Übungsblatt 2

Abgabe via Moodle.
Deadline Fr. 20ter Mai

Aufgabe 1 (*Rekurrenzen,* $4+4$ *Punkte*)

1. Gegeben sei folgende Rekurrenz:

$$T(n) \leq \begin{cases} 14 & \text{falls } n = 1, \\ 3n + 4 \cdot T(\lceil n/4 \rceil) & \text{falls } n > 1. \end{cases}$$

   Zeigen Sie durch vollständige Induktion, dass $T(n) \leq 20n^2 - 6n$, falls n eine Viererpotenz ist.

2. Gegeben sei folgende Rekurrenz:

$$T(n) = \begin{cases} c_0 n & \text{falls } n \leq n_0,\ n_0 > 20 \text{ geeignet gewählt}, \\ T(\lceil n/2 \rceil) + T(\frac{2}{5}n + 1) + c_1 n & \text{falls } n > n_0. \end{cases}$$

   Finden Sie eine Funktion f, so dass $T(n) = \Theta(f(n))$ gilt und beweisen Sie ihre Behauptung.

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

b) $B(1) := 1$ und für $n = 3^k, k \in \mathbb{N}$: $B(n) = 9B(n/3) + 4n$

c) $C(1) := 1$ und für $n = 4^k, k \in \mathbb{N}$: $C(n) = C(n/4) + n + 6$

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

Aufgabe 3 (*Invarianten,* $5+3$ *Punkte*)

1. Das Merging Problem ist folgendermaßen definiert:
   Gegeben: zwei aufsteigend sortierte Arrays $A[1..n_1]$, $B[1..n_2]$ von natürlichen Zahlen
   Gesucht: das aufsteigend sortierte Array $C[1..(n_1 + n_2) =: n]$ von natürlichen Zahlen, dass genau die Zahlen von A und B enthält

Der folgende Algorithmus löst das Problem:

1: **procedure** *merge*(A : Array $[1..n_1]$ of $\mathbb{N}_{\geq 0}$, B : Array $[1..n_2]$ of $\mathbb{N}_{\geq 0}$)
2: **precondition** $A[i] \leq A[j] \quad \forall i \leq j$ mit $i, j \in \{1, ..., n_1\}$
3: **precondition** $B[i] \leq B[j] \quad \forall i \leq j$ mit $i, j \in \{1, ..., n_2\}$
4: $A[n_1 + 1] := \infty$, $B[n_2 + 1] := \infty$
5: $n := n_1 + n_2$
6: $j_A := 1$, $j_B := 1$;
7: **for** $i := 1$ **to** n **do**
8: $C[i] = \min(A[j_A], B[j_B])$
9: **if** $A[j_A] < B[j_B]$ **then**
10: $j_A = j_A + 1$
11: **else**
12: $j_B = j_B + 1$
13: **invariant** $C[1..i]$ enthält genau $A[1..j_A - 1],\ B[1..j_B - 1]$
14: **invariant** $B[k] \leq A[j_A] \quad \forall k \in \{1..j_B - 1\}$, $A[k] \leq B[j_B] \quad \forall k \in \{1..j_A - 1\}$
15: **invariant** $C[1..i]$ ist sortiert
16: **assert** $j_A = n_1 + 1, j_B = n_2 + 1$
17: **postcondition** $C[i] \leq C[j] \quad \forall i \leq j, \quad i, j \in \{1, ..., n\}$
18: **postcondition** $C[1..n]$ enthält genau $A[1..n_1], B[1..n_2]$
19: **return** C

Beweisen sie die Korrektheit des vorgegebenen Algorithmus in dem Sie die vorgegebenen Invarianten und Assertions beweisen. Beweisen Sie außerdem, dass der vorgegebene Algorithmus linearen Zeitverbrauch hat.

2. Gegeben sei $n \in \mathbb{N}$. Geben Sie einen iterativen Algorithmus an, der $n!$ berechnet und beweisen Sie die Korrektheit Ihres Algorithmus über eine Invariante.

Aufgabe P2 (*Angry Professor, optional*)

Für die praktischen Übungen verwenden wir die Plattform `www.hackerrank.com`. Hier müssen Sie sich registrieren um an den Übungen teilzunehmen. Unter dem Link

`https://www.hackerrank.com/ads-i-praktische-uebung`

finden die praktischen Übungen in der Form eines Programmierwettbewerbs statt.

Die zweite Challenge heißt "Angry Professor". Hierbei geht es um einen Professor, dessen Studenten sehr undiszipliniert sind und oft zu spät kommen. Er hat entschieden seine Vorlesung nur noch dann zu halten, wenn k Studenten zu Vorlesungsbeginn (Zeitpunkt 0) anwesend sind. Ihre Aufgabe ist es ein Programm zu schreiben, dass entscheidet, ob die Vorlesung stattfindet oder nicht. Hierzu gibt es folgende Eingaben:

- die Anzahl der Studenten n,
- die Grenze k, so viele Studenten müssen zu Vorlesungsbeginn anwesend sein und
- die Ankunftszeiten der Studenten in einem Array a der Länge n.

Die Ausgabe ist entweder "YES", falls die Vorlesung stattfindet, oder "NO", falls nicht.

Eine genauere Beschreibung, sowie ein Beispiel dazu finden Sie auf HackerRank.