Prof. Dr. Stefan Göller
Dr. Da-Jung Cho
Daniel Kernberger

Einführung in die Informatik

WS 2019/2020

Übungsblatt 6
29.11.2019 - 5.12.2019

Abgabe: Bis zum 29.11. 18:00 Uhr über moodle. Reichen Sie pro Aufgabe, die Sie bearbeitet haben, genau eine Textdatei mit dem Namen `aufgabe_i.py`, (wobei `i` die Aufgabennummer ist) ein, welche die Lösung ihrer Gruppe enthält. Verwenden Sie die Vorlagen, die wir in Moodle hinterlegt haben. Beachten Sie den speziellen Dateinamen, der in Aufgabe 3 gefordert ist.

Aufgabe 1 (Optimaler Suchbaum beschleunigt) (20 Punkte):

Erweitern Sie den rekursiven Algorithmus aus der Vorlesung zur Berechnung eines optimalen Suchbaums mittels Memoization, d.h. insbesondere derart dass bereits berechnete optimale Suchbäume für Unterintervalle nicht nochmal neu berechnet werden. Erweitern Sie die Signatur Ihrer rekursiven Funktion `loeseRekursiv` so, dass das neben den Parameter `i` und `j` für die Intervallgrenzen noch einen Parameter `wb` übergeben bekommt, der manchen der Intervalle, beschrieben durch Paare Paar `(i,j)`, einen optimalen binären Suchbaum, wie in der Vorlesung eingeführt, zuweist. Orientieren Sie sich bei der Lösung an der beschleunigten Lösung zur Berechnung des optimalen Produktionswegs aus der Vorlesung.

Aufgabe 2 (Das erweiterte Streichholzspiel) (2+3+20=25 Punkte):

In der Vorlesung wurde das *Streichholzspiel* eingeführt. Dabei war ein Haufen von n Streichholzern gegeben. Ein Zug eines Spielers bestand darin entweder ein, zwei oder drei Streichhölzer vom Haufen zu entfernen. Derjenige Spieler, der den Haufen zuerst leert, hatte verloren.

Das *erweiterte Streichholzspiel* hat sieben nichtleere Haufen. Ein *Zug* besteht darin von genau einem nichtleeren Haufen ein oder mehrere (womöglich alle) Streichhölzer zu entfernen. Verloren hat derjenige Spieler, der den letzten nichtleeren Haufen leert. Erweitern Sie den Algorithmus mittels Memoization zur Berechnung der Gewinnstrategie im Streichholzspiel (aus der Vorlesung) auf das erweiterte Streichholzspiel. Befolgen Sie dazu folgende Punkte:

a) Stellen Sie dazu den aktuellen Spielstand als 7-Tupel mit Elementen vom Typ `int` dar, wobei der Eintrag an Position i beschreibt wieviel Streichhölzer auf Haufen i noch übrig sind (die Haufen haben daher, der Einfachheit halber, Indizes 0 bis 6).

b) Schreiben Sie eine Initialisierungfunktion `init(x)`, die einen Maximalwert x entgegennimmt und jeden Haufen mit einem Zufallswert aus `range`$(1, x+1)$ initialisiert. Die Funktion soll dann diesen Spielstand (wie bei Teilaufgabe a) beschrieben) zurück geben. Verwenden Sie das in der Vorlesung eingeführte Paket `random`.

c) Schreiben Sie, ähnlich wie in der Vorlesung, eine Funktion `strategie`, die einen Spielstand und ein geeignetes Memoisierungswörterbuch übergeben bekommt (so dass bereits berechnete Zwischenwerte nicht nochmal neu berechnet werden) und

   - -1 zurückgibt, falls der anziehende Spieler von diesem Spielstand aus keine Gewinnstrategie hat,
   - ein Paar (i, j) zurückgibt, falls der anziehende Spieler eine Gewinnstratie hat und ein möglicher Gewinnzug darin besteht, vom Haufen i genau j Streichhölzer zu entfernen.

Hinweis. Ihr Memoisierungswörterbuch soll als Schlüsselmenge Spielstände und als Werte entweder Tupel der Form (i, j) oder -1 annehmen.

Aufgabe 3 (Ineffzienter Code, Anzahl der Funktionsaufrufe) (15 Punkte):

Angelehnt an die Folie "Ineffizienter Code" aus der Vorlesung, berechnen Sie die Anzahl $r(i, n)$ von Aufrufen von `fib`(i) innerhalb bei der Berechnung von `fib`(n), für jedes $i \in \{0, \ldots, n\}$. Begründen Sie Ihre Antwort, und beweisen Sie diese mittels vollständiger Induktion in einer Abgabe mit dem Dateinamen `aufgabe_3.txt`.

Hinweise. Aus den Vorlesungsunterlagen können Sie $r(2, 5) = 3$ entnehmen. Bestimmen Sie $r(i, n)$ für $i = n$ und für $i = n - 1$. Stellen Sie dann eine Rekursionsgleichung für $r(i, n)$ für alle $0 \leq i < n - 1$ auf, formulieren Sie eine Vermutung und beweisen Sie diese Vermutung mittels Induktion nach $n - i$.

Um den genauen Wert für $r(i, n)$ zu berechnen, programmieren Sie — wenn Ihnen das hilft — eine neue Variante der rekursiven Funktion, die die Gesamtanzahl der Aufrufe von `fib`(i) innerhalb `fib`(n) (mit-)berechnet und vergleichen Sie die Ausgaben dieser Variante mit Ihrer Vermutung für $r(i, n)$. **Geben Sie Ihr Programm jedoch nicht ab, gefordert ist nur Ihre schriftliche Lösung in** `aufgabe_3.txt`**!**