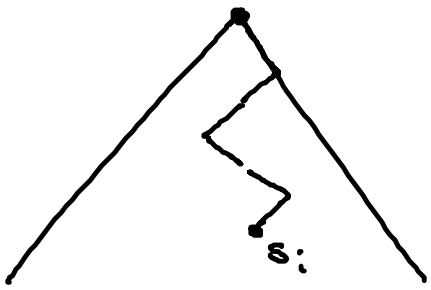
Optimale statische Suchbäume

Kenne alle Suchschlüssel $S = \{s_1, \dots, s_n\}$ und Zugriffshäufigkeiten $p_1, \dots, p_n$ mit denen auf $s_1, \dots, s_n$ zugegriffen wird.
Bsp: Daten auf CD, etc.

Ziel: Bestimme Suchbaum T für S und zugehörige p_i, so dass die **mittlere Zugriffszeit** klein ist.



Anzahl nötiger Vergleiche beim Suchen nach s_i beträgt

$h_T(s_i) + 1$

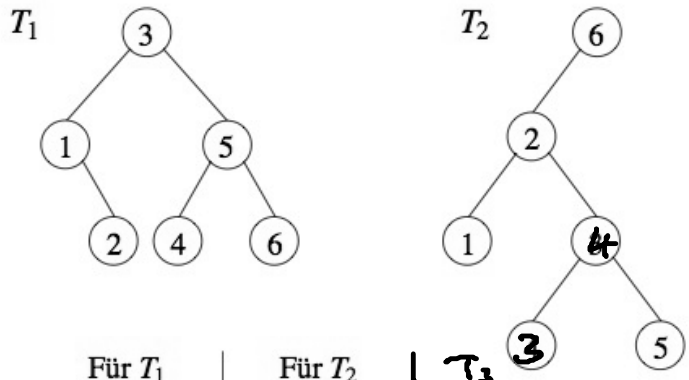
Mittlere Zugriffszeit:

$$E_p(T) := \frac{\sum_{i=1}^{n} p_i (h_T(s_i) + 1)}{\sum_{i=1}^{n} p_i} \qquad \left(p_i = \frac{p_i}{\sum_{j=1}^{n} p_j} \right)$$

$$= \sum_{i=1}^{n} p_i \cdot (h_T(s_i) + 1)$$

Beispiel:

Schlüssel s_i	1	2	3	4	5	6
Häufigkeit β_i in %	20	5	10	25	10	30



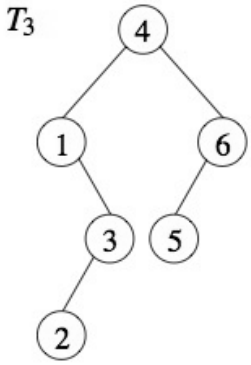


Abbildung 6.2: Der optimale statische Suchbaum für Bsp. 6.1.

Für T_1		Für T_2		T_3
$1 \cdot 10$	für 3	$1 \cdot 30$	für 6	$1 \cdot 25$
$2 \cdot 20$	für 1	$2 \cdot 5$	für 2	$2 \cdot 20$
$2 \cdot 10$	für 5	$3 \cdot 20$	für 1	$2 \cdot 30$
$3 \cdot 5$	für 2	$4 \cdot 10$	für 3	$3 \cdot 10$
$3 \cdot 25$	für 4	$3 \cdot 25$	für 4	$3 \cdot 5$
$3 \cdot 30$	für 6	$4 \cdot 10$	für 5	$4 \cdot 5$
250		~~270~~ 255		205

mittlere Zugriffszeit

2,5 ~~2,7~~ 2,55 2,05

Ziel:
- Finde Suchbaum T, so dass $E_p(T)$ minimal ist
- Entwickle einen Algorithmus, der dieses optimale T findet.

T ist optimaler statischer Suchbaum, falls

$E_p(T) \leq E_p(T')$ für alle Suchbäume T'

$$\Leftrightarrow \underbrace{\sum_{i=1}^{n} \beta_i \cdot (h_T(s_i)+1)}_{=: C(T)} \leq \underbrace{\sum_{i=1}^{n} \beta_i \cdot (h_{T'}(s_i)+1)}_{=: C(T')}$$

Betrachte also im Folgenden absolute Häufigkeiten β_i (statt p_i) und $C(T)$ (statt $E_p(T)$).

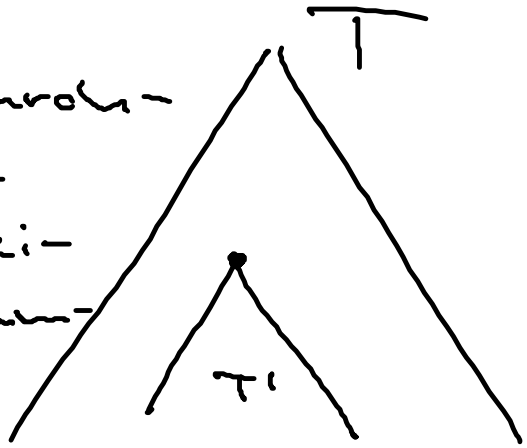
Struktur von optimalen stat. Suchb.

Prinzip der „optimalen Substruktur"
(z.B. kürzeste Wege, Huffman Codes)

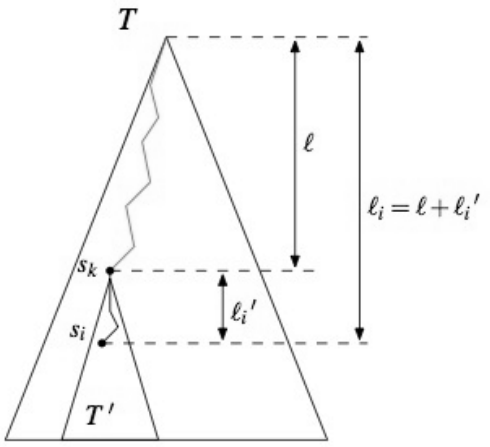
Satz 6.1 *Sei T ein Suchbaum für S und T' ein Teilbaum von T. Dann gilt:*

1. *Die Schlüsselmenge S' der Schlüssel in T' bildet eine konsekutive Teilsequenz (Intervall) $s_i < s_{i+1} < \cdots < s_j$ von $s_1 < s_2 < \cdots < s_n$.*
2. *„Prinzip der optimalen Substruktur"*
   Ist T optimal für S und $\beta_1, \ldots, \beta_n$, so ist T' optimal für die zugehörige Schlüsselmenge $S' = \{s_i, s_{i+1}, \ldots, s_j\}$ und die Häufigkeiten $\beta_i, \beta_{i+1}, \ldots, \beta_j$ ist.

Beweis: a) Der inorder-Durchlauf durch T liefert die Schlüssel in S in aufsteigender Sortierung (Suchbaumeigenschaft). Dieser Durchlauf besucht die Knoten in einem Teilbaum T' nacheinander ohne Unterbrechung durch Knoten außerhalb von T'.



5)



Definiere:

$$\left.\begin{aligned} \ell_i &:= l_T(s_i) + 1 \\ \ell_i' &:= l_{T'}(s_i) + 1 \\ \ell &:= l_T(s_k) \end{aligned}\right\} \ell_i = \ell_i' + \ell$$

$$C(T) = \sum_{q=1}^{n} p_q \cdot \ell_q$$

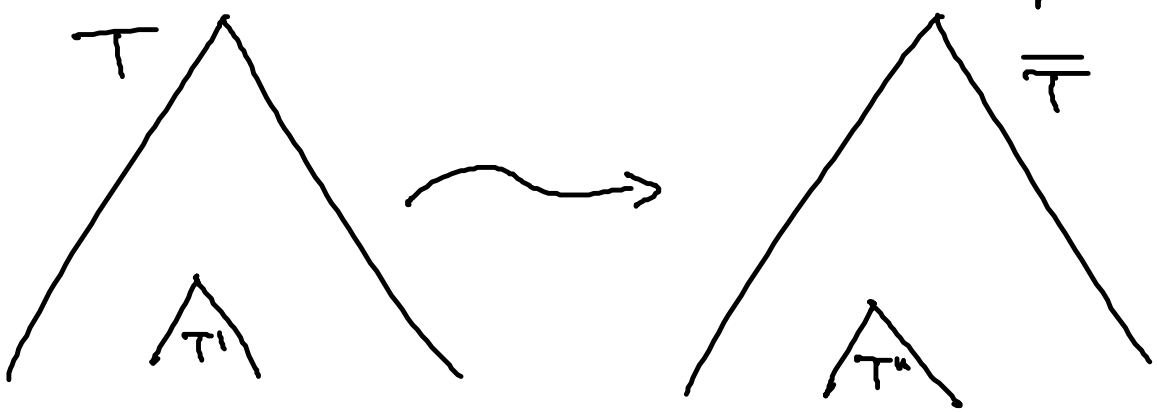
$$= \underbrace{\sum_{q=1}^{i-1} p_q \cdot \ell_q}_{=: V \text{ (vor } T')} + \sum_{q=i}^{j} p_q \cdot \ell_q + \underbrace{\sum_{q=j+1}^{n} p_q \cdot \ell_q}_{=: N \text{ (nach } T')}$$

$$= V + \sum_{q=i}^{j} p_q \cdot (\ell + \ell_q') + N$$

$$= \underbrace{V + N + \sum_{q=i}^{j} p_q \cdot \ell}_{\text{Konstante nach. von } T'} + \underbrace{\sum_{q=i}^{j} p_q \cdot \ell_q'}_{= C(T')}$$

$\Rightarrow C(T') \leq C(T'')$ für alle Suchb.

Sonst:



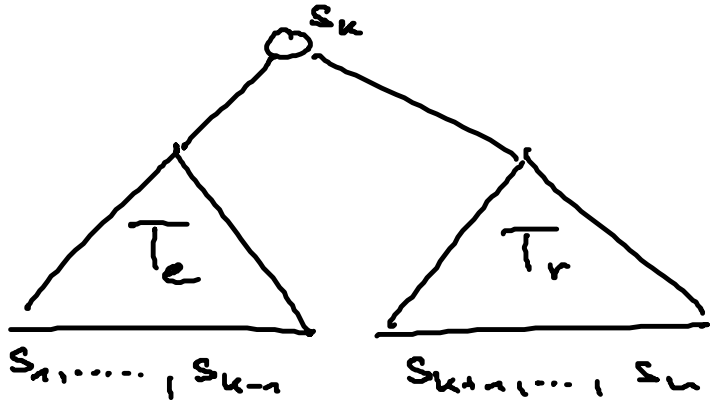
T'' für $s_{i+1}, \dots, s_j$
mit $p_{i+1}, \dots, p_j$

$$C(T) > C(\bar{T})$$

falls $C(T') > C(T'')$. □

Nutze die Einsicht aus dem Satz:

Falls s_k Wurzel des opt. stat. Suchbaumes ist, dann sieht er so aus:



T_l opt. stat. Suchbaum für $s_1, \dots, s_{k-1}$
$p_1, \dots, p_{k-1}$

T_r " " " " $s_{k+1}, \dots, s_n$
$p_{k+1}, \dots, p_n$

d.h.

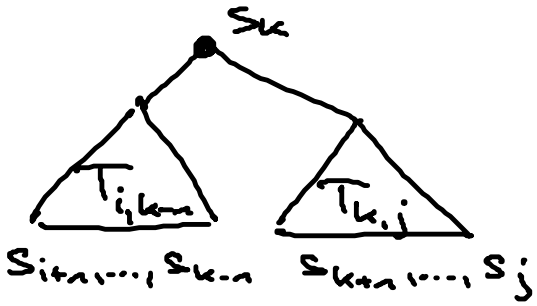
$$C(\text{opt. Suchbaum}) = \min_{k=1,\dots,n} \Big[C\big(\text{opt. Suchbaum für } s_1,\dots,s_{k-1}\big) + C\big(\text{opt. Suchbaum für } s_{k+1},\dots,s_n\big) + \sum_{i=1}^{n} p_i \Big] \quad (*)$$

Für Algorithmus bedeutet dies:

- Betrachte jede mögliche Wurzel s_k, $k=1,\dots,n$
- für diese Wahl, baue optimale Teilbäume an.
- Wurzel auswählen, die (*) minimiert.

Systematisiere diese Idee:

Es sei T_{ij} Suchbaum für $s_{i+1},\dots,s_j$. Wenn T_{ij} die Wurzel s_k hat, dann sind die Teilbäume $T_{i,k-1}$ und $T_{k,j}$



zugehörige Kosten:

..

$$C(T_{ij}) =: C_{ij} = p_{i+1} \cdot l^{ij}_{i+1} + \dots + p_j \cdot l^{ij}_j$$

$$= p_{i+1} \cdot (l^{ij}_{i+1} - 1) + \dots + 0 + \dots + p_j (l^{ij}_j - 1)$$

$$+ \underbrace{\sum_{q=i+1}^{j} p_q}_{=: w_{ij}}$$

$$\boxed{C_{ij} = C_{i,k-1} + C_{kj} + w_{ij}}$$

Rekursionsformel für T_{ij} und Wurzel k

Wende Prinzip der opt. Substruktur an:

$$C^{opt}_{ij} = \min_{i+1 \le k \le j} \left(C^{opt}_{i,k-1} + C^{opt}_{kj} + w_{ij} \right)$$

Idee des Algorithmus:

Berechne alle C^{opt}_{ij} und zugehörige T^{opt}_{ij} (opt. Teilbäume) nach aufst. Länge des Intervals $i+1, \dots, j$.

Beispiel: $S = \{1, \ldots, 6\}$

i:	1	2	3	4	5	6
β_i:	$\frac{1}{10}$	$\frac{1}{10}$	$\frac{1}{5}$	$\frac{1}{20}$	$\frac{3}{10}$	$\frac{1}{4}$

Intervalllänge 0:

$$C_{ii}^{opt} = 0 \quad \text{für alle } i$$

Intervalllänge 1:

$$\begin{aligned}
C_{0,1}^{opt} &= \omega_{0,1} + C_{0,0}^{opt} + C_{1,1}^{opt} = \omega_{0,1} = \beta_1 = 0,1 \\
C_{1,2}^{opt} &= \omega_{1,2} + C_{1,1}^{opt} + C_{2,2}^{opt} = \omega_{1,2} = \beta_2 = 0,1 \\
C_{2,3}^{opt} &= \omega_{2,3} + C_{2,2}^{opt} + C_{3,3}^{opt} = \omega_{2,3} = \beta_3 = 0,2 \\
C_{3,4}^{opt} &= \omega_{3,4} + C_{3,3}^{opt} + C_{4,4}^{opt} = \omega_{3,4} = \beta_4 = 0,05 \\
C_{4,5}^{opt} &= \omega_{4,5} + C_{4,4}^{opt} + C_{5,5}^{opt} = \omega_{4,5} = \beta_5 = 0,3 \\
C_{5,6}^{opt} &= \omega_{5,6} + C_{5,5}^{opt} + C_{6,6}^{opt} = \omega_{5,6} = \beta_6 = 0,25
\end{aligned}$$

Zugehörige opt. Teilbäume

Intervalllänge 2:

Schlüssel S_1, S_2:

$$C_{0,2}^{opt} = \min\left(C_{0,0}^{opt} + C_{1,2}^{opt} + \omega_{0,2},\ C_{0,1}^{opt} + C_{2,2}^{opt} + \omega_{0,2} \right)$$

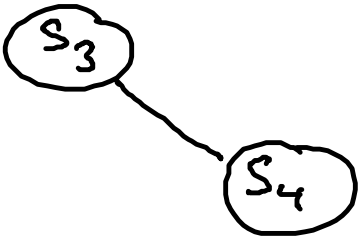
$0 \quad 0,1 \quad 0,2 \quad 0,1 \quad 0 \quad 0,2$

opt. Suchbäume:

Schlüssel S_2, S_3:

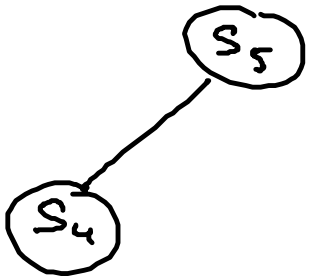
$C_{1,3}^{opt} = 0,4$

S_3, S_4:



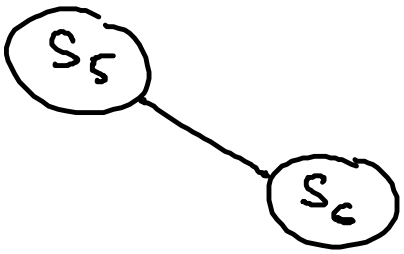
$0,3$

s_4, s_5 :



$0,4$

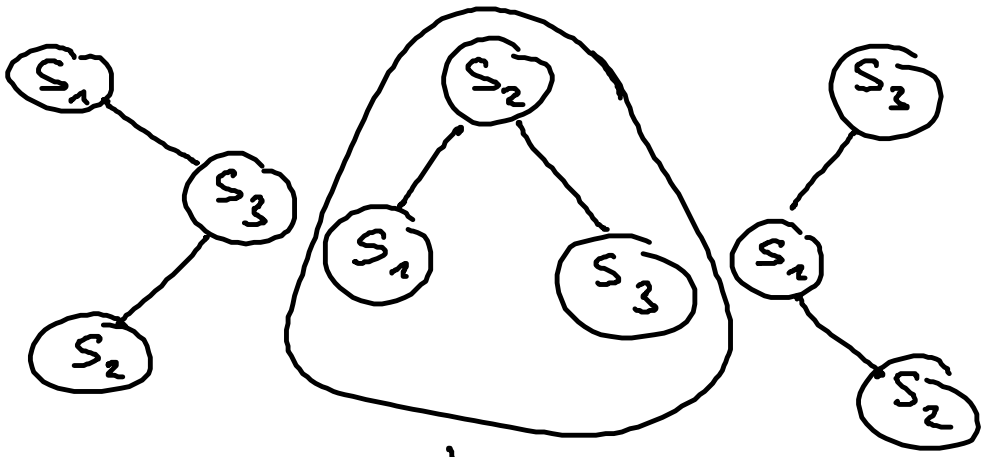
s_5, s_6 :



$0,8$

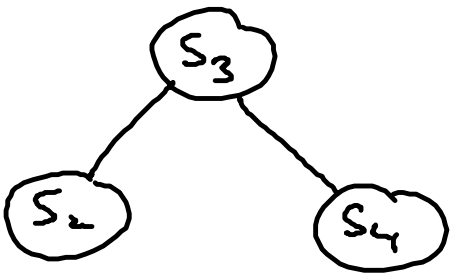
Intervalllänge 3:

s_1, s_2, s_3 :

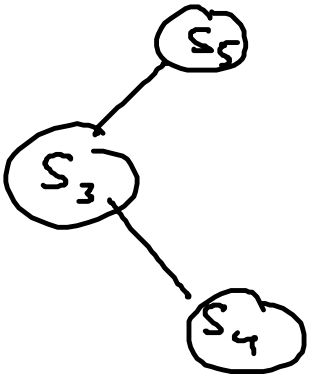


$C^{opt}_{0,3} = 97$

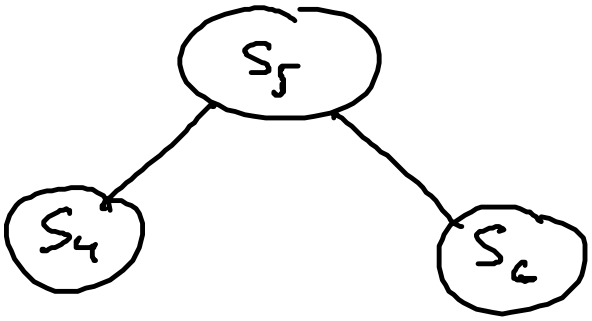
s_2, s_3, s_4 :



s_3, s_4, s_5 :



s_4, s_5, s_6 :

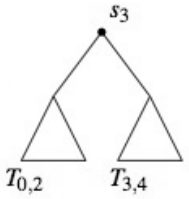


Intervalllänge 4:

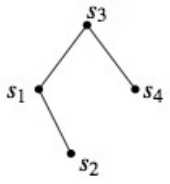
$$
\begin{aligned}
C_{0,4}{}^{opt} &= \omega_{0,4} + \min_{k=1,2,3,4} \left[C_{0,k-1}{}^{opt} + C_{k,4}{}^{opt} \right] \\
&= \omega_{o,4} + \min \left[C_{0,0}{}^{opt} + C_{1,4}{}^{opt}; C_{0,1}{}^{opt} + C_{2,4}{}^{opt}; C_{0,2}{}^{opt} + C_{3,4}{}^{opt}; C_{0,3}{}^{opt} + C_{4,4}{}^{opt} \right] \\
&= 0,45 + \min \left[0 + 0,5; 0,1 + 0,3; 0,3 + 0,05; 0,7 + 0 \right] \\
&= 0,45 + 0,35 = 0,8 \quad (\text{für } k = 3)
\end{aligned}
$$

$k = 3$

$T_{0,4}$ =



=

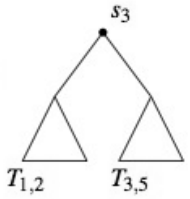


$$
\begin{aligned}
C_{1,5}{}^{opt} &= \omega_{1,5} + \min_{k=2,3,4,5} \left[C_{1,k-1}{}^{opt} + C_{k,5}{}^{opt} \right] \\
&= \omega_{1,5} + \min \left[C_{1,1}{}^{opt} + C_{2,5}{}^{opt}; C_{1,2}{}^{opt} + C_{3,5}{}^{opt}; C_{1,3}{}^{opt} + C_{4,5}{}^{opt}; C_{1,4}{}^{opt} + C_{5,5}{}^{opt} \right] \\
&= 0,65 + \min \left[0 + 0,85; 0,1 + 0,4; 0,4 + 0,3; 0,5 + 0 \right] \\
&= 0,65 + 0,5 = 1,15 \quad (\text{für } k = 3 \text{ oder } k = 5)
\end{aligned}
$$

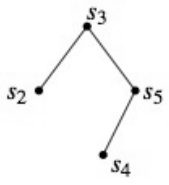
$k = 3$

In der Abbildung ist $k = 3$:

$T_{1,5}$ =



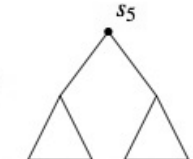
=



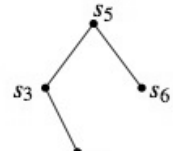
$$
\begin{aligned}
C_{2,6}{}^{opt} &= \omega_{2,6} + \min_{k=3,4,5,6} \left[C_{2,k-1}{}^{opt} + C_{k,6}{}^{opt} \right] \\
&= \omega_{2,6} + \min \left[C_{2,2}{}^{opt} + C_{3,6}{}^{opt}; C_{2,3}{}^{opt} + C_{4,6}{}^{opt}; C_{2,4}{}^{opt} + C_{5,6}{}^{opt}; C_{2,5}{}^{opt} + C_{6,6}{}^{opt} \right] \\
&= 0,8 + \min \left[0 + 0,9; 0,2 + 0,8; 0,3 + 0,25; 0,85 + 0 \right] \\
&= 0,8 + 0,55 = 1,35 \quad (\text{für } k = 5)
\end{aligned}
$$

$k = 5$

$T_{2,6}$ =



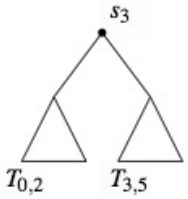
=



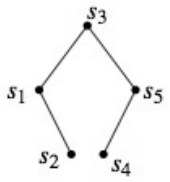
Intervalllänge 5:

$$
\begin{aligned}
C_{0,5}{}^{opt} &= \omega_{0,5} + \min_{k=1,2,3,4,5} \left[C_{0,k-1}{}^{opt} + C_{k,5}{}^{opt} \right] \\
&= \omega_{0,5} + \min \left[C_{0,0}{}^{opt} + C_{1,5}{}^{opt}; C_{0,1}{}^{opt} + C_{2,5}{}^{opt}; C_{0,2}{}^{opt} + C_{3,5}{}^{opt}; C_{0,3}{}^{opt} + C_{4,5}{}^{opt}; \right. \\
&\qquad \left. C_{0,4}{}^{opt} + C_{5,5}{}^{opt} \right] \\
&= 0,75 + \min [0+1,15; 0,1+0,85; 0,3+0,4; 0,7+0,3; 0,8+0] \qquad k = 3 \\
&= 0,75 + 0,7 = 1,45 \quad (\text{für } k = 3)
\end{aligned}
$$

$T_{0,5}$ =

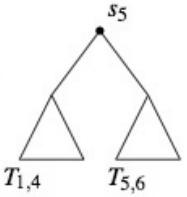


=

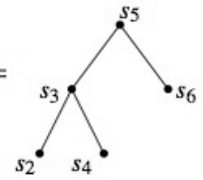


$$
\begin{aligned}
C_{1,6}{}^{opt} &= \omega_{1,6} + \min_{k=2,3,4,5,6} \left[C_{1,k-1}{}^{opt} + C_{k,6}{}^{opt} \right] \\
&= \omega_{1,6} + \min \left[C_{1,1}{}^{opt} + C_{2,6}{}^{opt}; C_{1,2}{}^{opt} + C_{3,6}{}^{opt}; C_{1,3}{}^{opt} + C_{4,6}{}^{opt}; C_{1,4}{}^{opt} + C_{5,6}{}^{opt}; \right. \\
&\qquad \left. C_{1,5}{}^{opt} + C_{6,6}{}^{opt} \right] \\
&= 0,9 + \min [0+1,35; 0,1+0,9; 0,4+0,8; 0,5+0,25; 1,15+0] \\
&= 0,9 + 0,75 = 1,65 \quad (\text{für } k = 5)
\end{aligned}
$$

$T_{1,6}$ =



=

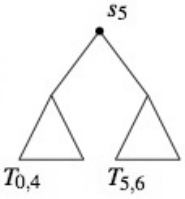


Intervalllänge 6:

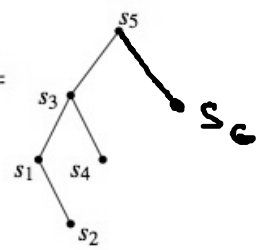
$$
\begin{aligned}
C_{0,6}{}^{opt} &= \omega_{0,6} + \min_{k=1,2,3,4,5,6} \left[C_{0,k-1}{}^{opt} + C_{k,6}{}^{opt} \right] \\
&= \omega_{0,6} + \min \left[C_{0,0}{}^{opt} + C_{1,6}{}^{opt}; C_{0,1}{}^{opt} + C_{2,6}{}^{opt}; C_{0,2}{}^{opt} + C_{2,6}{}^{opt}; C_{0,3}{}^{opt} + C_{4,6}{}^{opt}; \right. \\
&\qquad \left. C_{0,4}{}^{opt} + C_{5,6}{}^{opt}; C_{0,5}{}^{opt} + C_{6,6}{}^{opt} \right] \\
&= 1 + \min [0+1,65; 0,1+1,35; 0,3+0,9; 0,7+0,8; 0,8+0,25; 1,45+0] \\
&= 1 + 1,05 = 2,05 \quad (\text{für } k = 5)
\end{aligned}
$$

So ergibt sich dann schließlich der optimale statische Suchbaum für die gegebene Schlüsselmenge mit den zugehörigen Häufigkeiten zu

$T_{0,6}$ =



=



Dieser Alg. ist ein Beispiel für das Prinzip der dyn. Programmierung.

Aufwand des Algorithmus:

n Bäume	mit	1	Knoten	n
$n-1$	"	"	2	"	$n-1$
$n-2$	"	"	3	"	$n-2$
⋮				
2	"	"	$n-1$	"	2
1	"	"	n	"	1
					$\Theta(n^2)$

Aufwand für Berechnung eines dieser opt. Teilbäume: $O(n)$

$\Rightarrow$ Gesamtaufwand: $O(n^3)$