

12. Dynamische Programmierung

Memoisieren, Optimale Substruktur, Überlappende Teilprobleme, Abhängigkeiten, Allgemeines Vorgehen. Beispiele: Schneiden von Eisenstangen, Kaninchen, Editierdistanz

[Ottman/Widmayer, Kap. 7.1, 7.4, Cormen et al, Kap. 15]

Fibonacci Zahlen



(schon wieder)

$$F_n := \begin{cases} n & \text{wenn } n < 2 \\ F_{n-1} + F_{n-2} & \text{wenn } n \geq 2. \end{cases}$$

Analyse: warum ist der rekursive Algorithmus so langsam.

Algorithmus FibonacciRecursive(n)

Input: $n \geq 0$

Output: n -te Fibonacci Zahl

if $n < 2$ **then**

 | $f \leftarrow n$

else

 | $f \leftarrow \text{FibonacciRecursive}(n - 1) + \text{FibonacciRecursive}(n - 2)$

return f

Analyse

$T(n)$: Anzahl der ausgeführten Operationen.

- $n = 0, 1: T(n) = \Theta(1)$

Analyse

$T(n)$: Anzahl der ausgeführten Operationen.

■ $n = 0, 1: T(n) = \Theta(1)$

■ $n \geq 2: T(n) = T(n - 2) + T(n - 1) + c.$

Analyse

$T(n)$: Anzahl der ausgeführten Operationen.

■ $n = 0, 1: T(n) = \Theta(1)$

■ $n \geq 2: T(n) = T(n - 2) + T(n - 1) + c.$

$$T(n) = T(n - 2) + T(n - 1) + c \geq 2T(n - 2) + c \geq 2^{n/2}c' = (\sqrt{2})^n c'$$

Analyse

$T(n)$: Anzahl der ausgeführten Operationen.

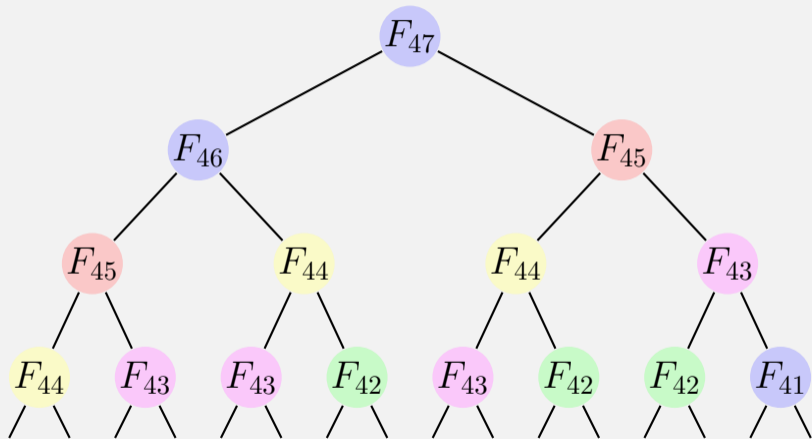
■ $n = 0, 1: T(n) = \Theta(1)$

■ $n \geq 2: T(n) = T(n - 2) + T(n - 1) + c.$

$$T(n) = T(n - 2) + T(n - 1) + c \geq 2T(n - 2) + c \geq 2^{n/2}c' = (\sqrt{2})^n c'$$

Algorithmus ist *exponentiell (!)* in n .

Grund, visualisiert



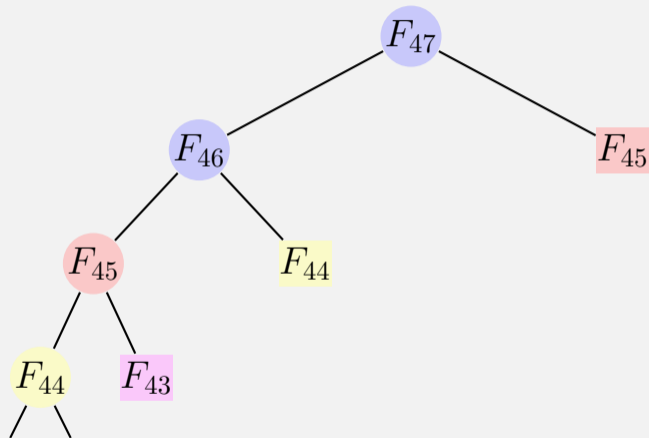
Knoten mit denselben Werten werden (zu) oft ausgewertet.

Memoization

Memoization (sic) Abspeichern von Zwischenergebnissen.

- Bevor ein Teilproblem gelöst wird, wird Existenz eines entsprechenden Zwischenergebnis geprüft.
- Existiert ein gespeichertes Zwischenergebnis bereits, so wird dieses verwendet.
- Andernfalls wird der Algorithmus ausgeführt und das Ergebnis wird entsprechend gespeichert.

Memoization bei Fibonacci



Rechteckige Knoten wurden bereits ausgewertet.

Algorithmus FibonacciMemoization(n)

Input: $n \geq 0$

Output: n -te Fibonacci Zahl

if $n \leq 2$ **then**

| $f \leftarrow 1$

else if $\exists \text{memo}[n]$ **then**

| $f \leftarrow \text{memo}[n]$

else

| $f \leftarrow \text{FibonacciMemoization}(n - 1) + \text{FibonacciMemoization}(n - 2)$

| $\text{memo}[n] \leftarrow f$

return f

Analyse

Berechnungsaufwand:

$$T(n) = T(n - 1) + c = \dots = \mathcal{O}(n).$$

denn nach dem Aufruf von $f(n - 1)$ wurde $f(n - 2)$ bereits berechnet.

Das lässt sich auch so sehen: Für jedes n wird $f(n)$ maximal einmal rekursiv berechnet. Laufzeitkosten: n Aufrufe mal $\Theta(1)$ Kosten pro Aufruf $n \cdot c \in \Theta(n)$. Die Rekursion verschwindet aus der Berechnung der Laufzeit.

Algorithmus benötigt $\Theta(n)$ Speicher.¹⁹

¹⁹Allerdings benötigt der naive Algorithmus auch $\Theta(n)$ Speicher für die Rekursionsverwaltung.

Genauer hingesehen ...

... berechnet der Algorithmus der Reihe nach die Werte $F_1, F_2, F_3,$
... verkleidet im *Top-Down* Ansatz der Rekursion.

Man kann den Algorithmus auch gleich *Bottom-Up* hinschreiben.
Das ist charakteristisch für die *dynamische Programmierung*.

Algorithmus FibonacciBottomUp(n)

Input: $n \geq 0$

Output: n -te Fibonacci Zahl

$F[1] \leftarrow 1$

$F[2] \leftarrow 1$

for $i \leftarrow 3, \dots, n$ **do**

$F[i] \leftarrow F[i - 1] + F[i - 2]$

return $F[n]$

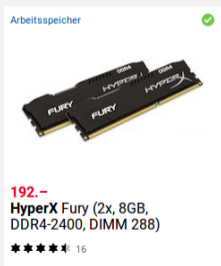
Dynamische Programmierung: Idee

- Aufteilen eines komplexen Problems in eine vernünftige Anzahl kleinerer Teilprobleme
- Die Lösung der Teilprobleme wird zur Lösung des komplexeren Problems verwendet
- Identische Teilprobleme werden nur einmal gerechnet

Dynamische Programmierung: Konsequenz

Identische Teilprobleme werden nur einmal gerechnet

⇒ Resultate werden zwischengespeichert



Wir tauschen Laufzeit
gegen Speicherplatz

Dynamic Programming: Beschreibung

- 1 Verwalte *DP-Tabelle* mit Information zu den Teilproblemen.
Dimension der Tabelle? Bedeutung der Einträge?

Dynamic Programming: Beschreibung

- 1 Verwalte *DP-Tabelle* mit Information zu den Teilproblemen.
Dimension der Tabelle? Bedeutung der Einträge?
- 2 Berechnung der *Randfälle*.
Welche Einträge hängen nicht von anderen ab?

Dynamic Programming: Beschreibung

- 1 Verwalte *DP-Tabelle* mit Information zu den Teilproblemen.
Dimension der Tabelle? Bedeutung der Einträge?
- 2 Berechnung der *Randfälle*.
Welche Einträge hängen nicht von anderen ab?
- 3 *Berechnungsreihenfolge* bestimmen.
In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

Dynamic Programming: Beschreibung

- 1 Verwalte *DP-Tabelle* mit Information zu den Teilproblemen.
Dimension der Tabelle? Bedeutung der Einträge?
- 2 Berechnung der *Randfälle*.
Welche Einträge hängen nicht von anderen ab?
- 3 *Berechnungsreihenfolge* bestimmen.
In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?
- 4 Auslesen der *Lösung*.
Wie kann sich Lösung aus der Tabelle konstruieren lassen?

Dynamic Programming: Beschreibung

- 1 Verwalte *DP-Tabelle* mit Information zu den Teilproblemen.
Dimension der Tabelle? Bedeutung der Einträge?
- 2 Berechnung der *Randfälle*.
Welche Einträge hängen nicht von anderen ab?
- 3 *Berechnungsreihenfolge* bestimmen.
In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?
- 4 Auslesen der *Lösung*.
Wie kann sich Lösung aus der Tabelle konstruieren lassen?

Laufzeit (typisch) = Anzahl Einträge der Tabelle mal Aufwand pro Eintrag.

Dynamic Programming: Beschreibung am Beispiel

1

Dimension der Tabelle? Bedeutung der Einträge?

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

2 Welche Einträge hängen nicht von anderen ab?

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

2 Welche Einträge hängen nicht von anderen ab?

Werte F_1 und F_2 sind unabhängig einfach "berechenbar".

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

2 Welche Einträge hängen nicht von anderen ab?

Werte F_1 und F_2 sind unabhängig einfach "berechenbar".

3 In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

2 Welche Einträge hängen nicht von anderen ab?

Werte F_1 und F_2 sind unabhängig einfach "berechenbar".

3 In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

F_i mit aufsteigenden i .

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

2 Welche Einträge hängen nicht von anderen ab?

Werte F_1 und F_2 sind unabhängig einfach "berechenbar".

3 In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

F_i mit aufsteigenden i .

4 Wie kann sich Lösung aus der Tabelle konstruieren lassen?

Dynamic Programming: Beschreibung am Beispiel

1 Dimension der Tabelle? Bedeutung der Einträge?

Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält n -te Fibonacci Zahl.

2 Welche Einträge hängen nicht von anderen ab?

Werte F_1 und F_2 sind unabhängig einfach "berechenbar".

3 In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

F_i mit aufsteigenden i .

4 Wie kann sich Lösung aus der Tabelle konstruieren lassen?

F_n ist die n -te Fibonacci-Zahl.

Dynamic Programming = Divide-And-Conquer ?

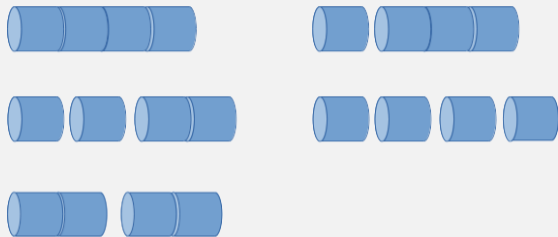
- In beiden Fällen ist das Ursprungsproblem (einfacher) lösbar, indem Lösungen von Teilproblemen herangezogen werden können. Das Problem hat *optimale Substruktur*.
- Bei Divide-And-Conquer Algorithmen (z.B. Mergesort) sind Teilprobleme unabhängig; deren Lösungen werden im Algorithmus nur einmal benötigt.
- Beim DP sind Teilprobleme nicht unabhängig. Das Problem hat *überlappende Teilprobleme*, welche im Algorithmus mehrfach gebraucht werden.
- Damit sie nur einmal gerechnet werden müssen, werden Resultate tabelliert. Dafür darf es *zwischen Teilproblemen keine zirkulären Abhängigkeiten* geben.

Schneiden von Eisenstäben

- Metallstäbe werden zerschnitten und verkauft.
- Metallstäbe der Länge $n \in \mathbb{N}$ verfügbar. Zerschneiden kostet nichts.
- Für jede Länge $l \in \mathbb{N}$, $l \leq n$ bekannt: Wert $v_l \in \mathbb{R}^+$
- Ziel: Zerschneide die Stange so (in $k \in \mathbb{N}$ Stücke), dass

$$\sum_{i=1}^k v_{l_i} \text{ maximal unter } \sum_{i=1}^k l_i = n.$$

Schneiden von Eisenstäben: Beispiel



Arten, einen Stab der Länge 4 zu zerschneiden (ohne Permutationen)

Länge	0	1	2	3	4
Preis	0	2	3	8	9

⇒ Bester Schnitt: 3 + 1 mit Wert 10.

Wie findet man den DP Algorithmus

- 0 Genaue Formulierung der gesuchten Lösung
- 1 Definiere Teilprobleme (und bestimme deren Anzahl)
- 2 Raten / Aufzählen (und bestimme die Laufzeit für das Raten)
- 3 Rekursion: verbinde die Teilprobleme
- 4 Memoisieren / Tabellieren. Bestimme die Abhängigkeiten der Teilprobleme
- 5 Lösung des Problems
Laufzeit = #Teilprobleme \times Zeit/Teilproblem

Struktur des Problems

- 0 **Gesucht:** r_n = maximal erreichbarer Wert von (ganzem oder geschnittenem) Stab mit Länge n .
- 1 **Teilprobleme:** maximal erreichbarer Wert r_k für alle $0 \leq k < n$
- 2 **Rate** Länge des ersten Stückes
- 3 **Rekursion**

$$r_k = \max \{v_i + r_{k-i} : 0 < i \leq k\}, \quad k > 0$$

$$r_0 = 0$$

- 4 **Abhängigkeit:** r_k hängt (nur) ab von den Werten $v_i, l \leq i \leq k$ und den optimalen Schnitten $r_i, i < k$
- 5 **Lösung** in r_n

Algorithmus RodCut(v, n)

Input: $n \geq 0$, Preise v

Output: bester Wert

$q \leftarrow 0$

if $n > 0$ **then**

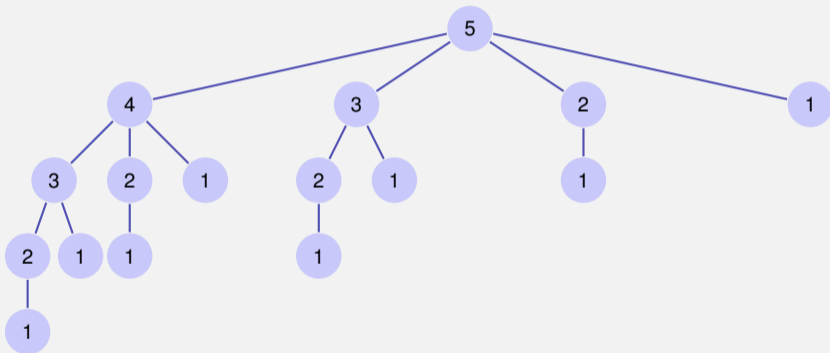
for $i \leftarrow 1, \dots, n$ **do**
 $q \leftarrow \max\{q, v_i + \text{RodCut}(v, n - i)\};$

return q

Laufzeit $T(n) = \sum_{i=0}^{n-1} T(i) + c \Rightarrow^{20} T(n) \in \Theta(2^n)$

²⁰ $T(n) = T(n-1) + \sum_{i=0}^{n-2} T(i) + c = T(n-1) + (T(n-1) - c) + c = 2T(n-1) \quad (n > 0)$

Rekursionsbaum



Algorithmus RodCutMemoized(m, v, n)

Input: $n \geq 0$, Preise v , Memoization Tabelle m

Output: bester Wert

$q \leftarrow 0$

if $n > 0$ **then**

if $\exists m[n]$ **then**

$q \leftarrow m[n]$

else

for $i \leftarrow 1, \dots, n$ **do**

$q \leftarrow \max\{q, v_i + \text{RodCutMemoized}(m, v, n - i)\};$

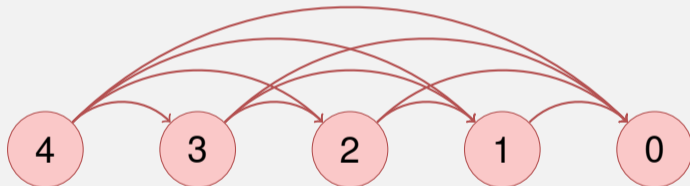
$m[n] \leftarrow q$

return q

Laufzeit $\sum_{i=1}^n i = \Theta(n^2)$

Teilproblem-Graph

beschreibt die Abhängigkeiten der Teilprobleme untereinander



und darf keine Zyklen enthalten

Konstruktion des optimalen Schnittes

- Während der (rekursiven) Berechnung der optimalen Lösung für jedes $k \leq n$ bestimmt der rekursive Algorithmus die optimale Länge des ersten Stabes
- Speichere die Länge des ersten Stabes für jedes $k \leq n$ in einer Tabelle mit n Einträgen.

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

1

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

2 Welche Einträge hängen nicht von anderen ab?

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert r_0 ist 0.

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert r_0 ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

.

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert r_0 ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

- 3 $r_i, i = 1, \dots, n$.

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert r_0 ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

- 3 $r_i, i = 1, \dots, n$.

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

4

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle der Grösse $n \times 1$. n -ter Eintrag enthält besten Wert eines Stabes der Länge n .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert r_0 ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

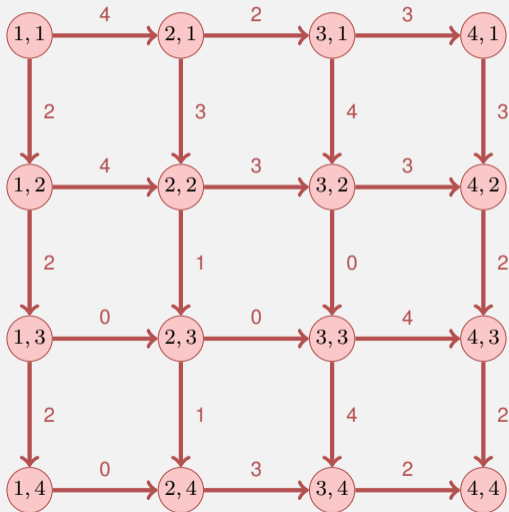
- 3 $r_i, i = 1, \dots, n$.

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

- 4 r_n ist der beste Wert für eine Stange der Länge n

Kaninchen!

Ein Kaninchen sitzt auf Platz $(1, 1)$ eines $n \times n$ Gitters. Es kann nur nach Osten oder nach Süden gehen. Auf jedem Wegstück liegt eine Anzahl Rüben. Wie viele Rüben sammelt das Kaninchen maximal ein?

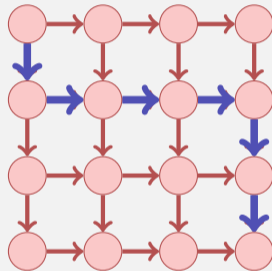


Kaninchen!

Anzahl mögliche Pfade?

- Auswahl von $n - 1$ Wegen nach Süden aus $2n - 2$ Wegen insgesamt.

⇒ Naiver Algorithmus hat keine Chance



Der Weg 100011
(1:nach Süden, 0:nach Osten)

Kaninchen!

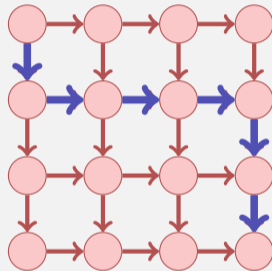
Anzahl mögliche Pfade?

- Auswahl von $n - 1$ Wegen nach Süden aus $2n - 2$ Wegen insgesamt.



$$\binom{2n - 2}{n - 1} \in \Omega(2^n)$$

⇒ Naiver Algorithmus hat keine Chance



Der Weg 100011
(1:nach Süden, 0:nach Osten)

Rekursion

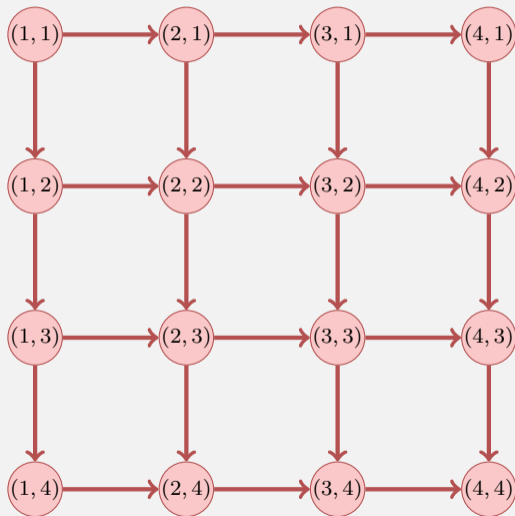
Gesucht: $T_{0,0} = \text{Maximale Anzahl Rüben von } (0, 0) \text{ nach } (n, n)$.

Sei $w_{(i,j)-(i',j')}$ Anzahl Rüben auf Kante von (i, j) nach (i', j') .

Rekursion (maximale Anzahl Rüben von (i, j) nach (n, n))

$$T_{ij} = \begin{cases} \max\{w_{(i,j)-(i,j+1)} + T_{i,j+1}, w_{(i,j)-(i+1,j)} + T_{i+1,j}\}, & i < n, j < n \\ w_{(i,j)-(i,j+1)} + T_{i,j+1}, & i = n, j < n \\ w_{(i,j)-(i+1,j)} + T_{i+1,j}, & i < n, j = n \\ 0 & i = j = n \end{cases}$$

Teilproblemabhängigkeitsgraph



Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

1

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

2 Welche Einträge hängen nicht von anderen ab?

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert $T_{n,n}$ ist 0.

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert $T_{n,n}$ ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

- 3

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert $T_{n,n}$ ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

- 3 $T_{i,j}$ mit $i = n \searrow 1$ und für jedes $i: j = n \searrow 1$, (oder umgekehrt: $j = n \searrow 1$ und für jedes $j: i = n \searrow 1$).

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert $T_{n,n}$ ist 0.

In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

- 3 $T_{i,j}$ mit $i = n \searrow 1$ und für jedes $i: j = n \searrow 1$, (oder umgekehrt: $j = n \searrow 1$ und für jedes $j: i = n \searrow 1$).

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

- 4

Bottom-Up Beschreibung am Beispiel

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle T der Grösse $n \times n$. Eintrag bei i, j enthält die maximale Anzahl Rüben von (i, j) nach (n, n) .

Welche Einträge hängen nicht von anderen ab?

- 2 Wert $T_{n,n}$ ist 0.

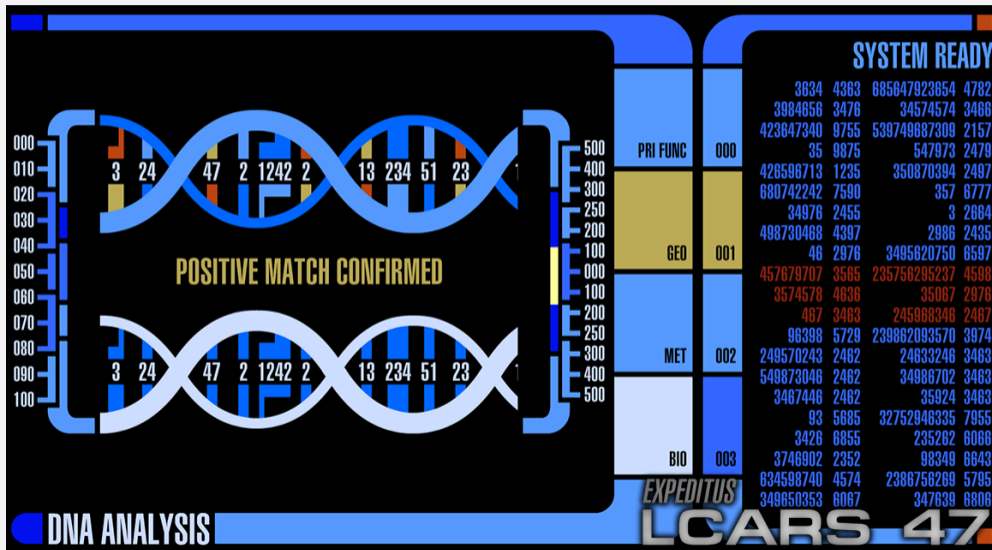
In welcher Reihenfolge können Einträge berechnet werden, so dass benötigte Einträge jeweils vorhanden sind?

- 3 $T_{i,j}$ mit $i = n \searrow 1$ und für jedes $i: j = n \searrow 1$, (oder umgekehrt: $j = n \searrow 1$ und für jedes $j: i = n \searrow 1$).

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

- 4 $T_{1,1}$ enthält die maximale Anzahl Rüben

DNA - Vergleich (Star Trek)



DNA - Vergleich

- DNA besteht aus Sequenzen von vier verschiedenen Nukleotiden **A**denin **G**uanin **T**hymine **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. Die folgenden Folien werden daher in der Vorlesung nicht behandelt.

[Längste Gemeinsame Teilfolge]

Teilfolgen einer Zeichenkette:

Teilfolgen(*KUH*): (), (*K*), (*U*), (*H*), (*KU*), (*KH*), (*UH*), (*KUH*)

Problem:

- **Eingabe:** Zwei Zeichenketten $A = (a_1, \dots, a_m)$, $B = (b_1, \dots, b_n)$ der Längen $m > 0$ und $n > 0$.
- **Gesucht:** Eine längste gemeinsame Teilfolge (LGT) von A und B .

[Längste Gemeinsame Teilfolge]

Beispiele:

$LGT(IGEL, KATZE) = E$, $LGT(TIGER, ZIEGE) = IGE$

Ideen zur Lösung?

T I E G E R
Z I E G E

[Rekursives Vorgehen]

Annahme: Lösungen $L(i, j)$ bekannt für $A[1, \dots, i]$ und $B[1, \dots, j]$ für alle $1 \leq i \leq m$ und $1 \leq j \leq n$, jedoch nicht für $i = m$ und $j = n$.

T I G E R
Z I E G E

Betrachten Zeichen a_m, b_n . Drei Möglichkeiten:

- 1 A wird um ein Leerzeichen erweitert. $L(m, n) = L(m, n - 1)$
- 2 B wird um ein Leerzeichen erweitert. $L(m, n) = L(m - 1, n)$
- 3 $L(m, n) = L(m - 1, n - 1) + \delta_{mn}$ mit $\delta_{mn} = 1$ wenn $a_m = b_n$ und $\delta_{mn} = 0$ sonst

[Rekursion]

$$L(m, n) \leftarrow \max \{L(m - 1, n - 1) + \delta_{mn}, L(m, n - 1), L(m - 1, n)\}$$

für $m, n > 0$ und Randfälle $L(\cdot, 0) = 0, L(0, \cdot) = 0$.

	\emptyset	Z	I	E	G	E
\emptyset	0	0	0	0	0	0
T	0	0	0	0	0	0
I	0	0	1	1	1	1
G	0	0	1	1	2	2
E	0	0	1	2	2	3
R	0	0	1	2	2	3

[Dynamic Programming Algorithmus LGT]

Dimension der Tabelle? Bedeutung der Einträge?

1

[Dynamic Programming Algorithmus LGT]

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle $L[0, \dots, m][0, \dots, n]$. $L[i, j]$: Länge einer LGT der Zeichenketten (a_1, \dots, a_i) und (b_1, \dots, b_j)

[Dynamic Programming Algorithmus LGT]

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle $L[0, \dots, m][0, \dots, n]$. $L[i, j]$: Länge einer LGT der Zeichenketten (a_1, \dots, a_i) und (b_1, \dots, b_j)

Berechnung eines Eintrags

- 2

[Dynamic Programming Algorithmus LGT]

Dimension der Tabelle? Bedeutung der Einträge?

- 1 Tabelle $L[0, \dots, m][0, \dots, n]$. $L[i, j]$: Länge einer LGT der Zeichenketten (a_1, \dots, a_i) und (b_1, \dots, b_j)

Berechnung eines Eintrags

- 2 $L[0, i] \leftarrow 0 \forall 0 \leq i \leq m, L[j, 0] \leftarrow 0 \forall 0 \leq j \leq n$. Berechnung von $L[i, j]$ sonst mit $L[i, j] = \max(L[i-1, j-1] + \delta_{ij}, L[i, j-1], L[i-1, j])$.

[Dynamic Programming Algorithmus LGT]

Berechnungsreihenfolge

3

[Dynamic Programming Algorithmus LGT]

Berechnungsreihenfolge

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

[Dynamic Programming Algorithmus LGT]

Berechnungsreihenfolge

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

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

4

[Dynamic Programming Algorithmus LGT]

Berechnungsreihenfolge

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

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

- 4 Beginne bei $j = m, i = n$. Falls $a_i = b_j$ gilt, gib a_i aus und fahre fort mit $(j, i) \leftarrow (j - 1, i - 1)$; sonst, falls $L[i, j] = L[i, j - 1]$ fahre fort mit $j \leftarrow j - 1$; sonst, falls $L[i, j] = L[i - 1, j]$ fahre fort mit $i \leftarrow i - 1$. Terminiere für $i = 0$ oder $j = 0$.

[Analyse LGT]

- 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:

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

Minimale Editierdistanz

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

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 $A_n \rightarrow B_m$ mit Kosten

Operation	Levenshtein	LGT ²¹	allgemein
c einfügen	1	1	ins(c)
c löschen	1	1	del(c)
Ersetzen $c \rightarrow c'$	$\mathbb{1}(c \neq c')$	$\infty \cdot \mathbb{1}(c \neq c')$	repl(c, c')

Beispiel

T I G E R
Z I E G E

T I _ G E R
Z I E G E _

T \rightarrow Z +E -R
Z \rightarrow T -E +R

²¹Längste gemeinsame Teilfolge – Spezialfall des Editierproblems

DP

- 0 $E(n, m)$ = minimale Anzahl Editieroperationen (ED Kosten) für $a_{1\dots n} \rightarrow b_{1\dots m}$
- 1 Teilprobleme $E(i, j)$ = ED von $a_{1\dots i}$ $b_{1\dots j}$. #TP = $n \cdot m$
- 2 Raten/Probieren Kosten $\Theta(1)$
- $a_{1\dots i} \rightarrow a_{1\dots i-1}$ (löschen)
 - $a_{1\dots i} \rightarrow a_{1\dots i}b_j$ (einfügen)
 - $a_{1\dots i} \rightarrow a_{1\dots i_1}b_j$ (ersetzen)
- 3 Rekursion

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

4 Abhängigkeiten



⇒ Berechnung von links oben nach rechts unten. Zeilen- oder Spaltenweise.

5 Lösung steht in $E(n, m)$

Beispiel (Levenshteinabstand)

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

	\emptyset	Z	I	E	G	E
\emptyset	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 $E[0, \dots, m][0, \dots, n]$. $E[i, j]$: Minimaler Editierabstand der Zeichenketten (a_1, \dots, a_i) und (b_1, \dots, b_j)

Bottom-Up DP Algorithmus ED]

Dimension der Tabelle? Bedeutung der Einträge?

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

Berechnung eines Eintrags

- 2

Bottom-Up DP Algorithmus ED]

Dimension der Tabelle? Bedeutung der Einträge?

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

Berechnung eines Eintrags

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

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

3

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

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

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

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

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

4

Bottom-Up DP Algorithmus ED

Berechnungsreihenfolge

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

Wie kann sich Lösung aus der Tabelle konstruieren lassen?

- 4 Beginne bei $j = m, i = n$. Falls $E[i, j] = \text{repl}(a_i, b_j) + E(i - 1, j - 1)$ gilt, gib $a_i \rightarrow b_j$ 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 $j = 0$.