Algorithmen und Datenstrukturen 1

ALGO1 · SoSe-2023 · tcs.uni-frankfurt.de/algo1/ · 2023-07-06 · *ed6968a*

Analyse von Algorithmen (Woche 3)

Eigenständige Vorbereitung:
Lies 📖 CLRS Kapitel 3 sowie 4.3–4.5 und schau dir das 📺 Video der Woche an. Beantworte dabei folgende Leitfragen:

a) Was misst eine Rekursionsgleichung?
b) Wie kann man die Laufzeit eines Programmes experimentell bestimmen?
c) Welche andere Größen als die Laufzeit können durch den asymptotischen Ausdruck $O(n^2)$ gemeint sein?

Zeichenlegende:

✏️ Schriftliche Aufgabe, die du fristgerecht in Moodle abgibst. In der Klausur wirst du alle Aufgaben schriftlich bearbeiten, daher ist das Feedback der Tutoren wichtig, damit du deine Schreibfähigkeiten verbessern kannst.
👍 Diese Art von Aufgabe musst du sicher können, um die Klausur zu bestehen.
🔑 Diese Art von Aufgabe musst du weitgehend können, um die Klausur zu bestehen.
🔭 Diese Art von Aufgabe musst du können, um eine gute Note zu erhalten.
🌈 Diese Aufgabe ist als Knobelspaß gedacht, der das algorithmische Verständnis vertieft.

Aufgabe 3.1 (Asymptotisches Wachstum 👍). Ordne die folgenden Funktionen in aufsteigender Reihenfolge nach ihrem asymptotischem Wachstum. Das heißt, $f(n)$ kommt vor $g(n)$, wenn $f(n) = o(g(n))$ gilt.

$$n \log n \quad n^2 \quad 2^n \quad n^3 \quad \sqrt{n} \quad n$$

Aufgabe 3.2 (Θ-Notation 👍). Vereinfache die folgenden Funktionen in Θ-Notation.

$$n^2 + n^3/2 \qquad 8 \log_2^7 n + 34 \log_2 n + \tfrac{1}{1000} n$$
$$2^n + n^4 \qquad 2^n \cdot 7 + 5 \log_2^3 n$$
$$\log_2 n + n\sqrt{n} \qquad n(n^2 - 18) \log_2 n$$
$$n(n-6) \qquad n \log_2^4 n + n^2$$
$$4\sqrt{n} \qquad n^3 \log_2 n + \sqrt{n} \log_2 n$$
$$99 \sin^2 n + 99 \cos^2 n \qquad \tfrac{100n}{\log n} + \tfrac{1}{n}$$

Aufgabe 3.3 (Looping Louie 👍). Analysiere die Laufzeit für die folgenden Funktionen in n und gib das Ergebnis in O-Notation an.

```
1  Loop1(n)
2  i = 1
3  while i <= n do
4    print "*"
5    i = 2*i
6  return
```

```
1  Loop2(n)
2  i = 1
3  while i <= n do
4    print "*"
5    i = 5*i
6  return
```

```
1  Loop3(n)
2  for i = 1 to n do
3    j = 1
4    while j <= n do
5      print "*"
6      j = j*2
7  return
```

Aufgabe 3.4 (Asymptotische Aussagen 👍). Sind die folgenden Aussagen wahr oder falsch?

$$\frac{1}{20}n^2 + 100n^3 = O(n^2) \qquad \frac{n^3}{1000} + n + 100 = \Omega(n^2)$$
$$\log_2 n + n = O(n) \qquad 2^n + n^2 = \omega(n)$$
$$2^{\log_2 n} = O(n) \qquad \log_4 n + \log_{16} n = \Theta(\log n)$$
$$n^3(n-1)/5 = \Theta(n^3) \qquad n^{1/4} + n^2 = \Theta(n)$$
$$\log_2^2 n + n = \Theta(n) \qquad 2^{\log_4 n} = \Theta(\sqrt{n})$$
$$n^{1.9} \log^9 n = o(n^2/\log n) \qquad n \cdot (n \bmod 7) = \Theta(n)$$
$$f(n) = \omega(g(n)) \Longrightarrow f(n) = \Omega(g(n)) \qquad f(n) = O(g(n)) \Longrightarrow f(n) = o(g(n))$$

Aufgabe 3.5 (Master der Asymptotik 👍). Löse die folgenden Rekursionsgleichungen in Θ-Notation als Funktion von n. Benutze Rekursionsbäume um die Gleichungen für A, B, C zu lösen, und für D, E, F benutze das Mastertheorem.

$$A(n) = 2A(n/4) + \sqrt{n} \qquad B(n) = 2B(n/4) + n \qquad C(n) = 2C(n/4) + n^2$$
$$D(n) = 8D(n/2) + 487n^2 \qquad E(n) = 2E(n/4) + n^{0.4} \qquad F(n) = 9F(n/3) + n^2$$

Aufgabe 3.6 (Verdopplungen 👍). Löse die folgenden Teilaufgaben.

a) (einfach) Algorithmus A benötigt genau $7n^3$ Operationen für eine Eingabe der Länge n. Wie viele Operationen mehr benötigt A bei doppelter Eingabelänge?

b) Betrachte die Laufzeiten für einen Algorithmus B:

Eingabelänge (Bits)	1000	2000	3000	4000	5000
Dauer [Sekunden]	5	20	45	80	125

Schätze die Laufzeit von B auf einer 6000 Bit langen Eingabe. Was ist vermutlich die asymptotische Laufzeit von B? Drück deine Vermutung in Abhängigkeit von Eingabelänge n in O-Notation aus.

c) Algorithmus C benötigt für jede Verdoppelung der Eingabelänge 3 Sekunden länger. Gib die asymptotischen Laufzeit von C in Abhängigkeit von Eingabelänge n in O-Notation an.

Aufgabe 3.7 (Asymptotische Eigenschaften). Löse die folgenden Teilaufgaben.

a) 👍 Seien $f(n)$ und $g(n)$ asymptotisch nicht negative Funktionen. Beweise, dass folgendes gilt: $\max(f(n), g(n)) = \Theta(f(n) + g(n))$.

b) 👍 Erkläre warum die Aussage „Die Laufzeit von Algorithmus D ist mindestens $O(n^2)$“ keinen Sinn ergibt.

c) 👍 Gilt $2^{n+1} = O(2^n)$? Gilt $2^{2n} = O(2^n)$? Beweise deine Behauptung.

d) 🔑 Beweise, dass $\log_2(n!) = O(n \log n)$ gilt.

e) 🔭 Beweise, dass $\log_2(n!) = \Omega(n \log n)$ gilt. Beweise, dass $\log_2(n!) = \Theta(n \log n)$ gilt.

Aufgabe 3.8 (Maximale Teilfelder). Sei $A' \in \mathbb{Z}^n$ als ein Feld $A[0, \ldots, n-1]$ gespeichert. Ein Teilfeld von A ist ein genau dann ein *maximales Teilfeld* $A[i..j]$ mit $0 \le i \le j \le n-1$, wenn die Summe $A[i] + A[i+1] + \cdots + A[j]$ maximal über alle möglichen Teilfelder ist. Löse die folgenden Aufgaben.

a) 👍 Schreib einen Algorithmus, der ein maximales Teilfeld von A in Laufzeit $O(n^3)$ findet.

b) 👍 Schreib einen Algorithmus, der ein maximales Teilfeld von A in Laufzeit $O(n^2)$ findet.

c) 🔑 (schwer) Schreib einen *Divide and Conquer* Algorithmus, der ein maximales Teilfeld von A in Laufzeit $O(n \log n)$ findet.

d) 🔭 (sehr schwer) Schreib einen Algorithmus, der einen maximales Teilfeld von A in Laufzeit $O(n)$ findet.

Aufgabe 3.9 (Fehlende Bitstrings ✏️). Seien n, k, ℓ positive ganze Zahlen mit $n = 2^\ell - k$. Die Elemente von $\{0,1\}^\ell$ heißen *Bitstrings der Länge ℓ*. Ein Wesen namens Regloh besitzt ein unsortiertes Feld $A[1..n]$ von n *unterschiedlichen* Bitstrings der Länge ℓ; das heißt, genau k Bitstrings der Länge ℓ kommen *nicht* in A vor. Regloh erlaubt Zugriff auf das Feld ***nur*** über eine Funktion FetchBit(i, j), welche das $j$te Bit des Strings $A[i]$ zurückliefert.

a) 🔑 Im Fall $k = 1$ fehlt dem Feld A genau ein Bitstring. Beschreibe einen Algorithmus, der für eine Eingabe mit $k = 1$ den fehlenden Bitstring ausgibt und dabei nur $O(n)$ oft auf FetchBit zugreift.

b) 🔭 (schwer) Beschreibe einen Algorithmus, der für eine beliebige Eingabe mit $k \geq 1$ alle fehlenden Bitstrings ausgibt und dabei nur $O(n \log k)$ oft auf FetchBit zugreift.

Beispiel für b). Im Fall $k = 3, n = 5, \ell = 3$ und $A =$`["100", "010", "110", "000", "111"]` wäre FetchBit$(2, 2) = 1$ und FetchBit$(4, 1) = 0$, und der Algorithmus soll `011`, `101` und `001` ausgeben.

Hinweis: In dieser Aufgabe liegt der Fokus auf der Laufzeitanalyse: Dabei soll die Zahl der Zugriffe auf FetchBit beschränkt werden, d.h. alle anderen Schritte, die der Algorithmus ausführt, zählen diesmal nicht mit.

Anforderungen an schriftliche Abgaben. In der Klausur gibt es zum Bestehen notwendige 🔑-Aufgaben, in denen du die Korrektheit und die Laufzeit eines Algorithmus beweisen musst. In den schriftlichen Abgaben übst du, wie man das richtig macht. Es gibt bewusst nur eine schriftliche Aufgabe pro Woche, damit du dich intensiv mit dem Schreiben beschäftigen kannst. Achte hierbei auf die folgenden Aspekte:

Autor:innen. Alle Autor:innen des Dokuments sind namentlich genannt.

Schreibstil. Dein Text sollte durchgehend in grammatikalisch korrekten und ganzen Sätzen geschrieben sein. Achte darauf, dass alle Sätze möglichst kurz sind und keine zu verschachtelten Nebensätze enthalten. Wissenschaftssprache ist dann am besten, wenn sie stilistisch gesehen *einfach* ist. *If you prefer to submit your solutions in English,* ***please do so****.*

Struktur. Dein Text muss nachvollziehbar und zielgerichtet strukturiert sein. Die Lösung zu einer Algorithmenaufgabe ist typischerweise wie folgt strukturiert:

a) Grobe Idee: Zunächst beschreibst du in einer **kurzen** Einleitung die Idee des Algorithmus in natürlicher Sprache auf. Ein kurzer Absatz genügt in den meisten Fällen.
b) Formale Beschreibung: Du beschreibst den Algorithmus in Pseudocode oder Code.
c) Korrektheitsbeweis: Du beweist die Korrektheit des Algorithmus.
d) Laufzeitanalyse: Du analysierst die Laufzeit des Algorithmus.

Insbesondere sollten diese vier Aspekte klar getrennt sein und nicht ineinander fließen.

Korrektheit und Präzision. Dein Text darf keine Widersprüche oder unzulässigen Folgerungen beinhalten, er muss stets präzise und eindeutig sein. **Stell dir bei jedem Satz die Frage, wie ein Leser ihn missverstehen könnte, und formuliere den Satz so lange um, bis keine Missverständnisse mehr möglich sind.**

Zielorientiert. Dein Text muss durchgehend zielgerichted und kompakt sein. Stell dir bei jedem Satz die Frage, ob er nicht doch weg gelassen oder gekürzt werden kann. Irrelevante Ausführungen führen in der Klausur selbst dann zu Punktabzug, wenn sie nicht falsch sind.

Leserliche Handschrift. (falls relevant) Die Handschrift ist durchgehend ohne Mühe lesbar.

Mathematik. Mathematische Ausdrücke sind korrekt gesetzt (z.B. n^2 anstatt `n^2`) und die mathematischen Ausdrücke sind grammatikalisch korrekt in die Satzstruktur eingebettet.

Wissenschaftssprache. Der sprachliche Ausdruck ist sachorientiert mit treffender Wortwahl und präzisen Formulierungen. Fachbegriffe werden einheitlich, präzise und auf den Inhalt der Kurses bezogen verwendet.