Algorithmen und Programmierung 3, WS 2019/2020 — 1. Übungsblatt

Abgabe bis Freitag, 25. Oktober 2019, 12:00 Uhr, in die Fächer der Tutoren
Abzugeben sind diesmal die Aufgaben 1, 2, und 6. Null-Punkte-Aufgaben sind zur freiwilligen Vertiefung gedacht und werden nicht abgegeben.

---

1. (i) Vorübung: Wachstum von Laufzeiten, 4 Punkte
   Sortieren Sie die folgenden Laufzeiten aufsteigend: Wenn $f(n) = O(g(n))$ ist, aber nicht $g(n) = O(f(n))$, dann soll $f(n)$ vor $g(n)$ kommen.

   (a) n^3 (b) $\log_2 n$ (c) $1{,}8^n$ (d) n (e) 3^n (f) $\sqrt{n}$ (g) $n(\log_2 n)^2$ (h) n^2

   (ii) Beschleunigung der Hardware, 4 Punkte
   Wenn ein Programm eine Laufzeit $f(n)$ aus Aufgabe (i) hat und die Computer in zehn Jahren um den Faktor 1000 schneller werden, dann kann man Probleme *welcher Größe* in derselben Zeit lösen, in der man heute Probleme der Größe (1) $n = 20$, (2) $n = 1000$ lösen kann?

2. (2 Punkte) Die Computer werden immer schneller, die Speicherelemente immer billiger. Wird der Entwurf von effizienten Algorithmen angesichts dieser Entwicklung in Zukunft an Bedeutung verlieren? Bei welchen Computeranwendungen werden Effizienzfragen eine größere/kleinere Rolle spielen?

   Diskutieren Sie diese Fragen. (mindestens 5–10 Zeilen)

3. Korrektheit von Programmen, 0 Punkte

   Das folgende Programmstück ist ein Versuch, Sortieren durch Einfügen zu implementieren.

```
    void insertionSort(float[] a)
    { float x;
      for (int i = 0; i<a.length; i++)
      { † x = a[i];
        // Sortiere x zwischen a[0..i-1] ein
        // Bestimme zunächst die richtige Position j:
(*)     int j=i-1; † while (j>=0 && a[j-1]>x) j-- † ;
        // Nun verschiebe einen Teil des Feldes und füge x an dieser Stelle ein:
      † for (int k = i-1; k>=j; k--) a[k+1]=a[k];
        a[j]=x; †
      }
    }
```

   Stellen Sie dieses Programm richtig und fügen Sie an den durch † bezeichneten Stellen in Ihrem Programm Schleifeninvarianten in der Form vom Zusicherungen (assertions) ein, aus denen die Korrektheit des Programmes hervorgeht. (Ein formaler Beweis ist nicht erforderlich, aber die Zusicherungen müssen aussagekräftig sein und vor allem zutreffen.)

4. (0 Punkte) Kann man Sortieren durch Einfügen schneller machen, indem man die Einfügestelle `j` schneller findet als oben in Zeile (∗), zum Beispiel durch binäre Suche? Wie ist es im besten und im schlechtesten Fall?

5. (0 Punkte) Was passiert bei (i) *Sortieren durch Einfügen*, (ii) *Sortieren durch Auswahl*, wenn die gegebene Folge bereits (a) aufsteigend oder (b) absteigend sortiert ist? (c) Was passiert, wenn alle Elemente gleich sind? Ist es möglich, dass man bloß $O(n)$ Zeit benötigt?

6. Programmierexperiment zum Sortieren, 10 Punkte

   Schreiben Sie ein Programm in Java zum Sortieren von Zahlen, und testen Sie es mit 20 Millionen zufällig erzeugten Gleitkommazahlen einfacher Genauigkeit (`float`).

   - Programmieren Sie das Verfahren Quicksort. Wenn die Länge der zu sortierenden Teilfolge in der Rekursion unter eine Schranke b sinkt, dann soll aber auf eine einfacheres Sortierverfahren ihrer Wahl umgeschaltet werden (z. B. Sortieren durch Einfügen, Sortieren durch Auswahl, Bubblesort).
   - Das Pivotelement, das bei Quicksort zum Zerlegen der Folge verwendet wird, soll in jeder Teilfolge zufällig ausgewählt werden.
   - Messen Sie die Laufzeit. Zählen Sie auch die *Anzahl der Vergleiche* zwischen den Eingabezahlen mit und geben Sie sie aus.
   - Experimentieren Sie mit mindestens drei verschiedenen Werten von b, und versuchen Sie experimentell, den besten Wert zu finden.
   - Die Eingabewerte sollen gleichverteilt im Intervall $[0, 1]$ erzeugt werden.
   - Machen Sie jeweils (mindestens) zehn Durchläufe mit verschiedenen Datensätzen, und dokumentieren Sie die Ergebnisse. Geben Sie auch die Durchschnittswerte Ihrer Programmläufe an.

   Zusatzaufgabe (0 Punkte): Bestimmen Sie jeweils die Anzahl der mehrfach vorkommenden Werte: Wieviele Zahlen kommen mindestens zweimal in der Eingabe vor?

   Anleitung zur Abgabe der Programmieraufgabe: Laden Sie die dokumentierten Quelltexte (zum Beispiel als `zip`-komprimierten Ordner) auf der Whiteboard-Seite der Veranstaltung[1] bis zum Abgabetermin (Freitag, 25. Oktober 2019, 12:00 Uhr) hoch. Fügen Sie eventuell eine `README`-Datei hinzu, wenn der Aufruf des Programmes und die Benützung nicht selbsterklärend ist.

   Geben Sie zusätzlich den ausgedruckten Quellcode und die Ergebnisse der Testläufe auf Papier ab.

7. Vergleich von asymptotischen Laufzeiten, 0 Punkte

   Welche der beiden folgenden Laufzeiten $f(n)$ und $g(n)$ ist für große Werte von n schneller, und welche für kleine n? Bei welchem Wert von n ändert sich die Antwort?

   (a) $f(n) = 10n(\log_2 n)^2$, $g(n) = 2n^{3/2}$
   (b) $f(n) = 5 \cdot 2^n$, $g(n) = 100n^2 \log_2 n$

8. Wiederholung der O-Notation, 0 Punkte

   (a) Beweisen Sie: Wenn $f(n) = O(g(n))$ ist, dann ist $f(n) + g(n) = O(g(n))$.
   (b) Beweisen Sie: $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$ für $f(n), g(n) > 0$.
   (c) Welche der folgenden Aussagen sind richtig (Begründen Sie Ihre Antworten):
   $n\sqrt{n} = O(n(\log n)^2)$; $n(\log n)^2 = O(n\sqrt{n})$; $n \log n \cdot (\log\log n) = O(n^2)$;
   $n^2 = O(n \log n \cdot (\log\log n))$; $n^2 \cdot 2^n = O(2^{n+2})$; $n^2 \cdot 2^n = O(3^n)$; $3^n = O(n^2 \cdot 2^n)$;

[1] `https://mycampus.imp.fu-berlin.de/x/sL9Pfe`