

3. Übungsblatt

zur Vorlesung

Grundzüge der Informatik I

Abgabe über Ilias bis zum 26.4. 14:00 Uhr.
Besprechung in Kalenderwoche 18.

Aufgabe 1 Schleifeninvariante (1 + 5 + 1 Punkte)

Betrachten Sie den folgenden Algorithmus, der als Eingabe ein Array $A = [a_1, a_2, \dots, a_n]$ der Länge n mit $a_i \in \mathbb{N}$, $1 \leq i \leq n$ erhält.

BerechneWert(A, n):

1. $p = 0$
2. **for** $i = 1$ **to** n **do**
3. $p = p + A[i]$
4. **return** p/n

- a) Stellen Sie eine Behauptung auf, welchen Wert der Algorithmus in Abhängigkeit des Eingabearrays berechnet.
- b) Formulieren Sie eine Schleifeninvariante, die zu Beginn jeder Iteration der *for*-Schleife (Zeilen 2 und 3) für die Variable p gilt. Beweisen Sie diese mittels vollständiger Induktion.
- c) Verwenden Sie die Schleifeninvariante aus Aufgabenteil b), um zu zeigen, dass die Behauptung aus Aufgabenteil a) korrekt ist.

Aufgabe 2 Korrektheit (4 Punkte)

Gegeben sei der folgende Algorithmus zur Berechnung der Potenz a^b für zwei Zahlen $a \in \mathbb{N}_{\geq 0}$ und $b \in \mathbb{N}_{\geq 0}$.

Potenz(a, b):

1. **if** $b = 0$ **then**
2. **return** 1
3. **return** $a \cdot \text{Potenz}(a, b - 1)$

Beweisen Sie mithilfe von vollständiger Induktion die Korrektheit des Algorithmus. Zeigen Sie dazu einen Induktionsanfang für eine geignete Induktionsvariable, formulieren Sie eine Induktionsannahme und zeigen Sie unter dieser einen Induktionsschritt.

1.) a) Der Algorithmus berechnet den Durchschnitt aller Werte in A,
also $(\sum_{i=0}^n a_i)/n$

b) Schleifeninvariante:

$S(i)$: Vor Iteration i hat p den Wert $\sum_{j=0}^{i-1} A[j]$ mit $1 \leq i \leq n+1$

Beweis durch Induktion

IndAnfang ($i=1$):

Vor Iteration 1 ist $p = 0 = \sum_{j=0}^0 A[j]$ (Z 1)

IndAnnahme

$S(j)$ gilt für $1 \leq j \leq i$

IndSchritt ($i \rightarrow i+1$)

Vor Iteration $i+1$ wurde Iteration i ausgeführt, also Z 3

$$p = p + A[i] = \left(\sum_{j=0}^{i-1} A[j] \right) + A[i] = \sum_{j=0}^{i-1} A[j]$$

laut IndAnn gilt $S(j)$, also

$$p = \sum_{j=0}^{i-1} A[j]$$

, da $j \leq i$:

c) Nach b) gilt für alle Iterationen i mit $1 \leq i \leq n+1$ $S(i)$,

insbesondere gilt $S(n+1)$. Iteration $n+1$ ist der Austritt aus der Schleife.

Also gilt nach $S(n+1)$ vor Ausführung von Z 4 $p = \sum_{j=0}^n A[j]$

Da in Z 4 p/n zurückgegeben wird, berechnet der Algo $(\sum_{j=0}^n A[j])/n$

□

2.)

Beh Potenz(a, b) berechnet a^b

Bew durch Induktion über b

Sei a beliebig

Ind Anf $b=0$

Da $b=0$, wird die Bedingung in Z 1 erfüllt und Z 2

wird ausgeführt und Z 1 wird zurück gegeben.

$$1 = a^0 = a^b \quad \checkmark$$

Ind Annahme

Potenz(a, i) berechnet a^i für $0 \leq i < b$

Ind Schritt ($b > 0$)

Da $b > 0$ wird die Bedingung in Z 1 nicht erfüllt und Z 4

wird ausgeführt und $a \cdot$ Potenz($a, b-1$) wird zurückgegeben;

dann Ind Ann berechnet Potenz($a, b-1$) a^{b-1} , ^{mit $b-1 < b$} also wird zurückgegeben:

$$a \cdot a^{b-1} = a^b \quad \checkmark$$



Aufgabe 3 *Merge-Operation (3 + 2 Punkte)*

Betrachten Sie die Funktion $\text{Merge}(A, p, q, r)$ aus der Vorlesung. Diese fügt die sortierten Teilarrays $A[p..q]$ und $A[q+1..r]$, $1 \leq p \leq q < r$, zum sortierten Teilarray $A[p..r]$ zusammen und soll dabei $O(n)$ Zeitschritte benötigen, wobei $n = r - p + 1$ ist.

- a) Spezifizieren Sie die Merge-Funktion in Pseudocode. Geben Sie außerdem eine intuitive Erklärung zu Ihrem Pseudocode an.
- b) Analysieren Sie die asymptotische Worst-Case-Laufzeit Ihres Algorithmus.

Aufgabe 4 *Teile & Herrsche (4 Punkte)*

Gegeben sei ein Feld A mit n natürlichen Zahlen. Entwickeln Sie einen rekursiven Teile-und-Herrsche-Algorithmus, der für die Eingabe A die Anzahl der gerade Zahlen in A berechnet. Geben Sie ihren Algorithmus in Pseudocode an und kommentieren Sie diesen.

3.) Betrachten Sie die Funktion $\text{Merge}(A, p, q, r)$ aus der Vorlesung. Diese fügt die sortierten Teilarrays $A[p..q]$ und $A[q+1..r]$, $1 \leq p \leq q < r$, zum sortierten Teilarray $A[p..r]$ zusammen und soll dabei $O(n)$ Zeitschritte benötigen, wobei $n = r - p + 1$ ist.

- Spezifizieren Sie die Merge-Funktion in Pseudocode. Geben Sie außerdem eine intuitive Erklärung zu Ihrem Pseudocode an.
- Analysieren Sie die Worst-Case-Laufzeit Ihres Algorithmus.

Beispiellosung:

(a) $\text{Merge}(A, p, q, r)$: Feld zum Zusammenfügen von $A_1 = A[p..q]$ und $A_2 = A[q+1..r]$ (b)

1. $B = \text{new array}[1 \dots (r - p + 1)]$
2. $i = p$ aktuelle Position in A_1
3. $j = q + 1$ aktuelle Position in A_2
4. **for** $k = 1$ **to** $r - p + 1$ **do**
5. **if** $i \leq q$ **and** ($j > r$ **or** $A[i] \leq A[j]$)
 then
6. $B[k] = A[i]$ An noch nicht abgearbeitet und nächstes Element in A_1
7. $i = i + 1$
8. **else**
9. $B[k] = A[j]$ in B an aktuelle Position einbauen
10. $j = j + 1$
11. $A[p \dots r] = B[1 \dots (r - p + 1)]$

Laufzeit:

- $r - p + 1 = n$
- 1
- 1
- $n + 1$
- $3 \cdot n$

$$\begin{array}{c} \text{S in beliebige} \\ \text{Position in } A \\ \text{einfügen} \end{array} \quad \bullet n$$

$$6n + 3 \in \mathcal{O}(n)$$

4.)

 $\text{CountEven}(A, n)$ $\text{return Count}(A, 1, n)$ $\text{Count}(A, \ell, r)$ $\text{if } (\ell = r)$ $\text{if } (A[\ell] \% 2 = 0)$ $\text{return } 1$ else $\text{return } 0$ else

$$m = \left\lfloor \frac{r-\ell}{2} \right\rfloor + \ell$$

 $\text{return Count}(A, \ell, m) + \text{Count}(A, m+1, r)$