Übersicht

Übersicht

Textalgorithmen – Stringmatching

Eingabe: Zwei Strings v und w

Frage: Kommt v als Unterstring in w vor?

Genauer:

$u, v \in \Sigma^*$, wobei Σ ein Alphabet ist.

Gibt es $x, y \in \Sigma^*$, so daß $xvy = w$ gilt?

Freie Monoide

Definition

Ein Alphabet Σ ist eine nichtleere, endliche Menge von Symbolen.

Ein Monoid $(M, e, \circ)$ ist eine Halbgruppe $(M, \circ)$ mit einem neutralem Element e.

Ein Monoid $(M, e, \circ)$ ist von $\Sigma \subseteq M$ frei erzeugt, wenn sich jedes $w \in M$ eindeutig als $w = a_1 \circ \cdots \circ a_n$ darstellen läßt, wobei $a_i \in \Sigma$.

Wir bezeichnen das von einem Alphabet Σ frei erzeugte Monoid mit Σ^*.

RWTH AACHEN UNIVERSITY

Homomorphismen

Definition

Es seien $(M_1, e_1, \circ)$ und $(M_2, e_2, \cdot)$ zwei Monoide.

Eine Abbildung $h\colon M_1 \to M_2$ ist ein (Monoid-)Homomorphismus, falls

1. $h(x \circ y) = h(x) \cdot h(y)$ für alle $x, y \in M_1$
2. $h(e_1) = e_2$

Ein bijektiver Homomorphismus ist ein Isomorphismus.

Freie Monoide

Theorem

Ist ein Monoid $(M, e, \circ)$ von einer Basis $B \subseteq M$ frei erzeugt, dann ist ein Homomorphismus auf M durch seine Bilder auf B bereits eindeutig bestimmt.

Beweis.

Es sei $h : M \to X$ ein Homomorphismus.
Dann ist $h(w) = h(a_1 \circ \ldots \circ a_n) = h(a_1) \cdot \ldots \cdot h(a_n)$. □

Theorem

Ein von einem Alphabet Σ frei erzeugtes Monoid ist bis auf Isomorphismus eindeutig bestimmt.

Beweis.

Es seien $(M_1, e_1, \circ)$ und $(M_2, e_2, \cdot)$ von Σ frei erzeugte Monoide.

Es sei $h\colon M_1 \to M_2$ ein Homomorphismus, den wir vermöge $h(a) = a$ für $a \in \Sigma$ festlegen.

Behauptung: h ist ein Isomorphismus.

Wir müssen beweisen, daß h injektiv und surjektiv ist.

- $u \neq v \Rightarrow u_1 \dots u_n \neq v_1 \dots v_m \Rightarrow h(u_1) \dots h(u_n) \neq h(v_1) \dots h(v_m) \Rightarrow h(u) \neq h(v)$
- $u \neq v \Leftarrow u_1 \dots u_n \neq v_1 \dots v_m \Leftarrow h(u_1) \dots h(u_n) \neq h(v_1) \dots h(v_m) \Leftarrow h(u) \neq h(v)$

AACHEN UNIVERSITY

Stringmatching

Sei Σ ein Alphabet.

1. $u = a_1 \dots a_n \in \Sigma^*$ mit $a_i \in \Sigma$
2. $v = b_1 \dots b_m \in \Sigma^*$ mit $b_i \in \Sigma$

Frage: Kommt v in u vor?

Genauer: Gibt es ein j, so daß $b_i = a_{j+i}$ für alle $1 \leq i \leq m$?

Direkter Algorithmus: Probiere alle j.

a	b	r	a	c	a	d	a	b	r	a

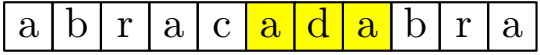
Fenster der Länge m gleitet über u.

Laufzeit $\Theta(n \cdot m)$

RWTH AACHEN UNIVERSITY

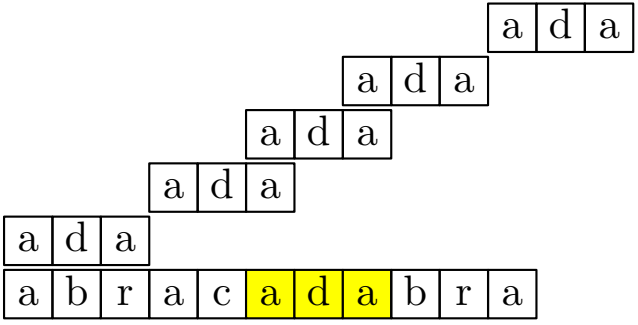
Stringmatching

Wie kann der naive Algorithmus verbessert werden?



1. Von hinten vergleichen
2. Fenster möglichst weit verschieben
3. v vor der Suche analysieren und Hilfstabellen erstellen

Stringmatching



Verschiebe Fenster um $\delta(x)$, wenn x der erste Mismatch von hinten ist.

Wie berechnen wir $\delta(x)$?

Algorithmus von Boyer und Moore (vereinfacht)

Algorithmus

```
function Boyer − Moore1(u, v) integer :
i := |v|; j := |v|;
repeat
  if u[i] = v[j]
  then i := i − 1; j := j − 1
  else
    if|v| − j + 1 > delta(u[i]) then i := i + |v| − j + 1
                               else i := i + delta(u[i]) fi;
    j := |v|
  fi
until j < 1 or i > |u|;
return i + 1
```

AACHEN UNIVERSITY

Algorithmus von Boyer und Moore

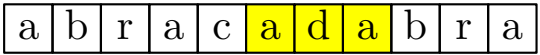
Diese einfache Variante hat Worst-Case-Laufzeit $O(|u| \cdot |v|)$.

Praktisch ist sie aber sehr bewährt und zeigt oft die Laufzeit $O(|u|/|v|)$.

Durch eine Modifikation läßt sich die Worst-Case-Laufzeit $O(|u| + |v|)$ erzielen.

In der Praxis lohnt sich das allerdings nicht.

Algorithmus von Rabin und Karp



Gegeben u und v mit $|v| = m$.
Wähle eine geeignete Hashfunktion $\Sigma^* \to \mathbf{N}$.

1. Berechne $h(v)$ und $h(u[i \dots i+m-1])$ für $i = 1, \dots, |u| - m + 1$.
2. Vergleiche v mit $u[i \dots i+m-1]$ falls Hashwerte übereinstimmen.

Bei einer guten Hashfunktion unterscheiden sich die Hashwerte fast immer, wenn die Strings verschieden sind.

Wie lange dauert es alle $h(u[i \dots i+m-1])$ zu berechnen?

Algorithmus von Rabin und Karp

Wähle eine Hashfunktion, so daß alle

$$h(u[i \dots i+m-1])$$

in $O(m+|u|)$ Schritten berechnet werden können.

Zum Beispiel:

$$h(a_1 \dots a_n) = \left(\sum_{i=1}^{n} q^i a_i \right) \bmod p,$$

wobei p und q große Primzahlen sind.

Es gilt

$$h(a_2 \dots a_{n+1}) = \left(h(a_1 \dots a_n)/q - a_1 + q^n a_{n+1} \right) \bmod p.$$

RWTH AACHEN UNIVERSITY

Übersicht

Editdistanz

Gegeben seien zwei Zeichenketten u und v.

Wir dürfen u auf drei Arten ändern:

1. Ersetzen: Ersetze ein Symbol in u durch ein anderes
2. Löschen: Entferne ein Symbol aus u (Länge wird kleiner)
3. Einfügen: Füge ein Symbol an beliebiger Stelle in u ein (Länge wird größer)

Frage: Wieviele solche Operationen sind mindestens nötig, um u zu v zu verwandeln.

Anwendung: Wie ähnlich sind die beiden Zeichenketten (z.B. DNA-Strings)?

Editdistanz

Lösung durch dynamisches Programmieren.

GACGTCAGCTTACGTACGATCATTTGACTACG

GACGTCAGCTACAGTAGATCATTTGACTACG