17. Dynamic Programming II

Editierdistanz, Algorithmus von Bellman-Ford
[Cormen et al, Kap. 24.1]]

Minimale Editierdistanz

Editierdistanz von zwei Zeichenketten $A_n = (a_1, \ldots, a_n)$, $B_m = (b_1, \ldots, b_m)$.

Editieroperationen:

- Einfügen eines Zeichens
- Löschen eines Zeichens
- Änderung eines Zeichens

Frage: Wie viele Editieroperationen sind mindestens nötig, um eine gegebene Zeichenkette A in eine Zeichenkette B zu überführen.

TIGER ZIGER ZIEGER ZIEGE

Minimale Editierdistanz

Gesucht: Günstigste zeichenweise Transformation $A_n \to B_m$ mit Kosten

Operation	Levenshtein	LGT[24]	allgemein
c einfügen	1	1	ins(c)
c löschen	1	1	del(c)
Ersetzen $c \to c'$	$\mathbb{1}(c \neq c')$	$\infty \cdot \mathbb{1}(c \neq c')$	repl(c, c')

Beispiel

T	I	G	E	R		T	I	_	G	E	R		T→Z	+E	-R
Z	I	E	G	E		Z	I	E	G	E	_		Z→T	-E	+R

[24] Längste gemeinsame Teilfolge – Spezialfall des Editierproblems

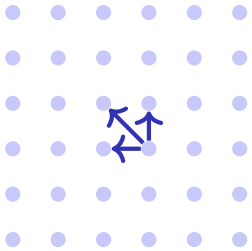
DP

0. $E(n, m)$ = minimale Anzahl Editieroperationen (ED Kosten) für $a_{1\ldots n} \to b_{1\ldots m}$
1. Teilprobleme $E(i, j)$ = ED von $a_{1\ldots i}$. $b_{1\ldots j}$. #TP $= n \cdot m$
2. Raten/Probieren Kosten $\Theta(1)$
   - $a_{1..i} \to a_{1\ldots i-1}$ (löschen)
   - $a_{1..i} \to a_{1\ldots i} b_j$ (einfügen)
   - $a_{1..i} \to a_{1\ldots i-1} b_j$ (ersetzen)
3. Rekursion

$$E(i,j) = \min \begin{cases} \text{del}(a_i) + E(i-1, j), \\ \text{ins}(b_j) + E(i, j-1), \\ \text{repl}(a_i, b_j) + E(i-1, j-1) \end{cases}$$

DP

4. Abhängigkeiten



   $\Rightarrow$ Berechnung von links oben nach rechts unten. Zeilen- oder Spaltenweise.
5. Lösung steht in $E(n, m)$

Beispiel (Levenshteinabstand)

$$E[i,j] \leftarrow \min\left\{E[i-1,j]+1, E[i,j-1]+1, E[i-1,j-1]+\mathbb{1}(a_i \neq b_j)\right\}$$

	∅	Z	I	E	G	E
∅	0	1	2	3	4	5
T	1	1	2	3	4	5
I	2	2	1	2	3	4
G	3	3	2	2	2	3
E	4	4	3	2	3	2
R	5	5	4	3	3	3

Editierschritte: von rechts unten nach links oben, der Rekursion folgend.
Bottom-Up Beschreibung des Algorithmus: Übung

Bottom-Up DP Algorithmus ED

Dimension der Tabelle? Bedeutung der Einträge?

1.

Bottom-Up DP Algorithmus ED

Dimension der Tabelle? Bedeutung der Einträge?

1. Tabelle $E[0,\ldots,m][0,\ldots,n]$. $E[i,j]$: Minimaler Editierabstand der Zeichenketten $(a_1,\ldots,a_i)$ und $(b_1,\ldots,b_j)$

Bottom-Up DP Algorithmus ED

1. **Dimension der Tabelle? Bedeutung der Einträge?**

   Tabelle $E[0,\ldots,m][0,\ldots,n]$. $E[i,j]$: Minimaler Editierabstand der Zeichenketten $(a_1,\ldots,a_i)$ und $(b_1,\ldots,b_j)$

2. **Berechnung eines Eintrags**

Bottom-Up DP Algorithmus ED

Dimension der Tabelle? Bedeutung der Einträge?

1. Tabelle $E[0,\ldots,m][0,\ldots,n]$. $E[i,j]$: Minimaler Editierabstand der Zeichenketten $(a_1,\ldots,a_i)$ und $(b_1,\ldots,b_j)$

Berechnung eines Eintrags

2. $E[0,i] \leftarrow i \; \forall 0 \leq i \leq m$, $E[j,0] \leftarrow i \; \forall 0 \leq j \leq n$. Berechnung von $E[i,j]$ sonst mit $E[i,j] = \min\{\mathsf{del}(a_i)+E(i-1,j), \mathsf{ins}(b_j)+E(i,j-1), \mathsf{repl}(a_i,b_j)+E(i-1,j-1)\}$

Bottom-Up DP Algorithmus ED

3. Berechnungsreihenfolge

Bottom-Up DP Algorithmus ED

3. **Berechnungsreihenfolge**

   Abhängigkeiten berücksichtigen: z.B. Zeilen aufsteigend und innerhalb von Zeilen Spalten aufsteigend.

Bottom-Up DP Algorithmus ED

3. **Berechnungsreihenfolge**

   Abhängigkeiten berücksichtigen: z.B. Zeilen aufsteigend und innerhalb von Zeilen Spalten aufsteigend.

4. **Rekonstruktion einer Lösung?**

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

3. Abhängigkeiten berücksichtigen: z.B. Zeilen aufsteigend und innerhalb von Zeilen Spalten aufsteigend.

Rekonstruktion einer Lösung?

4. Beginne bei $j = m$, $i = n$. Falls $E[i,j] = \text{repl}(a_i, b_j) + E(i-1, j-1)$ gilt, gib $a_i \to b_j$ aus und fahre fort mit $(j,i) \leftarrow (j-1, i-1)$; sonst, falls $E[i,j] = \text{del}(a_i) + E(i-1, j)$ gib $\text{del}(a_i)$ aus fahre fort mit $j \leftarrow j-1$; sonst, falls $E[i,j] = \text{ins}(b_j) + E(i, j-1)$, gib $\text{ins}(b_j)$ aus und fahre fort mit $i \leftarrow i-1$. Terminiere für $i = 0$ und $j = 0$.

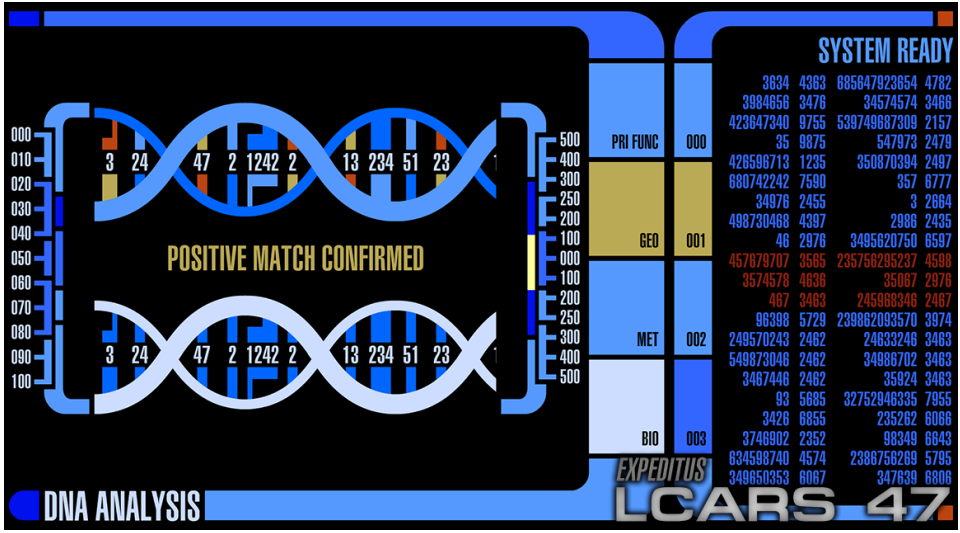
Analyse ED

- Anzahl Tabelleneinträge: $(m+1) \cdot (n+1)$.
- Berechnung jeweils mit konstanter Anzahl Zuweisungen und Vergleichen. Anzahl Schritte $\mathcal{O}(mn)$
- Bestimmen der Lösung: jeweils Verringerung von i oder j. Maximal $\mathcal{O}(n+m)$ Schritte.

Laufzeit insgesamt:

$$\mathcal{O}(mn).$$

DNA - Vergleich (Star Trek)



DNA - Vergleich

- DNA besteht aus Sequenzen von vier verschiedenen Nukleotiden **A**denin **G**uanin **T**hymin **C**ytosin
- DNA-Sequenzen (Gene) werden mit Zeichenketten aus A, G, T und C beschrieben.
- Ein möglicher Vergleich zweier Gene: Bestimme **Längste gemeinsame Teilfolge**

Das Problem, die längste gemeinsame Teilfolge zu finden ist ein Spezialfall der minimalen Editierdistanz.

Erinnerung Kürzeste Wege Algorithmus

1. Initialisiere d_s und π_s: $d_s[v] = \infty$, $\pi_s[v] =$ null für alle $v \in V$
2. Setze $d_s[s] \leftarrow 0$
3. Wähle eine Kante $(u, v) \in E$

   Relaxiere (u, v):
   if $d_s[v] > d[u] + c(u, v)$ then
   $d_s[v] \leftarrow d_s[u] + c(u, v)$
   $\pi_s[v] \leftarrow u$
4. Wiederhole 3 bis nichts mehr relaxiert werden kann.
   (bis $d_s[v] \leq d_s[u] + c(u, v) \quad \forall (u, v) \in E$)

Dynamic Programming Ansatz (Bellman)

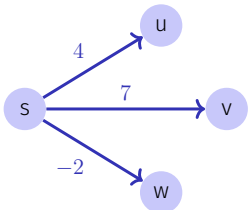
Induktion über Anzahl Kanten. $d_s[i, v]$: Kürzeste Weglänge von s nach v über maximal i Kanten.

$$d_s[i, v] = \min\{d_s[i-1, v], \min_{(u,v)\in E}(d_s[i-1, u] + c(u, v))$$

$$d_s[0, s] = 0, d_s[0, v] = \infty \; \forall v \neq s.$$

Dynamic Programming Ansatz (Bellman)

	s	$\cdots$	v	$\cdots$	w
0	0	∞	∞	∞	∞
1	0	∞	7	∞	-2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-1$	0	$\cdots$	$\cdots$	$\cdots$	$\cdots$



Algorithmus: Iteriere über letzte Zeile bis die Relaxationsschritte keine Änderung mehr ergeben, maximal aber $n-1$ mal. Wenn dann noch Änderungen, dann gibt es keinen kürzesten Pfad.

Algorithmus Bellman-Ford(G, s)

Input: Graph $G = (V, E, c)$, Startpunkt $s \in V$
Output: Wenn Rückgabe true, Minimale Gewichte d der kürzesten Pfade zu jedem Knoten, sonst kein kürzester Pfad.

foreach $u \in V$ **do**
    $d_s[u] \leftarrow \infty$; $\pi_s[u] \leftarrow$ null
$d_s[s] \leftarrow 0$;
for $i \leftarrow 1$ **to** $|V|$ **do**
    $f \leftarrow$ false
    foreach $(u, v) \in E$ **do**
        $f \leftarrow f \vee \mathrm{Relax}(u, v)$
    if $f =$ false **then return** true
return false;