Informatik II (D-ITET)
FS 2009
Prof. Dr. F. Mattern
Philipp Bolliger (bolligph@inf.ethz.ch)

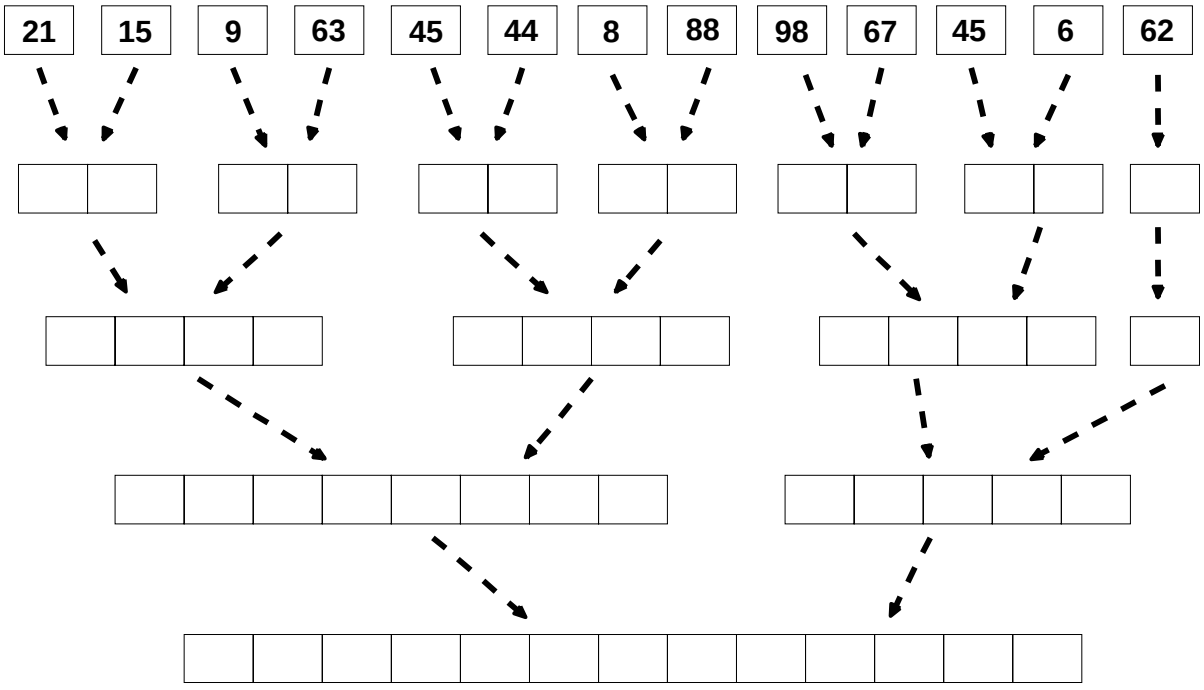
ETH Zürich
Institut für Pervasive Computing
Verteilte Systeme
http://www.vs.inf.ethz.ch

Übungsserie Nr. 10

Ausgabe: 6. Mai 2009
Abgabe: 13./14. Mai 2009

1. Aufgabe: (8 Punkte) Mergesort

(1a) *(2 Punkte)* Führen Sie einen Lauf von Mergesort manuell aus. Füllen Sie dazu das folgende Schema. Die erste Zeile enthält eine Sequenz ungeordneter Zahlen, die absteigend sortiert werden soll.



(1b) *(1 Punkt)* Welche Zeitkomplexität besitzt der Mergesort-Algorithmus (vgl. Skript Folie 429 ff.)?

(1c) *(1 Punkt)* Laden Sie sich die Klassen `MergeSort` und `TestMergeSort` von der Vorlesungswebseite herunter. Die Klasse MergeSort entält die Methode `mergeSort`, die eine absteigende Sortierung des als Eingabe übergebenen Arrays durchführt. Fügen Sie in der Klasse `TestMergeSort` geeignete Anweisungen ein, die Ihnen erlauben, die von Ihrem Rechner benötigte Zeit für die Sortierung eines Arrays von n Elementen zu messen[1].

(1d) *(2 Punkte)* Messen Sie nun die von Ihrem Rechner benötigte Laufzeit für die Sortierung eines Arrays von n Elementen. Lassen Sie n die Werte $1000, 5000, 25000, 50000, 100000, 500000$ und 1000000 annehmen und notieren Sie die Messergebnisse in der untenstehenden Tabelle. (*Hinweis: führen Sie jede Messung mehrmals aus und notieren Sie sich deren Mittelwert.*)

n	Laufzeit (in ms)
1000	
5000	
25000	
50000	
100000	
500000	
1000000	

(1e) *(2 Punkte)* Zeichnen Sie den Ablauf der oben gemessenen Laufzeit im Interval $n = [1000, 1000000]$. Stimmen Ihrer Messergebnisse mit der theoretischen Zeitkomplexität des Mergesort-Algorithmus (vgl. Teilaufgabe *(b)*) überein? Begründen Sie Ihre Antwort.

2. Aufgabe: (8 Punkte) Türme von Hanoi

In der Vorlesung wurde das Problem der "Türme von Hanoi" vorgestellt (Vorlesungsskript Folien 413-423) und eine rekursive Lösung angegeben.

(2a) *(1 Punkt)* Für jeden Schritt bei der Ausführung des rekursiven Algorithmus aus der Vorlesung (Folie 418) wird genau ein Turm nicht benötigt. Geben Sie für die 15 Schritte bei der Umschichtung eines Turms der Höhe 4 die Folge der Nummern derjenigen Türme an, die *nicht* angefasst werden.

(2b) *(2 Punkte)* Was fällt Ihnen auf? Können Sie aus dem Ergebnis aus Teilaufgabe *a)* eine Idee für einen kinderleichten nicht-rekursiven Algorithmus für das Problem entwickeln? Beschreiben Sie diesen Algorithmus in Pseudocode.

(2c) *(3 Punkte)* Implementieren Sie in Java (in einer eigenen Klasse `Hanoi`) den von Ihnen in Teilaufgabe *(2b)* entwickelten Algorithmus. Versehen Sie Ihren Code mit geeigneten Testausgaben, damit Ihre Implementierung einfach getestet werden kann.

[1] Benutzen Sie dazu die Methode `currentTimeMillis()` der Klasse `System`.

(2d) *(2 Punkte)* Kann man Ihre Implementierung aus Teilaufgabe *(c)* auch zur Lösung des Problems der Türme von Hanoi verwenden, falls der Anfangsturm die Höhe 5 besitzt? Begründen Sie Ihre Antwort.

3. Aufgabe: (8 Punkte) Reversi [Teil 4]

(3a) *(8 Punkte)* Realisieren Sie eine Bewertungsfunktion, die nach dem α-β-Verfahren arbeitet, ansonsten jedoch das gleiche leistet wie die reine MiniMax-Methode des letzten Übungsblattes.

(3b) *(Unbewertete Aufgabe)* Prüfen Sie für grössere Suchbäume, ob es sich lohnt, die Unterbäume eines Knotens so zu sortieren, dass die Wahrscheinlichkeit für α-β-Schnitte maximiert wird.

(3c) *(Unbewertete Aufgabe)* Analysieren Sie Spielbäume mit dem MiniMax-Verfahren und dem α-β-Verfahren. Wieviel Zeit gewinnen Sie mit dem α-β-Verfahren bzw. wieviele Spielbaumebenen können Sie in gleicher Zeit zusätzlich untersuchen?

Summe: 24 Punkte