



Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)

Dynamische Programmierung

Rekursiver Ansatz:

- Lösen eines Problems durch Lösen mehrerer kleinerer Teilprobleme, aus denen sich die Lösung für das Ausgangsproblem zusammensetzt

Phänomen:

- Mehrfachberechnungen von Lösungen

Methode:

- Lösungen zu Teilproblemen werden iterativ beginnend mit den Lösungen der kleinsten Teilprobleme berechnet (*bottom-up*).
- Speichern einmal berechneter Lösungen in einer Tabelle

Das Optimalitätsprinzip

Typische Anwendung für dynamisches Programmieren:
Optimierungsprobleme

Eine optimale Lösung für das Ausgangsproblem setzt sich aus *optimalen* Lösungen für kleinere Probleme zusammen.

Zusammenhang mit Teile & Herrsche

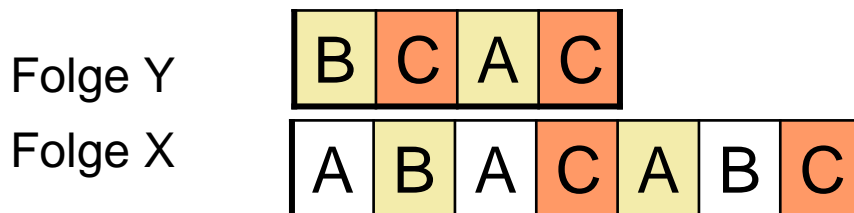
- Lösung eines Problems aus Lösungen zu Teilproblemen
- Aber: Lösungen zu Teilproblemen werden bei dynamischer Programmierung *nicht* rekursiv gelöst.

Längste gemeinsame Teilfolge

Definition:

- Seien $X=(x_1,\dots,x_m)$ und $Y=(y_1,\dots,y_n)$ zwei Teilfolgen, wobei $x_i, y_j \in A$ für ein endliches Alphabet A .
- Dann heißt Y **Teilfolge** von X , wenn es aufsteigend sortierte Indizes i_1,\dots,i_n gibt mit $x_{i_j} = y_j$ für $j = 1,\dots,n$.

Beispiel: (siehe Vollversion)



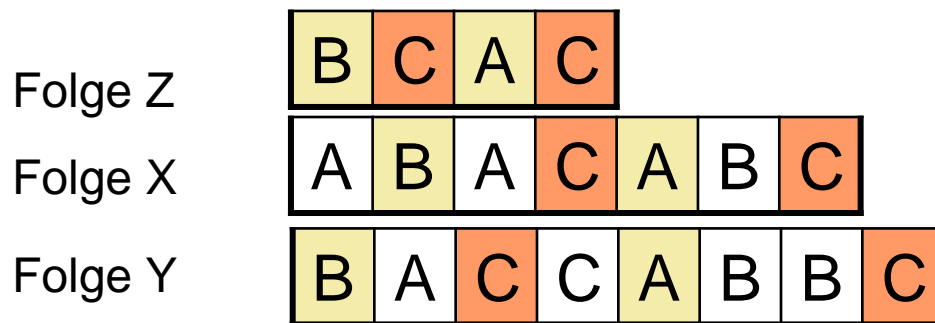
- Y ist Teilfolge von X
- Wähle $(i_1,i_2,i_3,i_4) = (2,4,5,7)$

Längste gemeinsame Teilfolge

Definition:

- Seien X, Y, Z Folgen über A .
- Dann heißt Z **gemeinsame Teilfolge** von X und Y , wenn Z Teilfolge sowohl von X als auch von Y ist.

Beispiel: (siehe Vollversion)



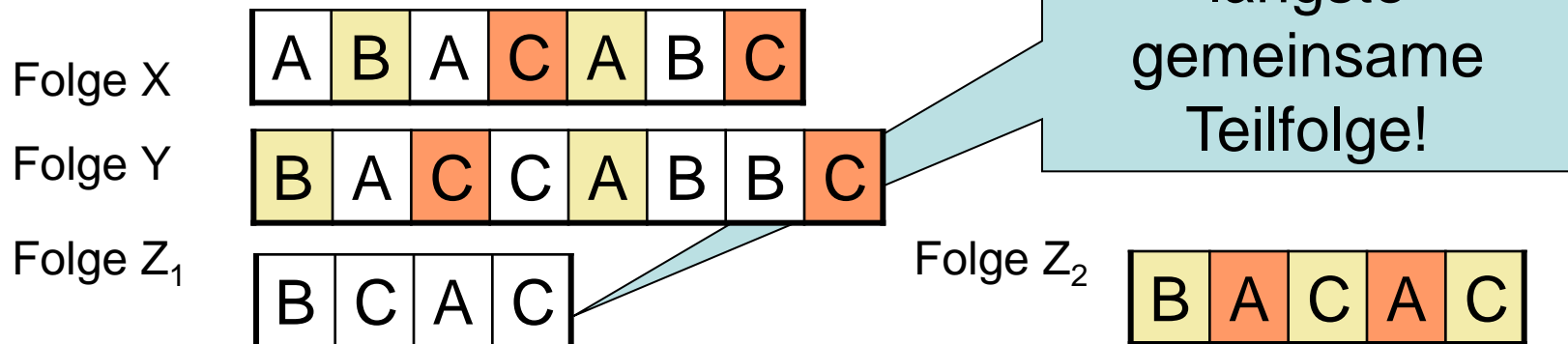
- Z ist gemeinsame Teilfolge von X und Y

Längste gemeinsame Teilfolge

Definition:

- Seien X, Y, Z Folgen über A .
- Dann heißt Z **längste gemeinsame Teilfolge** von X und Y , wenn Z gemeinsame Teilfolge von X und Y ist und es keine andere gemeinsame Teilfolge von X und Y gibt, die größere Länge als Z besitzt.

Beispiel: (siehe Vollversion)



Problem LCS

Eingabe:

- Folge $X=(x_1,\dots,x_m)$
- Folge $Y=(y_1,\dots,y_n)$

Ausgabe:

- Längste gemeinsame Teilfolge Z
(Longest **C**ommon **S**ubsequenz)

Beispiel:

Folge X

A	B	C	B	D	A	B
---	---	---	---	---	---	---

Folge Y

B	D	C	A	B	A
---	---	---	---	---	---

Einfacher Ansatz

Algorithmus:

- Erzeuge alle möglichen Teilfolgen von X
- Teste für jede Teilfolge von X, ob auch Teilfolge von Y
- Merke zu jedem Zeitpunkt bisher längste gemeinsame Teilfolge

Laufzeit:

- 2^m mögliche Teilfolgen
- Exponentielle Laufzeit!

Struktur von LCS

Satz 24:

Seien $X=(x_1,\dots,x_m)$ und $Y=(y_1,\dots,y_n)$ beliebige Folgen und sei $Z=(z_1,\dots,z_k)$ eine längste gemeinsame Teilfolge von X und Y . Dann gilt

1. Ist $x_m = y_n$, dann ist $z_k = x_m = y_n$ und (z_1,\dots,z_{k-1}) ist eine längste gemeinsame Teilfolge von (x_1,\dots,x_{m-1}) und (y_1,\dots,y_{n-1}) .
2. Ist $x_m \neq y_n$ und $z_k \neq x_m$, dann ist Z eine längste gemeinsame Teilfolge von (x_1,\dots,x_{m-1}) und Y .
3. Ist $x_m \neq y_n$ und $z_k \neq y_n$, dann ist Z eine längste gemeinsame Teilfolge von X und (y_1,\dots,y_{n-1}) .

Anwendung des
Optimalitätsprinzips

Struktur von LCS

Beweis:

(1) **Annahme:** $z_k \neq x_m$

Dann können wir $z_k = x_m$ hängen, um eine gemeinsame Teilfolge von X und Y der Länge $k+1$ zu erhalten. Widerspruch: Z ist eine *längste* gemeinsame Teilfolge von X und Y

$\Rightarrow z_k = x_m = y_n$

$\Rightarrow (z_1, z_2, \dots, z_{k-1})$ ist eine gemeinsame Teilfolge der Länge $k-1$ von $(x_1, x_2, \dots, x_{m-1})$ und $(y_1, y_2, \dots, y_{n-1})$.

Annahme: Es gibt eine gemeinsame Teilfolge W von $(x_1, x_2, \dots, x_{m-1})$ und $(y_1, y_2, \dots, y_{n-1})$, die mindestens Länge k hat. Dann erzeugt das Anhängen von $z_k = x_m$ an W eine gemeinsame Teilfolge von X und Y , dessen Länge mindestens $k+1$ hat. Widerspruch zur Optimalität von Z .

Struktur von LCS

Beweis:

(2) Falls $z_k \neq x_m$ dann ist Z eine gemeinsame Teilfolge von $(x_1, x_2, \dots, x_{m-1})$ und Y .

Annahme: Es gibt eine gemeinsame Teilfolge W von $(x_1, x_2, \dots, x_{m-1})$ und Y mit einer Länge größer k .

Dann ist W auch eine gemeinsame Teilfolge von X und Y . Widerspruch:
 Z ist längste gemeinsame Teilfolge von X und Y .

(3) Der Beweis ist analog zu (2)

Rekursion für Länge von LCS

Lemma 25:

Sei $C[i][j]$ die Länge einer längsten gemeinsamen Teilfolge von (x_1, \dots, x_i) und (y_1, \dots, y_j) . Dann gilt:

$$C[i][j] = \begin{cases} 0 & \text{falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{falls } i, j > 0 \text{ und } x_i = y_j \\ \max\{C[i-1][j], C[i][j-1]\} & \text{falls } i, j > 0 \text{ und } x_i \neq y_j \end{cases}$$

Beobachtung:

Rekursive Berechnung der $C[i][j]$ würde zu Berechnung immer wieder derselben Werte führen. Dieses ist ineffizient. Berechnen daher die Werte $C[i][j]$ iterativ, nämlich zeilenweise.

Berechnung der $C[i][j]$ Werte (siehe Vollversion)

LCS-Länge(X, Y)

1. $m \leftarrow \text{length}[X]$
2. $n \leftarrow \text{length}[Y]$
3. **new** array $C[0..m][0..n]$
4. **for** $i \leftarrow 0$ **to** m **do** $C[i][0] \leftarrow 0$
5. **for** $j \leftarrow 0$ **to** n **do** $C[0][j] \leftarrow 0$
6. **for** $i \leftarrow 1$ **to** m **do**
7. **for** $j \leftarrow 1$ **to** n **do**
8. ➤ Längenberechnung(X, Y, C, i, j)
9. **return** C

Berechnung der C[i][j] Werte (siehe Vollversion)

Längenberechnung(X, Y, C, i, j)

1. **if** $x_i = y_j$ **then** $C[i][j] \leftarrow C[i-1][j-1] + 1$
2. **else**
3. **if** $C[i-1][j] \geq C[i][j-1]$ **then** $C[i][j] \leftarrow C[i-1][j]$
4. **else** $C[i][j] \leftarrow C[i][j-1]$

$$C[i][j] = \begin{cases} 0 & \text{falls } i = 0 \text{ oder } j = 0 \\ C[i-1][j-1] + 1 & \text{falls } i, j > 0 \text{ und } x_i = y_j \\ \max\{C[i-1][j], C[i][j-1]\} & \text{falls } i, j > 0 \text{ und } x_i \neq y_j \end{cases}$$

(siehe Vollversion)

		j	0	1	2	3	4	5	6
		y_j	B	D	C	A	B	A	
0	x_i	0	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1	
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2	
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2	
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3	
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3	
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4	
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4	

Laufzeitanalyse

Lemma 26:

Der Algorithmus LCS-Länge hat Laufzeit $O(nm)$, wenn die Folgen X, Y Länge n und m haben.

Lemma 27:

Die Ausgabe der längsten gemeinsamen Teilfolge anhand der Tabelle hat Laufzeit $O(n+m)$, wenn die Folgen X, Y Länge n und m haben.

Vorgehensweise bei dynamischer Programmierung

1. Bestimme rekursive Struktur einer optimalen Lösung.
2. Entwerfe rekursive Methode zur Bestimmung des Wertes einer optimalen Lösung.
3. Transformiere rekursiv Methode in eine iterative (bottom-up) Methode zur Bestimmung des Wertes einer optimalen Lösung.
4. Bestimmen aus dem Wert einer optimalen Lösung und in 3. ebenfalls berechneten Zusatzinformationen eine optimale Lösung.

Zusammenfassung

Algorithmenentwurfstechnik:

- Oft bei Optimierungsproblemen angewandt

Einsatz:

- Bei rekursiven Problemlösungen, wenn Teillösungen mehrfach benötigt werden

Lösungsansatz:

- Tabellieren von Teilergebnissen

Vorteil:

- Laufzeitverbesserungen, oft polynomiell statt exponentiell