Handout für das Seminar über Algorithmen betreut von Prof. Dr. Helmut Alt, FU-Berlin

Editierabstand und der '4-Russen-Trick'

Marco Träger

23.06.2011

1 Editierabstand in $O(n \cdot m)$

1.1 Definitionen

Σ endliches Alphabet
$S, T \subset \Sigma^*$ endliche Strings, $|S| = n, |T| = m$

Eine Editiersequenz $\mathbb{E}_T^S$ ist eine Sequenz von 4 Operationen

- I Insert - einfügen eines Zeichens
- D Delete - löschen eines Zeichens
- R Replace - ersetzen eines Zeichens
- M Match - keine Operation

so dass $\mathbb{E}_T^S(S) = T$

Beispiele

	$S:$	H	E	L	L	O			$S:$	H	$\_$	E	L	L	O			$S:$	H	$\_$	E	L	L	O
Beispiel 1:	$T:$	B	A	L	L	$\_$	Beispiel 2:		$T:$	$\_$	B	A	L	L	$\_$	Beispiel 3:		$T:$	B	A	$\_$	L	L	$\_$
	$\mathbb{E}_{T_1}^S:$	R	R	M	M	D			$\mathbb{E}_{T_2}^S:$	D	I	R	M	M	D			$\mathbb{E}_{T_2}^S:$	R	I	D	M	M	D

Definitionen

$c(\mathbb{E}_T^S)$ Kosten der Editiersequenz $\mathbb{E}_T^S$

D(S,T) Minimale Kosten aller möglichen Editiersequenzen $\mathbb{E}_T^S$, auch 'Editierabstand' oder 'Lewenshtein-Abstand'

$\mathbb{E}_{T\,opt}^S$ eine Editiersequenz mit minimalen Kosten, es gilt also $c(\mathbb{E}_{T\,opt}^S) = D(S,T)$

$D(i,j)$ $:= D(S[1..i], T[1..j])$, $S[1..0] = \epsilon$

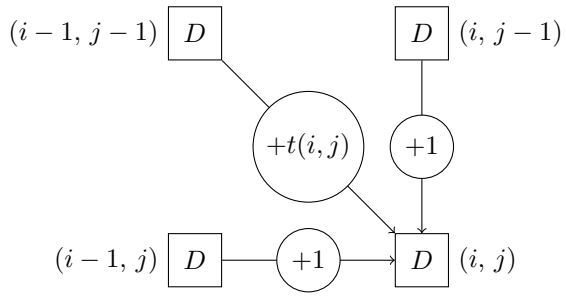
1.2 Rekursive Definition

$$\forall i : D(i,0) = i$$
$$\forall j : D(0,j) = j$$
$$\forall i,j \geq 1 : D(i,j) = \min \begin{Bmatrix} D(i-1,j)+1 \\ D(i,j-1)+1 \\ D(i-1,j-1)+t(i,j) \end{Bmatrix}$$
$$\text{wobei } t(i,j) = \begin{cases} 1 : S(i) \neq T(j) & \text{(mismatch)} \\ 0 : S(i) = T(j) & \text{(match)} \end{cases}$$

Korollar Der Editierabstand D(S,T) zwischen S und T kann bottom-up über die Methode der dynamischen Programmierung in Zeit $O(n \cdot m)$ berechnet werden. Diese Rechnung führt zu einer $(n+1) \times (m+1)$ Tabelle. Sie besitzt folgende Form:

	$\_$	S_1	S_2	$\cdots$	S_{n-1}	S_n
	0	1	2	$\cdots$	n-1	n
$\_$	1	$D(1,1)$	$D(2,1)$		$D(n-1,1)$	$D(n,1)$
T_1	2	$D(1,2)$	$D(2,2)$		$D(n-1,2)$	$D(n,2)$
	$\vdots$			$\ddots$		
T_{m-1}	$m-1$	$D(1,3)$	$D(2,3)$		$D(n-1,m-1)$	$D(n,m-1)$
T_m	m	$D(1,4)$	$D(2,4)$		$D(n-1,m)$	$D(n,m)$

wobei sich der Eintrag $D(i,j)$ an Stelle (i,j) aus den Einträgen an den Stellen $(i-1,j),(i,j-1),(i-1,j-1)$ bestimmen lässt.



1.3 Dynamische Programmierung

- initialisiere die erste Zeile und Spalte der Tabelle mit $D(i,0) = i$ und $D(0,j) = j$
- bestimme, beginnend mit der 2. Zeile, zeilenweise den Wert $D(i,j)$ mit Hilfe der Nachbarn $(i-1,j)$, $(i,j-1)$, $(i-1,j-1)$ und $S(i)$, $T(j)$
- lese $D(n,m)$ aus der Zelle (n,m) aus

Satz
Die Laufzeit des dynamischen Programmier-Algorithmus für den Editierabstand beträgt $O(n \cdot m)$

2 Der '4-Russen-Trick'

Diese Idee stammt von Arlazarov, Dinic, Kronrod, Faradzev aus einem Paper über Boolsche Matrixmultiplikation. [?] referenziert in [?, S. 302]

- gegeben ein dynamischer Programmier-Algorithmus
- berechne nicht die kleinsten Teilproblem als Anker
- sondern Teilproblem einer festen Größe t
- gibt es 'wenige' unterschiedliche Problem der Größe t kann dies ein Laufzeitverbesserung darstellen

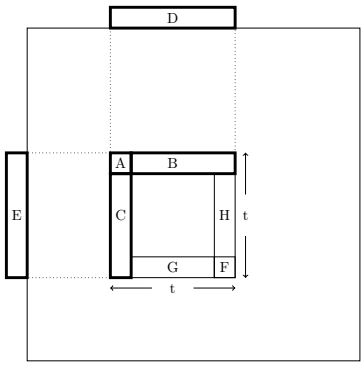
3 Editierabstand in $O(\frac{n^2}{\log n})$

3.1 t-Block

Ein $t \times t$ Ausschnitt der Tabelle zur Berechnung des Editierabstandes.

Lemma Ein t-Block der in Position (i_0, j_0) startet ist eine Funktion der Kosten des oberen linken Eintrags $[A]$, der 1. Zeile $[B]$ und der 1. Spalte $[C]$ des t-Blockes und der Teilstrings $S[i_0..i_0+t-1]$ $[D]$ und $T[j_0..j_0+t-1]$ $[E]$, hängen also nur von diesen ab, und liefert den unteren rechten Eintrag $[F]$, die letzte Zeile $[G]$ und die letzte Spalte $[H]$.

$\mathbb{B}(A, B, C, D, E) = (F, G, H)$ ist die so genannte t-Block-Funktion.



3.2 Exkurs: Memoization-Trie

Quelle: Blog von Conal Elliott [**?**]

Geben eine Funktion $f :: p_1 \times .. \times p_n \to A$ mit Parametern $p_1 \in \Sigma_1^* \cdots p_n \in \Sigma_n^*$ Wobei Σ_i konstante endliche Alphabete.

Der Memoization-Trie M_f ist ein Trie dessen Kanten mit Symbolen der Alphabete $|\Sigma_1| \cdots |\Sigma_n|$ beschriftet sind. Wobei wenn man das Wort $p_1 p_2 \cdots p_n$ auf dem Weg von der Wurzel zu einem Blatt ließt, das Blatt den Funktionswert $f(p_1, p_2, \cdots, p_n)$ enthält.
Dafür hat M_f zwei Operationen

- $M_f.memo(p_1, \cdots p_n) :=$ speichert $f(p_1 \cdots p_n) \in A$ im Trie M_f am Ende des Weges $p_1 \cdots p_n$
- $M_f.unmemo(p_1, \cdots p_n) :=$ ließt $f(p_1 \cdots p_n) \in A$ aus dem Trie M_f, indem dem Weg $p_1 \cdots p_n$ gefolgt wird

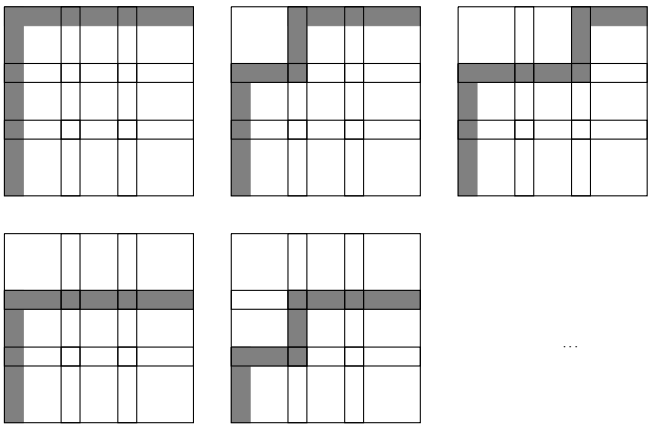
da alle Σ_i konstante endliche Alphabete sind, kostet

- $M_f.memo(p_1, \cdots, p_n)$ $O(|p_1 \cdots p_n| +$ Laufzeit von f)
- $M_f.unmemo(p_1, \cdots, p_n)$ $O(|p_1 \cdots p_n| +$ Größe der Ausgabe)

3.3 Block-Editierabstand-Algorithmus

$(|S| = |T| = n)$

- (Vorverarbeitung) wir berechnen den $M_{\mathbb{B}}.memo(..)$ der t-Blockfunktion $\mathbb{B}$ (festes t) für alle möglichen Eingaben
- überdecke die $(n+1) \times (n+1)$ Tabelle mit t-Blöcken, so dass sich jeweils die letzte Zeile (Spalte) eines Blockes mit der ersten des nächten überdeckt.
- initialisiere die erste Zeile und Spalte der Tabelle mit
  D(i, 0) = i und D(0,j) = j
- bestimme block-zeilenweise, mittels $M_{\mathbb{B}}.unmemo(..)$, die Werte der *letzte Zeile und Spalte* jedes t-Blocks
- lese $D(n, n)$ aus der Zelle (n, n) aus



Satz Der Block-Editierabstand-Algorithmus hat Laufzeit

$$O\left(\frac{n^2}{t} + P_t(n)\right)$$

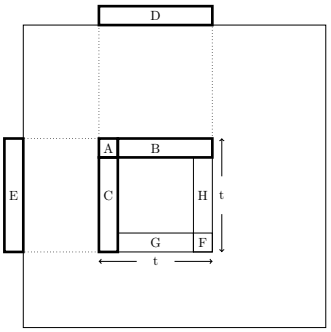
wobei $P_t(n)$ die Zeit ist $M_{\mathbb{B}}.memo(..)$ für alle möglichen Eingaben zu berechnen.

Beweisskitze Die Tabelle wurde mit $\Theta(\frac{n^2}{t^2})$ t-Blöcken überdeckt. Man beachte, das wenn $M_{\mathbb{B}}.memo(..)$ für alle möglichen Eingaben berechnet wurde, ist für alle $\Theta(\frac{n^2}{t^2})$ t-Blöcke $\mathbb{B}(A, B, C, D, E) = (F, G, H)$ mit Hilfe von $M_{\mathbb{B}}.unmemo(..)$ in $O(t)$ berechenbar ist.

Diese Überlegung führt zur gewüschten Laufzeit.

3.4 Berechnung der t-Block-Funktion $\mathbb{B}$ für alle möglichen Eingaben

Ein naiver Versuch



- $A \in \{0..n\}$
- $B \in \{0..n\}^{t-1}$
- $C \in \{0..n\}^{t-1}$
- S-Teilstring $D \in \Sigma^t$
- T Teilstring $E \in \Sigma^t$

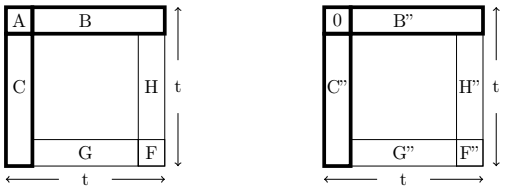
$$P_t(n) \in \quad O\left(\overbrace{(n+1)^t}^{A} \cdot \overbrace{(n+1)^{2(t-1)}}^{B+C} \cdot \overbrace{|\Sigma|^{2t}}^{D+E} \cdot \overbrace{t^2}^{\text{dyn. Prog.}}\right)$$

Lemma
Für jede Zelle (i,j) der Tabelle zur Bestimmung des Editierabstandes unterscheidet sich der Abstand $D(i, j)$ um maximal 1 vom linken, rechten, oberen und unteren Nachbarn der Zelle.

Korrolar
Damit ist eine Offset-Kodierung B' und C' von B und C möglich, die lediglich den Unterschied zum linken (oberen) Nachbarn speichert.
Damit sind $B', C' \in \{-1, 0, +1\}^{t-1}$ und es gibt 3^{t-1} Möglichkeiten für B' und analog für C'.

$$P_t(n) \in \quad O\left(\overbrace{(n+1)^t}^{A} \cdot \overbrace{3^{2(t-1)}}^{B'+C'} \cdot \overbrace{|\Sigma|^{2t}}^{D+E} \cdot \overbrace{t^2}^{\text{dyn. Prog}}\right)$$

Verbesserung
Anstatt den Memoization-Trie der Funktion $\mathbb{B}(A, B, C, D, E) = (F, G, H)$ zu berechnen,
berechnen wir die Memoization-Trie der Funktion $\mathbb{B}'(0, B'', C'', D, E) = (F'', G'', H'')$ indem wir von jedem Wert in B, C, D den Wert in A abziehen.
Wir erhalten (F, G, H) indem wir auf die jeden Eintrag der Ausgabe von $\mathbb{B}'$ wieder A addieren.



$$P_t(n) \in \quad O\left(\overbrace{3^{2(t-1)}}^{B''+C''} \cdot \overbrace{|\Sigma|^{2t}}^{D+E} \cdot \overbrace{t^2}^{\text{dyn. Prog}}\right)$$

3.5 Analyse

Satz Gegeben 2 Strings S, T mit $|S| = |T| = n$.
Der Editiertabstand $D(S, T)$ lässt sich in $O\left(\frac{n^2}{\log n}\right)$ berechnen.

Beweis Wir wissen aus den vorherigen Betrachtungen, die Laufzeit ist:

$$\overbrace{O\left(\frac{n^2}{t}\right)}^{\text{Algorithmus}} + \overbrace{O\left(3^{2(t-1)} \cdot |\Sigma|^{2t} \cdot t^2\right)}^{\text{Vorverarbeitung}}$$

setze nun $t := \frac{\log_{3|\Sigma|} n}{2}$, dann folgt die Behauptung.
[?]

Literatur

[ADKF70] V. Arlazarov, E. Dinic, M. Kronrod, and I. Faradzev. On economical construction of the transitive closure of a directed graph. 1970.

[Ell] Conal Elliott. Memoization in functional programming languages, http://conal.net/blog/tag/memoization/.

[Gus97] Dan Gusfield. *Algorithms on Strings, Trees and Sequences.* 1997. ISBN 0-521-58519-8. 302–307 pp.

[vSHzVvDFHin] Vorlesungsskript von Stephan Hell zur Vorlesung von Dr. Frank Hoffman. *Aktuelle Forschungsthemen der Algorithmik, String Matching Algorithmen.* WS 09/10, Freie Universität Berlin.