8.1 Mittlere Tiefe der Blätter in einem Entscheidungsbaum (2 Punkte)

Sei m die Anzahl der Blätter in einem binären Baum, l bzw. r die Anzahl der Blätter im linken/rechten Teilbaum. Die mittlere Tiefe $\bar{t}$ der Blätter in einem Entscheidungsbaum wurde in der Vorlesung vom 05.12.2012 wie folgt abgeschätzt:

$$\bar{t}(m) = \frac{l}{m}\bar{t}(l) + \frac{r}{m}\bar{t}(r) + 1 \tag{1}$$

$$\geq \bar{t}(\frac{m}{2}) + 1 \tag{2}$$

d.h. $\bar{t}(m)$ wird minimal für $l = r = \frac{m}{2}$. Zeigen Sie dies, indem Sie

a) (1P) das Minimum der Funktion f_m suchen:

$$f_m(x) = x \cdot \log_2 x + (m - x) \cdot \log_2(m - x)$$

b) (1P) durch Induktion zeigen, dass $\bar{t}(m) \geq \log_2 m$.

8.2 Hashing (7 Punkte)

a) (2P) Vergleichen Sie die Eigenschaften einer Hashtabelle mit denen eines Suchbaums. Gehen Sie dabei auf die Laufzeit der Operationen search/insert/delete ein und ob man in einer der Datenstrukturen Elemente speichern kann, die man in der anderen Struktur nicht speichern kann.
b) (1P) Weshalb sollte eine Hashfunktion möglichst schnell sein?
c) (1P) Recherchieren Sie, in welchem Zusammenhang langsame Hashfunktionen von Vorteil sind.
d) (1P) Sei h folgende Hashfunktion

$$h(x) = (a \cdot x + b) \mod p \mod m$$

mit $x, a, b, p, m \in \mathbb{N}_0, 1 \leq a < p, 0 \leq b < p, m < p, p$ prim. Geben Sie eine Funktion g an, die aus einem gegebenen Hash $h(x)$ und Kenntnis von a, b, p, m ein mit x kollidierendes Element findet, d.h. es soll $h(x) = h(g(h(x)))$ gelten.
e) (2P) Sei p prim. Überlegen Sie sich eine Hashfunktion h mit konstanter Ausführungszeit für einen Kalendereintrag K mit den Attributen Datum, Uhrzeit und Titel. Es soll dabei $0 \leq h(K) < p$, gelten. Geben Sie erst eine Beschreibung, inkl. eventuellen Anforderungen, die Sie an die Attribute stellen, und anschließend Pseudocode an, der Ihre Hashfunktion umsetzt.[1]

[1] Es wird erwartet, dass Ihre Hashfunktion sinnvoll ist und insbesondere nicht einfach eine Konstante zurück liefert. Einen Beweis, dass Ihre Hashfunktion die Elemente gleichverteilt auf die Tabelle verteilt, müssen Sie nicht liefern.

8.3 Sternenkarte (11 Punkte)

Gegeben sei eine Liste von 3D-Koordinaten $x_{i=0,\ldots,n-1}$ von $n = 1000$ Sternen in einem Universum mit

$$x_i = (x_{i0}, x_{i1}, x_{i2}) \in \{0, 1, \ldots, n-1\}^3 \subset \mathbb{Z}^3$$

Ziel dieser Aufgabe ist es, mit Hilfe einer Hashtabelle relativ einfach die Sterne zu finden, die sich in der Nähe eines gegebenen Sterns befinden.

a) (1P) Man möchte das Universum nun gleichmäßig in würfelförmige Zellen einteilen und eine Hashfunktion finden, die allen Sternen einer Zelle den gleichen Hashwert zuweist. Wie groß muss ein Würfel sein, wenn der Belegungsfaktor $\alpha = 1$ sein soll?

b) (1P) Stellen Sie eine Hashfunktion $h(x_i)$ auf, die allen x_i aus einer Zelle den gleichen Hashwert zuweist. Verwenden Sie als Primzahl $p = 2053$, falls Sie Hashwerte $\geq p$ erzeugen.

c) (1P) Laden Sie sich die Datei `StarMap.java` herunter und implementieren Sie h in der Methode `hashCode`.

d) (4P) Fügen Sie der Klasse `StarMap` ein Feld zur Speicherung der Hashtabelle hinzu und implentieren sie die Methode `calcHashTable`.

e) (4P) Die Methode `getNearestStars` soll mit Hilfe Ihrer Hashtabelle alle Sterne zurück liefern, deren Abstand zu einem gegebenen Stern kleiner ist als ein gegebener Radius. Implementieren Sie die Methode, durchsuchen Sie hierbei aber nur Zellen, die potentiell Sterne innerhalb des Radius enthalten können. Verwenden Sie die Maximumsnorm zur Abstandsbestimmung.