Discrete Mathematics in Computer Science

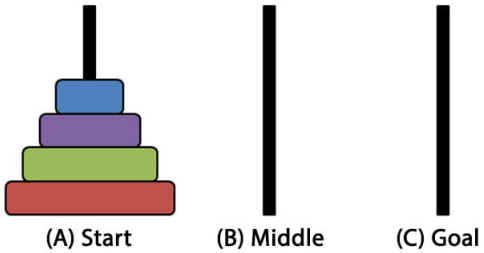
M. Helmert, G. Röger
S. Eriksson
Herbstsemester 2021

Universität Basel
Fachbereich Informatik

Übungsblatt 9

Abgabe: Donnerstag, 25. November 2021

Aufgabe 9.1 (2 Punkte)



Im Towers of Hanoi Puzzle gibt es 3 Stäbe A, B und C, und n Scheiben $\{1, \ldots, n\}$, wobei der Name einer Scheibe ihre Grösse repräsentiert. Am Anfang sind alle Scheiben in abnehmender Reihenfolge auf dem Stab A, wobei die grösste Scheibe zuunterst ist. Wir können eine Scheibe i von X nach Y bewegen, falls i zuoberst auf X liegt und Y entweder keine Scheiben hat oder die oberste Scheibe von Y grösser als i ist. Das Ziel ist, alle Scheiben auf Stab C zu bringen.
Wenn wir den folgenden Algorithmus mit $X = A$, $Y = B$ und $Z = C$ aufrufen, löst er das Problem rekursiv, indem er zuerst Scheiben $\{1, \ldots, n-1\}$ von X nach Y bewegt (wobei Z als temporäre Zwischenlagerung dient), dann n von X nach Z bewegt, und schlussendlich Scheiben $\{1, \ldots, n-1\}$ von Y nach Z bewegt (wobei X als temporäre Zwischenlagerung dient):

procedure TowersOfHanoi(n, X, Y, Z)
  if $n = 1$ **then**
    bewege Scheibe n von X nach Z
  else if $n > 1$ **then**
    TowersOfHanoi($n-1$, X, Z, Y) ▷ bewege Scheiben $\{1, \ldots, n-1\}$ von X nach Y
    bewege Scheibe n von X nach Z
    TowersOfHanoi($n-1$, Y, X, Z) ▷ bewege Scheiben $\{1, \ldots, n-1\}$ von Y nach Z
  end if
end procedure

Die Anzahl Bewegungen, die der Algorithmus braucht, um ein Towers of Hanoi Puzzle mit n Scheiben zu lösen, kann mit der folgenden rekursiven Funktion angegeben werden:

$$moves(0) = 0$$
$$moves(n) = 2 \cdot moves(n-1) + 1$$

Geben Sie die geschlossene Form von $moves(n)$ an und beweisen Sie Ihre Antwort.

Aufgabe 9.2 (3 Punkte)

(a) Wir betrachten eine naive rekursive Implementation von $F(n)$, welche die Fibonacci Nummern berechnet indem sie 0 für $n = 0$, 1 für $n = 1$, und $F(n-1) + F(n-2)$ für $n > 1$ zurückgibt. Wie oft wird die Funktion F aufgerufen wenn wir $F(n)$ für $n > 1$ berechnen? Sie müssen Ihre Antwort nicht begründen.

(b) Memoisierung ist eine Optimierungsmethode für rekursive Funktionen, in welcher Zwischenergebnisse in einer Tabelle gespeichert werden, um sie nicht wiederholt berechnen zu müssen:

```
procedure RecursiveFunction(x, table)
    if base case then
        return base value
    else if table enthält Eintrag für x then
        return table[x]
    else
        table[x] ← rekursive Aufrufe von RecursiveFunction
        return table[x]
    end if
end procedure
```

Wie ändert sich Ihre Antwort zu (a) wenn $F(n)$ Memoisierung anwendet? Sie müssen Ihre Antwort nicht begründen.

(c) Wir nehmen an, dass jeder Aufruf von F eine Nanosekunde benötigt, ausschliesslich der Zeit die in den rekursiven Aufrufen verbracht wird. Geben Sie für $n = 10$, $n = 40$ und $n = 50$ an, wie viel Zeit insgesamt gebraucht wird um $F(n)$ ohne und mit Memoisierung zu berechnen.

Hinweis: Um (a) und (b) zu beantworten empfehlen wir, die Funktion in Ihrer Lieblingsprogrammiersprache mit einem Funktionsaufrufszähler zu implementieren. So können Sie Anhaltspunkte kriegen wie viele Aufrufe für ein spezifisches n benötigt werden. Dadurch können Sie dann ein Muster finden.
Eine Implementierung ist allerdings freiwillig und soll nicht abgegeben werden (sie wird nicht bewertet).

Aufgabe 9.3 (3 Punkte)

Wir betrachten eine Fibonacci-ähnliche Sequenz die wie folgt definiert ist:

$$F'(0) = 4$$
$$F'(1) = 2$$
$$F'(n) = F'(n-1) + F'(n-2) \text{ für } n \geq 2$$

Das Ziel dieser Aufgabe ist eine geschlossene Form von F' mithilfe der Strategie aus Folie 9 (handout Version) von Kapitel D2 zu definieren. Da ein Fehler weitreichende Folgen hat, teilen wir die Aufgabe in drei Teile auf und geben die Lösung des vorherigen Teils für den nächsten Teil vor. Um Punkte zu erreichen müssen Sie den kompletten Rechenweg angeben.

(a) Leiten Sie eine Gleichung für die erzeugende Funktion g her, und vereinfachen Sie sie soweit wie möglich (insbesondere soll die vereinfachte Gleichung keine unendliche Summe mehr beinhalten).

Wie in der Vorlesung können Sie ohne Beweis annehmen, dass die Potenzreihe der F'-Zahlen für $|x| < \varepsilon$ für genügend kleine $\varepsilon > 0$ konvergiert.

(b) Lösen sie die Gleichung $g(x) = x^2 g(x) + x g(x) - 2x + 4$ nach $g(x)$ auf.

(c) Leiten sie die Potenzreihenrepräsentation für $g(x) = \frac{4-2x}{1-x-x^2}$ mithilfe der partiellen Bruchzerlegung her, und geben Sie daraus die geschlossene Form von L an.

Hinweis: Im Gegensatz zu den Fibonacci-Zahlen können Sie hier die partielle Bruchzerlegung direkt auf $g(x)$ anwenden.

Aufgabe 9.4 (2 Punkte)

Der folgende rekursive Algorithmus berechnete das grösste Element eines gegebenen Arrays wenn der Index des ersten Array-Elements als Parameter *left* und der Index des letzten Array-Elements als Parameter *right* benutzt wird.

```
procedure MAX(array, left, right)
    if left = right then
        return array[left]
    else
        middle ← ⌊(left + right)/2⌋
        max1 ← MAX(array, left, middle)
        max2 ← MAX(array, middle + 1, right)
        if max1 > max2 then
            return max1
        else
            return max2
        end if
    end if
end procedure
```

Berechnen Sie die asymptotische Laufzeit von MAX mithilfe des Master Theorems (Slide 27 Handout Version, Kapitel D2).

Regeln zur Abgabe:
Als Abgabe ist nur eine einzelne PDF-Datei (endend auf .pdf), welche mit LaTeX generiert wurde, zugelassen. Die Namen aller Gruppenmitglieder müssen oben auf der ersten Seite stehen. Die Seiten müssen entweder nummeriert sein, oder die Namen der Gruppenmitglieder müssen auf jeder Seite stehen. Die PDF-Datei muss im A4-Format sein (der Inhalt muss auf einen A4-Ausdruck passen).