Informatik II: Algorithmen und Datenstrukturen SS 2017

Vorlesung 1b, Mittwoch, 26. April 2017

(MergeSort, Divide and Conquer, Rekursion)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Code aus dem /public Verzeichnis im SVN

    Diese Code dürfen Sie grundsätzlich verwenden wie Sie möchten, siehe ausführliche Post dazu auf dem **Forum**
- Inhalt
  - MergeSort: ein besserer Sortieralgorithmus

    Wir betrachten eine iterative und eine rekursive Variante

    Für das ÜB1 sollen Sie die iterative Variante nehmen
  - Divide and Conquer: das allgemeine Prinzip dahinter
  - Rekursion: kurze Wiederholung für die, die es wieder vergessen oder noch nie richtig verstanden haben

MergeSort 1/6

- Mischen (Merge), Definition
  - Wir betrachten erstmal folgendes Teilproblem:
  - Gegeben zwei sortierte Folgen A und B
  - Berechne eine sortiere Folge C mit den Elementen aus A und B
  - Das ist einfacher als das Ganze neu zu sortieren, weil A und B ja schon sortiert sind

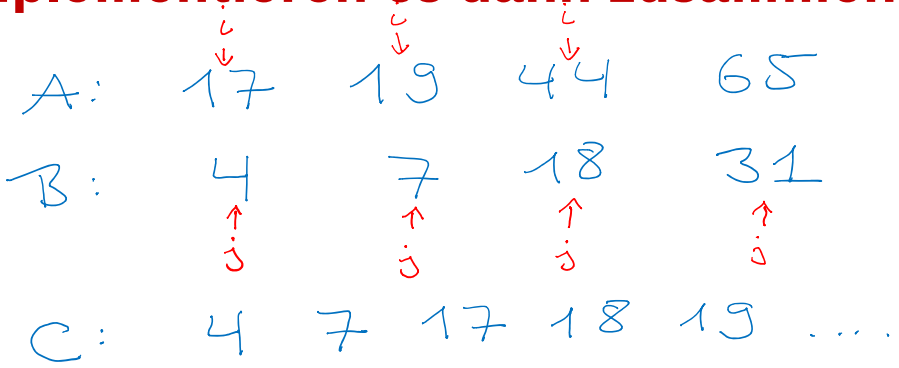
A: 17 19 44 65

B: 4 7 18 31

C: 4 7 17 18 19 31 44 65

MergeSort 2/6

- Mischen, Algorithmus
  - Wir merken uns in jedem Feld eine Position (i und j)
  - Zu Beginn i = j = 0
  - Jetzt gehen wir immer in dem Feld weiter nach rechts wo das kleinere Element von A[i] und B[j] steht
  - **Das schauen wir uns erst an einem Beispiel an und implementieren es dann zusammen**



MergeSort 3/6

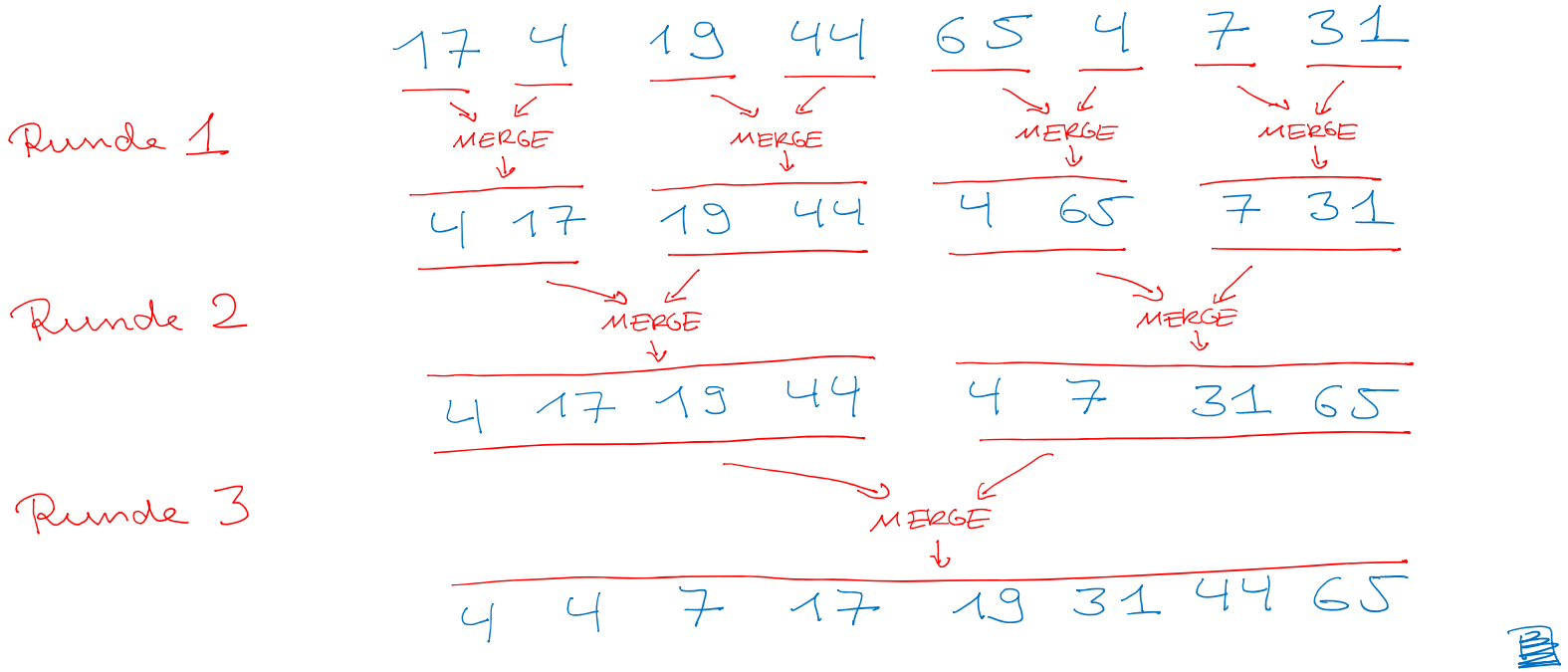
die sind trivial sortiert

- MergeSort, iterativer Algorithmus
  - **Grundidee:** wir benutzen Mischen, um aus kleinen sortierten Teilfelder größere sortierte Teilfelder zu machen
  - Wir beginnen mit Teilfeldern der Größe 1, die wir paarweise zu sortierten Teilfeldern der Größe 2 mischen
  - Diese sortierten Teilfelder der Größe 2 mischen wir dann paarweise zu sortierten Teilfelder der Größe 4
  - Und so weiter … bis das ganze Feld sortiert ist
  - Beispiel auf der nächsten Folie, mit **n = Zweierpotenz**

    Für das ÜB1 müssen Sie sich überlegen, wie es auch für beliebige n (= nicht unbedingt eine Zweierpotenz) geht

MergeSort 4/6

- MergeSort, iterativer Algorithmus, Beispiel



MergeSort 5/6

- MergeSort, rekursiver Algorithmus
  - Dasselbe Prinzip lässt sich auch rekursiv anwenden:

    Teile das Eingabefeld in zwei (fast) gleichgroße Teile

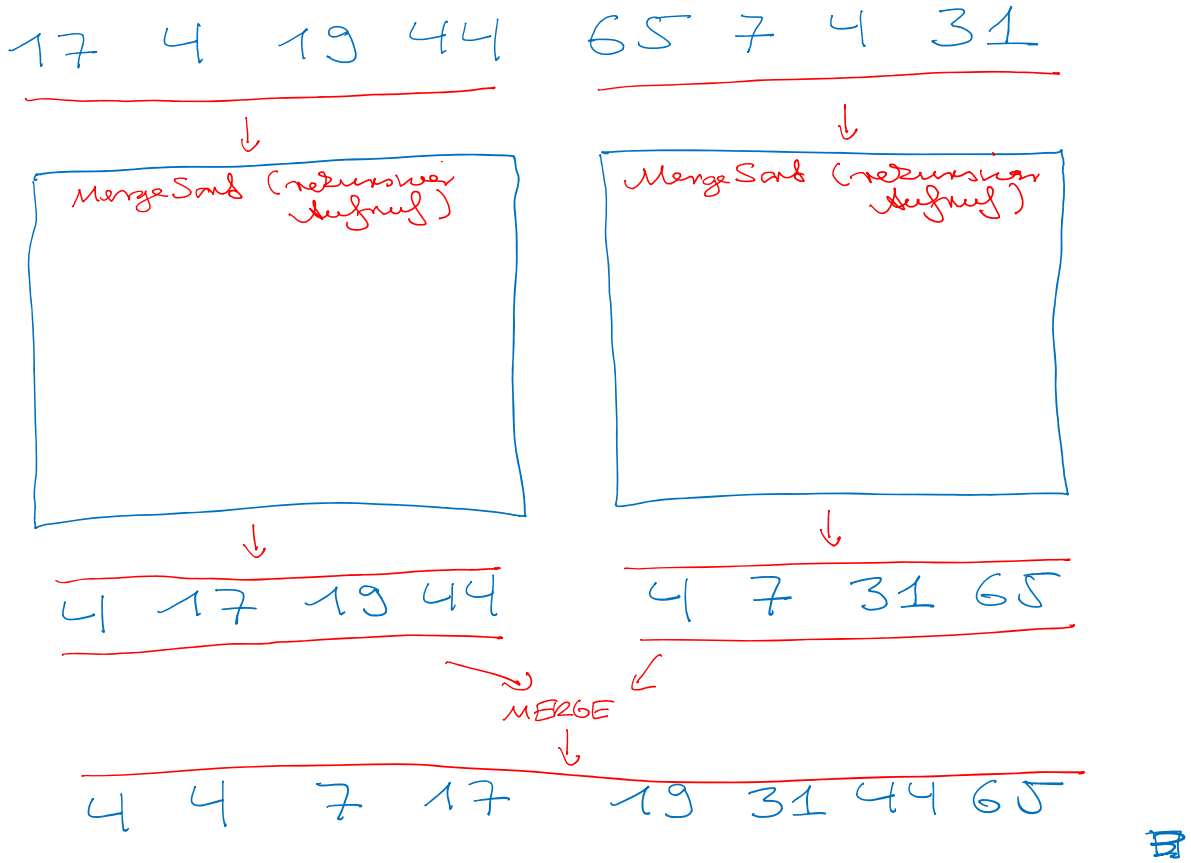
    Für jedes der beiden Teilfelder: falls größer als 1, sortiere das Teilfeld rekursiv (= mit derselben Funktion)

    Mische die beiden sortieren Hälften
  - Von der Implementierung her ist das **einfacher**
  - Aber es ist schwerer zu verstehen, was konkret passiert und es ist auch leichter (doofe) Fehler zu machen

    Deswegen bleiben wir für das ÜB1 erstmal bei der (aufwändigeren aber klareren) iterativen Variante

MergeSort 6/6

- MergeSort, rekursiver Algorithmus, Beispiel



Divide and Conquer 1/2

UNI FREIBURG

- Allgemeines Prinzip
  - Teile das gegebene Problem in kleinere Teile
  - Setze Lösungen für größere Teile aus Lösungen für kleinere Teile zusammen
  - Zwei Eigenschaften sind notwendig, damit das klappt:

    1. Das Zusammensetzen von Teillösungen ist einfacher, als das Problem von Grund auf zu lösen

    2. Das Problem ist für kleinste Teile einfach(er) zu lösen
  - Das führt in der Regel zu einer **rekursiven** Funktion
  - Man kann es aber auch (immer) **iterativ** implementieren
  - Je nach Implementierung, kann die rekursive oder die iterative Variante schneller sein

Divide and Conquer 2/2

- Etymologie
  - Latein: Divide et impera
  - Deutsch: Teile und herrsche
  - Französisch: Diviser pour régner
  - Spanisch: Divide y vencerás
  - Griechisch: Διαίρει και βασίλευε
  - Usw.
  - Das kommt eigentlich aus der Kriegsführung

    So wie der Campus hier übrigens auch … wir versuchen, das Beste daraus zu machen

Rekursion 1/2

- Ist im Prinzip ganz einfach
  - Eine Funktion ruft sich als Teil Ihres Codes selber wieder auf
  - Man muss nur darauf achten, dass das Ganze irgendwann aufhört, so wie das hier z.B. **nicht**

```
def eternalDamnation():
    eternalDamnation()
```

Rekursion 2/2

1, 1, 2, 3, 5, 8, 13, ...
F_1 F_2 F_3 F_4 F_5 F_6 F_7 ...

- Ein einfaches Beispiel
  - **Fibonacci-Zahlen** ... die sind ja rekursiv definiert:

    $F_1 = 1,\ F_2 = 1,\ F_n = F_{n-1} + F_{n-2}$ für alle $n \geq 3$
  - Die programmieren wir jetzt rekursiv ... und lassen uns dabei zum Verständnis die Rekursionsstufen ausgeben
  - Die rekursive Implementierung ist rein zu Lehrzwecken und sehr ineffizient, aus **zwei** Gründen:

    1. Durch die doppelte Rekursion wird derselbe Wert viele Male berechnet → iterativ wäre hier effizienter

    2. Es gibt außerdem eine geschlossene Formel für F_n mit einer konstanten Anzahl von Operationen

goldene Schnitt

$$F_n = \frac{1}{\sqrt{5}}\left(\varphi^n - (1-\varphi)^n\right) \quad \dots \quad \varphi = \frac{1+\sqrt{5}}{2} = 1.61803\dots$$

Literatur / Links

- MergeSort
  - Wikipedia: Merge sort
  - Mehlhorn/Sanders: 5.2 Mergesort
- Divide and Conquer
  - Wikipedia: Divide and conquer algorithm
  - Mehlhorn/Sanders: über das ganze Buch verteilt