



Dynamische Programmierung (3)

Konstruktion optimaler Suchbäume

Prof. Dr. S. Albers
Prof. Dr. Th. Ottmann

Rekursiver Ansatz: Lösen eines Problems durch Lösen mehrerer kleinerer Teilprobleme, aus denen sich die Lösung für das Ausgangsproblem zusammensetzt.

Phänomen: Mehrfachberechnungen von Lösungen

Methode: Speichern einmal berechneter Lösungen in einer Tabelle für spätere Zugriffe.

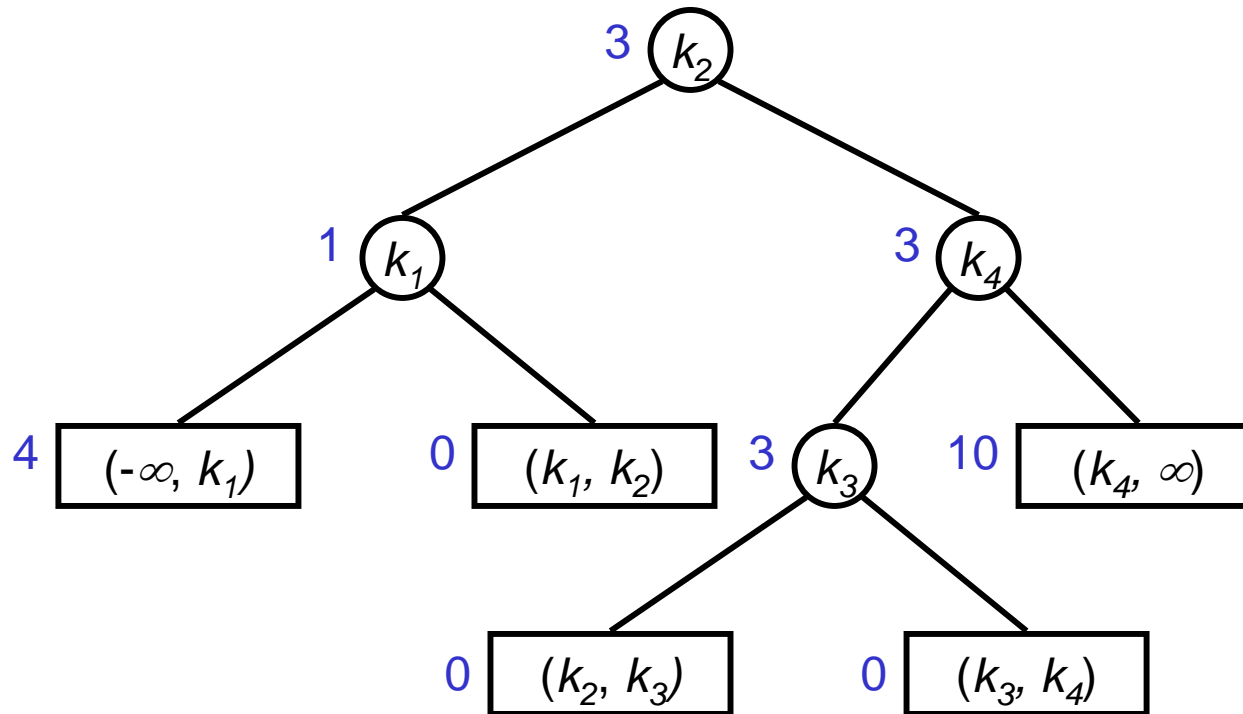
Das Optimalitätsprinzip

Typische Anwendung für dynamisches Programmieren:
Optimierungsprobleme

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

Konstruktion optimaler Suchbäume

$(-\infty, k_1)$ k_1 (k_1, k_2) k_2 (k_2, k_3) k_3 (k_3, k_4) k_4 (k_4, ∞)
 4 1 0 3 0 3 0 3 10



Gewichtete Pfadlänge:

$$3 \cdot 1 + 2 \cdot (1 + 3) + 3 \cdot 3 + 2 \cdot (4 + 10)$$

Konstruktion optimaler Suchbäume

Gegeben: Schlüsselmenge S

$$S = \{k_1, \dots, k_n\} \quad -\infty = k_0 < k_1 \leq \dots \leq k_n < k_{n+1} = \infty$$

a_j : (absolute) Häufigkeit, mit der nach k_j gesucht wird

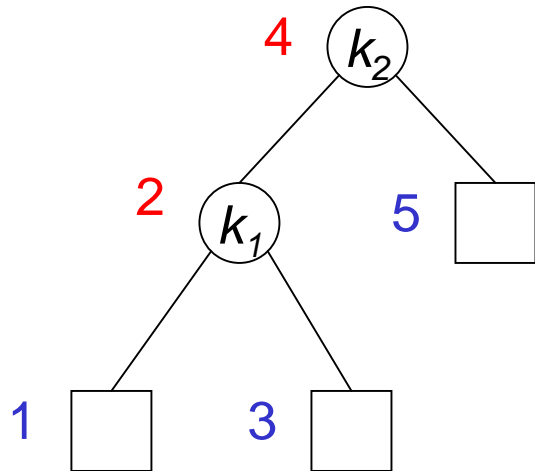
b_j : (absolute) Häufigkeit, mit der nach $x \in \{k_j, k_{j+1}\}$ gesucht wird

Gewichtete Pfadlänge $P(T)$ eines Suchbaumes T für S :

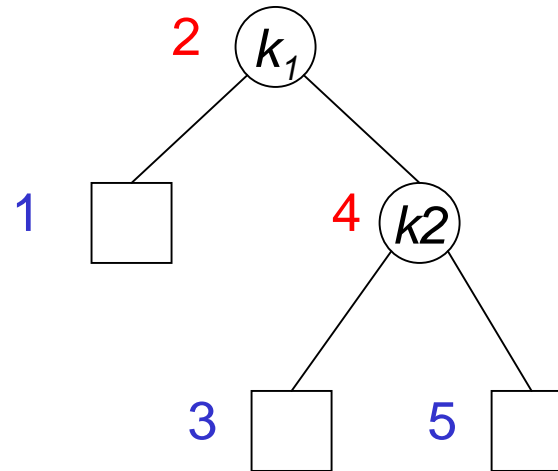
$$P(T) = \sum_{i=1}^n (\text{Tiefe}(k_i) + 1)a_i + \sum_{j=0}^n \text{Tiefe}((k_j, k_{j+1}))b_j$$

Gesucht: Ein Suchbaum für S mit minimaler Pfadlänge P

Konstruktion optimaler Suchbäume

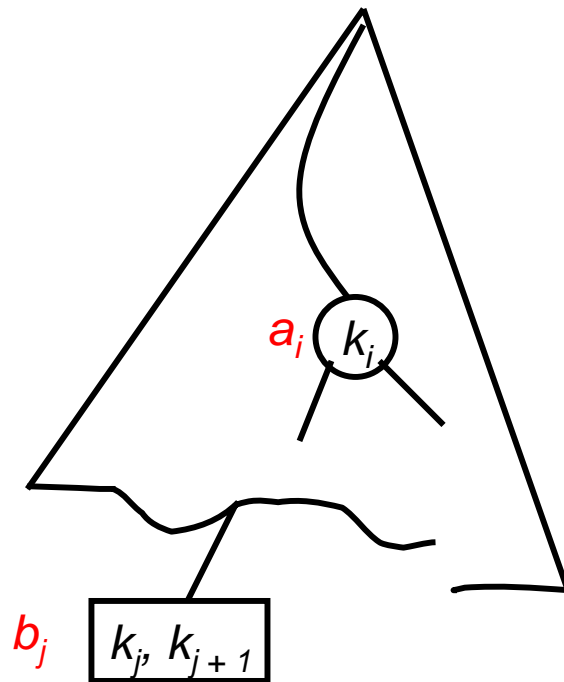


$$P(T_1) = 21$$



$$P(T_2) = 27$$

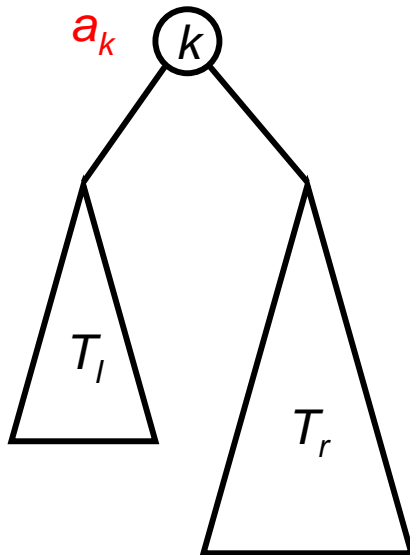
Konstruktion optimaler Suchbäume



Ein Suchbaum mit minimal möglicher gewichteter Pfadlänge ist ein optimaler Suchbaum

Konstruktion optimaler Suchbäume

T



$$P(T) = P(T_l) + W(T_l) + P(T_r) + W(T_r) + a_{\text{Wurzel}}$$

$$= P(T_l) + P(T_r) + W(T) \text{ mit}$$

$W(T) :=$ Gesamtgewicht der Knoten von T

Ist T Baum mit minimaler gewichteter Pfadlänge, so auch T_l und T_r

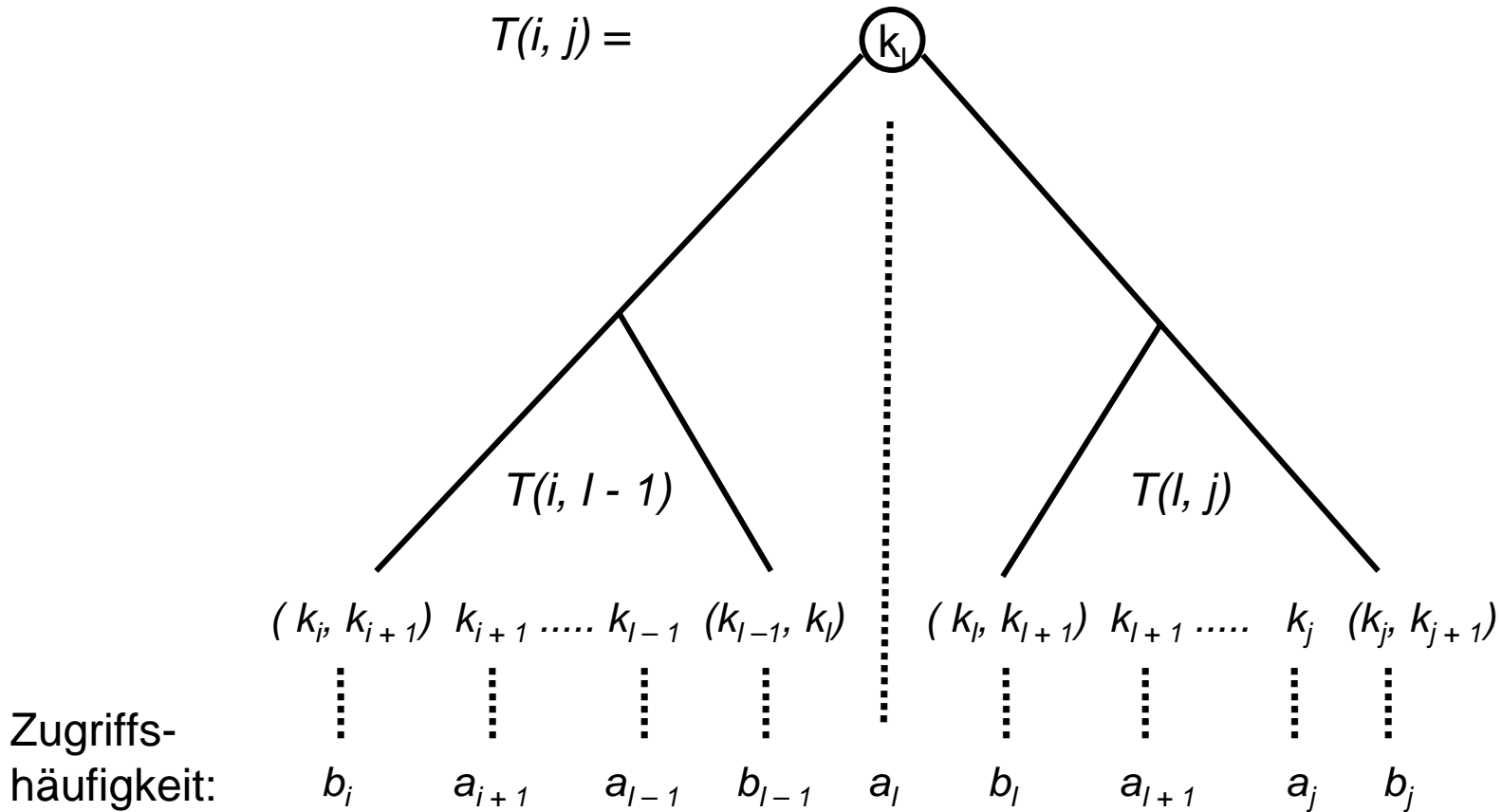
Sei

$T(i, j)$ optimaler Suchbaum für $(k_i, k_{i+1}) \dots (k_j, k_{j+1})$,

$W(i, j)$ das Gewicht von $T(i, j)$, also $W(i, j) = b_i + a_{i+1} + \dots + a_j + b_j$,

$P(i, j)$ die gewichtete Pfadlänge von $T(i, j)$.

Konstruktion optimaler Suchbäume



Konstruktion optimaler Suchbäume

$$W(i, i) = b_i \quad , \text{ für } 0 \leq i \leq n$$

$$W(i, j) = W(i, j - 1) + a_j + b_j \quad , \text{ für } 0 \leq i < j \leq n$$

$$P(i, i) = 0 \quad , \text{ für } 0 \leq i \leq n$$

$$P(i, j) = W(i, j) + \min_{i < l \leq j} \{ P(i, l - 1) + P(l, j) \}, \text{ für } 0 \leq i < j \leq n$$

$r(i, j)$ = diejenige Zahl, für die das Minimum angenommen wird

Konstruktion optimaler Suchbäume

Anfangsfälle

Fall 1: $h = j - i = 0$

$$T(i, i) = (k_i, k_{i+1})$$

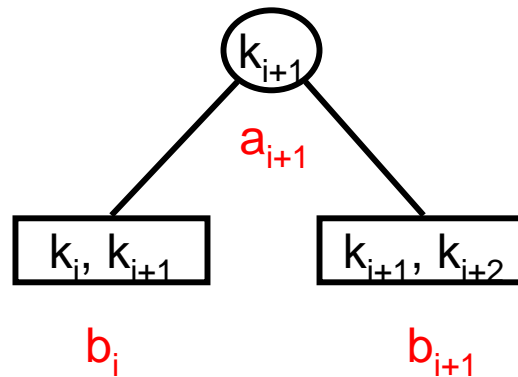
$$W(i, i) = b_i$$

$$P(i, i) = 0, \quad r(i, i) \text{ undefiniert}$$

Konstruktion optimaler Suchbäume

Fall 2: $h = j - i = 1$

$T(i, i+1)$



$$W(i, i+1) = b_i + a_{i+1} + b_{i+1} = W(i, i) + a_{i+1} + W(i+1, i+1)$$

$$P(i, i+1) = W(i, i+1)$$

$$r(i, i+1) = i+1$$

Berechnung der optimalen Pfadlänge mit dynamischer Programmierung



Fall 3: $h = j - i > 1$

for $h = 2$ **to** n **do**

for $i = 0$ **to** $(n - h)$ **do**

 { $j = i + h$;

 finde das (größte) l , $i < l \leq j$, für das

$P(i, l - 1) + P(l, j)$ minimal wird;

$P(i, j) = P(i, l - 1) + P(l, j) + W(i, j)$;

$r(i, j) = l$;

 }

Konstruktion optimaler Suchbäume

Definiere:

$$\begin{aligned}
 P(i, j) &:= \text{opt. Pfadlänge für} \\
 W(i, j) &:= \text{Summe von}
 \end{aligned}
 \left. \vphantom{\begin{aligned} P(i, j) \\ W(i, j) \end{aligned}} \right\} b_i a_{i+1} b_{i+1} \cdots a_j b_j$$

Dann ist

$$W(i, j) = \begin{cases} b_i & \text{falls } i = j \\ W(i, j-1) + a_j + W(j, j) & \text{sonst} \end{cases}$$

$$P(i, j) = \begin{cases} 0 & \text{falls } i = j \\ W(i, j) + \min_{i < l \leq j} \{P(i, l-1) + P(l, j)\} & \text{sonst} \end{cases}$$

→ Die gesuchte Lösung $P(0, n)$ kann in Zeit $O(n^3)$ und Platz $O(n^2)$ berechnet werden.

Konstruktion optimaler Suchbäume

Satz

Ein optimaler Suchbaum für n Schlüssel und $n + 1$ Intervalle mit gegebenen Zugriffshäufigkeiten kann in Zeit $O(n^3)$ und Platz $O(n^2)$ konstruiert werden.