Prof. Dr. Stefan Göller
Florian Bruse
Dr. Norbert Hundeshagen

Einführung in die Informatik

WS 2018/2019

Übungsblatt 8
13.12.2018-20.12.2018

Abgabe: Bis zum 20.12. 18:00 Uhr über moodle. Reichen Sie pro Aufgabe, die Sie bearbeitet haben, die vorgebenene Rumpfdatei ein. Verändern Sie diese Datei nicht weiter als angegeben. Andere Dateien oder unzulässig veränderte Rumpfdateien werden im Zweifelsfall nicht korrigiert.

Aufgabe 1 (Bisection Search mit Indexermittlung) (15 Punkte):

In der Vorlesung wurde ein verbesserter Algorithmus für Bisection Search vorgestellt:

```
def bisection2(L,e):
        def bisection_hilfe(L, e, links, rechts):
                if links >= rechts:
                        return links == rechts and L[links] == e
                mitte = (links + rechts) // 2
                if L[mitte] == e:
                        return True
                elif L[mitte] > e:
                        return bisection_hilfe(L,e,links,mitte-1)
                else:
                        return bisection_hilfe(L,e,mitte+1,rechts)
        if len(L) == 0:
                return False
        else:
                return bisection_hilfe(L,e,0,len(L)-1)
```

Passen Sie diesen Algorithmus so an, dass er nicht zurückgibt, *ob* ein Element `e` in der Liste `L` enthalten ist, sondern die Position dieses Elements, falls es es enthalten ist, und -1, falls nicht. Falls das Element mehrfach enthalten ist, reicht es, wenn Sie eine passende Position zurückgeben.

Für diese Aufgabe dürfen Sie den kompletten Inhalt der Funktion `bisection2` verändern, inklusive der Köpfe etwaiger Hilfsfunktionen (wie `bisection_hilfe`). Den Kopf der Funktion `bisection2` dürfen Sie nicht verändern.

Aufgabe 2 (Rekursionsgleichungen) (5+10+10 Punkte):

Das folgende, rekursive Programm berechnet die n-te Fibonacci-Zahl:

```
def fib(n):
        if n==0 or n==1:
                return 1
        else:
                return fib(n-1)+fib(n-2)
```

Bei Eingabe $n > 1$ benötigt das Programm fünf Schritte und ruft sich selbst mit den Eingaben $n-1$ und $n-2$ auf. Auf den Eingaben $n = 0$ und $n = 1$ terminiert das Programm in je zwei Schritten. Das motiviert das folgende System von Rekurrenzgleichungen für die Laufzeit des Programms:

$$\begin{aligned} t_0 &= 2 \\ t_1 &= 2 \\ t_n &= 5 + t_{n-1} + t_{n-2} \qquad \text{falls } n \geq 2 \end{aligned}$$

Eine genaue Überführung in eine geschlossene Form erscheint zu aufwändig. Wir vereinfachen uns zunächst die Berechnung, indem wir alle konstanten Schrittzahlen (also hier 2 und 5) in eine Konstante c überführen, deren genauer Wert dann im Folgenden nicht wichtig ist, solange alle durch c vertretenen Werte in $\mathcal{O}(1)$ sind.

Außerdem gilt für alle $n \geq 2$, dass $t_{n-1} \geq t_{n-2}$. Nach Definition gilt für $n > 2$, dass $t_{n-1} = c + t_{n-2} + t_{n-3} \geq t_{n-2}$. Für $n = 2$ gilt, dass $t_{2-1} = t_1 = c = t_0$. Damit gilt, dass

$$t_n = c + t_{n-1} + t_{n-2} \leq c + t_{n-1} + t_{n-1} = c + 2t_{n-1}$$

für alle $n \geq 2$. Als obere Schranke für den exakten Wert der geschlossenen Form können wir die geschlossene Form des folgenden, einfacheren Systems von Rekurrenzgleichungen

berechnen:

$$\begin{aligned} t'_n &= c && \text{falls } n \leq 1 \\ t'_n &= c + 2t'_{n-1} && \text{falls } n \geq 2 \end{aligned}$$

Aus den obigen Überlegungen folgt, dass $t'_n \geq t_n$ für alle $n \in \mathbb{N}$ gilt. Eine geschlossene Form ohne rekursive Definitionen für ein ähnliches System von Gleichungen wurde in der Vorlesung bereits angegeben. Wir passen die Lösung an zu $t'_n = c \cdot (2^n - 1)$ für $n \geq 1$. Dies lässt sich per Induktion wie folgt beweisen:

Als Induktionsstart beweisen wir den Fall $t'_1 = c = c \cdot (2^1 - 1)$ direkt. Für den Induktionsschritt nehmen wir als Induktionshypothese (IH) an, dass die Korrektheit der Gleichung $t'_n = c \cdot (2^n - 1)$ bereits bewiesen haben. Zu zeigen ist jetzt, dass auch $t'_{n+1} = c \cdot (2^{n+1} - 1)$ gilt. Nach Definition gilt $t'_{n+1} = c + 2t'_n$. Nach der Induktionshypothese gilt $t'_n = c \cdot (2^n - 1)$. Wir erhalten die Gleichung

$$t'_{n+1} = c + 2t'_n =_{IH} c + 2c \cdot (2^n - 1) = 2c \cdot 2^n + 2c - c = c \cdot (2^{n+1} - 1)$$

was zu beweisen war. Damit ist die Korrektheit der angegebenen geschlossenen Form bewiesen.

Wir benutzen diese geschlossene Form und die Tatsache, dass $t'_n \geq t_n$ für alle $n \in \mathbb{N}$ gilt, um zu folgern, dass $t_n \in \mathcal{O}(2^n)$ gilt, und damit das Programm `fib(n)` in Zeit in $\mathcal{O}(2^n)$ läuft.

Wir betrachten nun das folgende Programm, welches das einfache Streichholzspiel aus der Vorlesung löst, indem es berechnet, ob der anziehende Spieler eine Gewinnstrategie hat[1].

```
def loese(n):
        if n==0 or n == 2:
                return True
        elif n == 1:
                return False
        else:
                return not loese(n-1) or not loese(n-2) not loese(n-3)
```

a) Beschreiben Sie die Laufzeit des Programms und stellen Sie ein exaktes System von Rekurrenzgleichungen für die Laufzeit (in Abhängigkeit von n) von `loese(n)` auf.

b) Vereinfachen Sie das System geeignet, indem Sie wie oben die Terme nach oben abschätzen. Geben Sie dann eine geschlossene Form für ihr neues System an.

[1]Zur Erinnerung: Beim Streichholzspiel nehmen zwei Spieler abwechselnd je ein, zwei oder drei Streichhölzer von einem Haufen Streichhölzer. Wer das letzte Holz nimmt, veliert. Das Spiel beginnt mit n Streichhölzern.

Hinweis: Statt einer Konstanten c wie im Beispiel oben kann es auch sinnvoll sein, ein Vielfaches der Konstanten (wie $2c$ oder $3c$) in die Abschätzungen einzusetzen.

c) Beweisen Sie die Korrektheit dieser geschlossenen Form per Induktion. Schließen Sie danach geeignet auf eine obere Schranke für die worst-case-Laufzeit von `loese(n)`.

Aufgabe 3 (Rekurrenzgleichungen) (8+8+4 Punkte):

Für den Kurs "Einführung in die Informatik" soll der Meister im Kirschkernweitspucken ermittelt werden. Eine Partie läuft dabei folgendermaßen ab: Zwei Kontrahenden stellen sich nebeneinander und spucken einen Kirschkern möglichst weit. Wer weiter gespuckt hat, hat gewonnen. Wir nehmen an, dass so eine Partie, inklusive Aufstellung usw. genau eine Minute dauert, und niemals unentschieden endet.

Zur Ermittlung des Meisters sind nun verschiedene Formate möglich:

i) (*King of the Hill*) Gibt es nur einen Teilnehmer, hat dieser gewonnen. Ansonsten wird zu Beginn zufällig eine vorläufige Reihung der Spieler ausgelost. Der schlechteste tritt nun gegen den zweitschlechtesten an. Der Verlierer dieser Partie ist ausgeschieden, der Sieger tritt gegen den drittschlechtesten an, usw., bis der vorletzte noch teilnehmende Spieler gegen den Besten antritt. Der Gewinner dieser Partie ist der Meister.

ii) (*Single Elimination*) Gibt es nur einen Teilnehmer, hat dieser gewonnen. Ansonsten wird das Teilnehmerfeld zufällig in zwei gleichgroße Hälften aufgeteilt. Jede dieser Hälften ermittelt nach der Single-Elimination-Methode ihren besten Teilnehmer. Der Sieger der Partie zwischen diesen beiden besten ist der Meister.

iii) (*Round Robin*) Bei diesem Modus wird der Sieger nach Punkten ermittelt. Gibt es nur einen Teilnehmer, hat dieser mit 0 Punkten gewonnen. Ansonsten wird zufällig ein Teilnehmer A ausgelost. Die übrigen Teilnehmer spielen jetzt untereinander ihren Besten B aus, der dabei eine Punktzahl erhält. Dann spielt A gegen jeden der übrigen Teilnehmer, außer B. Für einen Sieg erhält A je einen Punkt, für einen Niederlage keinen. Dann spielt A gegen B. Der Sieger der Partie erhält einen zusätzlichen Punkt. Derjenige von A und B, der mehr Punkte hat, hat gewonnen. Bei Gleichstand entscheidet das Los.

iv) (*Kasseler Methode*) Gibt es nur einen Teilnehmer, hat dieser gewonnen. Bei zwei Teilnehmern entscheidet eine einzelne Partie über den Meister. Bei mehr als zwei Teilnehmern wird ein Teilnehmer A zufällig per Los bestimmt. Die übrigen Teilnehmer spielen nun nach der Kasseler Methode ihren Besten B aus. Danach spielen alle

Teilnehmer außer B, aber inklusive A, nach der Kasseler Methode ihren Besten C aus. Schließlich spielt B gegen C. Der Gewinner dieser Partie ist der Meister.

Wir gehen im Folgenden davon aus, dass die Anzahl der Teilnehmer eine Zweierpotenz ist. Ihre Aufgabe ist nun:

a) Stellen Sie für alle vorgeschlagenen Formate Rekurrenzgleichungen auf, die die Anzahl der benötigten Partien in Abhängigkeit von der Teilnehmergröße auf entsprechende Partiezahlen für kleinere Turniere zurückführen.

b) Geben Sie für jede Rekurrenzgleichung je eine geschlossene Form an. Begründen Sie kurz, warum dies die korrekte Form ist. Sie brauchen keinen Induktionsbeweis zu führen.

c) Berechnen Sie anschließend, wie lange es bei jedem Modus dauern würde, den Meister für die "Einführung in die Informatik" zu bestimmen. Gehen Sie von 32 Teilnehmern aus. Eine Partie dauert genau eine Minute. Da es nur eine Arena gibt, kann immer nur eine Partie gleichzeitig stattfinden. *Bonus*: Rechnen Sie die Turnierdauer ggf. in Stunden, Tage, Jahre um. Ein Jahr hat genau 365 Tage.

d) (Sternchen) Überlegen Sie sich, wie sich die benötigte Zeiten für die Turniere in Abhängigkeit von der Teilnehmerzahl jeweils verkürzen lassen würden, wenn beliebig viele Partien parallel ausgeführt werden könnten (natürlich kann ein Spieler immer nur an einer Partie gleichzeitig teilnehmen). Dazu können Sie die Funktionen $\mathit{floor}(n)$ und $\mathit{ceil}(n)$ nutzen, welche eine Zahl zur jeweils nächsten ganzen Zahl ab- bzw. aufrunden. Beachten Sie auch, dass bspsw. $\mathit{floor}(n/2) + \mathit{ceil}(n/2) = n$ gilt.

Hinweis: Für die Aufgabenteile a) und b) kann es für das Verständnis hilfreich sein, jeweils für einige kleine Eingaben das entsprechende Turnier einmal voll aufzubauen.