

## 22. Dynamic Programming III

---

FPTAS [Ottman/Widmayer, Kap. 7.2, 7.3, Cormen et al, Kap. 15,35.5], Optimale Suchbäume [Ottman/Widmayer, Kap. 5.7]

# Approximation

Sei ein  $\varepsilon \in (0, 1)$  gegeben. Sei  $I_{\text{opt}}$  eine bestmögliche Auswahl.  
Suchen eine gültige Auswahl  $I$  mit

$$\sum_{i \in I} v_i \geq (1 - \varepsilon) \sum_{i \in I_{\text{opt}}} v_i.$$

Summe der Gewichte darf  $W$  natürlich in keinem Fall überschreiten.

# Andere Formulierung des Algorithmus

**Bisher:** Gewichtsschranke  $w \rightarrow$  maximaler Nutzen  $v$

Umkehrung Nutzen  $v \rightarrow$  minimales Gewicht  $w$

$\Rightarrow$  **Alternative Tabelle:**  $g[i, v]$  gibt das minimale Gewicht an, welches

■ eine Auswahl der ersten  $i$  Gegenstände ( $0 \leq i \leq n$ ) hat, die

■ einen Nutzen von genau  $v$  aufweist ( $0 \leq v \leq \sum_{i=1}^n v_i$ ).

# Berechnung

## Initial

- $g[0, 0] \leftarrow 0$
- $g[0, v] \leftarrow \infty$  (Nutzen  $v$  kann mit 0 Gegenständen nie erreicht werden.).

## Berechnung

$$g[i, v] \leftarrow \begin{cases} g[i-1, v] & \text{falls } v < v_i \\ \min\{g[i-1, v], g[i-1, v-v_i] + w_i\} & \text{sonst.} \end{cases}$$

aufsteigend nach  $i$  und für festes  $i$  aufsteigend nach  $v$ .

Lösung ist der grösste Index  $v$  mit  $g[n, v] \leq w$ .

# Beispiel

$$E = \{(2, 3), (4, 5), (1, 1)\}$$

		$\xrightarrow{v}$									
		0	1	2	3	4	5	6	7	8	9
$i \downarrow$	$\emptyset$	0	$\infty$								
	$(2, 3)$	0	$\infty$	$\infty$	2	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$(4, 5)$	0	$\infty$	$\infty$	2	$\infty$	4	$\infty$	$\infty$	6	$\infty$
	$(1, 1)$	0	1	$\infty$	2	3	4	5	$\infty$	6	7

Auslesen der Lösung: wenn  $g[i, v] = g[i - 1, v]$  dann Gegenstand  $i$  nicht benutzt und bei  $g[i - 1, v]$  weiterfahren, andernfalls benutzt und bei  $g[i - 1, b - v_i]$  weiterfahren.

# Der Approximationstrick

Pseudopolynomielle Laufzeit wird polynomiell, wenn vorkommenden Werte in Polynom der Eingabelänge beschränkt werden können.

Sei  $K > 0$  *geeignet* gewählt. Ersetze die Nutzwerte  $v_i$  durch “gerundete Werte”  $\tilde{v}_i = \lfloor v_i/K \rfloor$  und erhalte eine neue Eingabe  $E' = (w_i, \tilde{v}_i)_{i=1\dots n}$ .

Wenden nun den Algorithmus auf Eingabe  $E'$  mit derselben Gewichtsschranke  $W$  an.

# Idee

**Beispiel**  $K = 5$

Eingabe Nutzwerte

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ..., 98, 99, 100

→

0, 0, 0, 0, 1, 1, 1, 1, 1, 2, ..., 19, 19, 20

Offensichtlich weniger unterschiedliche Nutzwerte

# Eigenschaften des neuen Algorithmus

- Auswahl von Gegenständen aus  $E'$  ist genauso gültig wie die aus  $E$ .  
Gewicht unverändert!
- Laufzeit des Algorithmus ist beschränkt durch  $\mathcal{O}(n^2 \cdot v_{\max}/K)$   
( $v_{\max} := \max\{v_i | 1 \leq i \leq n\}$ )

# Wie gut ist die Approximation?

Es gilt

$$v_i - K \leq K \cdot \left\lfloor \frac{v_i}{K} \right\rfloor = K \cdot \tilde{v}_i \leq v_i$$

Sei  $I'_{opt}$  eine optimale Lösung von  $E'$ . Damit

$$\begin{aligned} \left( \sum_{i \in I_{opt}} v_i \right) - n \cdot K &\stackrel{|I_{opt}| \leq n}{\leq} \sum_{i \in I_{opt}} (v_i - K) \leq \sum_{i \in I_{opt}} (K \cdot \tilde{v}_i) = K \sum_{i \in I_{opt}} \tilde{v}_i \\ &\stackrel{I'_{opt} \text{ optimal}}{\leq} K \sum_{i \in I'_{opt}} \tilde{v}_i = \sum_{i \in I'_{opt}} K \cdot \tilde{v}_i \leq \sum_{i \in I'_{opt}} v_i. \end{aligned}$$

# Wahl von $K$

Forderung:

$$\sum_{i \in I'} v_i \geq (1 - \varepsilon) \sum_{i \in I_{\text{opt}}} v_i.$$

Ungleichung von oben:

$$\sum_{i \in I'_{\text{opt}}} v_i \geq \left( \sum_{i \in I_{\text{opt}}} v_i \right) - n \cdot K$$

Also:  $K = \varepsilon \frac{\sum_{i \in I_{\text{opt}}} v_i}{n}.$

# Wahl von $K$

Wähle  $K = \varepsilon \frac{\sum_{i \in I_{\text{opt}}} v_i}{n}$ . Die optimale Summe ist aber unbekannt, daher wählen wir  $K' = \varepsilon \frac{v_{\text{max}}}{n}$ .<sup>40</sup>

Es gilt  $v_{\text{max}} \leq \sum_{i \in I_{\text{opt}}} v_i$  und somit  $K' \leq K$  und die Approximation ist sogar etwas besser.

Die Laufzeit des Algorithmus ist beschränkt durch

$$\mathcal{O}(n^2 \cdot v_{\text{max}}/K') = \mathcal{O}(n^2 \cdot v_{\text{max}}/(\varepsilon \cdot v_{\text{max}}/n)) = \mathcal{O}(n^3/\varepsilon).$$

---

<sup>40</sup>Wir können annehmen, dass vorgängig alle Gegenstände  $i$  mit  $w_i > W$  entfernt wurden.

# FPTAS

Solche Familie von Algorithmen nennt man **Approximationsschema**: die Wahl von  $\varepsilon$  steuert Laufzeit und Approximationsgüte.

Die Laufzeit  $\mathcal{O}(n^3/\varepsilon)$  ist ein Polynom in  $n$  und in  $\frac{1}{\varepsilon}$ . Daher nennt man das Verfahren auch ein voll polynomielles Approximationsschema **FPTAS - Fully Polynomial Time Approximation Scheme**

## 22.2 Optimale Suchbäume

---

# Optimale binäre Suchbäume

Gegeben: Suchwahrscheinlichkeiten  $p_i$  zu jedem Schlüssel  $k_i$  ( $i = 1, \dots, n$ ) und  $q_i$  zu jedem Intervall  $d_i$  ( $i = 0, \dots, n$ ) zwischen Suchschlüsseln eines binären Suchbaumes.  $\sum_{i=1}^n p_i + \sum_{i=0}^n q_i = 1$ .

Gesucht: Optimaler Suchbaum  $T$  mit Schlüsseltiefen  $\text{depth}(\cdot)$ , welcher die erwarteten Suchkosten

$$\begin{aligned} C(T) &= \sum_{i=1}^n p_i \cdot (\text{depth}(k_i) + 1) + \sum_{i=0}^n q_i \cdot (\text{depth}(d_i) + 1) \\ &= 1 + \sum_{i=1}^n p_i \cdot \text{depth}(k_i) + \sum_{i=0}^n q_i \cdot \text{depth}(d_i) \end{aligned}$$

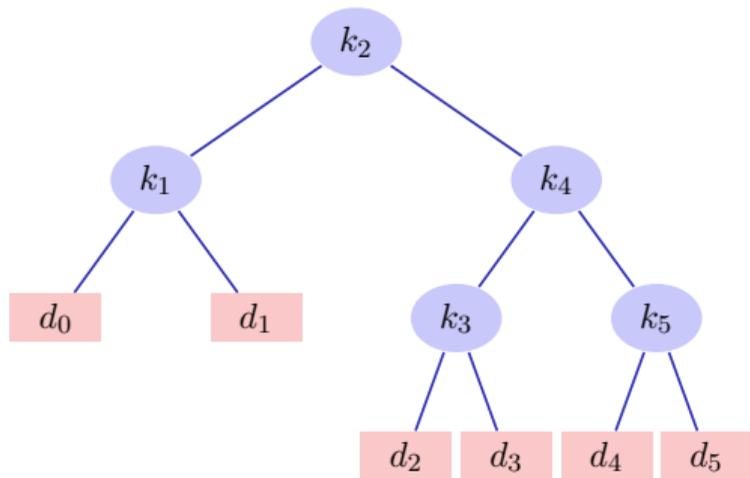
minimiert.

# Beispiel

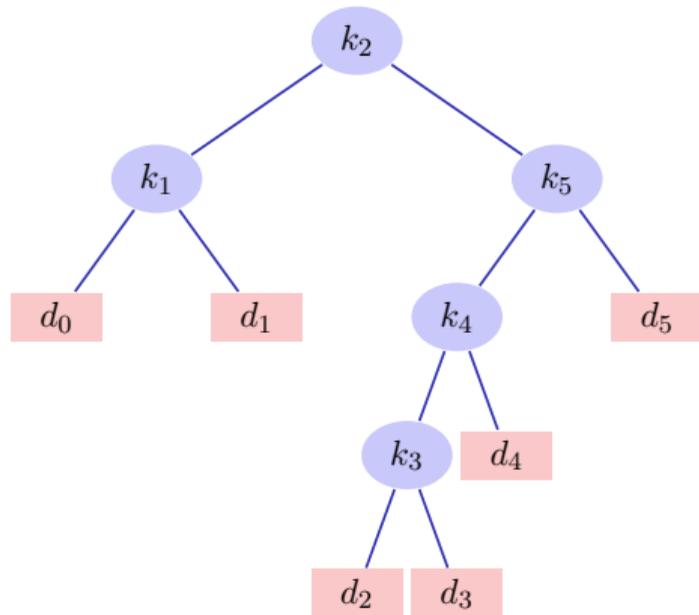
## Erwartete Häufigkeiten

$i$	0	1	2	3	4	5
$p_i$		0.15	0.10	0.05	0.10	0.20
$q_i$	0.05	0.10	0.05	0.05	0.05	0.10

# Beispiel



Suchbaum mit erwarteten Kosten  
2.8



Suchbaum mit erwarteten Kosten  
2.75

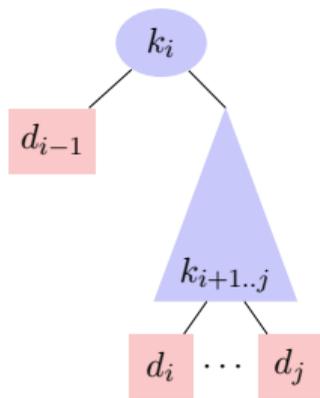
# Struktur eines optimalen Suchbaumes

- Teilsuchbaum mit Schlüsseln  $k_i, \dots, k_j$  und Intervallschlüsseln  $d_{i-1}, \dots, d_j$  muss für das entsprechende Teilproblem optimal sein.<sup>41</sup>
- Betrachten aller Teilsuchbäume mit Wurzel  $k_r$ ,  $i \leq r \leq j$  und optimalen Teilbäumen  $k_i, \dots, k_{r-1}$  und  $k_{r+1}, \dots, k_j$

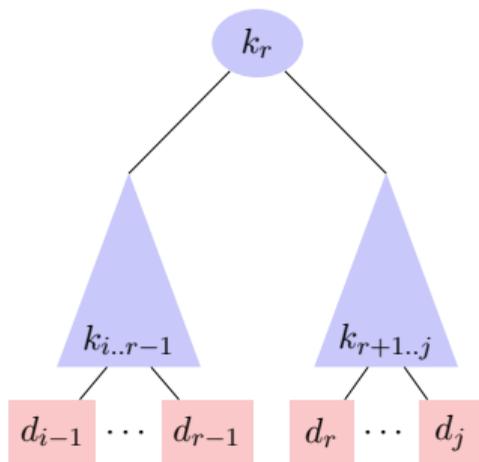
---

<sup>41</sup>Das übliche Argument: wäre er nicht optimal, könnte er durch eine bessere Lösung ersetzt werden, welche die Gesamtlösung verbessert.

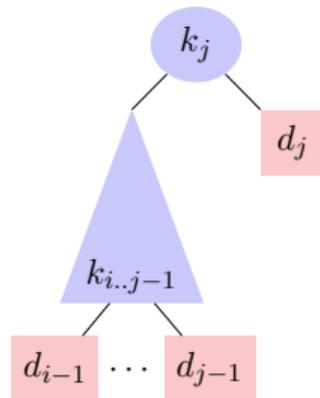
# Teilsuchbäume



leerer linker  
Teilsuchbaum



Teilsuchbäume links  
und rechts nichtleer



leerer rechter  
Teilsuchbaum

# Erwartete Suchkosten

Sei  $\text{depth}_T(k)$  die Tiefe des Knotens im Teilbaum  $T$ . Sei  $k_r$  die Wurzel eines Teilbaumes  $T_r$  und  $T_{L_r}$  und  $T_{R_r}$  der linke und rechte Teilbaum von  $T_r$ . Dann

$$\text{depth}_T(k_i) = \text{depth}_{T_{L_r}}(k_i) + 1, \quad (i < r)$$

$$\text{depth}_T(k_i) = \text{depth}_{T_{R_r}}(k_i) + 1, \quad (i > r)$$

# Erwartete Suchkosten

Seien  $e[i, j]$  die Kosten eines optimalen Suchbaumes mit Knoten  $k_i, \dots, k_j$ .

Basisfall:  $e[i, i - 1]$ , erwartete Suchkosten  $d_{i-1}$

Sei  $w(i, j) = \sum_{l=i}^j p_l + \sum_{l=i-1}^j q_l$ .

Wenn  $k_r$  die Wurzel eines optimalen Teilbaumes mit Schlüsseln  $k_i, \dots, k_j$ , dann

$$e[i, j] = p_r + (e[i, r - 1] + w(i, r - 1)) + (e[r + 1, j] + w(r + 1, j))$$

mit  $w(i, j) = w(i, r - 1) + p_r + w(r + 1, j)$ :

$$e[i, j] = e[i, r - 1] + e[r + 1, j] + w(i, j).$$

# Dynamic Programming

$$e[i, j] = \begin{cases} q_{i-1} & \text{falls } j = i - 1, \\ \min_{i \leq r \leq j} \{e[i, r - 1] + e[r + 1, j] + w[i, j]\} & \text{falls } i \leq j \end{cases}$$

# Berechnung

Tabellen  $e[1 \dots n + 1, 0 \dots n]$ ,  $w[1 \dots n + 1, 0 \dots m]$ ,  $r[1 \dots n, 1 \dots n]$  Initial

■  $e[i, i - 1] \leftarrow q_{i-1}$ ,  $w[i, i - 1] \leftarrow q_{i-1}$  für alle  $1 \leq i \leq n + 1$ .

Berechnung

$$w[i, j] = w[i, j - 1] + p_j + q_j$$

$$e[i, j] = \min_{i \leq r \leq j} \{e[i, r - 1] + e[r + 1, j] + w[i, j]\}$$

$$r[i, j] = \arg \min_{i \leq r \leq j} \{e[i, r - 1] + e[r + 1, j] + w[i, j]\}$$

für Intervalle  $[i, j]$  mit ansteigenden Längen  $l = 1, \dots, n$ , jeweils für  $i = 1, \dots, n - l + 1$ . Resultat steht in  $e[1, n]$ , Rekonstruktion via  $r$ . Laufzeit  $\Theta(n^3)$ .

# Beispiel

$i$	0	1	2	3	4	5
$p_i$		0.15	0.10	0.05	0.10	0.20
$q_i$	0.05	0.10	0.05	0.05	0.05	0.10

$e$

$j$							
0	0.05						
1	0.45	0.10					
2	0.90	0.40	0.05				
3	1.25	0.70	0.25	0.05			
4	1.75	1.20	0.60	0.30	0.05		
5	2.75	2.00	1.30	0.90	0.50	0.10	
	1	2	3	4	5	6	$i$

$w$

$j$							
0	0.05						
1	0.30	0.10					
2	0.45	0.25	0.05				
3	0.55	0.35	0.15	0.05			
4	0.70	0.50	0.30	0.20	0.05		
5	1.00	0.80	0.60	0.50	0.35	0.10	
	1	2	3	4	5	6	$i$

$r$

$j$						
1	1					
2	1	2				
3	2	2	3			
4	2	2	4	4		
5	2	4	5	5	5	
	1	2	3	4	5	$i$