Höhere Algorithmik, WS 2014/15 — 1. Übungsblatt

Vor- und Wiederholungsübungen, ohne Abgabe

---

1. Laufzeitvergleich

   Die folgende Tabelle gibt die Laufzeiten von verschiedenen Algorithmen für eine Eingabe der Größe n auf einem bestimmten Rechner an.

Verfahren	Laufzeit in ms	Programmieraufwand
Algorithmus A	$0{,}001n!$	1 Stunde
Algorithmus B	$0{,}01n^2$	1 Tag
Algorithmus C	$0{,}1\, n \log_2 n$	1 Woche
Algorithmus D	$0{,}5n$	10 Wochen

   Welchen Algorithmus würden Sie wählen, wenn das Programm für Eingaben der Größe (a) $n = 10$, (b) $n = 1000$, (c) $n = 10\,000\,000$ bis zum Jahr 2020 (i) einmal pro Jahr, (ii) täglich, (iii) 10-mal pro Sekunde laufen müsste. Was ändert sich, wenn man auf eine dreimal schnellere Hardware umsteigen würde?

2. Wiederholungsübungen zur O-Notation

   (a) Zwei Algorithmen A und B haben die (feste) Laufzeit $f(n)$ bzw. $g(n)$, mit $g(n) = O(f(n) \log n)$. Welche der folgenden Aussagen lässt sich daraus schließen?

      i. Für alle n größer als ein gewisses n_0 ist A schneller als B.
      ii. Für alle n größer als ein gewisses n_0 ist B schneller als A.
      iii. Keine der beiden Aussagen lässt sich daraus folgern.

   (b) Beantworten Sie die gleiche Frage für die Voraussetzung $g(n) = \Omega(f(n) \log n)$.

   (c) Ein Algorithmus hat im schlimmsten Fall die Laufzeit $\Theta(n^2)$. Ist es möglich, dass er für manche Eingaben in $O(n)$ Zeit läuft?

   (d) Ein Algorithmus hat im schlimmsten Fall die Laufzeit $\Theta(n^2)$. Ist es möglich, dass er für alle Eingaben in $O(n)$ Zeit läuft?

   (e) Welche der folgenden Aussagen ist wahr? (a) $3^n = O(2^n)$, (b) $\log 3^n = O(\log 2^n)$, (c) $3^n = \Omega(2^n)$, (d) $\log 3^n = \Omega(\log 2^n)$.

   (f) Ordnen Sie folgende Funktionen nach der Geschwindigkeit des Wachstums, und fassen Sie Funktionen, die asymptotisch gleich stark wachsen ($f(n) = \Theta(g(n))$), zusammen:

   $$2^n, \quad n^2 \log_2(n^2), \quad (\log_2 n)^2, \quad n^2 + n - 1, \quad 4^n + n^2, \quad \log_2 n,$$
   $$n^2 \log_2 n, \quad 2^{\sqrt{n}}, \quad 5n^2 + n \log_2 n, \quad \log_3 n, \quad n \log_2(2^n), \quad n + 2^{2n}.$$

3. Wiederholungsübung zu Graphen

   Betrachten Sie den gerichteten Graphen $G = (V, E)$ mit Knotenmenge $V = \{v_1, v_2, \dots, v_{12}\}$ und Kantenmenge

   $$E = \{\, (v_i, v_j) \mid 1 \le i, j \le 12,\ j - i = 2 \text{ oder } \mathrm{ggT}(i, j) = 2 \,\}$$

   wobei ggT der größte gemeinsame Teiler ist.

   Wieviele Kanten hat der Graph? Finden Sie den kürzesten Weg von v_1 zu v_8. Wie viele kürzeste Wege von v_1 zu v_8 gibt es? Ist der Graph stark zusammenhängend? Bestimmen Sie seine starken Zusammenhangskomponenten.

4. Wiederholungsübung zur Wahrscheinlichkeit

   Beim folgenden Spiel wird ein Würfel viermal geworfen. Beim zweiten, dritten, und vierten Wurf bekommen Sie jedesmal 5 Euro, wenn die Augenzahl höher als alle vorhergehenden Würfe ist. Wie groß ist die Wahrscheinlichkeit, dass beim dritten Wurf ein neuer Höchststand erreicht wird? Wie groß ist der Erwartungswert des Gewinns? Würde sich das Spiel lohnen, wenn Sie dafür 10 Euro Einsatz zahlen müssten?

5. Wiederholungsfrage zu Sortieralgorithmen

   Welche Algorithmen zum Sortieren kennen Sie? Welche Laufzeit und welchen Speicherbedarf haben diese Algorithmen?

6. Wiederholungsfrage zu Graphenalgorithmen

   Welche Algorithmen zum Bestimmen des kürzesten Weges in einem Graphen kennen Sie? Welche Laufzeit und welchen Speicherbedarf haben diese Algorithmen? Für welche Arten von Graphen und welche Daten sind sie anwendbar?

7. Wiederholungsaufgaben zu Rekursionen

   (a) Bestimmen Sie die Lösung der folgenden Rekursionsgleichung für $T(n)$, und beweisen Sie Ihre Antwort durch vollständige Induktion:

   $$T(n) = \begin{cases} 1, & \text{für } n = 1, \\ T(n-1) + 4, & \text{für } n > 1. \end{cases}$$

   (b) Bestimmen Sie für die Lösung $S(n)$ der folgenden Rekursionsgleichung eine obere Schranke der Form

   $$S(n) \leq Cn \log_2 n$$

   für eine geeignete Konstante C, und beweisen Sie Ihre Antwort durch vollständige Induktion:

   $$S(n) = \begin{cases} 3n, & \text{für } n \leq 5, \\ 5n + 2 \cdot S(\lfloor \frac{n}{2} \rfloor), & \text{für } n > 5. \end{cases}$$

   Hinweis: Sie müssen C groß genug wählen, dass der Induktionsbeweis klappt.

   (c) Wie ändert sich die Konstante C, wenn in der Rekursion der Ausdruck $5n + 2 \cdot S(\lfloor \frac{n}{2} \rfloor)$ durch $5n + S(\lfloor \frac{n}{4} \rfloor) + S(\lfloor \frac{3n}{4} \rfloor)$ ersetzt wird?

8. Speicherknappheit (zum Knobeln)

   In einem Feld $a[1], a[2], \ldots, a[n]$ der Länge n sind Werte aus der Menge $\{1, 2, \ldots, n-1\}$ gespeichert. Nach dem Schubfachprinzip gibt es dann mindestens einen Wert, der mehr als einmal vorkommt. Finden Sie einen möglichst effizienten Algorithmus, der einen mehrfach vorkommenden Wert ausgibt, der dabei auf die Eingabe nur lesend zugreift und mit konstantem „zusätzlichen“ Speicher auskommt.

   Das bedeutet: Es gibt nur eine feste Zahl von Registern (die Anzahl ist unabhängig von n), die geschrieben oder gelesen werden können. Jedes dieser Register kann eine ganze Zahl zwischen 0 und n speichern. Die gängigen Operationen auf diesen Registern können in konstanter Zeit durchgeführt werden. Das Eingabefeld kann gelesen, aber *nicht verändert* werden.

   Eine Laufzeit von $O(n \log n)$ ist nicht schwierig, es geht aber sogar in $O(n)$ Zeit!