AD-Übung zum 19. November

Arne Beer, MN 6489196
Merve Yilmaz, MN 6414978
Sascha Schulz, MN 6434677

20. November 2013

1. (a) $h(k) = k \bmod 11 \Rightarrow 11\mathbb{N} + 10$ auf Index10,
da $11 \cdot n \bmod 11 = 0$, $n \in \mathbb{N}$ und nach einer Offset verschiebung durch die Addition der 10. Index erreicht ist.

(b) $h(k) = 2k \bmod 11 \Rightarrow 11\mathbb{N} + 5$ auf Index10,
da $2(an + b) \bmod 11 = 10$ mit $n \in \mathbb{N}$
gefordert ist. Nach anwenden des Distributivgesetztes:
$(2an + 2b) \bmod 11 = 10$.
Da 11 eine Primzahl ist, lässt sich diese nich durch die Multiplikation von natürlichen Zahlen erzeugen, folglich sollte
$2an \bmod 11 = 0$
sein, sodass die Offset-Verschiebung wie in Aufgabe 1 durch die Addition vorgenommen wird.

(c) $h(k) = k^2 + 10 \bmod 11 \Rightarrow 11\mathbb{N} + 0$ auf Index10,
da für alle k als Vielfache von 11 gilt:
$k^2 \bmod 11 = 0$.
Wobei durch die anschließende Addition von 10 jeder Vielfache von 11 auf Index10 abgebildet wird. $b = 0$ ist erforderlich, da auch der Index 0 auf Index10 abgebildet wird.

(d) $h(k) = 3^k - 1 \bmod 11 \Rightarrow \emptyset$ auf Index10,
Damit $h(k)$ auf Index10 abbildet, ist erforderlich, dass 3^k einen Vielfachen von 11 bilden. Dies würde bedeuten, dass eine Zahl, welche in der Primzahlzerlegung aus k-Fach 3en besteht zudem durch die Primzal 11 teilbar ist. Da 11 und 3 beides Primzahlen sind ist dies schlicht nicht möglich.

2. Zu Beginn wird $n!$ mit n^n verglichen.

$$\frac{n \cdot n \cdot n \cdot ... \cdot n \cdot n}{n \cdot (n-1) \cdot (n-2) \cdot ... \cdot 2 \cdot 1}$$

Es wird deutlich, dass $n!$ asymptotisch langsamer wächst als n^n. Anschließend vergleichen wir $n!$ mit $\left(\frac{n}{2}\right)^{\frac{n}{2}}$.

$$\frac{n \cdot (n-1) \cdot (n-2) \cdot ... \cdot (n-\frac{n}{2}) \cdot (n-\frac{n}{2}-1) \cdot ... \cdot 2 \cdot 1}{\frac{n}{2} \cdot \frac{n}{2} \cdot \frac{n}{2} \cdot ... \frac{n}{2} \cdot 1 \cdot ... \cdot 1 \cdot 1}$$

Es wird deutlich, dass $n!$ asymptotisch schneller wächst als $\left(\frac{n}{2}\right)^{\frac{n}{2}}$.

Aufgrund dieser Feststellungen wird nun der Logarithmus von $\left(\frac{n}{2}\right)^{\frac{n}{2}}$ und n^n gebildet und mit dem von $n!$ verglichen.

$$\log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right) = \frac{n}{2}\log\left(\frac{n}{2}\right)$$
$$= \frac{1}{2}n\log\left(\frac{1}{2}n\right)$$
$$\log(n^n) = n\log n$$

Damit ist klar, dass die Logarithmen von $\left(\frac{n}{2}\right)^{\frac{n}{2}}$ und n^n beide in $\theta(n\log n)$ sind. Aus unserem obigen Vergleich wissen wir, dass $n!$ schneller als $\left(\frac{n}{2}\right)^{\frac{n}{2}}$ und langsamer als n^n wächst. Daraus ergibt sich:

$$\frac{1}{2}n\log(\frac{1}{2}n) \in \theta(n\log n) \leq \log(n!) \leq n\log n \in \theta(n\log n)$$

Da $\log(n!)$ asymptotisch sowohl schneller als auch langsamer als $n\log n$ wachsen muss, liegt $\log(n!)$ damit folgerichtig in $\theta(n\log n)$.

3. (a) $T(n) = 2\cdot T(\frac{n}{2}) + a\cdot n$
$= \Theta(n\cdot\log n)$
Die Bestimmung des Medians betraegt $O(n)$. Die zu sortierenden Elemente werden in zwei Teile aufgeteilt unter Verwendung des Medians.

(b) Diese Variante wird in der Praxis nicht benutzt, da die Konstante a zum Auffinden des Medians sehr groß ist.

(c) TODO: Kein Loesungsansatz gefunden.

4. (a) Wir stellen uns einen binären Entscheidungsbaum vor. Ist das Ergebnis des Münzwurfs eine 0, wählen wir den linken Ast, alternativ bei 1 den rechten Ast. Dies führen wir fort, bis zur Ebene k, in dieser Befinden sich 2^k Blätter, wobei jedes Blatt einem Element unseres Arrays entspricht.

Der Baum hat somit eine Tiefe von $k = log_2(n)$, mit $n = 2^k$. Die Anzahl der nötigen Münzwürfe liegt folglich bei $O(log(n))$, da mit jedem Wurf eine Ebene tiefer erreicht wird. Da die Anzahl der Ebenen endlich ist (k Stück), terminiert dies stets. Da genau ein Pfad im Baum zu genau einem Element im Array führt, besitzt jedes Element die exakt gleiche Wahrscheinlichkeit gezogen zu werden.

(b) Wir wandeln den zuvor vollen binären Entscheidungsbaum in einen vollständigen Baum um, sodass weiterhin jedes Blatt des Baumes ein Element des Array repräsentiert.

Dies hat zur Folge, das es Pfade geben kann, für die genau ein Münzwurf weniger für die Entscheidung benötigt wird, da sie sich nicht auf der untersten (unvollständigen) Ebene befinden, sondern auf der vorherigen.

Die maximale Tiefe des Baumes und somit die Anzahl der benötigten Münzwürfe bleibt unverändert $O(log(n))$, da maximal $\lceil log_2(n)$ Würfe benötigt werden und das Ceiling in der Landau-Notation nicht berücksichtigt werden muss, da dies die Addition von $x \in [0,1]$ bedeutet, wobei das Ceiling erst für 3+ Elemente notwendig ist (da es sich zuvor eh um einen vollen Baum handelt). In diesen Fällen ist bereits $log_2(n) > 1$.

(c) Um zu gewährleisten, dass für jeden Eintrag exakt die selbe Wahrscheinlichkeit ($\frac{1}{n}$) gegen ist, müssen zur Wahr eines Elementes exakt gleich viele Münzwürfe benötigt werden (Bei der asymptotischen Betrachtung ist es jedoch fraglich, ob nicht bereits die zuvor skizzierte Lösung 'exakte' Wahrscheinlichkeiten liefert im Sinne von +/- 1 Münzwurf auf $O(log(n))$ Würfe).

Um dies zu gewährleisten gehen wir initial von einem vollständigen binären Baum aus, den wir anschließend modifizieren.

Bei einem vollständigen Baum befinden sich in der untersten Ebene $2^{log(n)}$ Blätter. Von links an wird jedem Element ein Blatt zugeordnet, bis alle Elemente injektiv abgebilet sind.

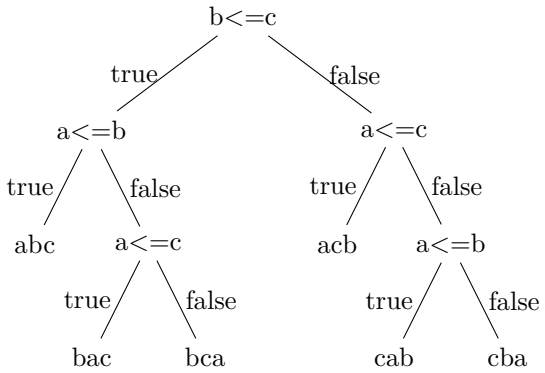
Die nun potentiell übrig gebliebenden Blätter werden rekursiv entfernt, sofern auf zwei Blätter eines Elternknotens nicht abgebildet wurde. Diese 'toten Blätter' verweisen auf den Wurzelknoten - es muss erneut von vorn mit Münzwürfen begonnen werden, um die Gleichverteilung der Wahrscheinlichkeit unter keinen Umständen zu gefährden, potentiell auf Kosten der Laufzeit - bis kein totes Blatt mehr erreicht wurde.

Die maximale Anzahl dieser 'toten Blätter beträgt im Worst Case $log_2(n) - 1$ (es wird genau 1 Blatt des Teilbaumes verwendet, welches vom Wurzelelement nach rechts geht, ab da befindet sich auf jeder Ebene 1 totes Blatt), im Best Case 0 (vollständiger Baum).

Ich erhoffe mir, dass im Erwartungswert auf Grund von Wahrscheinlichkeitsrechnung und Kombinatorik - welcher ich in diesem Augenblick nicht fähig bin - $O(log(n))$ Münzwüfe benötigt werden um ein Element zu bestimmen, jedoch zufällig - durch das Erreichen eines toten Blattes, diese Anzahl schwanken kann.

Zugegeben, eine kreative Lösung. Finde ich aber interessanter als in der Literatur nach fremden Ideen zu suchen um schlicht zu beweisen, dass ich motiviert bin, Zeit aufzuwenden um dann eine Fremdlösung zu präsentieren.

5. (a) BerechnungsbaUm der durchgeführten Vergleichsabläufe für Eingabe [a,b,c]



(b) Anzahl der Blätter bei Eingabe von [a, b, c, d]: 4!

Anzahl der Blätter bei Eingabe von [a, b, c, d, . . . , w, x, y, z]: 24!

6. Zunächst wird mit Hilfe von „Median of Medians" das k't-grösste Element gefunden und alle Elemente, die größer sind als k, aus der zu sortierenden Liste entfernt. Wir setzen vorraus, dass dies eine Laufzeit von $\mathcal{O}(n)$ hat.

Danach wird mit einem deterministischen Quicksort-Verfahren sortiert, bspw. aus a) in $\mathcal{O}(n)$. Zusammen also läge man damit in $\mathcal{O}(n + k \log k)$. Hierfür gilt:

$$\frac{n + k \log k}{n \log k} = \frac{1}{\log k} + \frac{k}{n}$$

Aus $k \ll n$ folgt, dass der letzte Bruch annähernd 0 ist, und $\frac{1}{\log k}$ geht für $k \to \infty$ gegen 0. Somit ist gezeigt, dass $(n + k \log k)$ für $k \ll n$ in $\mathcal{O}(n \log k)$ liegt.