Grundlagen: Algorithmen und Datenstrukturen

Woche 12

Tobias Eppacher

School of Computation, Information and Technology

14. Juli 2025

Inhalt

Aufgaben

E-Aufgaben

Hausaufgaben

Aufgabe 12.1 - Dirty Double Hashing

Für diese Aufgabe verwenden wir ein modifiziertes Double Hashing, welches beim Löschen von Elementen die abhängigen Kollisionen nicht neu hasht, sondern einfach einen “gelöscht”- Platzhalter einfügt. Die Größe der Hashtabelle ist $m = 11$. Die Schlüssel der Elemente sind die Elemente selbst.

$$h(x, i) = (h_1(x) + i \cdot h_2(x)) \mod m$$
$$h_1(x) = 3x \mod m$$
$$h_2(x) = 1 + (x \mod 10)$$

a) Unter welchen zwei Umständen kann `find` ergebnislos abbrechen?

b) Können bei dieser Vorgehensweise die Platzhalter beim Einfügen überschrieben werden?

c) Wir fügen $n > 0$ Elemente in eine anfangs leere Hashtabelle ausreichender Größe ein und löschen diese wieder. Was ist die Worst-Case Laufzeit einer folgenden `find`-Operation? Begründen Sie Ihre Antwort kurz.

d) Führen Sie die folgenden Operationen aus. Tragen Sie die überprüften Hashwerte ein.

`insert`:	$4, 15, 6, 10$
`delete`:	$4, 10$
`insert`:	$10, 1$

Aufgabe 12.1 - Dirty Double Hashing (a-c)

a) `find`-Abbruchgründe?

b) `insert`: Platzhalter überschreiben?

c) Worst-Case `find` nach `insert` und `delete` von $n > 0$ Elementen?

Aufgabe 12.1 - Dirty Double Hashing (d)

	4	15	6	10	1
$i = 0$					
$i = 1$					
$i = 2$					

	0	1	2	3	4	5	6	7	8	9	10
ins(4)											
ins(15)											
ins(6)											
ins(10)											
del(4)											
del(10)											
ins(10)											
ins(1)											

Aufgabe 12.2 - Multiple Choice (a)

Wir erweitern den Bubblesort-Algorithmus, indem vor **jeder** Vergleichsoperation eine Funktion `isSorted` aufgerufen wird. Diese überprüft, ob das gesamte Feld bereits sortiert ist, indem jedes Paar von benachbarten Elementen uberprüft wird. Dazu wird das gesamte zu sortierende Feld einmal komplett durchlaufen. Ist das Feld sortiert, wird das modifizierte Sortierverfahren sofort beendet.
Es bezeichne f die **Worst-Case**-Laufzeit des Original-Bubblesort-Algorithmus und g die des modifizierten Bubblesort-Algorithmus. Was gilt?

- ☐ $f \in \Theta(g)$
- ☐ $f \notin \Theta(g)$, aber $f \in \mathcal{O}(g)$
- ☐ $f \notin \Theta(g)$ und $g \in \mathcal{O}(f)$
- ☐ weder $g \in \mathcal{O}(f)$ noch $f \in \mathcal{O}(g)$

Aufgabe 12.2 - Multiple Choice (b)

Kreuzen Sie in den Zeilen (1) bis (3) jeweils das stärkste passende Symbol an. D. h. wenn z.B. $\Delta = o$ (bzw. $\Delta = \Theta$) möglich ist, wählen Sie $\Delta = o$ (bzw. $\Delta = \Theta$) und nicht $\Delta = O$. Falls die Funktionen unvergleichbar sind, kreuzen Sie u. (”unvergleichbar“) an. Setzen Sie also in jeder Zeile genau ein Kreuz

Bsp:	$n \in \Delta(n^2)$	☐ o	☐ $\mathcal{O}$	☐ Θ	☐ ω	☐ Ω	☐ u.
(1)	$7\log_2(n) \in \Delta(4\log_2(n^2))$	☐ o	☐ $\mathcal{O}$	☐ Θ	☐ ω	☐ Ω	☐ u.
(2)	$n^3 \in \Delta(n^8(n \mod 2))$	☐ o	☐ $\mathcal{O}$	☐ Θ	☐ ω	☐ Ω	☐ u.
(3)	$4^n \in \Delta(2^{4n})$	☐ o	☐ $\mathcal{O}$	☐ Θ	☐ ω	☐ Ω	☐ u.

Aufgabe 12.2 - Multiple Choice (c & d)

c) Welche Aussagen sind wahr?

- ☐ Jeder AVL-Baum ist zugleich ein binärer Suchbaum.
- ☐ Jeder binäre Suchbaum ist zugleich ein Binärer Heap.
- ☐ Jeder Binäre Heap ist zugleich ein AVL-Baum.
- ☐ Jeder Binäre Heap ist zugleich ein binärer Suchbaum.

d) Welche Operation muss in einem Binären Min-Heap beim Ausführen von `decreaseKey` (Verringern eines Schlüssel-Wertes) u.U. aufgerufen werden, um die Heap-Invariante wiederherzustellen?

☐ `bubbleUp` ☐ `bubbleDown` ☐ `insert` ☐ `delete`

Aufgabe 12.2 - Multiple Choice (e & f)

e) Die durchschnittliche Laufzeit von Algorithmen (*Average Case*) ist

- ☐ uninteressant, da man ausschließlich am *Worst-Case* interessiert ist.
- ☐ oft nur mit großem Aufwand zu berechnen.
- ☐ stets am besten geeignet, um den passenden Algorithmus zu wählen.

f) Der MergeSort-Algorithmus

- ☐ ist für alle Eingaben schneller als jede gute Implementierung von InsertionSort.
- ☐ ist für bestimmte Eingabeklassen signifikant schneller als eine deterministische Implementierung von Quicksort.
- ☐ ist im Schnitt um einen in der Eingabe linearen Faktor schneller als Quicksort.
- ☐ sortiert eine Eingabe im Best Case in linearer Zeit.

Aufgabe 12.2 - Multiple Choice (g)

g) Beim Hashing mit *linear probing*

- ☐ werden Elemente, deren Schlüssel auf den gleichen Wert gehasht werden, in einer Liste abgelegt.
- ☐ ist die Hashfunktion nicht besonders wichtig, da auf ineffiziente Listen verzichtet wird.
- ☐ ist das Löschen von Elementen kompliziert, da Löcher in der Hashtabelle das Auffinden von anderen Elementen verhindern können.

Aufgabe 12.2 - Multiple Choice (h)

h) Der PermutationSort-Algorithmus sortiert ein Feld, indem er die Elemente wiederholt umordnet und jede so entstandene Anordnung auf Sortiertheit prüft.
Es werden systematisch alle möglichen Anordnungen durchprobiert. Wir gehen im Folgenden davon aus, dass PermutationSort ein Feld ohne Duplikate sortiert.

- ☐ Für die *Worst-Case*-Laufzeit f von PermutationSort gilt $f \in \mathcal{O}(n^4)$.
- ☐ Für die *Average-Case*-Laufzeit f von PermutationSort gilt $f \in \Theta(n * n!)$.
- ☐ Fur die *Worst-Case*-Laufzeit f von PermutationSort gilt $f \in \mathcal{O}(n^n)$.
- ☐ Mit einer sehr geringen, positiven Wahrscheinlichkeit terminiert PermutationSort für bestimmte Eingaben niemals.
- ☐ PermutationSort hat eine lineare *Best-Case*-Laufzeit.

Aufgabe 12.3 - Graph Algorithmen

Gegeben sei der folgende Graph. Bearbeiten Sie mit diesem folgende Teilaufgaben:

h) Nennen Sie die Reihenfolge, in der eine Tiefensuche, die bei Knoten 0 gestartet wird, die Knoten des Graphen besucht, wenn die Nachbarn eines Knotens in aufsteigender Reihenfolge entsprechend der natürlichen Zahlenordnung bearbeitet werden.

i) Nennen Sie die Reihenfolge, in der eine Breitensuche, die bei Knoten 0 gestartet wird, die Knoten des Graphen besucht, wenn die Nachbarn eines Knotens in aufsteigender Reihenfolge entsprechend der natürlichen Zahlenordnung bearbeitet werden.

j) Kann der Graph topologisch sortiert werden? Begründen Sie Ihre Antwort und nennen Sie, falls möglich, die topologische Sortierung der Knoten, die sich aus der Reihenfolge ergibt, in der die von Ihnen in Aufgabe a) durchgeführte Tiefensuche die Knoten fertig bearbeitet hat.

Aufgabe 12.2 - Multiple Choice

E-Aufgaben

Aufgabe 11.5 - Bad Hash

Intuition zu Hashfunktion

Aufgabe 11.6 - Linear Probing

Abgeänderte Aufgabe aus Tutorium

Hausaufgaben

Hausaufgabe 9 - AB-Baum
(Deadline: 09.07.2025)

Hausaufgabe 10 - Simple Hashing with Chaining
(Deadline: 16.07.2025)

Hausaufgabe 11 - Double Hashing
(Deadline: 23.07.2025)

Fragen?

- Nach Übung gerne bei mir melden
- Tutoriumschannel oder DM an mich auf Zulip
- Vorlesungschannels von GAD auf Zulip (insbesondere bei Hausaufgaben)

Feedback oder Verbesserungsvorschläge?
Gerne nach dem Tutorium mit mir quatschen oder DM auf Zulip

Bis nächste Woche!