Aufgabensammlung ADS-Repetitorium 2021

Pseudocode – Groß-Oh-Notation – Rekursion vs. Iteration – Sortieren

Aufgabe 1: Pseudocode – Spot the Error

Die folgenden Algorithmen berechnen nicht das, was sie sollen. Erklären Sie, was der Fehler ist und schreiben Sie den richtigen Algorithmus auf. Geben Sie auch die asymptotische Worst-Case-Laufzeit des korrigierten Algorithmus in Θ-Notation an.

(a) Der Algorithmus soll $f(n) = n$ berechnen.

Algorithmus 1: int Algo1(int n)

```
1 zähler = 0
2 for i = 1 to n do
3 |  return zähler +1
```

(b) Der Algorithmus soll $f(n) = \sum_{i=0}^{n} i$ berechnen

Algorithmus 2: int Algo2(int n)

```
1 zähler = 0
2 for i = 1 to n do
3 |  zähler+1
4 return zähler
```

(c) Der Algorithmus soll $f(n) = n!$ berechnen.

Algorithmus 3: int Algo3(int n)

```
1 return Algo3(n − 1) ·n
```

(d) Der Algorithmus soll **true** zurückgeben, wenn i im Array A enthalten ist, sonst **false**.

Algorithmus 4: boolean Algo4(int i, int[] A, int $l = 0$)

```
1 if A.length == l then
2 |  return false
3 else
4 |  return (i == A[l]) or Algo4(i, l + 1)
```

Aufgabe 2: Vereinigung

Geben Sie in gut kommentiertem Pseudocode einen Algorithmus an, der als Eingabe zwei aufsteigend sortierte Felder A und B erhält. die Ausgabe soll ein Feld C sein, das jede Zahl aus A und B genau einmal enthält. Die Laufzeit soll $O(n)$ sein, wobei n = A.length + B.length.

Aufgabe 3: Algorithmen und Laufzeiten

(a) Was berechnet der Algorithmus?
Wie viele Vergleiche, Additionen und Multiplikationen werden in Abhängigkeit von n ausgeführt?

Algorithmus 5: SomeAlgo(n)

```
1 int j = 0; int s = 1; int S = 0
2 while j < n do
3 |   S = S + s
4 |   j = j + 1
5 |   s = s · 2
6 return S
```

(b) Sei folgender Algorithmus zur Berechnung des Produkts $i \cdot (i+1) \cdot \ldots \cdot (j-1) \cdot j$ für natürliche Zahlen i und j mit $i < j$ gegeben:

Algorithmus 6: int Produkt(int j, int i)

```
1 return Fakultaet(j)/Fakultaet(i − 1)
```

Algorithmus 7: int Fakulaet(int x)

```
1 if x == 0 then
2 |   return 1
3 return x · Fakultaet(x − 1)
```

Begründen Sie kurz, warum der Algorithmus Produkt korrekt ist. Geben Sie die Worst-Case-Laufzeit von Produkt in Abhängigkeit von i und j an.

Aufgabe 4: Sortieralgorithmen

(a) Sortieren Sie das Feld A = [4, 3, 7, 2, 0, 9 ,8, 1, 5, 6] mit InsertionSort. Geben Sie nach jeder Iteration der äußeren Schleife das Feld an.

(b) MergeSort arbeitet rekursiv. Geben Sie für das Feld C = [9, 4, 1, 3, 5, 2, 6, 0, 8, 7] den Rekursionsbaum von MergeSort an. In jedem Knoten soll der jeweilige Aufruf von MergeSort und die zu sortierende Teilliste stehen, jeweils *vor* der Sortierung.

(c) Sortieren Sie das Feld D = [6, 4, 7, 9, 2, 3, 1, 5, 0, 8] mit einem vereinfachten QuickSort. Dieser schreibt alle Elemente, die kleiner als das Pivotelement sind, links und alle größeren rechts neben das Pivotelement. Zeichnen Sie den Rekursionsbaum. Schreiben Sie in jeden Knoten das zu sortierende Teilfeld nach dem Aufruf von Partition und markieren Sie das Pivotelement, die Wurzel sieht also so aus: [6, 4, 7, 0, 2, 3, 1, 5, <u>8</u>, 9]. *Achtung: Das ist nicht der VL-Algorithmus, aber die Idee ist die gleiche!*

Aufgabe 5: Induktionsbeweis eines Algorithmus

Betrachten Sie den in Aufgabe 7 vorgestellten Algorithmus zur Berechnung der Fakultät.

(a) Geben Sie einen Algorithmus in Pseudocode an, der die Fakultät rekursiv berechnet.

(b) Beweisen Sie mittels vollständiger Induktion die Korrektheit Ihrer rekursiven Variante.

Aufgabe 6: Vollständige Induktion

Zeigen Sie die folgenden Aussagen mittels vollständiger Induktion.

(a) Für jede natürliche Zahl n ist 3 ein Teiler von $n^3 - n$.

(b) Die Fibonacci-Folge ist eine rekursiv definierte Zahlenfolge. Dabei ist $F(0) = 0$ und $F(1) = 1$. Die n-te Fibonacci-Zahl für ein $n > 1$ ist dann $F(n-1) + F(n-2)$. Die Berechnungsvorschrift dauert für große n jedoch sehr lange. Mit der Formel von Moivre-Binet kann die n-te Fibonacci-Zahl direkt ausgerechnet werden. Beweisen Sie die Richtigkeit der Formel:

$$F(n) = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

(c) Auf einem quadratischen Schachbrett mit einer Seitenlänge von mehr als drei Feldern kann der Springer jedes Feld von jedem anderen Feld erreichen. Dafür hat er beliebig viele Züge zur Verfügung.

Aufgabe 7: Schleifeninvariante

Gegeben sei der folgende Algorithmus, der die Fakultät einer Zahl k berechnet.

Algorithmus 9: int fakultät(int k)

```
1 f = j = k
2 while j > 1 do
3 |   j = j − 1
4 |   f = f · j
5 return f
```

(a) Geben Sie eine geeignete Invariante an.

(b) Zeigen Sie mit Hilfe der in a) aufgestellten Invariante die Korrektheit des Algorithmus.

Aufgabe 8: Kosinus rekursiv

Es seien die folgenden Theoreme für $x, y \in \mathbb{R}$ gegeben:

Additionstheorem: $\cos(x+y) = 2\cos x \cos y - \cos(x-y)$

Periodizität: $\cos(x + 2\pi) = \cos x$

Wir betrachten im Folgenden nur positive Winkel. Die Kosinusfunktion lässt sich für kleine Winkel x durch $\cos x \approx 1 - x^2/2$ annähern. Diese Näherung gelte für $x < 0,001$. Die Operatoren $+, -, \cdot, /$ und mod werden in konstanter Zeit ausgewertet.

(a) Geben Sie einen rekursiven Algorithmus zur Berechnung des Kosinus in Pseudocode an, der $\cos x$ für $x > 0$ mithilfe des Additionstheorems berechnet. Die gegebene Näherung soll als Abbruchkriterium der Rekursionsgleichung dienen. Die Laufzeit des Algorithmus soll in $\Theta(\log x)$ liegen.

(b) Nun soll die Worst-Case-Laufzeit durch Periodizität verbessert werden. Wie kann dies geschehen? Welche Laufzeit ergibt sich dann?

(c) Was passiert, wenn Sie für $x < 0,001$ die Näherung $\cos x \approx 1$ verwenden?

Aufgabe 9: Ähnliche Zahlen

Sei A ein Feld der Länge $n > 1$ von zufälligen Zahlen, wobei Zahlen mehrfach vorkommen dürfen.

(a) Geben Sie einen Algorithmus in Pseudocode an, der zwei Zahlen A[i] und A[j] mit $i \neq j$ sucht, sodass $|A[i]-A[j]|$ minimal ist. Der Algorithmus soll die Indizes beider Zahlen ausgeben.

(b) Begründen Sie die Korrektheit Ihres Algorithmus, indem Sie die Korrektheit der inneren Schleife mit einer Invariante zeigen.

Aufgabe 10: Noch mehr Korrektheitsbeweise und Rekursion

Betrachten Sie den folgenden Algorithmus.

Algorithmus 12: int doSomethingSimple(**int** A[ ], **int** $i = 1$)

```
Data: Feld mit natürlichen Zahlen A, natürliche Zahl i
Result: Ein Wert, der mit A zusammenhängt
1 if i == A.length then
2     return A[i]
3 k = doSomethingSimple(A, i + 1);
4 if k > A[i] then
5     return k
6 else
7     return A[i]
```

(a) Beschreiben Sie in einem Satz, was der Algorithmus macht.

(b) Beweisen Sie die Korrektheit des Algorithmus.

(c) Geben Sie einen Algorithmus an, der äquivalent zu doSomethingSimple ist, ohne Rekursion zu verwenden.

(d) Geben Sie eine Schleifeninvariante für Ihren inkrementellen Algorithmus an.

(e) Beweisen Sie die Korrektheit Ihres Algorithmus mit der von Ihnen aufgestellten Schleifeninvariante.

Aufgabe 11: $\mathcal{O}$-, Θ- und Ω-Notation

Beweisen oder widerlegen Sie die Behauptungen. Arbeiten Sie mit der Definition aus der Vorlesung.

(a) $f(n) = \frac{1}{2}n - 2 \in \Omega(\log_2 n)$

(b) $f(n) = n^n + n^2 \in O(n^{n-1})$

(c) $f(n) = \frac{n^4-4n^2}{2n+7} \notin O(n^3)$

(d) $f(n) = \log_3(n^5 \cdot 9^{n^2}) \in \Omega(n \log_3 n)$

(e) $f(n) = \log_a n \in \Theta(\log_b n)$ für beliebige $a, b > 1$

(f) $f(n) = \frac{1}{100}n^2 + n \sin n \in \Theta(n^2)$

(g) $f(n) = n^4 - 10n^3 + 2n \in O(n^3)$

(h) $f(n) = \frac{9}{n} \notin \Omega(\frac{1}{\sqrt{n}})$

Aufgabe 12: Worst-Case-Laufzeiten für Algorithmen

Wir betrachten folgendes Problem. Aus einem Eingabefeld A von ganzen Zahlen sollen alle Tupel (i, j) mit $i < j$ ausgegeben werden, sodass A[i] + A[j] ein Vielfaches von 10 ist.

(a) Zeigen Sie, dass jeder Algorithmus, der dieses Problem löst, eine Worst-Case-Laufzeit von $\Omega(n)$ hat.

(b) Kann man eine asymptotisch größere (und damit bessere) untere Schranke angeben? Beweisen Sie Ihre Behauptung. Sie müssen dafür entweder zeigen, dass die Laufzeit für *alle* Listen in $\Omega(n)$ liegt bzw. es *eine* Familie von Listen der Länge n gibt, für die die Laufzeit größer ist.

Aufgabe 13: Flugsicherheit

Im Flugverkehr müssen die Flugzeuge gewisse Abstände einhalten. Gegeben ist eine unsortierte Liste von Flugzeugen. Jedes Flugzeug a hat drei Attribute, nämlich $a.x$, $a.y$ und $a.z$. Diese Attribute geben die Koordinaten im Luftraum an. Sie sollen einen Algorithmus angeben, der `true` ausgibt, falls sich zwei Flugzeuge näher als den Abstand d kommen. Die Laufzeit Ihres Algorithmus soll $\mathcal{O}(n \log n)$ sein. Angenommen, Wurzelberechnungen sind zeitintensiv. Durch welche einfache Änderung muss der Algorithmus weniger Wurzelberechnungen durchführen?

Aufgabe 14: Suppentöpfe

Sie kennen das. Man will sich eine Nudelsuppe kochen, findet aber nicht den passenden Deckel für den Topf, da alle Deckel und Töpfe durcheinandergekommen sind. Da Sie immer auf Ihre Töpfe geachtet haben wissen Sie, dass zu jedem Topf ein Deckel vorhanden ist.

(a) Sie möchten ein *beliebiges* passendes Deckel-Topf-Paar finden. Wie viele Vergleiche sind dafür im besten Fall nötig?

(b) Geben Sie einen Algorithmus in Pseudocode an, der ein Feld T mit Topfgrößen und ein Feld mit Deckelgrößen D entgegennimmt. Die Ausgabe soll aus zwei Indizes i und j bestehen, sodass $\mathsf{D}[i] = \mathsf{T}[j]$. Wie viele Vergleiche braucht Ihr Algorithmus am schlechtesten Fall, um ein solches Paar zu finden? Können Sie Ihren Algorithmus verbessern, sodass er im schlechtesten Fall weniger Vergleiche braucht?

(c) Nun haben Sie genug von der Unordnung und möchten zu jedem Topf den passenden Deckel finden. Wie gehen Sie vor, um jedem Topf einen passenden Deckel zuzuordnen? Sie dürfen dabei nur Topf mit Topf und Deckel mit Deckel vergleichen. Verwenden Sie $\Theta(n \log n)$ Vergleiche.

(d) Lösen Sie nun Teilaufgabe c), aber diesmal sollen nur Vergleiche zwischen je einem Topf und einem Deckel verwendet werden. Die Anzahl der Vergleiche soll wieder in $\Theta(n \log n)$ liegen. Welchem Verfahren aus der Vorlesung ähnelt Ihre Vorgehensweise?