „belli“

Suchen Sie das Wort

```
belli
```

im Text

```
It is a dark time for the Rebellion.
```

0.4. "barbara"

Suchen Sie das Wort

```
barbara
```

im Text

```
abbabarabarbarbara
```