Image Analysis Lab
Jun.-Prof. Dr.
Olaf Ronneberger
Claudius Korzen

Algorithmen und Datenstrukturen (ESE + IEMS) WS 2014 / 2015

http://lmb.informatik.uni-freiburg.de/lectures/

UNI FREIBURG

Übungsblatt 5

Abgabe für ESE: bis Donnerstag, den 27. November um 10:00 Uhr

Abgabe für IEMS: bis Donnerstag, den 11. Dezember um 10:00 Uhr

Sei $U = \{0, 1, 2, \ldots, N-1\}$ ein Universum von N möglichen Schlüsseln, p eine Primzahl $\geq |U|$, und m die Größe der Hashtabelle. Wir haben in der Vorlesung gesehen, dass die Klasse der Hashfunktionen $\mathcal{H} = \Big\{h_{a,b} : a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\}\Big\}$ mit

$$h_{a,b}(x) = ((ax + b) \bmod p) \bmod m \tag{1}$$

c-universell ist. In dieser Aufgabe sollen Sie mit einem Programm empirisch nachprüfen, dass diese Eigenschaft gilt. Die Aufgabe erfordert nicht viel Code, sondern vor allem Verständnis davon, was Universalität bedeutet.

Aufgabe 1 (5 Punkte)

Schreiben Sie eine Methode *meanBucketSize*, die für eine gegebene Schlüsselmenge S und eine gegebenene Hashfunktion h, die *mittlere Bucketgröße* von h angewandt auf S ermittelt. Das ist gerade die mittlere Anzahl von Elementen, die Sie im Bucket untersuchen müssen, um einen gesuchten Schlüssel zu finden. Achtung: Sie müssen dabei nur über die Buckets mitteln, die mindestens einen Schlüssel enthalten. Warum?

Tipps:

- Implementieren Sie die Hashfunktion als Klasse mit den member-Variablen *parameter_a_, parameter_b_, prime_number_, universe_size_, hash_table_size_*, einer Methode *int apply(int x)* und einer Methode *void setRandomParameters()*
- *meanBucketSize* bekommt ein int-Array und eine Hashfunktion übergeben, also in C++ z.B.: *double meanBucketSize(const std::vector<int>& key_set, const HashFunction& hash_func)*

Aufgabe 2 (5 Punkte)

Schreiben Sie eine Methode *estimateCForSingleSet*, die für eine gegebene Schlüsselmenge S für 1000 zufällige Hashfunktionen die mittlere Bucketgröße berechnet, daraus den bestmöglichen Wert für c berechnet und zurück gibt. Benutzen Sie dazu die Formel aus der Vorlesung $E(|S_i|) \leq 1 + c \cdot |S|/m$. In C++ könnte die Methode folgendermaßen aussehen:
double estimateCForSingleSet(const std::vector<int>& key_set, HashFunction hash_func)

Aufgabe 3 (5 Punkte)

Schreiben Sie eine Methode *estimateCForMultipleSets*, die eine gegebene Anzahl n von Schlüsselmengen einer gegebenen Größe k zufällig erzeugt (keine Duplikate innerhalb einer Schlüsselmenge!), und für jede dieser n Schlüsselmengen jeweils das bestmögliche c mit der Funktion aus Aufgabe 2 ermittelt. Berechnen Sie davon den mittleren, den minimalen und den maximalen Wert. In C++ könnte die Funktion so aussehen:
void estimateCForMultipleSets(int num_sets, int set_size, HashFunction hash_func,
double mean, double* min, double* max)*

Aufgabe 4 (5 Punkte)

Schreiben Sie ein Programm, das für $|U| = 100$, $m = 10$, $p = 101$, $n = 1000$ zufällig gewählte Schlüsselmengen der Größe $k = 20$ die in Aufgabe 3 genannten Werte berechnet. Simulieren sie dann eine nicht-universelle Hashfunktion, indem Sie $p = 10$ setzen und berechnen Sie auch dafür die in Aufgabe 3 genannten Werte. Tragen Sie diese Werte in die auf der Homepage verlinkte Tabelle ein (folgen Sie beim Eintragen den Anweisungen dort). Erklären Sie in Ihrer *erfahrungen.txt* kurz, ob Ihre Ergebnisse Sinn machen und warum.

Committen Sie, wie gehabt, Ihren Code in das SVN, in einen neuen Unterordner *uebungsblatt_05*, sowie, ebendort, Ihr Feedback in einer Textdatei *erfahrungen.txt*. Insbesondere: Wie lange haben Sie ungefähr gebraucht? An welchen Stellen gab es Probleme und wieviel Zeit hat Sie das gekostet?