Prof. Dr.-Ing Georg J. Schneider

Datenstrukturen und Algorithmen

Übung 7 Hashing

Aufgabe 1

Erstellen einer Hashtabelle

Aufgabenstellung

Fügen Sie die Schlüsselfolge 9, 23, 10, 19, 17, 16 gemäß der Hashfunktion
h(k) = k modulo m
in eine geschlossene Hashtabelle mit m = 7 und b = 1 ein. Stellen Sie das Ergebnis grafisch dar. Verwenden Sie bei Kollisionen

a) eine lineare Kollisionsstrategie mit
$h_i(k) = (h(k) + i) \bmod m$

b) eine quadratische Kollisionsstrategie mit
$h_i(k) = (h(k) + i^2) \bmod m$

Vergleichen Sie die beiden Ergebnisse.

Aufgabe 2

Hashing mit Löschen von Werten

Bei einer Streuspeicherung für integer-Zahlen k sei die folgende Hashfunktion gegeben:
$h(k) = h_0(k) = k \bmod 9$.
Zur Kollisionsbehandlung werden die Funktionen
$h_i(k) = (h(k) + i^2) \bmod 9$
verwendet (quadratisches Sondieren).
Protokollieren Sie die besuchten Zellen einer Hashtabelle (Feld mit Feldindex von 0 bis 8), falls folgende Einfüge-/Suche- und Löschoperationen nacheinander auf die anfangs leere Hashtabelle angewendet werden:
insert(3), insert(21), delete(3), insert(28), member(10), delete(3)

Aufgabe 3

Implementierung einer Hashtabelle

Schreiben Sie ein Python Programm, das die Operationen insert() und delete() auf einer Hashtabelle implementiert und dabei den Besuch einer Zelle mitprotokolliert.

Kennzeichnen Sie ein gelöschtes Feld entsprechend. Verwenden Sie zur Kollisionsbehandlung **quadratisches Sondieren mit alternierendem Vorzeichen**, und zwar:

$h_{2i-1}(x) = (h(x) + i^2) \bmod m$ für i ungerade, d.h. i = 1, 3, 5, …

$h_{2i}(x) = (h(x) - i^2) \bmod m$ für i gerade, d.h. i = 2, 4, 6, …

Aufgabe 4

Perfektes Hashing

Ist eine Menge von Schlüsseln im Voraus bekannt (und damit konstant), so kann man versuchen, die Hashfunktion h so zu wählen, dass sie injektiv ist, d.h. dass es keine Kollisionen gibt (perfektes Hashing genannt).
Entwickeln Sie eine perfekte Hashfunktion (A=1, B=2, .., Z=26), die möglichst einfach sein soll und die nachfolgenden 10 Flughafen-Codes in eine möglichst kleine Tabelle abbildet:

- München (MUC)
- Stuttgart (STR)
- Palma (PMI)
- Luxemburg (LUX)
- Hamburg (HAM)
- Friedrichshafen (FDH)
- Valencia (VLC)
- Köln (CGN)
- Malaga (AGP)
- Faro (FAO)

Schreiben Sie dazu ein Programm mit einer Hashfunktion, die den Wert der jeweiligen Buchstaben mit unterschiedlichen Faktoren multipliziert und kollisionsfrei auf die 10 Behälter verteilt.