Algorithmen und Datenstrukturen
WS 2025
von Luxburg/Bordt

Präsenzübung 4

Besprechung: 3.11.25 – 7.11.25

Aufgabe 1: Verschiedene Hashfunktionen (mündlich, keine Punkte)

Gegeben seien folgende Hashfunktionen für eine Hashtabelle der Größe $s = 13$:

- $h_1(k) = k \mod 13$
- $h_2(k) = \lfloor k/13 \rfloor \mod 13$
- $h_3(k) = (3k + 7) \mod 13$

a) Fügen Sie die Schlüssel $26, 39, 52, 65, 78$ mit jeder der drei Hashfunktionen ein (verwenden Sie verkettete Listen zur Kollisionsbehandlung). Zeichnen Sie die resultierende Hashtabelle für jede Hashfunktion. Welche Funktion verteilt die Schlüssel am besten?

b) Identifizieren Sie für jede Hashfunktion eine problematische Eingabesequenz von mindestens 4 Schlüsseln, die alle auf dieselbe Position gehasht werden (maximale Kollisionen).

c) Diskutieren Sie: Was macht eine "gute" Hashfunktion aus? Nennen Sie mindestens drei wichtige Eigenschaften und begründen Sie, warum diese wichtig sind.

Aufgabe 2: Hashing (mündlich, keine Punkte)

Wir haben in der Vorlesung Hashing mit verketteten Listen als auch Hashing mit offener Adressierung kennengelernt. Hashing mit offener Adressierung haben wir dabei nur im Spezialfall von "linearem Sondieren" betrachtet, wo ein Element in die nächste freie Position, die auf die von der Hashfunktion eigentlich vorgesehene Position folgt, geschrieben wird.

Allgemeiner kann Hashing mit offener Adressierung wie folgt behandelt werden: Jede Position in der Hashtabelle kann höchstens ein Element speichern. Kollisionen werden wie folgt behandelt: Zu Beginn jeder INSERT-Operation eines neuen Elements mit Key k wird ein Kollisionszähler $i = 0$ gesetzt, welcher von der Hashfunktion $h(k, i)$ berücksichtigt wird: Ist Position $h(k, i)$ bereits belegt, dann wird i inkrementiert und die nächste Position $h(k, i)$ getestet, solange bis ein freier Platz gefunden wurde.

Beantworten Sie die folgenden beiden Fragen jeweils für den Fall von verketteten Listen als auch für den Fall von offener Adressierung.

a) Wieviele Elemente können maximal in einem Hashtable der Größe s gespeichert werden?

b) Sei m die Anzahl einzufügender Elemente. Was sind Vor-/Nachteile von $m \gg s$ und $m \ll s$?

Sei H nun eine leere Hashtabelle der Größe $s = 7$ mit den Positionen $0, \ldots, 6$. Fügen Sie die folgenden Elemente der Reihe nach in H ein: $9, 12, 2, 31, 10, 4$.

(c) ... unter Kollisionsbehandlung mittels verketteter Listen und $h(k) = k \mod 7$

(d) ... mittels offener Adressierung und linearem Sondieren: $h(k, i) = (k + i) \mod 7$

(e) ... mittels offener Adressierung und "doppeltem Hashing": $h(k, i) = (h_1(k) + i \cdot h_2(k)) \mod 7$ für $h_1(k) = k \mod 7$ und $h_2(k) = 1 + (k \mod 6)$

Aufgabe 3: Anwendungen von Hashtabellen (mündlich, keine Punkte)

In der Vorlesung haben wir Hashtabellen als abstrakte Datenstruktur kennengelernt. Auch in der Praxis werden Hashtabellen sehr viel benutzt. Oft heißen die Datenstrukturen in den verschiedenen Programmiersprachen aber nicht "Hashtabelle", sondern sie haben einen anderen Namen. Diskutieren Sie mit Ihrem Partnern und im Tutorium:

1. Kennen Sie Beispiele von Hashtabellen in verschiedenen Programmiersprachen, z.B. in Python, Java oder C++?

2. Wenn Sie eine Datenstruktur verwenden oder implementieren müssen, woran können Sie erkennen ob eine Hashtabelle hilfreich sein könnte?