

Tutorium

Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 7

Aufgabe 7.1 – 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 (mit ☹ zu markieren).

Die Größe der Hashtabelle ist $m = 11$. Die Schlüssel der Elemente sind die Elemente selbst. Verwenden Sie die folgenden Hashfunktionen:

$$h(x, i) = (h(x) + i * h'(x)) \bmod 11$$

$$h(x) = 3x \bmod 11$$

$$h'(x) = 1 + (x \bmod 10)$$

Aufgabe 7.1 – Dirty Double Hashing

- a) Können bei dieser Vorgehensweise die Platzhalter beim Einfügen überschrieben werden?
- b) Unter welchen zwei Umständen kann die Find-Operation ergebnislos abbrechen?

Aufgabe 7.1 – Dirty Double Hashing

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 folgende Operationen in der gegebenen Reihenfolge aus (tragen Sie auch alle überprüften Positionen ein):

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

1. Operation: insert(4): Positionen: 1

0	1	2	3	4	5	6	7	8	9	10
	4									

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

2. Operation: insert(15):

Positionen: 1, 7

0	1	2	3	4	5	6	7	8	9	10
	4						15			

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

3. Operation: insert(6): Positionen: 7, 3

0	1	2	3	4	5	6	7	8	9	10
	4		6				15			

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

4. Operation: insert(10): Positionen: 8

0	1	2	3	4	5	6	7	8	9	10
	4		6				15	10		

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

5. Operation: delete(4): Positionen: 1

0	1	2	3	4	5	6	7	8	9	10
	👁		6				15	10		

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10



delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

6. Operation: delete(10):

Positionen: 8

0	1	2	3	4	5	6	7	8	9	10
			6				15			

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

7. Operation: insert(10): Positionen: 8

0	1	2	3	4	5	6	7	8	9	10
	👉		6				15	10		

Aufgabe 7.1 – Dirty Double Hashing

insert: 4, 15, 6, 10

delete: 4, 10

insert: 10, 1

	4	15	6	10	1
i = 0	1	1	7	8	3
i = 1	6	7	3	9	5
i = 2	0	2	10	10	7

8. Operation: insert(1): Positionen: 3, 5

0	1	2	3	4	5	6	7	8	9	10
	👉		6		1		15	10		

Aufgabe 7.2 – Mergesort

Sortieren Sie die Zahlenfolge 523, 126, 67, 1, 500, 34, 21, 229, 9, 123, 13 mit MergeSort. Geben Sie für jede Rekursionsebene jeweils für das Aufspalten der Teilsequenzen und für das Verschmelzen der sortierten Teilsequenzen einen Zwischenschritt an (d.h. bei dieser Eingabesequenz insgesamt circa acht Zwischenschritte), sodass Ihr Vorgehen nachvollzogen werden kann.

Aufgabe 7.2 – Mergesort

Wie viele Rekursionsebenen gibt es im Allgemeinen bei MergeSort (wobei wir den initialen Aufruf von MergeSort nicht als eigene Rekursionsebene zählen)? In welcher Größenordnung liegt asymptotisch der Aufwand für jede Rekursionsebene?

Aufgabe 7.3 – Quicksort

Die Laufzeit von Quicksort hängt von der Wahl des Pivotelements ab. Der Algorithmus aus der Vorlesung wählt immer das letzte Element als Pivotelement.

Wird in den folgenden Teilaufgaben ein anderes Element ausgewählt, vertauschen Sie dieses zunächst mit dem letzten Element und gehen Sie dann wie in der Vorlesung gezeigt vor.

Geben Sie jeweils ein Array der Länge 10 an, bei dem es besonders schlecht ist (d.h., es kommt zu vielen Vertauschungen) und ein Array der Länge 10, bei dem es besonders gut ist (wenige Vertauschungen), immer als Pivotelement

- a) das erste Element zu wählen.
- b) den Median aus dem ersten, letzten und mittleren Element zu wählen, d.h. bei einem Array A der Länge n den Median aus den Elementen $A[0]$, $A[\lfloor (n-1)/2 \rfloor]$ und $A[n-1]$. Der Median ist der mittlere Wert: Z.B. ist es bei $[2, 1, 10]$ der Wert 2 (also nicht der Mittelwert). Ein zweielementiges Array wird direkt ohne Pivotelement sortiert, d.h. $[2, 1]$ wird durch eine Vertauschung zu $[1, 2]$.

Begründen Sie jeweils Ihre Antwort.

Aufgabe 7.3 – Quicksort

a) das erste Element zu wählen.

Aufgabe 7.3 – Quicksort

a) das erste Element zu wählen.

Aufgabe 7.3 – Quicksort

- b) den Median aus dem ersten, letzten und mittleren Element zu wählen, d.h. bei einem Array A der Länge n den Median aus den Elementen $A[0]$, $A[\lfloor (n-1)/2 \rfloor]$ und $A[n-1]$. Der Median ist der mittlere Wert: Z.B. ist es bei $[2, 1, 10]$ der Wert 2 (also nicht der Mittelwert). Ein zweielementiges Array wird direkt ohne Pivotelement sortiert, d.h. $[2, 1]$ wird durch eine Vertauschung zu $[1, 2]$.

Aufgabe 7.3 – Quicksort

- b) den Median aus dem ersten, letzten und mittleren Element zu wählen, d.h. bei einem Array A der Länge n den Median aus den Elementen $A[0]$, $A[\lfloor (n-1)/2 \rfloor]$ und $A[n-1]$. Der Median ist der mittlere Wert: Z.B. ist es bei $[2, 1, 10]$ der Wert 2 (also nicht der Mittelwert). Ein zweielementiges Array wird direkt ohne Pivotelement sortiert, d.h. $[2, 1]$ wird durch eine Vertauschung zu $[1, 2]$.

Aufgabe 7.3 – Quicksort

- c) Sortieren Sie das Array $[5, 4, 6, 7, 3, 2, 8, 9, 1]$ mit Quicksort. Wählen Sie als Pivot-Element jeweils das letzte Element bzw. den Median **wie in Teil b)**.

Aufgabe 7.3 – Quicksort

- c) Sortieren Sie das Array $[5, 4, 6, 7, 3, 2, 8, 9, 1]$ mit Quicksort. Wählen Sie als Pivot-Element jeweils das letzte Element bzw. den Median **wie in Teil b)**.