1 Optimaler binärer Suchbaum

Elemente $e_1 < e_2 < \ldots < e_m$ mit Zugriffswarscheinlichkeiten $p_1, p_2, \ldots, p_m$. Finde Suchbaum mit minimalem *erwarteten* Vergleichen.

$$
\begin{aligned}
w_{i,j} &:= \sum_{k=i}^{j} p_k \\
e_{i,j} &:= \min_{i \le r \le j} (e_{i,r-1} + e_{r+1,j}) + 1 \cdot w_{i,j} \qquad \text{Erwartungswert der Vergleiche im Unterbaum mit } e_i, \ldots, e_j
\end{aligned}
$$

Anfang der dynamischen Programmierung: $e_{i,i} = w_{i,i} = p_i$

k	1	2	3	4
p_k	0.4	0.1	0.2	0.3

$w_{i,j}$	1	2	3	4
1	0.4	0.5	0.7	1
2		0.1	0.3	0.6
3			0.2	0.5
4				0.3

$e_{i,j}$	1	2	3	4
1	0.4	$0.6^{(1)}$	$1.1^{(1)}$	$1.9^{(3)}$
2		0.1	$0.4^{(3)}$	$1.0^{(3,4)}$
3			0.2	$0.7^{(4)}$
4				0.3

2 Amortisierte Analyse

2.1 Delayed Insertion

- `insert` einfach in verkettete Liste in Zeit $O(1)$
- Bei `find`, `remove` erste Listenelemente in z.B. einen AVL-Baum einfügen, danach ensptrechend die Operation des Baumes benutzten, die in $O(\log n)$ Zeit machbar sein sollen
- Länge der Liste l_i, enthaltene Element in gesamter Struktur n_i

Naiv: `find`, `remove` bis zu n Elementen benötigt $O(n \log n)$
Abschätzen der Operationen `insert` und `find`

$$
\begin{aligned}
t_{insert} &\le 1 \\
t_{find} &\le \underbrace{l_i \cdot \log(n_i + 1) \cdot c}_{\text{Liste einfügen}} + \underbrace{\log(n_i + 1)}_{\texttt{find}}
\end{aligned}
$$

Ansatz $\boxed{\Phi_i = c l_i \log(n_i + 1)}$ erfüllt

- $\Phi_0 = 0$ wegen $l_0 = 0$
- $\Phi_i \ge 0$ für alle i

und ergibt:

$$
\begin{aligned}
t_{insert} + \Delta\Phi &\le 1 + \Phi_{i+1} - \Phi_i = 1 + c l_{i+1}(\log n_{i+1} + 1) - c l_i \log(n_i + 1) \\
&= 1 + c(l_i + 1)\log(n_i + 2) - c l_i \log(n_i + 1) \\
&= c \log n_i + O(1)
\end{aligned}
$$

$$
\begin{aligned}
t_{find} + \Delta\Phi &\le c l_i \log(n_i + 1) + \log(n_i + 1) + \Phi_{i+1} - \Phi_i \\
&= c l_i \log(n_i + 1) + \log(n_i + 1) + c l_{i+1} \log(n_{i+1} + 1) - c l_i \log(n_i + 1) | l_{i+1} = 0 \\
&= \log(n_i + 1) = \log n_i + O(1)
\end{aligned}
$$

Insgesamt ergibt sich

$$
\sum_{i=0}^{n-1} t_i \le \left(\sum_{i=0}^{n-1} t_i \right) + \Phi_n - \Phi_0 = \sum_{i=0}^{n-1} t_i + \Phi_{i+1} - \Phi_i \le n \cdot (c \log n_i + O(1)) \le cn \log n + O(n)
$$

Und damit durchschnittlich pro Operation $\boxed{c \log n + O(1) = O(\log n)}$
Shortcut: jedes `insert` wird nur hinausgezögert, kann also nur $O(\log n)$ sein.