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

Algorithmen und Datenstrukturen (ESE + IEMS) WS 2013 / 2014

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

UNI FREIBURG

Übungsblatt 5

Abgabe für ESE: bis Dienstag, den 3. Dezember um 10:00 Uhr

Abgabe für IEMS: bis Donnerstag, den 12. 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 $\mathcal{H} = \{h_{a,b} : a, b \in U\}$, wobei $h_{a,b}(x) = ((ax+b) \bmod p) \bmod m$, 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 *maxBucketSize*, die für eine gegebene Schlüsselmenge S und eine gegebenene Hashfunktion h, die *maximale Bucketgröße* von h angewandt auf S berechnet, also $\max_{i=0,\ldots,m-1} |S_i|$, wobei $S_i = \{x \in S : h(x) = i\}$.

Tipps:

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

Aufgabe 2 (5 Punkte)

Schreiben Sie eine Methode *ratioGoodHashFunctions*, die für eine gegebene Schlüsselmenge S und eine Konstante c den Anteil (= eine Zahl zwischen 0 und 1) der *c-guten* Hashfunktionen in $\mathcal{H}$ berechnet. Eine Hashfunktion heißt dabei c-gut für S, wenn die maximale Bucketgröße $\leq c \cdot |S|/m$ ist. In C++ könnte die Methode folgendermaßen aussehen:
double ratioGoodHashFunctions(const std::vector<int>& key_set, double bucket_size_factor)

Aufgabe 3 (5 Punkte)

Schreiben Sie eine Methode *statisticsGoodHashFunctions*, die eine gegebene Anzahl n von Schlüsselmengen einer gegebenen Größe k zufällig erzeugt (keine Duplikate innerhalb einer Schlüsselmenge!), und für jede dieser n Schlüsselmengen den Anteil aus Aufgabe 2 berechnet. Berechnen Sie den Durchschnitt dieser n Anteile, sowie die folgenden Perzentile: 50%, 90%, 99%, 99.9%. Dabei sei das p%

Perzentil einer Folge von n nicht-negativen Zahlen definiert als der maximale Wert x, so dass p% aller dieser n Werte $\geq x$ sind. In C++ könnte die Funktion so aussehen:
void statisticsGoodHashFunctions(int num_sets, int set_size, double bucket_size_factor, double mean, double* perc50, double* perc90, double* perc99, double* perc99_9)*

Aufgabe 4 (5 Punkte)

Schreiben Sie ein Programm, das für $|U| = 100$, $m = 10$, $p = 101$, $n = 1000$ zufällig gewählte Schlüsselmengen der Größe $k = 20$ und $c = 2$, die in Aufgabe 3 genannten Perzentile sowie den Durchschnitt berechnet. 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?