DHBW Mannheim

2. Semester Cyber Security

Algorithmen und Komplexität

N.W. & J.T

16. Juni 2020

Aufgabe Nr. 1

Eigenschaften der Groß-O-Notation

1. Geben Sie die Definition der $\mathcal{O}$-Notation an.
2. Es sei $f \in \mathcal{O}(h_1)$ und $g \in \mathcal{O}(h_2)$. Zeigen Sie, dass $f \cdot g \in \mathcal{O}(h_1 \cdot h_2)$ gilt.
3. Beweisen Sie, dass für alle $f : \mathbb{N} \to \mathbb{R}_+$ gilt, dass $f \in \mathcal{O}(f)$.
4. Beweisen Sie, dass für $f,\ g : \mathbb{N} \to \mathbb{R}_+$ und $d \in \mathbb{R}_+$ gilt, dass
   $g \in \mathcal{O}(f) \to d \cdot g \in \mathcal{O}(f)$.
5. Beweisen Sie, dass für $f,\ g,\ h : \mathbb{N} \to \mathbb{R}_+$ gilt, dass
   $f \in \mathcal{O}(h) \ \wedge g \in \mathcal{O}(h) \to f + g \in \mathcal{O}(h)$.
6. Beweisen Sie, dass für $f,\ g,\ h : \mathbb{N} \to \mathbb{R}_+$ gilt, dass
   $f \in \mathcal{O}(g) \ \wedge g \in \mathcal{O}(h) \to f \in \mathcal{O}(h)$.
7. Angenommen $f,\ g,\ h : \mathbb{N} \to \mathbb{R}_+$. Außerdem wird angenommen, dass der Grenzwert von
   $$\lim_{n\to\infty} \frac{f(n)}{g(n)}$$
   existiert. Beweisen sie, dass dann auch $f \in \mathcal{O}(g)$ gilt.
8. Es seien $f,\ g \in \mathbb{R}_+$. Geben Sie die Definition $f \sim g$ an.

Aufgabe Nr. 2

Groß-O-Notation

1. Zeigen Sie, dass $n^2 \in \mathcal{O}(2^n)$ ist.
2. Zeigen Sie auch, dass $n^3 \in \mathcal{O}(2^n)$ gilt.
3. Zeigen Sie: $\log_2(n) \in \mathcal{O}(\ln(n+1))$
4. Zeigen Sie, dass $\ln^2(n) \in \mathcal{O}(\sqrt{n})$ gilt.
5. Versuchen Sie zu zeigen, dass $n^\alpha \in \mathcal{O}(2^n)$, wenn angenommen werden kann, dass $\alpha \in \mathbb{N}$ vorausgesetzt ist.

Aufgabe Nr. 3

Rekurrenzgleichungen

Lösen Sie folgende Rekurrenzgleichungen:

1. $x_{n+2} = x_{n+1} + x_n$ für welche gilt: $x_0 = 0$ und $x_1 = 1$
2. $x_{n+2} = 4 \cdot x_{n+1} - 4 \cdot x_n + 1$ für welche gilt: $x_0 = 1$ und $x_1 = 3$
3. $a_{n+2} = \frac{1}{6} \cdot a_{n+1} + \frac{1}{6} \cdot a_n$ für welche gilt: $a_0 = 0$ und $a_1 = \frac{5}{6}$
4. $a_{n+2} = -\frac{1}{2} \cdot a_{n+1} + \frac{1}{2} \cdot a_n$ für welche gilt: $a_0 = 2$ und $a_1 = 1$
5. $a_{n+2} = a_{n+1} + 2 \cdot a_n + 1$ für welche gilt: $a_0 = 0$ und $a_1 = -\frac{1}{2}$
6. $a_{n+2} = a_n + 2$ für welche gilt: $a_0 = 2$ und $a_1 = 1$
7. $a_{n+2} = 2 \cdot a_n - a_{n+1}$ für welche gilt: $a_0 = 0$ und $a_0 = 3$
8. $a_{n+2} = 7 \cdot a_{n+1} - 10 \cdot a_n$ für welche gilt: $a_0 = 0$ und $a_0 = 3$
9. $a_{n+1} = 2^n \cdot a_n$ für welche gilt: $a_1 = 1$
10. Stellen Sie mit dem Ansatz $a_k := f(2^k)$ eine Rekurrenzgleichung auf und lösen Sie diese.
    $f(n) = 2 \cdot f(n \backslash 2) + \log_2(n)$
    Es gelten folgende Anfangsbedingungen: $a_0 = 0$ und $a_1 = 1$

Aufgabe Nr. 4

Master Theorem

1. Geben Sie die Definition des Master-Theorems an.
2. Schätzen Sie mit Hilfe des Master-Theorems die Komplexität von f ab.
   $f(n) = 2 \cdot f(n\backslash 2) + n$
3. Schätzen Sie $g(n) = 4 \cdot g(n\backslash 3) + (\frac{2}{3})^2 \cdot n$ mit Hilfe des Master-Theorems ab.
4. Schätzen Sie $g(n) = 4 \cdot g(n\backslash 5) + (\frac{3}{2})^3 \cdot n^2$ mit Hilfe des Master-Theorems ab.
5. Schätzen Sie $g(n) = 4 \cdot g(n\backslash 3) + 2 \cdot n^{\log_3(4)} + n$ mit Hilfe des Master-Theorems ab.

Aufgabe Nr. 5

1. Geben Sie die Gleichung an, mit der wir Merge Sort definiert haben. Alternativ ist auch der Pythoncode von Merge Sort akzeptabel.

2. Wir haben die Funktion, die für zwei sortierte Listen L_1, L_2 berechnet, wie viele Vergleichsoperationen beim Aufruf von $merge(L_1, L_2)$ geschehen, mit $cmpCount$ bezeichnet. Geben Sie die Ungleichung an, die wir für das Verhältnis zwischen $\#L_1, \#L_2$ und $cmpCount(L_1, L_2)$ aufgestellt haben.

3. Schätzen Sie mit Hilfe des Master-Theorems die Komplexität von $MergeSort$ ab.

Aufgabe Nr. 6

Sortierproblem

1. Definieren Sie:
   (a) Partielle Ordnung
   (b) Lineare Ordnung
   (c) Quasiordnung
   (d) Totale-Quasiordnung
2. Geben Sie die Definition des Sortierproblems an.

Aufgabe Nr. 7

Sortieralgorithmen

1. Insertion Sort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
2. Selection Sort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
3. Merge Sort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
   (d) Komplexität im Master-Theorem
4. Quicksort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität
5. Counting Sort
   (a) Formale Defintion mit allen Phasen
   (b) Implementierung
   (c) Komplexität
6. Radix Sort
   (a) Vorgehensweise
   (b) Implementierung
7. Heapsort
   (a) Formale Defintion
   (b) Implementierung
   (c) Komplexität

Aufgabe Nr. 8

ADTs + Set und Maps

1. Geben Sie die formale Definition von ADTs an.
2. Geben Sie die formale Definition des ADT Stack an.
3. Beschreiben Sie welche Aufgabe ein Generator in einem ADT übernimmt.
4. Beschreiben Sie den Shunting-Yard-Algorithmus.
5. Geben Sie die formale Definition des ADT Map an.
6. Was haben geordnete Binärbäume, AVL Bäume, Tries und co. miteinander zu tun, was verbindet sie?
7. Geben Sie die formale Definition und die Komplexität von geordneten Binärbäumen an.
8. Geben Sie die formale Definition und die Komplexität von AVL Bäumen an.
9. Geben Sie die formale Definition und die Komplexität von Tries an.
10. Beschreiben Sie, was eine Prioritätswarteschlange ist.
11. Geben Sie die formale Definition von Prioritätswarteschlangen an.
12. Geben Sie die formale Definition und die Komplexität des Heaps an.