

SATZ 5.17 (Stabilität)

Mergesort und Countingsort sind stabil, Quicksort je nach Umsetzung der Pivotregel.

Beweis: Selbst!

Damit:

ALGORITHMUS 5.18

INPUT:  $n$  Zahlen mit je  $d$  Ziffern, die  $k$  verschiedene Werte annehmen können,  $A[1], \dots, A[n]$

OUTPUT: Sortierter Array

RADIX-SORT ( $A, d$ )

1 FOR ( $i=1$ ) TO  $d$

1.1 DO Countingsort für  $A$  auf Stelle  $i$

SATZ 5.19

Radixsort liefert ein korrektes Ergebnis in Zeit  $\Theta(d(n+k))$ .

Beweis:

Für je zwei Zahlen muss am Ende die Reihenfolge stimmen; diese entspricht der Ordnung der höchsten Ziffern, die verschieden sind. Die Sortierung nach diesen Ziffern liefert also die richtige Reihenfolge, die sich wegen der Stabilität nicht mehr ändert.

Die Laufzeit folgt aus Satz 5.15.

□

## S.7.3 Spaghettisort

(52)

Nicht ganz ernst gemeint!

### ALGORITHMUS S.20

Input:  $n$  Zahlen  $A[1], \dots, A[n]$

Output: Sortierter Array

1. FOR ( $i=1$ ) TO  $n$ 
  - 1.1 DO (schneide ein Spaghettio der Länge  $A[i]$  zurecht)
2. Stoße alle Spaghettis auf den Tisch
3. FOR ( $i=1$ ) TO  $n$ 
  - 3.1 DO (wähle die längste verbleibende Nudel)

„Analoger Algorithmus“ - nicht digital!

### „SATZ“ S.21

Algorithmus S.20 sortiert in Linearzeit!

Beweis: Scheint klar!

Problematisch: Rechnermodell, Rechenoperationen, Speicherplatz, Ausführbarkeit (und Genauigkeit!) für große  $n$ ...

# 5.8 Paralleles Sortieren

Idee: Nicht nur einzelne Paare können verglichen werden, sondern viele (disjunkte) Paare zugleich!

Dazu zunächst sequentiell:

## ALGORITHMUS 5.22 (Bubblesort)

INPUT: Array  $A[1], \dots, A[n]$

OUTPUT: Sortierter Array

```

1 FOR (i=1) TO n
  1.1 DO FOR (j=n) DOWNTO i+1
    1.1.1 DO IF ( $A[j] < A[j-1]$ )
      1.1.1.1 Vertausche  $A[j]$  und  $A[j-1]$ 

```

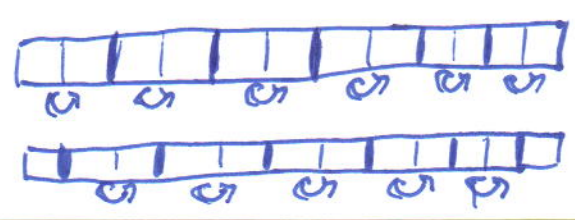
## SATZ 5.23

Bubblesort ist ein korrektes, stabiles Sortierverfahren. Die Laufzeit ist  $\Theta(n^2)$ .

Beweis: Einfach!

Also nicht so toll...

Aber: Lässt sich gut parallelisieren!



ALGORITHMUS 5.24 (Paralleles Bubblesort / odd-even-sort)

INPUT: n Zahlen  $A[1], \dots, A[n]$   
OUTPUT: Sortierter Array

- 1 FOR (i=1) TO n
  - 1.1 vergleiche und ggf. tausche gerade Paare parallel
  - 1.2 vergleiche und ggf. tausche ungerade Paare parallel

SATZ 5.25

Paralleles Bubblesort sortiert die Zahlen in n parallelen Runden.  
Die Gesamtzahl der Vergleiche ist  $\Theta(n^2)$ .

Beweis: Selbst!

Das funktioniert auch zwei- und mehrdimensional!

5.9 Unruste Sortierverfahren

ALGORITHMUS 5.26 (Bogosort)

INPUT: Array  $A[1], \dots, A[n]$   
OUTPUT: Sortierter Array

- 1 WHILE (Array unsortiert)
  - 1.1 Permutiere zufällig
  - 1.2 Überprüfe Sortiertheit

SATZ 5.27

Bogosort hat erwartete Laufzeit  $O(n \cdot n!)$ .

Beweis: Überprüfen dauert  $O(n)$ , die erwartete Zahl der Runden ist  $O(n!)$ , weil die Wahrscheinlichkeit in jeder Runde  $\frac{1}{n!}$  ist, die sortierte Reihenfolge zu erhalten.

Noch schlimmer:

ALGORITHMUS 5.28 (Bogobogosort)

INPUT:  $n$  Zahlen  $A[1], \dots, A[n]$

OUTPUT: Sortierter Array

- 1.  $i := 2$
- 2. WHILE ( $i \leq n$ ) DO
  - 2.1 wähle Zufallspermutation der Elemente  $A[i], \dots, A[n]$
  - 2.2 IF (Permutation ist sortiert)
    - 2.2.1  $i := i + 1$
  - 2.3 ELSE
    - 2.3.1  $i := 2$

SATZ 5.29

Bogobogosort hat erwartete Laufzeit  $O\left(n \cdot \prod_{i=2}^n i!\right)$ .

Beweis:

Jede Permutation hat Wahrscheinlichkeit  $\frac{1}{i!}$ , sortiert zu sein. Man braucht eine Glückssträhne für alle  $i = 2, \dots, n$

In einem geeigneten Berechnungsmodell lässt sich das aber trotzdem parallel in Linearzeit implementieren! (56)

### ALGORITHMUS 5.30 (Quantum Bogosort)

INPUT :  $n$  Zahlen  $A[1], \dots, A[n]$   
OUTPUT : Sortierter Array in geeignetem Universum

1. Permutiere zufällig  $\leftarrow$  Quantenbits
2. ~~IF~~ IF (Permutation unsortiert)
  - 2.1 zerstöre Universum

### SATZ 5.31

Quantum Bogosort liefert ein Universum, in dem nach  $O(n)$  alle Zahlen als sortiert verifiziert sind.

Beweis:

- Erfordert :
- Quantenphysikalisches Weltmodell  
( $\rightarrow$  Zufallsereignisse ergeben Parallelwelten)
  - Mechanismus zur Zerstörung des Universums in  $O(1)$

Rest selbst! (Bitte nicht experimentell...)

□

### KOROLLAR 5.32

Quantum Bogobogosort liefert ein Universum, in dem nach  $O(n^2)$  alle Zahlen als sortiert verifiziert sind.