Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger

17. November 2016

Algorithmen & Datenstrukturen — Blatt 9 — HS 16

Abgabe: Am Donnerstag, den 24.11.2016, vor Beginn der Vorlesung um 10 Uhr im Eingangsbereich vor ML D28. Bitte heften Sie Ihre Blätter zusammen und benutzen Sie dieses Blatt als Deckblatt. Füllen Sie auch die ersten zwei der untenstehenden Felder aus.

Übungsstunde (Raum & Zeit): _____

Abgegeben von: _____

Korrigiert von: _____

erreichte Punkte: ____

Aufgabe 9.1 *Offene Hashverfahren.*

Wir betrachten offenes Hashing mit einer Hashtabelle der Grösse p für eine Primzahl p.

a) Entscheiden Sie, welche der folgenden Funktionen als Hashfunktionen brauchbar sind und welche nicht, und begründen Sie Ihre Entscheidung.

- $h(k) =$ Quersumme von k
- $h(k) = k(1 + p^3) \bmod p$
- $h(k) = \lfloor p(rk - \lfloor rk \rfloor) \rfloor$, $r \in \mathbb{R}^+ \backslash \mathbb{Q}$

b) Fügen Sie die Schlüssel 17, 6, 5, 8, 11, 28, 14, 15 in dieser Reihenfolge in eine Hashtabelle der Grösse 11 ein. Benutzen Sie offenes Hashing mit der Hashfunktion $h(k) = k \bmod 11$ und führen Sie zur Kollisionsauflösung jeweils

(i) lineares Sondieren

(ii) quadratisches Sondieren

(iii) Double Hashing mit $h'(k) = 1 + (k \bmod 9)$

aus. Geben Sie auch jeweils die Anzahl der Kollisionen an. Welche Variante ist für die obige Situation die beste?

c) Welches Problem tritt auf, wenn der Schlüssel 17 aus den Hashtabellen aus b) entfernt werden soll, und wie kann es gelöst werden? Welche Probleme ergeben sich, wenn sehr viele Schlüssel aus einer Hashtabelle gelöscht werden?

Bitte wenden.

d) Als Sondierungsmethode werde nun *Double Hashing* mit der Hashfunktion $h(k) = k \bmod p$ benutzt. Im Folgenden sei q die grösste Primzahl kleiner als p, $h'(k)$ die zweite Hashfunktion und $s(j,k)$ die Sondierungsfunktion. Die vollständige Hashfunktion bei der j-ten Sondierung sei $h(k) - jh'(k) \bmod p$ bzw. $h(k) - s(j,k) \bmod p$. Entscheiden Sie, welche der folgenden Wahlen von $h'(k)$ bzw. $s(j,k)$ sinnvoll sind und welche nicht, und begründen Sie Ihre Entscheidung.

- $h'(k) = \lceil \ln(k+1) \rceil \bmod q$
- $s(j,k) = k^j \bmod p$
- $s(j,k) = ((k \cdot j) \bmod q) + 1$

e) Welchen Vorteil hat Double Hashing gegenüber quadratischem Sondieren?

Aufgabe 9.2 *Kuckucks-Hashing (Cuckoo Hashing).*

Cuckoo Hashing ist ein Hashverfahren mit der Besonderheit, dass für die Suche und das Entfernen eines Schlüssels in *jedem* Fall nur konstante Zeit benötigt wird. Um das zu erreichen, werden zwei Tabellen T_1 und T_2 gleicher Grösse und zwei Hashfunktionen h_1 und h_2 verwendet. Ein neuer Schlüssel k wird an Position $h_1(k)$ in T_1 gespeichert. Falls es dabei zu einer Kollision kommt, wird der bisher gespeicherte Schlüssel k' verdrängt und an Position $h_2(k')$ in T_2 eingefügt. Verdrängt er dabei einen weiteren Schlüssel, dann wird dieser wiederum in T_1 eingefügt, usw.

Es gibt Situationen, in denen dieses Verfahren nicht terminiert. Nach einer gewissen Anzahl von Einfügeoperationen für verdrängte Schlüssel tritt also dieselbe Tabellenkonfiguration wie zu Beginn auf. In dieser Aufgabe soll ein solcher Fall illustriert werden.

a) Gegeben seien zwei Tabellen mit je fünf Einträgen sowie zwei Hashfunktionen $h_1(k) = k \bmod 5$ und $h_2(k) = \lfloor k/5 \rfloor \bmod 5$. Fügen Sie die Schlüssel $27, 2, 32$ in dieser Reihenfolge in zu Beginn leere Hashtabellen ein.

b) Finden Sie nun einen weiteren Schlüssel, bei dem das oben beschriebene Verfahren nicht terminiert (in diesem Fall würde man zwei neue Tabellen anlegen und die gespeicherten Schlüssel mit zwei neuen Hashfunktionen in den neuen Tabellen speichern).

Aufgabe 9.3 *Queue mit Stacks implementieren.*

Ein gewöhnlicher Stack unterstützt die folgenden zwei Operationen in Zeit $\Theta(1)$:

- PUSH(x): legt das Objekt x auf dem Stack ab.
- POP: gibt das *zuletzt* zum Stack hinzugefügte Objekt aus und entfernt es.

Beschreiben Sie, wie mit zwei Stacks eine Queue implementiert werden kann, sodass die folgenden Operationen nur amortisierte Zeit $\Theta(1)$ benötigen.

- ENQUEUE(x): hängt das Objekt x an das Ende der Queue.
- DEQUEUE: gibt das Objekt aus, das sich *am Anfang* der Queue befindet, und entfernt es.