Projekt 4: Hashtechniken

Aufgabe 1 (von Mises' Geburtstagsparadoxon)

Kollisionen kommen häufiger vor, als man intuitiv vermutet. Bekannt ist dieses Fakt als das *von Mises' Geburtstagsparadoxon*: Befinden in einem Raum mindestens 23 Personen, dann ist die Wahrscheinlichkeit, dass zwei oder mehr Personen am selben Tag Geburtstag haben, größer als 50 %.

Lit.: R. von Mises. Über Aufteilungs- und Besetzungs-Wahrscheinlichkeiten. Revue de la Faculté des Sciences de l'Université d'Istanbul, N.S. 4. 1938-39, S. 145-63

Übertragen auf das Kollisionsproblem bedeutet dies:

> Bei einer Hashtabelle T mit 365 Slots und 23 zu speichernden Schlüsseln ist die Wahrscheinlichkeit einer Kollision größer als 50% .

Berechnen Sie die Wahrscheinlichkeit P(n) für ein oder mehrere Kollisionen, wenn man n Schlüssel in eine Hashtabelle T der Größe 365 einfügt.

Programmieren Sie die Funktion P(n), und geben Sie in einem Koordinatensystem die Ergebnisse (Wahrscheinlichkeiten) aus für n = 0, 5, 10, ..., 80 Schlüssel (x-Achse). Lesen Sie den Wert für n=23 aus der Kurve ab. Lösungshinweis:

> Sei Q(n) die Wahrscheinlichkeit, dass n Schlüssel zufällig in einer Hashtabelle T der Größe m=365 gespeichert werden und keine Kollision auftritt. Sei P(n) die Wahrscheinlichkeit für mindestens eine Kollision. Dann gilt:
>
> $$P(n) = 1-Q(n)$$
>
> da 1 minus die Wahrscheinlichkeit für keine Kollision gleich der Wahrscheinlichkeit für mindestens eine Kollision ist.
>
> Es gilt: Q(1) = 1. Überlegen Sie sich Q(2), Q(3), ... Sie erhalten eine Differenzgleichung, die Sie durch Entfaltung lösen können.

Aufgabe 2 (Erwartungswerte)

Vergleichen Sie für die Hash-Techniken Linear Probing, Quadratic Probing und uniformes Hashing die Erwartungswerte für die Anzahl der Schlüsselvergleiche bei **erfolgreicher** und **nicht-erfolgreicher Suche**. Berechnen Sie dazu die Erwartungswerte für die Füllungsgrade a = 50 %, a = 90 %, a = 95 % und a = 100 %.

Aufgabe 3 (Universelles Hashing)

Implementieren Sie die Methode „Universelles Hashing“.

a) Entwerfen und implementieren Sie ein Software-Modul, mit dem Zufallsschlüssel erzeugt werden können. Die Verteilung der Schlüssel soll beeinflussbar sein, zum Beispiel 10% der Schlüssel liegen im Intervall [2001 .. 4000], 40% liegen in [4001 .. 4500] usw. Stellen Sie die Verteilung der Schlüssel grafisch dar. Außerdem sollen Schlüssel erzeugt werden können, die durch eine vorgegebene Zahl teilbar sind.

b) Entwickeln Sie eine **Experimentierumgebung**, in der die „Universelles Hashing“-Parameter p, m und n modifiziert werden können:

p : Dekomposition eines Schlüssels in p+1 Bytes
m : Größe der Hashtabelle T = {0, ..., m-1}
n : Anzahl der Schlüssel im Universum U. n = |U|

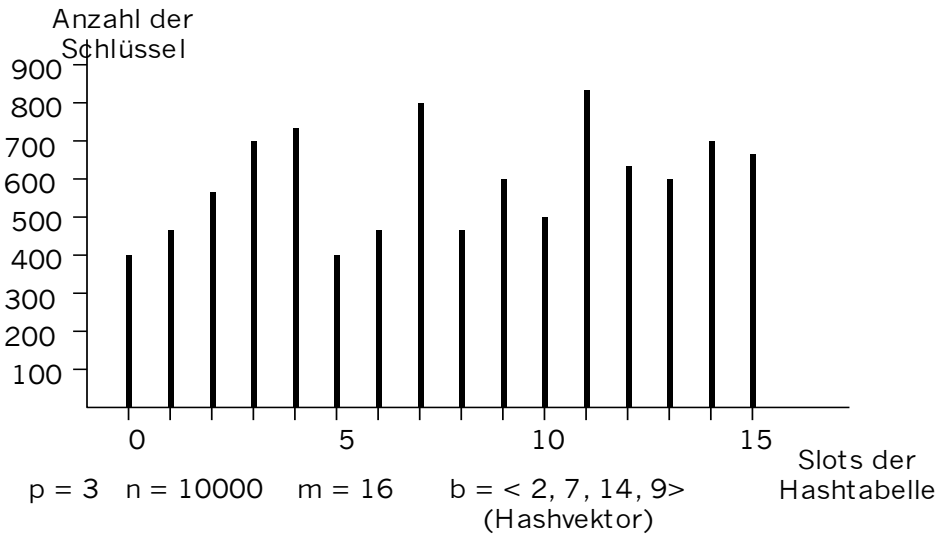
Es soll möglich sein, den Hashvektor $b = \langle b_0, ..., b_p \rangle$ interaktiv einzugeben und auch per Zufallsgenerator zu erzeugen.

c) Wenden Sie die Methode „Universelles Hashing“ auf Zufallsschlüssel an. Führen Sie **Experimente** durch, indem Sie die obigen Parameter variieren. Protokollieren Sie die Experimente.

Hinweise zu den Experimenten:

- **p** vergrößern und verkleinern. Worauf hat dies Einfluss?
- **m** vergrößern und verkleinern.
- Untersuchen Sie die Fälle **n ≤ m** und **n** >> **m**.
- **m** als Primzahl und als 2-er Potenz
  - **Variation der Schlüssel**:
    - durch 2 teilbar
    - durch 3 teilbar
    - durch 5 teilbar
    - ungerade Schlüssel
    - gleich verteilt
    - mit bestimmten Häufigkeiten in verschiedenen Intervallen (siehe a))

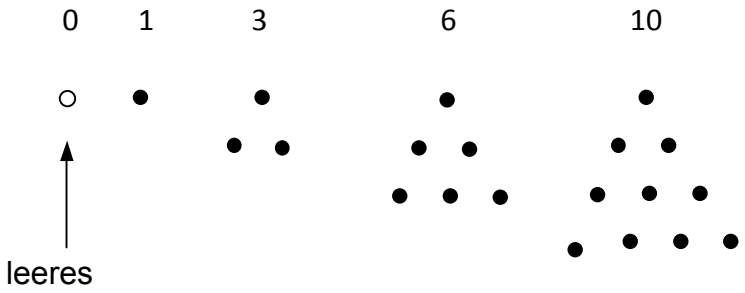
Stellen Sie die **Ergebnisse** der Experimente grafisch dar. Zum Beispiel lässt sich die Verteilung der Schlüssel auf die Slots **visualisieren**:



- Dokumentieren Sie Ihre **Vorgehensweise** und die **Testergebnisse**.
- Erklären/interpretieren Sie die Testergebnisse.
- Stellen Sie die Testergebnisse grafisch dar.

Aufgabe 4 (Triangle-Hashing)

Eine Variante des Quadratic Probing ist das so genannte Triangle-Hashing (Dreieckszahlen-Hashing). Dabei wird die Sondierungsfolge durch die Zahlenfolge 0, 1, 3, 6, 10, 15, ... gebildet, wobei die Sondierungsposition 0 die erste Hashadresse h(k) der Sondierungsfolge ist. Diese Zahlen werden Dreieckszahlen genannt, weil sie die Anzahl der Punkte in einem Dreieck wiedergeben:



Alle Schlüssel mit derselben Start-Hashadresse unterliegen dem Secondary Clustering. In der Vorlesung haben wir folgende Performance-Ergebnisse für Secondary Clustering wie folgt angegeben:

Erfolgreiche Suche: $$C_n \approx 1 + \ln(\frac{1}{1-a}) - \frac{a}{2}$$

Erfolglose Suche: $$C_{n'} \approx \frac{1}{1\text{-}a} - a + \ln(\frac{1}{1-a})$$

Implementieren Sie „Triangle-Hashing“ und messen Sie die Performance für Hashtabellen der Größe m, wobei m eine 2er-Potenz ist, z.B. m = 1024. Vergleichen Sie Ihre gemessene Performance mit den theoretischen Werten für C_n und C_n'. Führen Sie dazu eine geeignete Anzahl von Experimenten durch.

Programmieren Sie eine grafische Darstellung, so dass die beim Einfügen entstehenden Cluster am Bildschirm verfolgt werden können, ähnlich wie im folgenden Bild für Linear Probing gezeigt.