Aufgaben zur Veranstaltung
Algorithmen und Datenstrukturen, SS 2022

H. Pflug, J. Dietel FH Aachen, Campus Jülich; IT Center, RWTH Aachen

Präsenzaufgaben 14 21.06.2022

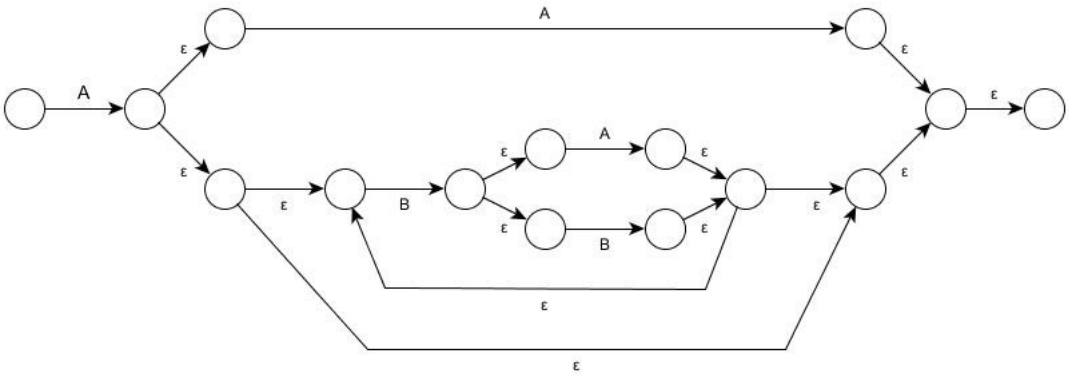
Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

Aufgabe 1: Fragen

a) Welcher schnelle Sortieralgorithmus ist stabil?
b) Welche Bedingung muss ein Feld erfüllen, damit man darin ein Element mit O(log n) finden kann?
c) Nennen Sie ein Sortierverfahren, dessen Laufzeitkomplexität im *average case* $O(n^2)$ und im *best case* O(n) ist.
d) Nennen Sie eine Optimierung des Quick-Sort-Verfahrens bezüglich der Wahl des Pivot-Elements.
e) Nennen Sie eine in der Vorlesung vorgestellte Anwendung des Entwurfsprinzips „dynamische Programmierung“.
f) Kreuzen Sie an, welche der nachfolgenden Zeichenketten vom regulären Ausdruck **(ab|ba)*(b*|a*)** erkannt wird.

Zeichenkette	wird erkannt
abbbab	[ ]
baabba	[ ]
aabbbb	[ ]
aaabab	[ ]

g) Welchem regulären Ausdruck entspricht dieser nichtdeterministische endliche Automat?



Aufgabe 2:

Sortieren Sie das folgende Feld mit Quick-Sort in aufsteigender Reihenfolge. Dabei sollte als Pivot-Element immer das am weitesten rechts liegende Element gewählt werden. Teilfelder von 2 Elementen brauchen Sie nicht mehr mit Quick-Sort zu sortieren - hier dürfen Sie die beiden Elemente gegebenenfalls einfach tauschen. Bitte markieren Sie in jedem Schritt das Pivot-Element, die Vertauschungen und die Grenzen.

8	*3*	*2*	*10*	*14*	*6*	*19*	*12*	*5*	*13*

Aufgabe 3:

Wie Aufgabe 2, nur mit einem etwas größeren Feld.

14	*10*	*6*	*8*	*18*	*19*	*3*	*13*	*2*	*5*	*12*	*15*	*1*	*4*	*9*

Aufgabe 4:

Analysieren Sie den folgenden Code und bearbeiten Sie die unten aufgeführten Teilaufgaben.

```
public void insert(int data) {
  TreeNode temp=root;
  while(temp!=null) {
    if (temp.value>data) {
      temp=temp.left;
    } else {
      temp=temp.right;
    }
  }
  temp = new TreeNode(data);
}
```

a) Die Funktion soll eine in der Vorlesung vorgestellte Funktionalität implementieren. Um welche Funktionalität handelt es sich?
b) Was bewirkt die Funktion stattdessen? Warum?
c) Um Normalfälle korrekt verarbeiten zu können, muss der Code an mehreren Stellen korrigiert werden. Geben Sie einen verbesserten Code an.
d) Was müsste man außerdem am Code ändern, wenn mehrfache Werte verboten sind? Beschreiben Sie die Lösung in Textform. Achten Sie auch auf sinnvolle Rückmeldungen an den Benutzer.

Aufgabe 5:

Ein typisches Beispiel einer klausurrelevanten Aufgabe aus der theoretischen Informatik ist das Folgende:

Lösen Sie folgende lineare Rekursionsgleichung mit Hilfe von Erzeugenden Funktionen:

$$T(n)=\begin{cases} 4\cdot T(n-3)+4\cdot T(n-2)-1\cdot T(n-1) & \text{falls } n>2 \\ 1 & \text{sonst} \end{cases}$$

a) Sehen Sie sich die Aufgabe und die zugehörige Musterlösung genau an.
   - Eine gedruckte Musterlösung finden Sie auch im ILIAS-Ordner der Algorithmen-Vorlesung unter der Woche 14 (ue-3LV.pdf). Es handelt sich um Aufgabe 1a.
   - Optional:
     - Die passenden Vorlesungsfolien finden Sie im ILIAS-Kurs zur Theoretischen Informatik unter *15-Rekursiongleichungen.* Relevant sind die Folien 16-40. Das Beispiel (Fibonacci-Folge) ist ein Standardbeispiel mit allerdings recht krummen Ergebnissen.
     - Haben Sie noch nicht genug? Das Fibonacci-Beispiel ist auch in folgenden Youtube-Videos erklärt:

       https://www.youtube.com/watch?v=aA0HDsa1Mu0
       https://www.youtube.com/watch?v=gwoFNE6QWVg

b) Lösen Sie anschließend analog folgende Aufgabe:

   Gegeben sei die Rekursionsgleichung

$$T(n)=\begin{cases} 0 & \text{falls } n=0 \\ 1 & \text{falls } n=1 \\ 7\cdot T(n-1)-12\cdot T(n-2) & \text{sonst} \end{cases}$$

   Benutzen Sie die Technik der Erzeugenden Funktion, um eine geschlossene Form für *T(n)* zu finden.

   Zur Selbstkontrolle: Das Ergebnis ist $T(n)=4^n-3^n$.