Dynamische Programmierung

Basierend auf dem Bellmansches Optimalitätsprinzip:

„Optimale Teillösungen bilden eine optimale Lösung.“

LgT

Geg.: Wörter a, b
Ges.: Längste gemeinsame Teilfolge LgT(a, b)

Idee: $C(i, j) \equiv LgT(a_1 \ldots a_i, b_1 \ldots b_j)$ sei bekannt

$$C(i, 0) = C(0, j) = 0$$

$$C(i, j) = \begin{cases} C(i-1, j-1) + 1 & \text{falls } a_i = b_j \\ \max\{C(i-1, j), C(i, j-1)\} & \text{falls } a_i \neq b_j \end{cases}$$

KgO

Geg.: Wörter a, b
Ges.: Kürzeste gemeinsame Oberfolge KgO(a, b)

Idee: $C(i, j) \equiv KgO(a_1 \ldots a_i, b_1 \ldots b_j)$ sei bekannt

$$C(i, 0) = i \quad C(0, j) = j$$

$$C(i, j) = \begin{cases} C(i-1, j-1) + 1 & \text{falls } a_i = b_j \\ \min\{C(i-1, j) + 1, C(i, j-1) + 1\} & \text{falls } a_i \neq b_j \end{cases}$$

Editierdistanz

Geg.: Wörter a, b
Ges.: Geringste Anzahl Operationen (Löschen, Einfügen, Umbenennen) zur Überführung von Wort a nach b (= Maß für die Ähnlichkeit von a und b)

Idee: $C(i, j) \equiv Edd(a_1 \ldots a_i, b_1 \ldots b_j)$ sei bekannt
$C(i-1, j-1)$ - Umbenennen von a_i nach b_j
$C(i-1, j)$ - Löschen von a_i
$C(i, j-1)$ - Einfügen von b_j

$$C(i, 0) = i \quad C(0, j) = j$$

$$C(i, j) = \begin{cases} C(i-1, j-1) & \text{falls } a_i = b_j \\ \min\{C(i-1, j-1) + 1, C(i-1, j) + 1, C(i, j-1) + 1\} & \text{falls } a_i \neq b_j \end{cases}$$

TSP

Geg.: Kostenmatrix K mit $K(a, b)$ = Kosten der „Strecke“ a nach b
Ges.: Optimale (günstigste) Rundreise

Idee: Optimaler Pfad $P_n(i, V)$ der Länge n vom Knoten 1 zum Knoten i bekannt, wobei $\forall\, v \in V$ gilt: $v \notin P_n$

$P_n(i, \varnothing) = K(i, 1)$ schließen des Kreises
$P_n(i, V) = \min\{K(i, v) + P_{n+1}(v, V \setminus v) \mid \forall\, v \in V\}$ expandieren des Pfades

Iterierte Matrizenmultiplikation

Geg.: n Matrizen $A_i \in N^{i-1 \times i}$
Ges.: Berechnung von $B = A_1 \times A_2 \times \ldots \times A_n$ durch minimale Anzahl Multiplikationen M

Idee: A_i mit 0 Multiplikationen berechenbar
$A_i \times A_{i+1}$ mit $(i-1) \times i \times (i+1)$ Multiplikationen berechenbar ($p_{i-1} \cdot p_i \cdot p_{i+1}$)
$A_i \times A_j \times A_k$ durch $A_i \times (A_j \times A_k)$ oder $(A_i \times A_j) \times A_k$ berechenbar …

$$M(i, j) = \begin{cases} 0 & \text{falls } i \geq j \\ \min\{M(i, k) + M(k+1, j) + p_{i-1} \cdot p_k \cdot p_j \mid i \leq k < j\} & \text{falls } i < j \end{cases}$$

Optimaler binärer Suchbaum

Geg.: Knoten a_i und dazu gehörige Häufigkeiten b_i $(i = 1, \ldots, n)$
Ges.: Optimaler Suchbaum S mit minimalen Kosten c ~ Häufigkeiten × Tiefe $\forall$ Knoten a_i

Idee: $C(i, j) \equiv S(a_i \ldots a_j)$ sei minimaler Suchbaum welcher die Knoten a_i bis a_j enthält

$$C(i, j) = \begin{cases} 0 & \text{falls } i > j \\ \sum_{n=i}^{j} c_n + \min\{C(i, k-1), C(k+1, j) \mid i \leq k \leq j\} & \text{falls } i \leq j \end{cases}$$