Informatik II für Verkehrsingenieure
Edit-Distanz

Janis Voigtländer

Technische Universität Dresden

Sommersemester 2007

Überblick

Problemstellung

Formalisierung

Effiziente Berechnung

Retracing

Dateivergleich

```
void sort(int L,int R)
{ int i,j,w,x,k;
  i=L; j=R; k=(L+R)/2; x=a[k];
  do
    { while (a[i]<x) i++;
      while (a[j]>x) j--;
      ...
```

```
void sort(int L,int R)
{ int i,j,w,x,k;
  i=L; j=R;
  k=(L+R)/2; x=a[k];
  do
    { while (a[i]<x) i++;
      while (a[j]<x) j--;
      ...
```


Dateivergleich

```
void sort(int L,int R)               | void sort(int L,int R)
{ int i,j,w,x,k;                     | { int i,j,w,x,k;
  i=L; j=R; k=(L+R)/2; x=a[k];       |   i=L; j=R;
  do                                 |   k=(L+R)/2; x=a[k];
    { while (a[i]<x) i++;            |   do
      while (a[j]>x) j--;            |     { while (a[i]<x) i++;
      ...                            |       while (a[j]<x) j--;
                                     |       ...
```

Problem: Ab der dritten Zeile würden die Dateien als komplett verschieden interpretiert!

Ziel: intelligenterer Vergleich

Problemstellung

Abstraktion:
- lediglich Vergleich einzelner Zeichen, nicht ganzer Zeilen
- als mögliche Veränderungen nur Löschung, Einfügung oder Änderung eines Zeichens
- Anzahl solcher Operationen als Vergleichsmaß

Beispiel: Eingabe „abcbcba" und „acbda"
$\rightsquigarrow$ mögliche Erklärungen des Unterschieds:

a	b	c	b	c	b	a
a	c	b	d	a	–	–

a	b	c	b	c	b	–	a
a	–	–	–	c	b	d	a

a	b	c	b	c	b	a
a	–	c	b	d	a	–

Formalisierung

Gegeben: `char s[n], t[m];`

Gesucht: Distanz d zwischen `s` und `t` als minimale Anzahl von Edit-Operationen

Idee: zunächst Entscheidung, ob `s[0]` und `t[0]` einander gegenübergestellt, oder eines der beiden gelöscht/eingefügt werden soll $\rightsquigarrow$ 3 Fälle:

- ```
  s[0]  ··· (s[1]...s[n-1])
  t[0]  ··· (t[1]...t[m-1])
  ```
- ```
  s[0]  ··· (s[1]...s[n-1])
   –    ··· (t[0]...t[m-1])
  ```
- ```
   –    ··· (s[0]...s[n-1])
  t[0]  ··· (t[1]...t[m-1])
  ```
  ```

Rekursionsgleichungen (I)

Gesucht: allgemein Distanz $d_{i,j}$ zwischen `s[i]...s[n-1]` und `t[j]...t[m-1]`

Ansatz: Fallunterscheidung (wie eben) ergibt:

- `s[i]` ⋯ (`s[i+1]...s[n-1]`)
  – ⋯ (`t[j]...t[m-1]`)
  $\rightsquigarrow 1 + d_{i+1,j}$
- – ⋯ (`s[i]...s[n-1]`)
  `t[j]` ⋯ (`t[j+1]...t[m-1]`)
  $\rightsquigarrow 1 + d_{i,j+1}$
- `s[i]` ⋯ (`s[i+1]...s[n-1]`)
  `t[j]` ⋯ (`t[j+1]...t[m-1]`)
  $\rightsquigarrow \begin{cases} d_{i+1,j+1} & \text{wenn } \texttt{s[i]==t[j]} \\ 1 + d_{i+1,j+1} & \text{sonst} \end{cases}$

Insgesamt ist das Minimum zu wählen!

Rekursionsgleichungen (II)

Sonderfälle: $i = n$ oder $j = m$:

- – ... –
  `t[j]...t[m-1]`
  $\rightsquigarrow m - j$
- `s[i]...s[n-1]`
  – ... –
  $\rightsquigarrow n - i$

Insgesamt:

$$d_{i,j} = \begin{cases} m - j & \text{wenn } i = n \\ n - i & \text{wenn } j = m \\ min\,\{1 + d_{i+1,j}\,;\, 1 + d_{i,j+1}\,;\, d_{i+1,j+1}\} & \text{wenn } \texttt{s[i]==t[j]} \\ 1 + min\,\{d_{i+1,j}\,;\, d_{i,j+1}\,;\, d_{i+1,j+1}\} & \text{sonst} \end{cases}$$

Beispiel

Eingabe: `s[0]...s[6]`=abcbcba und `t[0]...t[4]`=acbda

Berechnung:

$$
\begin{aligned}
d_{0,0} &= min\,\{1 + d_{1,0} \,;\, 1 + d_{0,1} \,;\, d_{1,1}\} \\
&= min\,\{1 + 1 + min\,\{d_{2,0} \,;\, d_{1,1} \,;\, d_{2,1}\} \,;\, 1 + d_{0,1} \,;\, d_{1,1}\} \\
&= min\,\{2 + min\,\{d_{2,0} \,;\, d_{1,1} \,;\, d_{2,1}\} \,; \\
&\qquad\quad 1 + 1 + min\,\{d_{1,1} \,;\, d_{0,2} \,;\, d_{1,2}\} \,;\, d_{1,1}\} \\
&= min\,\{2 + min\,\{d_{2,0} \,;\, d_{1,1} \,;\, d_{2,1}\} \,; \\
&\qquad\quad 2 + min\,\{d_{1,1} \,;\, d_{0,2} \,;\, d_{1,2}\} \,; \\
&\qquad\quad 1 + min\,\{d_{2,1} \,;\, d_{1,2} \,;\, d_{2,2}\}\} \\
&= min\,\{2 + min\,\{1 + min\,\{d_{3,0} \,;\, d_{2,1} \,;\, d_{3,1}\} \,;\, d_{1,1} \,;\, d_{2,1}\} \,; \\
&\qquad\quad 2 + min\,\{d_{1,1} \,;\, d_{0,2} \,;\, d_{1,2}\} \,; \\
&\qquad\quad 1 + min\,\{d_{2,1} \,;\, d_{1,2} \,;\, d_{2,2}\}\} \\
&= \cdots
\end{aligned}
$$

Problem: mehrfache Berechnung der gleichen Werte

Lösung: dynamische Programmierung

Dynamische Programmierung

Idee:

- Speicherung aller $d_{i,j}$ in Tabelle
- Berechnung in geeigneter Reihenfolge

Beispiel:

	a	b	c	b	c	b	a	
a	3	3	3	2	2	3	4	5
c	4	3	2	2	1	2	3	4
b	5	4	3	2	2	1	2	3
d	6	5	4	3	2	1	1	2
a	6	5	4	3	2	1	0	1
	7	6	5	4	3	2	1	0

$$
d_{i,j} = \begin{cases}
m - j & \text{wenn } i = n \\
n - i & \text{wenn } j = m \\
min\,\{1 + d_{i+1,j} \,;\, 1 + d_{i,j+1} \,;\, d_{i+1,j+1}\} & \text{wenn } \texttt{s[i]==t[j]} \\
1 + min\,\{d_{i+1,j} \,;\, d_{i,j+1} \,;\, d_{i+1,j+1}\} & \text{sonst}
\end{cases}
$$

Mögliche Reihenfolgen

	a	b	b	a	
c					
a					
b			0	1	2
a	3	2	1	0	1
	4	3	2	1	0

	a	b	b	a	
c				3	4
a				2	3
b			0	1	2
a			1	0	1
			2	1	0

	a	b	b	a	
c					
a					3
b				1	2
a			1	0	1
		3	2	1	0

	a	b	b	a	
c					4
a					3
b					2
a		2	1	0	1
	4	3	2	1	0

In C:

```
int min(int x, int y, int z)
{ ... };

int d[n+1][m+1];
int i,j;

for (i=0; i<=n; i++) d[i][m]=n-i;
for (j=0; j<m; j++) d[n][j]=m-j;

for (j=m-1; j>=0; j--)
  for (i=n-1; i>=0; i--)
    { if (s[i]==t[j])
        d[i][j]=min(1+d[i+1][j],1+d[i][j+1],d[i+1][j+1]);
      else
        d[i][j]=1+min(d[i+1][j],d[i][j+1],d[i+1][j+1]);
    }
```

Retracing (I)

Problem: Auffinden der Anordnung für die minimale Distanz

Ansatz: „Herkunft“ der minimalen Werte in Tabelle festhalten

Beispiel:

	a		b		c		b		c		b		a		
a	3		3		3	←	2		2		3		4		5
		↖		↖	↑	↖		↖	↑		↑		↑	↖	↑
c	4	←	3	←	2		2	←	1		2		3		4
						↖				↖	↑		↑		↑
b	5	←	4	←	3	←	2		2	←	1		2		3
				↖				↖		↖		↖	↑		↑
d	6	←	5	←	4	←	3	←	2	←	1		1		2
		↖		↖		↖		↖		↖		↖	↑		↑
a	6	←	5	←	4	←	3	←	2	←	1	←	0		1
		↖												↖	↑
	7	←	6	←	5	←	4	←	3	←	2	←	1	←	0

Lösung: Verfolgen aller Wege von $d_{0,0}$ nach $d_{n,m}$

Retracing (II)

Beispiel:

	a		b		c		b		c		b		a		
a	3		3		3	←	2		2		3		4		5
		↖		↖	↑	↖		↖	↑		↑		↑	↖	↑
c	4	←	3	←	2		2	←	1		2		3		4
						↖				↖	↑		↑		↑
b	5	←	4	←	3	←	2		2	←	1		2		3
				↖				↖		↖		↖	↑		↑
d	6	←	5	←	4	←	3	←	2	←	1		1		2
		↖		↖		↖		↖		↖		↖	↑		↑
a	6	←	5	←	4	←	3	←	2	←	1	←	0		1
		↖												↖	↑
	7	←	6	←	5	←	4	←	3	←	2	←	1	←	0

Ablesen:

a b c b c b a
a – c b – d a

a b c b c b a
a – c b d – a