1. (10 Punkte) Wir nennen eine Menge $\mathcal{H}$ von Funktionen $h : U \to \{0, 1, \ldots, t-1\}$ *pseudo-universell,* wenn für jedes $x \in U$ und jedes $i \in \{0, 1, \ldots, t-1\}$ gilt

$$\frac{|\{h \in \mathcal{H} \mid h(x) = i\}|}{|\mathcal{H}|} \leq \frac{1}{t}.$$

   Es sei nun $S \subset U$ mit $|S| = n$.

   (a) Zeigen Sie, dass bei zufälliger Wahl einer Hashfunktion aus einer pseudo-universellen Familie $\mathcal{H}$ für jedes $i \in \{0, 1, \ldots, t-1\}$ die erwartete Anzahl von Elementen von S, die durch h auf i abgebildet werden, höchstens n/t ist. Die erwartete Größe jedes "Hash-buckets" ist also n/t.

   (b) Zeigen Sie, dass trotz der kleinen erwarteten Hash-bucket-Größen pseudo-universelle Hasfunktionen keine gute Idee sind.

   Geben Sie eine Familie $\mathcal{H}$ an, die zwar pseudo-universell ist, für die aber die Operationen SEARCH und DELETE wesentlich mehr als $\Omega(n/t)$ Operationen brauchen.

2. (10 Punkte) In der Vorlesung haben wir eine Klasse von universellen Hashfunktionen beschrieben, die folgendermaßen gestaltet war: t ist eine Primzahl; das Universum U hat die Form $\{0, \ldots, t-1\}^r$. Für jedes $\overline{a} = \langle a_0, \ldots, a_{r-1} \rangle \in \{0, \ldots, t-1\}^r$ definieren wir eine Hashfunktion $h_{\overline{a}} : U \to \{0, \ldots, t-1\}$ durch

$$h_{\overline{a}}(x) = \left(\sum_{0 \leq i < r} a_i x_i \right) \bmod t.$$

   Dies ergibt eine Menge $\mathcal{H}$ von t^r Hashfunktionen. Um eine davon zufällig auszuwählen muss man $r\lceil \log_2 t \rceil$ zufällige Bits zur Verfügung haben. Wir wollen nun die notwendige Anzahl der zufälligen Bits reduzieren.

   Für jedes $a \in \{0, \ldots, t-1\}$ betrachte die Funktion

$$g_a(x) = \left(\sum_{0 \leq i < r} a^i x_i \right) \bmod t.$$

   Die Menge $\mathcal{G} = \{g_a \mid 0 \leq a < t\}$ umfasst nur t Funktionen. Eine zufällige Auswahl braucht also nur noch $\lceil \log_2 t \rceil$ zufällige Bits.

   Aber, ist $\mathcal{G}$ universell? Wenn nein, ist G eine c–universelle Klasse von Hashfunktionen? Für welches kleinste c ist dies der Fall?

3. (15 Punkte) Wenn man universelles Hashing verwendet, dann ist die erwartete Suchzeit $O(1 + n/t)$, wobei n die Größe der derzeit gespeicherten Menge darstellt und t die Hashtafelgröße. Wenn n sehr viel größer als t wird (oder sehr viel kleiner), dann kann man n/t nicht mehr als konstant auffassen. Deswegen trachtet man danach, die Hashtafelgröße t immer ans n anzupassen. Das kann man analog zu der in der Vorlesung vorgestellten Methode zum Anpassen der Größen von dynamischen Feldern machen: wenn $n = 2t$ (oder wenn $n = t/2$) dann passiert ein "Rehash", in dem eine neue Hashtafel der Größe $t' = 2t$ (bzw. $t' = t/2$) verwendet wird.

   Dieser Ansatz hat den möglichen Nachteil, dass man genau genommen nicht mehr sagen kann, dass, sagen wir, jede Einfügeoperation konstante erwartete Laufzeit hat. Denn jede Einfügeoperation, bei der die Mengengröße n eine Zweierpotenz erreicht, braucht vorhersagbar $\Theta(n)$ Zeit (wir nehmen an, dass t als Zweierpotenz initialisert wird).

   Entwicklen Sie eine Methode, die garantiert, dass jede Operation konstante erwartete Laufzeit hat und dass mit hoher Wahrscheinlichkeit immer nur $\Theta(n_c)$ Speicherplatz verwendet wird, wobei n_c die Kardinalität der gerade gespeicherten Menge darstellt.

SCHÖNE FEIERTAGE!