Technische Universität München
Institut für Informatik
Lehrstuhl für Effiziente Algorithmen
Volker Heun

WS 1999/2000
Übungsblatt 5
7. Dez. 1999

Grundlegende Algorithmen

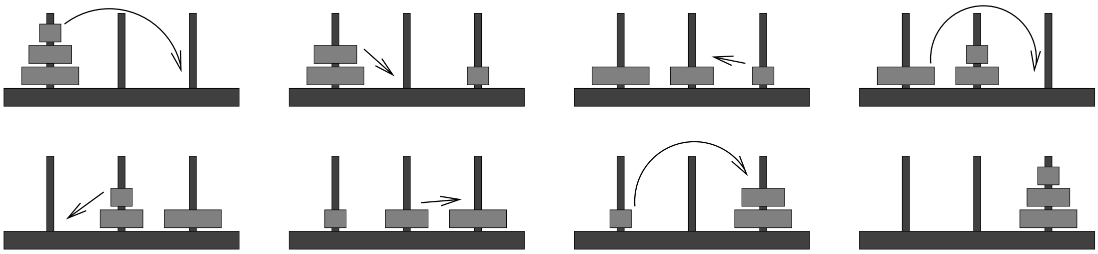
Abgabetermin: 14. Dezember

Aufgabe 1

Betrachten wir das folgende Spiel (Türme von Hanoi), das aus drei Stäben A, B und C besteht, die senkrecht im Boden befestigt sind. Weiter gibt es n kreisförmige Scheiben mit einem Loch im Mittelpunkt, so daß man sie auf die Stäbe stecken kann. Dabei haben die Scheiben verschiedene Radien, je eine mit Radius $i \in [1 : n]$. Zu Beginn stecken alle Scheiben auf dem Stab A, wobei immer eine kleinere auf einer größeren liegt.

Das Ziel des Spiels ist es nun, die Scheiben so umzuordnen, daß sie in der gleichen Reihenfolge auf dem Stab C liegen. Dabei darf immer nur eine Scheibe bewegt werden und es darf nie eine größere auf einer kleineren Scheibe liegen.

Ein Beispiel für 3 Scheiben finden Sie im folgenden Bild:



Entwerfen Sie einen rekursiven Algorithmus zur Lösung dieses Problems und analysieren die dessen Zeitkomplexität (Anzahl Bewegungen der Scheiben).

Können Sie einen Algorithmus finden, der ohne Rekursion auskommt?

Unter Emacs können sie sich die Türme von Hanoi mit `M-x hanoi <Enter>` für $n = 6$ anschauen bzw. mit `M-n M-x hanoi <Enter>` für beliebiges (nicht zu großes n) anschauen.

Bitte wenden!

Aufgabe 2

Ein *erweiterter binärer Baum* ist ein Baum, in dem jeder Knoten entweder zwei oder keine Kinder hat.

Zeigen Sie, daß in jedem erweiterten binären Baum die Anzahl der Blätter um eins größer ist als die Anzahl der inneren Knoten.

Für welche $n \in \mathbb{N}$ gibt es erweiterte binäre Bäume mit n Knoten.

Erinnerung: Ein Knoten mit keinem bzw. mindestens einem Kind heißt Blatt bzw. innerer Knoten.

Aufgabe 3

Man kann einen neuen Heap auch wie folgt aufbauen. Am Anfang ist ein Baum mit einem Knoten (die Wurzel) ein Heap. Nun ergänzt man in jedem Schritt den fast vollständigen binären Baum um ein Blatt. Den Inhalt dieses Blattes läßt man nun in dem Baum soweit durch Vertauschungen nach oben wandern, bis die Heap-Eigenschaft wiederhergestellt ist.

Beschreiben Sie den Algorithmus zum Aufbauen detailliert (z.B. in Pseudo-Code) und analysieren Sie dessen Komplexität (Anzahl der benötigten Vergleiche).

Hinweis: Vergessen Sie bitte nicht, das Programm ausreichend zu kommentieren.