Algorithmen und Datenstrukturen

Wintersemester 2020/21 — **Übungsblatt 3** — Prof. Dr. U. Schöning, T. Büchler

Abgabe bis 27.11. **13:00 Uhr**, Besprechung: 30.11-04.12. — Institut für Theoretische Informatik

---

Aufgabe 3.1 *2+2=4 Punkte.*

In der Vorlesung wurde ein $\mathcal{O}(n)$-Zeit-Algorithmus zur Bestimmung des k-kleinsten Elements einer Menge vorgestellt. Dieser unterteilt die Eingabe in Teilmengen der Größe $g = 5$.

(a) Zeigen Sie die Worst-Case-Laufzeit des Verfahrens im Sinne der $\mathcal{O}$-Notation für die Teilmengengröße $g = 7$. Stellen Sie hierfür die Rekursionsgleichung für die Anzahl der Vergleiche $V(n)$ für das Verfahren auf.

(b) Durch Nutzen dieses Algorithmus kann die Worst-Case Laufzeit von Quicksort von $\mathcal{O}(n^2)$ auf $\mathcal{O}(n \log n)$ gesenkt werden. Argumentieren Sie logisch, wieso das funktioniert. Und zeigen Sie die Laufzeit mithilfe einer Rekursionsgleichung.

Aufgabe 3.2 *3 Punkte.*

Gegeben sei folgendes Array: $a = [13, 5, 9, 18, 3, 8, 3, 8, 19, 15, 11, 1]$. Führen Sie die aus der Vorlesung bekannte Prozedur **select** aus um das viert-kleinste Element zu finden. Wählen Sie als Pivotelement immer das linkeste (bzw. das Element an der linken Grenze des betrachteten Teilarrays). Die Einsortierung des Pivotelements soll stabil erfolgen. Geben Sie für jeden (rekursiven) Aufruf der Funktion die Parameter und den Zustand des Array a an.

Aufgabe 3.3 *2+1=3 Punkte.*

Dieser Algorithmus erwartet als Eingabe ein Array a der Länge n, das nur Zahlen aus $\mathbb{N}_0$ enthält. Dabei können Zahlen im Array a mehrmals vorkommen. Die Ausgabe ist ein Array c der Länge n.

(a) Was berechnet dieser Algorithmus?

(b) Unter welcher Bedingung benötigt, der Algorithmus $\mathcal{O}(n)$ Rechenschritte?

```
m ← 0 ;
for i ← 0 to n − 1 do
    if m < a[i] then  m ← a[i] ;
Erstelle Array b[0 . . . m] ← (0, 0, . . . , 0);
for i ← 0 to n − 1 do
    b[a[i]] ← b[a[i]] + 1 ;
Erstelle Array c[0 . . . (n − 1)];
j ← 0 ;
for i ← 0 to m do
    while b[i] > 0 do
        c[j] ← i und j ← j + 1 ;
        b[i] ← b[i] − 1 ;
return c;
```

Aufgabe 3.4 *1+1+1+3=6 Punkte.*

Man kann den "Grad an Unsortiertheit"(GaU) eines Arrays a mit n Elementen angeben, indem man die Anzahl der Paare (i, j) mit $1 \leq i < j \leq n$ und $a[i] > a[j]$ zählt. Bei einem sortierten Array ist der GaU = 0.

a) Wie groß kann der GaU im worst-case sein?

b) Wie ändert sich der GaU wenn BubbleSort eine Vertausch-Aktion durchführt?

c) Wie sind die Elemente in a angeordnet, wenn der GaU maximal ist?

d) Geben Sie den Pseudocode eines Algorithmus an, der den GaU in a berechnet.

Aufgabe 3.5 *4 Punkte.*

Lösen Sie die Programmier-Aufgabe "Marathon".
Geben Sie Ihren Domjudge-Teamnamen bei Ihrer Abgabe an, damit Ihnen Ihre Lösung zugeordnet werden kann.