Prof. Dr. Leif Kobbelt
Stefan Dollase, Ira Fesefeldt, Alexandra Heuschling, Gregor Kobsik

Lösung - Übung 3

Tutoraufgabe 1 (Laufzeitanalyse):

Gegeben sei ein Algorithmus, der für ein Array von Integern überprüft, ob die Einträge aufsteigend sortiert sind:

```
1 def is_sorted(a):
2     i = 1
3     while i < len(a):
4         if a[i - 1] > a[i]:
5             return False
6         i += 1
7     return True
```

Bei Betrachtung der Laufzeit wird angenommen, dass die Vergleiche jeweils eine Zeiteinheit benötigen (siehe Zeilen 3 und 4). Die Laufzeit aller weiteren Operationen wird vernachlässigt.
Sei $n \in \mathbb{N}$ die Länge des Arrays a.

a) Bestimmen Sie in Abhängigkeit von n die Best-case Laufzeit $B(n)$. Begründen Sie Ihre Antwort. Geben Sie für jedes n ein Beispielarray an, bei dem diese Laufzeit erreicht wird.

b) Bestimmen Sie in Abhängigkeit von n die Worst-case Laufzeit $W(n)$. Begründen Sie Ihre Antwort. Geben Sie für jedes n ein Beispielarray an, bei dem diese Laufzeit erreicht wird.

c) Bestimmen Sie in Abhängigkeit von n die Average-case Laufzeit $A(n)$. Begründen Sie Ihre Antwort. Hierzu nehmen wir für $n \geq 2$ folgende Verteilung der Eingaben an:

- $\Pr(\mathtt{a}[0] = 1) \;=\; 1$,
- $\Pr(\mathtt{a}[\mathtt{n}-1] = 0) \;=\; 1$ und
- für alle $\mathtt{i} \in \{1, \ldots, n-2\}$ gilt: $\Pr(\mathtt{a}[\mathtt{i}] = 0) \;=\; \Pr(\mathtt{a}[\mathtt{i}] = 1) \;=\; 0.5$.

Der erste Eintrag ist also immer 1 und der letzte Eintrag ist immer 0. Die übrigen Einträge sind mit gleicher Wahrscheinlichkeit entweder 0 oder 1. Insbesondere ist die Wahrscheinlichkeit, dass das Array aufsteigend sortiert ist immer 0. Für $n < 2$ ist die Verteilung der Eingaben nicht relevant.

Hinweise:

- Wir gehen davon aus, dass die Indizierung von Arrays mit 0 beginnt.
- Geben Sie die geforderten Laufzeiten in geschlossener Form (d.h. ohne Summenzeichen, $\Pr(\cdots)$ oder ähnliches) an.

Lösung

a) Der Vergleich in Zeile 3 wird auf jeden Fall einmal ausgeführt. Wir unterscheiden zwei Fälle:

- Falls $n < 2$, so gilt die Schleifenbedingung nicht und der Algorithmus terminiert sofort nach insgesamt einem Vergleich. Beispieleingaben: `a=[]` (für $n = 0$) und `a=[1]` (für $n = 1$).
- Falls $n \geq 2$ gilt die Schleifenbedingung und wir führen zusätzlich den Vergleich in Zeile 4 aus. Falls $\mathtt{a}[0] > \mathtt{a}[1]$ terminiert der Algorithmus nach insgesamt zwei Vergleichen. Beispieleingaben: `E=[2, 1, ...]`

Wir erhalten also:

$$B(n) = \begin{cases} 1 & \text{, falls } n < 2 \\ 2 & \text{, falls } n \geq 2 \end{cases}$$

b) Wir unterscheiden wieder zwei Fälle:

- Falls $n < 2$, gilt wie bei **a)**, dass der Algorithmus nach insgesamt einem Vergleich terminiert. Beispieleingabe: `a = []` (für $n = 0$) und `a = [1]` (für $n = 1$).
- Falls $n \geq 2$ wird die Worst-case Laufzeit genau dann erreicht, wenn das Array bereits sortiert ist. Nur dann wird die Schleifenbedingung (Zeile 3) genau n mal überprüft und die `if`-Bedingung (Zeile 4) genau $n-1$ mal. Beispieleingabe: `a = [1, 2, 3, ..., n]`

Wir erhalten also:

$$W(n) = \begin{cases} 1 & \text{, falls } n < 2 \\ 2n-1 & \text{, falls } n \geq 2 \end{cases}$$

c) Wir unterscheiden wieder zwei Fälle:

- Falls $n < 2$, gilt wie bei **a)**, dass der Algorithmus nach insgesamt einem Vergleich terminiert.
- Falls $n \geq 2$, betrachten wir für jedes $k \in \{1, \ldots, n-1\}$ den Fall dass

$$\mathtt{a[k]} = 0 \text{ und für alle } j < k\colon \mathtt{a[j]} = 1.$$

Hinweis:

- Zur einfachen Berechnung ist es hier wichtig, dass die Fälle disjunkt sind, d.h. die Eingaben überschneiden sich nicht für verschiedene k. Nur so können wir die Wahrscheinlichkeit und die Laufzeit für jedes k einzeln bestimmen und am Ende aufsummieren.

In so einem Fall stellen wir fest, dass die `if`-Bedingung in Zeile 4 zum ersten mal gilt, wenn $\mathtt{i} = k$ ist. Bis dahin durchlaufen wir die Schleife genau k mal und in jedem Durchlauf werden zwei Vergleiche vorgenommen. Die Laufzeit beträgt also $2 \cdot k$.

Als nächstes berechnen wir die Wahrscheinlichkeit eines solchen Falles.

- Falls $k = n-1$, gilt $\mathtt{a} = [1, 1, \ldots, 1, 0]$. Da die Wahrscheinlichkeit eines einzelnen Eintrages unabhängig von den restlichen Einträgen ist, hat diese Eingabe die Wahrscheinlichkeit:

$$\underbrace{\Pr(\mathtt{a[0]} = 1)}_{=1} \cdot \underbrace{\Pr(\mathtt{a[1]} = 1) \cdot \ldots \cdot \Pr(\mathtt{a[n-2]} = 1)}_{=0.5 \cdot \ldots \cdot 0.5 = 0.5^{n-2}} \cdot \underbrace{\Pr(\mathtt{a[n-1]} = 0)}_{=1} = 0.5^{n-2}$$

- Falls $k < n-1$, gilt $\mathtt{a} = [1, 1, \ldots, 1, 0, \bullet, \ldots, \bullet]$, wobei $\bullet$ für einen beliebigen Eintrag steht. Wieder ist die Wahrscheinlichkeit eines einzelnen Eintrages unabhängig von den restlichen Einträgen. Somit ist die Wahrscheinlichkeit für eine solche Eingabe:

$$\underbrace{\Pr(\mathtt{a[0]} = 1)}_{=1} \cdot \underbrace{\Pr(\mathtt{a[1]} = 1) \cdot \ldots \cdot \Pr(\mathtt{a[k-1]} = 1) \cdot \Pr(\mathtt{a[k]} = 0)}_{=0.5 \cdot \ldots \cdot 0.5 \cdot 0.5 = 0.5^k} = 0.5^k$$

Hinweis:

- Wenn wir richtig gerechnet haben, müssten sich die Wahrscheinlichkeiten aller Fälle ($k \in \{1, \ldots, n-1\}$) zu 1 aufsummieren. Wir testen dies (und nutzen die bekannte[a] Gleichung zur geometrische Reihe):

$$\begin{aligned}0.5^{n-2} + \sum_{k=1}^{n-2} 0.5^k &= 0.5^{n-2} + \left(\sum_{k=0}^{n-2} 0.5^k\right) - 0.5^0 \\ &= 0.5^{n-2} + \frac{1 - 0.5^{n-1}}{1 - 0.5} - 1 \\ &= 0.5^{n-2} + 2 \cdot \left(1 - 0.5^{n-1}\right) - 1 \\ &= 0.5^{n-2} + 2 - 0.5^{n-2} - 1 \;=\; \mathbf{1}\end{aligned}$$

Zuletzt summieren wir die Produkte aus Laufzeit und Wahrscheinlichkeit für jedes k auf (und nutzen die etwas unbekanntere Variante[b] der geometrischen Reihe):

$$\begin{aligned}2 \cdot (n-1) \cdot 0.5^{n-2} + \sum_{k=1}^{n-2} 2 \cdot k \cdot 0.5^k &= 2 \cdot \left((n-1) \cdot 0.5^{n-2} + \sum_{k=0}^{n-2} k \cdot 0.5^k\right) \\ &= 2 \cdot \left((n-1) \cdot 0.5^{n-2} + \frac{(n-2) \cdot 0.5^n - (n-1) \cdot 0.5^{n-1} + 0.5}{(0.5-1)^2}\right) \\ &= 2 \cdot \left((n-1) \cdot 0.5^{n-2} + (n-2) \cdot 0.5^{n-2} - (n-1) \cdot 0.5^{n-3} + 0.5^{-1}\right) \\ &= 2 \cdot \left((n-1) \cdot 0.5^{n-2} + (n-2) \cdot 0.5^{n-2} - (n-1) \cdot 0.5^{n-2} \cdot 2 + 2\right) \\ &= 2 \cdot \left(\left((n-1) + (n-2) - 2(n-1)\right) \cdot 0.5^{n-2} + 2\right) \\ &= 2 \cdot \left(\left(n - 1 + n - 2 - 2n + 2\right) \cdot 0.5^{n-2} + 2\right) \\ &= 2 \cdot \left(-1 \cdot 0.5^{n-2} + 2\right) \\ &= 4 - 0.5^{n-3}\end{aligned}$$

Wir erhalten also:

$$A(n) = \begin{cases} 1 & \text{, falls } n < 2 \\ 4 - 0.5^{n-3} & \text{, falls } n \geq 2 \end{cases}$$

[a] https://de.wikipedia.org/wiki/Geometrische_Reihe#Berechnung_der_(endlichen)_Partialsummen_einer_geometrischen_Reihe
[b] https://de.wikipedia.org/wiki/Geometrische_Reihe#Verwandte_Summenformel_1

Tutoraufgabe 2 (Rekursionsgleichung):

Finden Sie für die Rekursionsgleichung F, die wir als

$$F(1) = 1 \text{ und } F(n) = F(n-1) + n^2 + n \text{ für } n > 1$$

definieren, ein Polynom $g : \mathbf{R}_+ \to \mathbf{R}_+$ als obere Abschätzung an, welche nicht rekursiv definiert ist. Beweisen Sie anschließend per vollständiger Induktion, dass ihr Vorschlag tatsächlich eine obere Abschätzung ist, also dass $F(n) \leq g(n)$ ist. Geben Sie schließlich auch eine Abschätzung der Funktion g in O-Notation an.

Lösung

Wir haben die Vermutung, dass die Formel sich kubisch verhalten könnte. Daher schätzen wir die Rekursionsgleichung durch die kubische Formel $g(n) = an^3 + bn^2 + cn + d$ ab. Hierfür benötigen wir jedoch noch Werte für alle Koeffizienten. Diese versuchen wir während des Beweis für die vollständige Induktion zu sammeln.
Für den Induktionsanfang haben wir: $F(1) = 1 \leq a + b + c + d$.
Nun nehmen wir an, die Aussage gelte für ein beliebiges aber festes n als unsere Induktionshypothese.

Für den Induktionsschritt haben wir:

$$\begin{aligned} F(n+1) &= F(n) + (n+1)^2 + (n+1) \\ &\leq an^3 + bn^2 + cn + d + (n+1)^2 + (n+1) && \text{(Induktions Hypothese)} \\ &\leq an^3 + bn^2 + cn + d + n^2 + 2n + 1 + n + 1 && \text{(Vereinfachen)} \\ &= an^3 + (b+1)n^2 + (c+3)n + d + 2 && \text{(Vereinfachen)} \end{aligned}$$

Auf der anderen Seite haben wir:

$$\begin{aligned} g(n+1) &= a(n+1)^3 + b(n+1)^2 + c(n+1) + d \\ &= an^3 + 3an^2 + 3an + a + bn^2 + 2bn + b + cn + c + d && \text{(Auflösen)} \\ &= an^3 + (3a+b)n^2 + (3a+2b+c)n + a + b + c + d && \text{(Polynom Normalform herbeiführen)} \end{aligned}$$

Damit nun $F(n+1) \leq g(n+1)$ ist, müssen folgende Bedingungen gelten:

$$\begin{aligned} 1 &\leq a + b + c + d \\ b + 1 &\leq 3a + b \\ c + 3 &\leq 3a + 2b + c \\ d + 2 &\leq a + b + c + d \end{aligned}$$

Dies ist zum Beispiel für die Belegung $a = \frac{1}{3}$, $b = 1$, $c = \frac{2}{3}$ und $d = -1$ der Fall. Damit bekommen wir dann den Induktionsbeweis:
Induktionsanfang: $F(1) = 1 = \frac{1}{3} + 1 + \frac{2}{3} - 1$
Induktionshypothese gelte für ein beliebiges, aber festes n.
Induktionsschritt:

$$\begin{aligned} g(n+1) &= \frac{1}{3}(n+1)^3 + (n+1)^2 + \frac{2}{3}(n+1) - 1 \\ &= \frac{1}{3}n^3 + n^2 + n + \frac{1}{3} + n^2 + 2n + 1 + \frac{2}{3}n + \frac{2}{3} - 1 && \text{(Auflösen)} \\ &= \frac{1}{3}n^3 + n^2 + \frac{2}{3}n^2 - 1 + n^2 + 3n + 2 && \text{(Neu Anordnen)} \\ &= F(n) + n^2 + 3n + 2 && \text{(Induktionshypothese)} \\ &= F(n) + (n+1)^2 + (n+1) && \text{(Neu Anordnen)} \\ &= F(n+1) && \text{(Definition)} \end{aligned}$$

Damit haben wir nun dass $F(n) \in O(n^3)$ ist da auch $g(n) \in O(n^3)$ ist.

Tutoraufgabe 3 (Master Theorem):

Benutzen Sie das Master Theorem um für folgende Rekursionsgleichungen $T(n)$ eine möglichst gute O-Klasse anzugeben, in der $T(n)$ liegt.

a)

$$T(1) = 1$$
$$T(n) = 16 \cdot T(\frac{n}{2}) + n^5 + \log_4(n) \qquad \text{for } n > 1$$

b)

$$T(1) = 2$$
$$T(n) = 27 \cdot T(\frac{n}{3}) + n^3 + n^2 \qquad \text{for } n > 1$$

c)

$$T(1) = 3$$
$$T(n) = 26 \cdot T(\frac{n}{5}) + n \cdot \log_2 n \qquad \text{for } n > 1$$

Lösung

a) Wir wählen $p = 5$, dann ist $n^5 + \log_4(n) \in O(n^5)$ und $16 < 32 = 2^5$. Somit ist $T(n) \in O(n^5)$ nach dem Master Theorem.

b) Wir wählen $p = 3$, dann ist $n^3 + n^2 \in O(n^3)$ und $27 = 3^3$. Somit ist $T(n) \in O(n^3 \cdot \log(n))$ nach dem Master Theorem.

c) Wir wählen $p = 2$, dann ist $n \cdot \log_2(n) \in O(n^2)$ und $26 > 25 = 5^2$. Somit ist $T(n) \in O(n^{\log_5(26)})$ nach dem Master Theorem.