Algorithmen und
Berechnungskomplexität I
Wintersemester 2017/18

Abgabe: 12.12.2017, 12.00 Uhr
Besprechung: KW 51

UNIVERSITÄT BONN

PD Dr. Elmar Langetepe
David Kübel
Institut für Informatik

Übungsblatt 9

Aufgabe 9.1: (2+2+2 Punkte)

(a) Wir werfen einen roten und einen blauen Spielwürfel. Bestimmen Sie die Wahrscheinlichkeit, dass der rote Würfel mindestens eine 3 und der blaue Würfel eine gerade Zahl zeigt.

(b) Wir werfen einen roten und einen blauen Spielwürfel. Bestimmen Sie die Wahrscheinlichkeit, dass einer der beiden Würfel mindestens eine 3 und der andere Würfel eine gerade Zahl zeigt.

(c) Wir werfen n-mal hintereinander einen gezinkten Würfel, bei dem die Wahrscheinlichkeit, eine Zahl i zu würfeln, gleich p_i ist. Bestimmen Sie, wie oft im Erwartungswert eine Primzahl gewürfelt wird.

Aufgabe 9.2: (2+2+2 Punkte)

Gegeben sei eine Hashtabelle der Größe $m = 7$ und die Hashfunktion $h(x) = x \mod 7$. Fügen Sie die Schlüsselwerte $10, 22, 31, 4, 15, 8$ in gegebener Reihenfolge in die Hashtabelle ein. Es genügt dabei, das Endergebnis aller Einfügeoperationen anzugeben. Ermitteln Sie außerdem die durchschnittliche und maximale Anzahl benötigter Schritte. Verwenden Sie zur Kollisionsbehandlung folgende Strategien:

(a) Verkettete Listen

(b) Lineares Sondieren

(c) Quadratisches Sondieren

Hinweis: Beim linearen Sondieren wird zur Berechnung des Hashwertes die Funktion $lin(x, i) = (h(x) + i) \mod m$ *ausgewertet, wobei* i *bei* 0 *beginnend solange erhöht wird, bis es nicht mehr zu einer Kollision kommt. Beim quadratischen Sondieren wird stattdessen die Funktion* $quad(x, i) = (h(x) \pm i^2) \mod m$ *verwendet. Vor dem Erhöhen von* i *wird* $quad(x, i)$ *zunächst für* $+$ *und (im Falle einer Kollision) dann für* $-$ *ausgewertet.*

Aufgabe 9.3: (6 Punkte)

Betrachten Sie eine Union-Find-Datenstruktur, die durch verkettete Listen repräsentiert wird und zu Anfang leer ist. Wie sieht die Union-Find-Datenstruktur aus, die durch Anwendung der folgenden Folge von Anweisungen entsteht? Wir erlauben dabei, dass eine Menge nicht nur durch ihren Repräsentanten angesprochen wird, sondern alternativ durch einen explizit vergebenen Bezeichner.

1. Make-Set(1)
2. Make-Set(2)
3. Make-Set(3)
4. Make-Set(4)
5. Make-Set(5)
6. Make-Set(6)
7. A := Union(1,2)
8. B := Union(3,4)
9. C := Union(B,6)
10. D := Union(A,C)

Geben Sie auch die Zwischenergebnisse nach den Schritten 7 bis 9 an!

Aufgabe 9.4: (4+2 Punkte)

Eine Funktion $t\colon V \to \{1, \ldots, |V|\}$ auf den Knoten eines gerichteten Graphen $G = (V, E)$ heißt *topologische Sortierung*, falls $t(v) < t(w)$ für alle Kanten (v, w) von G gilt. Veranschaulicht bedeutet das, dass alle Kanten von links nach rechts verlaufen, wenn wir jeden Knoten v an Position $t(v)$ auf der Zahlengerade eintragen.

(a) Betrachten Sie den folgenden Algorithmus zur Ermittlung einer topologischen Sortierung.

Topsort (G, t)

1. Setze count = 0.
2. **while** (G ist nicht leer) **do**
3. Finde einen Knoten v in G mit Ingrad 0.
4. **if** ($v =$ **null**) **then return false**
5. Erhöhe count um 1.
6. Setze $t(v) =$ count.
7. Entferne v und alle Kanten der Gestalt (v, w) aus G.
8. **end while**
9. **return true**

Beweisen Sie, dass TOPSORT eine *topologische Sortierung* von G erzeugt, falls G keinen Kreis besitzt.

Hinweis: Zeigen Sie zunächst, dass jeder azyklische Graph einen Knoten mit Ingrad 0 besitzt. Dazu können Sie die Kontraposition dieser Aussage zeigen: Wenn der Ingrad jedes Knotens mindestens 1 ist, dann besitzt der Graph einen Kreis. Überlegen Sie sich einen konstruktiven Beweis dieser Aussage.

(b) Beweisen Sie, dass G genau dann eine topologische Sortierung besitzt, wenn G keinen Kreis enthält.

Aufgabe 9.5: (2+2+2 Zusatzpunkte)

Betrachten Sie folgenden deterministischen Algorithmus, der für eine Menge A von n paarweise verschiedenen Schlüsseln, rekursiv das i-t größte Element bestimmt:

Select(A, i)

1. Ist $|A| = 1$, gebe das einzige Element direkt zurück. Führe andernfalls folgende Schritte aus.
2. Verteile die n Elemente von A auf $\lfloor n/5 \rfloor$ 5er-Gruppen und eine mit $r \equiv (n \mod 5)$ Elementen.
3. Sortiere in jeder $5er$-Gruppe mit INSERTIONSORT und bestimme ihren (Unter-)Median m_i.
4. Bestimme $M :=$ SELECT$(\{m_1, m_2, \ldots m_l\}, \lfloor l/2 \rfloor)$ für $l = \lceil n/5 \rceil$.
5. Ordne in Linearzeit alle Elemente um M, s. d. M das k-t größte Element von A ist.
6. **if** $i = k$
7. **return** M
8. **else if** $i > k$
9. Seien A' die Elemente aus A, die größer M sind.
10. **return** SELECT$(A', i - k)$
11. **else**
12. Seien A' die Elemente aus A, die kleiner M sind.
13. **return** SELECT(A', i)

(a) Zeigen Sie, dass SELECT$(A, \lceil n/2 \rceil)$ den (Unter-)Median der Menge A berechnet.

(b) Zeigen Sie, dass die Laufzeit des Algorithmus in $O(n)$ liegt.

(c) Begründen Sie, warum bei einer Aufteilung in 3er statt 5er-Gruppen die Laufzeit nicht mehr in $O(n)$ liegt.