

Quicksort und Suchbäume

In dieser Notiz soll dargestellt werden, daß es einen tieferen Zusammenhang zwischen *quicksort* und dem bekannten Konzept der binären Suchbäume gibt. Das quantitative Verhalten von *quicksort* leitet sich demnach aus Eigenschaften der Weglänge von Such- bzw. Prioritätsbäumen ab. Dies gibt einen kleinen Eindruck von der in jüngster Zeit aufgekommenen Analyse­methode der “randomisierten Suchbäume”.

1 Suchbäume und Prioritätsbäume

- Ist $M \subset \mathbf{Z}$ eine (endliche) Menge, so ist die Menge \mathcal{BST}_M der *binären Suchbäume* (“binary search trees”) über M rekursiv definiert durch

$$- \mathcal{BST}_\emptyset := \{\emptyset\}$$

$$- \text{falls } \sharp M \geq 1 : \mathcal{BST}_M := \bigcup_{m \in M} (\mathcal{BST}_{M^{<m}} \times \{m\} \times \mathcal{BST}_{M^{>m}})$$

Dabei bezeichnet \emptyset den “leeren Baum”, und $M^{<m} := \{x \in M; x < m\}$, $M^{>m} := \{x \in M; x > m\}$.

Eine binärer Suchbaum über M ist also ein mit den Elementen von M an den Knoten beschrifteter Baum, bei dem für jeden Knoten gilt: die Beschriftung dieses Knotens ist echt größer (kleiner) als alle Beschriftungen von Knoten des an ihm angehängten linken (rechten) Teilbaums.

Ist $\sharp M = n$, so enthält \mathcal{BST}_M genau $c_n = \frac{1}{n+1} \binom{2n}{n}$ Elemente (CATALAN-Zahlen!), denn jeder binäre Baum mit n Knoten lässt sich auf genau eine Weise mit den Elementen von M so beschriften, daß ein binärer Suchbaum entsteht.

- Ist $M \subset \mathbf{Z}$ eine (endliche) Menge, so ist die Menge \mathcal{BPT}_M der *binären Prioritätsbäume* (“binary priority trees”) über M rekursiv definiert durch

$$- \mathcal{BPT}_\emptyset := \{\emptyset\}$$

$$- \text{falls } \sharp M \geq 1 \text{ und } m = \min M :$$

$$\mathcal{BPT}_M := \bigcup_{M' \subseteq M \setminus \{m\}} (\mathcal{BPT}_{M'} \times \{m\} \times \mathcal{BPT}_{M \setminus \{m\} \setminus M'})$$

Ein binärer Prioritätsbaum über M ist also ein mit den Elementen von M an den Knoten beschrifteter Baum, bei dem für jeden Knoten gilt: die Beschriftung dieses Knotens ist echt kleiner als alle Beschriftungen von Knoten der an ihm angehängten Teilbäume.

Ist $\sharp M = n$, so enthält \mathcal{BPT}_M genau $n!$ Elemente — es ist einfach, eine Bijektion zwischen den Permutationen von M und den Elementen von \mathcal{BPT}_M anzugeben.

- Ist $M \subset \mathbf{Z}$ eine endliche Menge und $\pi : M \rightarrow \mathbf{Z}$ eine injektive Abbildung (“Prioritätsfunktion”), so gibt es genau ein $t \in \mathcal{BST}_M$ derart, daß $\pi(t) \in \mathcal{BPT}_{\pi(M)}$. Hierbei bezeichnet $\pi(t)$ den mit den Elementen der Menge $\pi(M)$ beschrifteten Baum, der aus t dadurch hervorgeht, daß man an jedem Knoten die Beschriftung m durch ihre Priorität $\pi(m)$ ersetzt. Dieser eindeutig

bestimmte Baum $t \in \mathcal{BST}_M$ wird für spätere Verwendung mit $t_{M,\pi}$ bezeichnet. Sein Bild $\pi(t_{M,\pi}) \in \mathcal{BPT}_{\pi(M)}$ wird mit t_π abgekürzt.

- Ist t irgendein Baum (mit Wurzel), so bezeichnet man als *Weglänge* $w(t)$ von t die Summe aller Längen von Wegen, die von irgendeinem einem Knoten von t zur Wurzel führen, also die Summe aller Distanzen von Baumknoten zur Wurzel. Eine äquivalente Formulierung:

$$w(t) = \#\{ (x, y) ; x, y \text{ Knoten in } t \text{ mit } x \prec_t y \}$$

wobei \prec_t die (strikte) Vorgängerordnung in t bezeichnet.

Für Binärbäume t gilt offensichtlich

$$w(t) = \begin{cases} 0 & \text{falls } t = \emptyset \\ w(t') + w(t'') + \#t' + \#t'' & \text{falls } t = (t', s, t'') \end{cases}$$

2 Mittlere Weglänge in Prioritätsbäumen

- Auf Grund ihrer rekursiven Definition lassen binäre Prioritätsbäume folgende Zerlegung zu:

ist $t \in \mathcal{BPT}_{\{1,\dots,n+1\}}$, so entspricht diesem Baum ein-eindeutig ein Tripel (M', t_1, t_2) mit $M' \subseteq \{2, \dots, n+1\}$, $t_1 \in \mathcal{BPT}_{\{1,\dots,\#M'\}}$ und $t_2 \in \mathcal{BPT}_{\{1,\dots,n-\#M'\}}$

Dabei entspricht die Menge M' der Menge der Beschriftungen des linken Teilbaumes von t , entsprechend ist $\{2, \dots, n+1\} \setminus M'$ die Menge der Beschriftungen des rechten Teilbaumes von t . Aus der Kenntnis von M' und t_1 (bzw. t_2) kann man den linken (bzw. rechten) Teilbaum von t durch entsprechende ordnungstreue Umbenennung rekonstruieren.

- Es geht nun um die *mittlere Weglänge* für binäre Prioritätsbäume mit n Knoten. Sei also

$$w_n := \sum_{t \in \mathcal{BPT}_{\{1,\dots,n\}}} w(t)$$

Aus der gerade beschriebenen Zerlegungseigenschaft und dem im vorigen Abschnitt erwähnten Verhalten der Weglängen beim Übergang zu den Teilbäumen folgt

$$w_{n+1} = \sum_{j=0}^n \binom{n}{j} \sum_{t_1, t_2} (w(t_1) + w(t_2) + \#t_1 + \#t_2)$$

wobei die Summe \sum_{t_1, t_2} über alle Paare $(t_1, t_2) \in \mathcal{BPT}_{\{1,\dots,j\}} \times \mathcal{BPT}_{\{1,\dots,n-j\}}$ läuft.

Beachtet man nun $\#t_1 + \#t_2 = n$ und $bpt_k := \#\mathcal{BPT}_{\{1,\dots,k\}} = k!$, so erhält man

$$\begin{aligned} w_{n+1} &= \sum_{j=0}^n \binom{n}{j} (w_j \#bpt_{n-j} + \#bpt_n w_{n-j} + n bpt_j bpt_{n-j}) \\ &= n! \sum_{j=0}^n \left(\frac{w_j}{j!} + \frac{w_{n-j}}{(n-j)!} + n \right) \end{aligned}$$

Für die mittleren Weglängen $\bar{w}_n := w_n/n!$ gilt also

$$\bar{w}_{n+1} = n + \frac{2}{n+1} \sum_{j=0}^n \bar{w}_j$$

und dies ist gerade die *quicksort*-Rekursion!

3 Quicksort und binäre Suchbäume

- Ist $M \subset \mathbf{Z}$ eine endliche Menge und L eine Umordnung (permutation) von M , so gehört zu jeder Exekution $\text{quicksort}(L)$ ein binärer Suchbaum $\tau \in \mathcal{BST}_M$, der “splitter-Baum”. Induktiv beschrieben: zu der leeren Liste gehört der leere Baum, zu einer Liste $L = [a]$ der Länge 1 gehört der Baum mit einem Knoten, der mit a beschriftet ist. Für eine Liste L der Länge ≥ 2 wird $\text{split}(L)$ aufgerufen — dies liefert ein splitting $[L', s, L'']$, wobei das Element s der splitter ist. Der splitter-Baum dieser Exekution besteht nun aus der Wurzel s und den beiden Teilbäumen τ' und τ'' , die zu den Exekutionen von $\text{quicksort}(L')$ bzw. $\text{quicksort}(L'')$ gehören. Wegen der randomisierten Arbeitsweise von split werden verschiedenen Aufrufe $\text{quicksort}(L)$ i.a. verschiedene splitter-Bäume erzeugen.
- Die Anzahl der Vergleichsoperationen bei einem splitting $L \rightarrow [L', s, L'']$ ist

$$\#L - 1 = \#L' + \#L'' = \#\tau' + \#\tau''$$

Per Induktion erkennt man, daß die Anzahl der Vergleichsoperationen bei einem Aufruf $\text{quicksort}(L)$ gerade die Weglänge des zugehörigen splitter-Baumes ist:

$$\#\{(x, y) ; x \prec_{\tau} y\} = w(\tau)$$

- Betrachten wir nun folgende Variante von *quicksort*, genannt *deterministisches quicksort mit Prioritätsfunktion*:
Wieder sei $M \subset \mathbf{Z}$ (endliche) Menge und L irgendeine Permutation von M . Ferner sei $\pi : M \rightarrow \mathbf{Z}$ eine Prioritätsfunktion.

$\text{dqs}(L, \pi)$: der Ablauf ist wie bei $\text{quicksort}(L)$, aber mit dem Unterschied, daß bei jedem Aufruf $\text{split}(\dots)$ das Element höchster Priorität (= kleinster Wert von π) der zu splittenden Teilliste zum splitter gemacht wird.

Der zur Exekution von $\text{dqs}(L, \pi)$ gehörende splitter-Baum ist $t_{M, \pi}$! (Siehe ersten Abschnitt für die Definition von $t_{M, \pi}$). Daher gilt:

Die Anzahl der Vergleichsoperationen beim Aufruf $\text{dqs}(L, \pi)$ ist gleich der Weglänge von $t_{M, \pi}$ (und das ist auch die Weglänge von τ_{π}) — diese Anzahl hängt also nur von der Prioritätsfunktion π ab, nicht von L .

- Betrachten wir schließlich folgende Version von *quicksort*, genannt *quicksort mit randomisierter Priorität*:

Wieder sei $M \subset \mathbf{Z}$ (endliche) Menge und L irgendeine Permutation von M .

$\text{rqs}(L)$: falls $\#M = n$ ist, wird zunächst irgendeine Prioritätsfunktion $\pi : M \rightarrow \{1, \dots, n\}$ gewählt, mit gleicher Wahrscheinlichkeit für alle $n!$ solchen Funktionen.. Dann wird $\text{dqs}(L, \pi)$ exekutiert.

Die Anzahl der Vergleichsoperationen bei Exekution von $\text{rqs}(L)$ ist also gleich der Weglänge $w(t_\pi)$, und wenn π über alle Prioritätsfunktionen variiert, läuft t_π über $\mathcal{BPT}_{\{1, \dots, n\}}$. Daher gilt:

Für jede (!) feste Permutation L von M ist die mittlere Anzahl von Vergleichsoperationen bei Aufruf von $\text{rqs}(L)$ gleich der mittleren Weglänge $\mathbf{E}[w(t), t \in \mathcal{BPT}_{\{1, \dots, \#M\}}]$.

Diese Aussage überträgt sich dann natürlich auch auf die Situation, wo über alle möglichen Permutationen L von M gemittelt wird.

- Abschließend ist zu bemerken, daß *randomized quicksort* rqs das bekannte *quicksort* simuliert, indem die Zufallsauswahl der splitter, die ja üblicherweise bei Aufruf von `split` vorgenommen wird, in die vorab bestimmte Prioritätsfunktion π verlegt wird. Gleichverteilung bei Auswahl von π überträgt sich auf die Gleichverteilung bei Auswahl der splitter. Deshalb gilt:

Für jede (!) feste Permutation L von M ist die mittlere Anzahl von Vergleichsoperationen bei Aufruf von $\text{quicksort}(L)$ gleich der mittleren Weglänge $\mathbf{E}[w(t), t \in \mathcal{BPT}_{\{1, \dots, \#M\}}]$

Diese Aussage überträgt sich dann natürlich auch auf die Situation, wo über alle möglichen Permutationen L von M gemittelt wird.