KIT
Institut für Theoretische Informatik

Jun.-Prof. Henning Meyerhenke, Jun.-Prof. Dennis Hofheinz
Chtistian Staudt, Christoph Striecks

2. Übungsblatt zu Algorithmen I im SS 2015

https://crypto.iti.kit.edu/algo-sose15
{staudt,striecks}@kit.edu

Aufgabe 1 (*$\mathcal{O}$-Kalkül, 7 Punkte*)

Beweisen oder widerlegen Sie:

a) $n = \Theta(\sqrt{n}), \qquad n^2 = \mathrm{o}(n^3), \qquad n^3 = \Omega\left(n^2\right).$ (*1 Punkt*)

b) $(n+1)! = \mathcal{O}(n!), \qquad 2^{n+1} = \Theta(2^n).$ (*1 Punkt*)

c) $f(n) = \mathcal{O}(g(n)) \wedge g(n) = \mathcal{O}(h(n)) \;\Rightarrow\; f(n) = \mathcal{O}(h(n)).$ (*1 Punkt*)

d) $f(n) = \mathcal{O}(n) \Rightarrow f(n)^2 = \mathcal{O}(n^2) \qquad f(n)g(n) = \mathcal{O}(f(n)g(n)).$ (*2 Punkte*)

e) Für feste $a \in \mathbb{R}$, $b \in \mathbb{R}_+$ sei $f : \mathbb{N} \to \mathbb{R}, n \mapsto (n+a)^b$. Es gilt $f(n) = \Theta(n^b)$. (*2 Punkte*)

Aufgabe 2 (*Anwendung Mastertheorem, $1+1+1+1$ Punkte*)

Zeigen Sie mit Hilfe des Master-Theorems scharfe asymptotische Schranken für folgende Rekurrenzen:

a) $A(1) := 1$ und für $n = 2^k, k \in \mathbb{N}$: $A(n) = \sqrt{2}A(n/2) + \tilde{c}^2 n$

b) $B(1) := 4$ und für $n = 4^k, k \in \mathbb{N}$: $B(n) = 4B(n/4) + 4n$

c) $C(1) := 3$ und für $n = 2^k, k \in \mathbb{N}$: $C(n) = 8C(n/2) + n$

d) $D(1) := 2$ und für $n = 5^k, k \in \mathbb{N}$: $D(n) = 3(D(n/5) + 2n + 1) + 3$

Aufgabe 3 (*Asymptotisches Sortieren, $3+2$ Punkte*)

Sortieren Sie die folgenden Funktionen von der asymptotisch kleinsten zur asymptotisch größten. Schreiben Sie $f(n) \ll g(n)$, falls $f(n) = o(g(n))$ und $f(n) \equiv g(n)$ falls $f(n) = \Theta(g(n))$. Geben Sie für die Paare $\{\log n, n/\log n\}$ und $\{n^{\log n}, 2^n\}$ einen Beweis für ihre Relation an. Ansonsten werden keine Beweise benötigt.

$\left(1+\frac{2}{n}\right)^n$	n^{100}	$\log n$	2^n
$n^{1+\varepsilon}$	1	$n^{\log n}$	$n/\log n$
n	n^n	$\log_{100000} n$	$42\, n \log n$
$\log(n^2)$	$\sqrt{\log n}$	$n!$	n^2

Hierbei sei ε eine Konstante mit $1 > \varepsilon > 0$.

Aufgabe 4 (*Doktor Meta, optional*)

Der ebenso geniale wie mächtige Wissenschaftler und Superbösewicht Doktor Meta ist erzürnt. Sein Assistent hat ihn verraten und ein Mitglied seiner Armee gesichtsloser, durchnummerierter Schergen ausgeschaltet. Damit seine unkonkreten Pläne nicht in Gefahr geraten, muss er wissen, um welchen Schergen es sich handelte. Er lässt sie alle vor dem Haifischtank zusammen rufen und steht nun vor der Aufgabe, den Fehlenden zu bestimmen.
Ihnen seien $n-1$ Personen gegeben. Nehmen sie vereinfachend an, dass $n = 2^k$ für ein $k \in \mathbb{N}$ gilt. Jeder Person ist eine Zahl zwischen 1 und n zugeteilt. Keine Zahl ist doppelt vergeben, eine fehlt. Zudem ist jeder Person nur die eigene Zahl bekannt. Sei `Person` ein Datentyp mit dem Attribut *zahl*, welches eine natürliche Zahl ist. Wir können demnach eine Person als Datentyp `Person` darstellen. (Die Zahl einer Person p vom Datentyp `Person` steht Ihnen in dem Feld $p.zahl$ zur Verfügung.) Personen sind gespeichert in der Datenstruktur `Gruppe`, die über dem Datentyp `Person` definiert ist. Eine Gruppe `Gruppe` unterstützt die folgenden Operationen:

- Iteration über alle Personen in der Gruppe: `for each` p, wobei p vom Typ `Person` ist. Die Iteration geschieht in keiner bestimmten Reihenfolge.
- Eine Gruppe lässt sich aufteilen in eine linke und eine rechte (Teil-)Gruppe. Bei der Initialisierung sind beide Teilgruppen leer, bis ihnen Personen zugewiesen werden. G.linkeGruppe: `Gruppe` liefert die linke Gruppe, G.rechteGruppe: `Gruppe` liefert die rechte. Beide Methoden benötigen konstante ($\mathsf{O}(1)$) Zeit. Die so erhaltenen Gruppen sind selbst Gruppen und können selbst wieder unterteilt werden.
- G.schickeLinks(p: `Person`), G.schickeRechts(p: `Person`): Kopiere eine Person p vom Typ `Person` (die in G enthalten sein muss) in die linke bzw. rechte (Teil-)Gruppe. Beide Methoden arbeiten in konstanter Zeit und haben keinen Rückgabewert.
- Die Größe einer Gruppe G vom Typ `Gruppe` kann mit $|G|$ erfragt werden; dies ist ebenfalls in konstanter Zeit möglich.

a) Geben Sie einen nicht-trivialen rekursiven Algorithmus in Pseudocode an, der die fehlende Zahl in $\mathcal{O}(n)$ Schritten bestimmt.

b) Geben Sie für den Algorithmus eine Rekursion für die Laufzeitabschätzung an.

c) Lösen Sie die Rekursion und beweisen Sie damit, dass Ihr Algorithmus in Zeit $\mathcal{O}(n)$ läuft.

Ausgabe: Mittwoch, 22.4.15
Abgabe: Montag, 4.5.15 (wegen des 1.-Mai-Feiertags), 12:45 im Briefkasten im Untergeschoss von Gebäude 50.34