

# Algorithmen in Zellularautomaten

## 9. Sortieren in zweidimensionalen ZA

Thomas Worsch

Fakultät für Informatik  
Karlsruher Institut für Technologie

Sommersemester 2017

## Ziele

Problemstellung: Sortieren von Quadraten mit  $\sqrt{n} \times \sqrt{n}$  Werten

- ▶ Was ist ein sortiertes Quadrat?
- ▶ Algorithmus Shearsort – in Zeit  $\Theta(\sqrt{n} \log n)$
- ▶ Algorithmus von Schnorr und Shamir – in Zeit  $\Theta(\sqrt{n})$

# Überblick

- Problemdefinition
- Shearsort
- Algorithmus von Schnorr und Shamir

# Was ist zweidimensionales Sortieren?

## Was ist zweidimensionales Sortieren?

|     |    |    |    |    |    |   |          |    |    |    |    |    |   |
|-----|----|----|----|----|----|---|----------|----|----|----|----|----|---|
| So: | 1  | 2  | 3  | 4  | 5  | ? | oder so: | 1  | 2  | 6  | 7  | 15 | ? |
|     | 6  | 7  | 8  | 9  | 10 |   |          | 3  | 5  | 8  | 14 | 16 |   |
|     | 11 | 12 | 13 | 14 | 15 |   |          | 4  | 9  | 13 | 17 | 22 |   |
|     | 16 | 17 | 18 | 19 | 20 |   |          | 10 | 12 | 18 | 21 | 23 |   |
|     | 21 | 22 | 23 | 24 | 25 |   |          | 11 | 19 | 20 | 24 | 25 |   |

|          |    |    |    |    |   |   |          |    |    |    |    |    |   |
|----------|----|----|----|----|---|---|----------|----|----|----|----|----|---|
| oder so: | 1  | 2  | 3  | 4  | 5 | ? | oder so: | 1  | 2  | 3  | 4  | 5  | ? |
|          | 16 | 17 | 18 | 19 | 6 |   |          | 10 | 9  | 8  | 7  | 6  |   |
|          | 15 | 24 | 25 | 20 | 7 |   |          | 11 | 12 | 13 | 14 | 15 |   |
|          | 14 | 23 | 22 | 21 | 8 |   |          | 20 | 19 | 18 | 17 | 16 |   |
|          | 13 | 12 | 11 | 10 | 9 |   |          | 21 | 22 | 23 | 24 | 25 |   |

## Problemstellung

- ▶  $A$  endlich und total geordnet.
- ▶ **Gesucht:** zweidimensionaler Zellularautomat, der jedes Muster  $e : \mathbb{N}_m \times \mathbb{N}_m \rightarrow A$  in das sortierte Muster  $s : \mathbb{N}_m \times \mathbb{N}_m \rightarrow A$  überführt, wobei gilt:
  1. Für alle  $a \in A$  ist  $N_a(e) = N_a(s)$ .
  2. Falls  $i \in \mathbb{N}_m$  ungerade, gilt für  $j \in \mathbb{N}_{m-1}$ :  $s(i, j) \leq s(i, j + 1)$ .
  3. Falls  $i \in \mathbb{N}_m$  gerade, gilt für  $j \in \mathbb{N}_{m-1}$ :  $s(i, j) \geq s(i, j + 1)$ .
  4. Falls  $i \in \mathbb{N}_{m-1}$  ungerade, gilt  $s(i, n) \leq s(i + 1, n)$ .
  5. Falls  $i \in \mathbb{N}_{m-1}$  gerade, gilt  $s(i, 1) \leq s(i + 1, 1)$ .
- ▶ Das ist „Sortieren in Schlangenlinienform“.

## Ideen?

Wie könnte man in Schlangenlinienform sortieren?

# Überblick

- Problemdefinition
- Shearsort
- Algorithmus von Schnorr und Shamir



## Algorithmus (Shearsort)

$1 + 2 \lceil \log \sqrt{n} \rceil$  Phasen:

## Algorithmus (Shearsort)

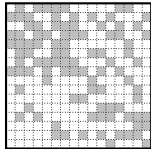
$1 + 2 \lceil \log \sqrt{n} \rceil$  Phasen:

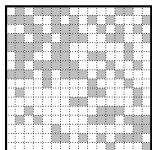
**ungerade Phasen:** jede *Zeile* für sich sortieren, und zwar

- ▶ ungerade Zeilen nach rechts aufsteigend und
- ▶ gerade Zeilen nach links aufsteigend.

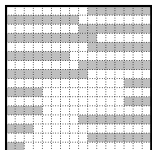
**gerade Phasen:** jede *Spalte* für sich sortieren, und zwar

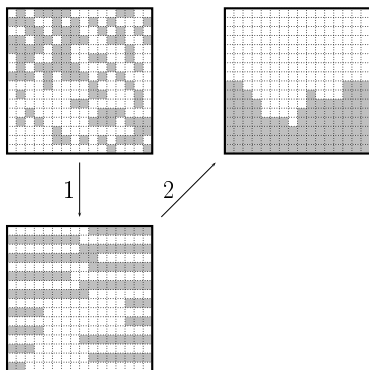
- ▶ kleine Werte nach oben.

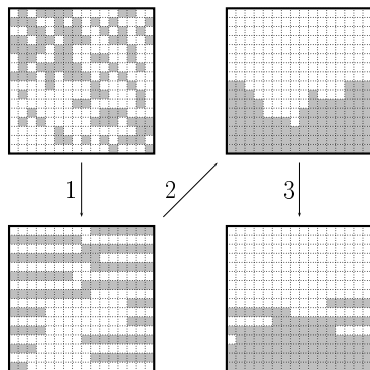


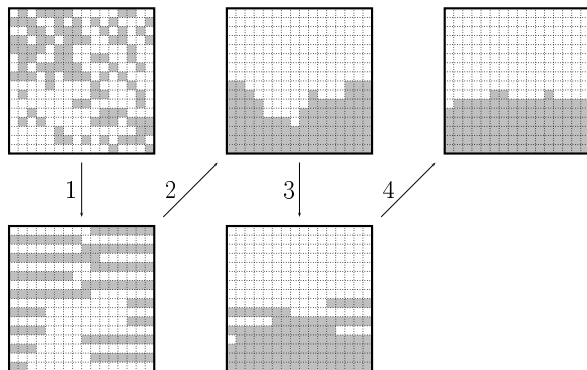


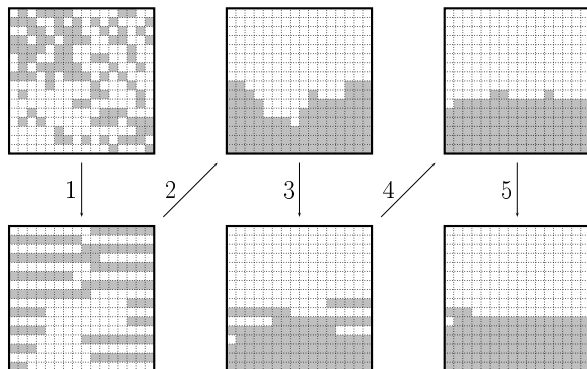
1 ↓



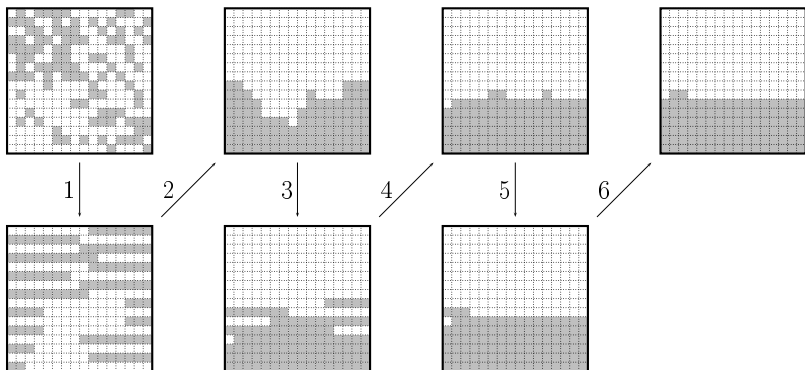


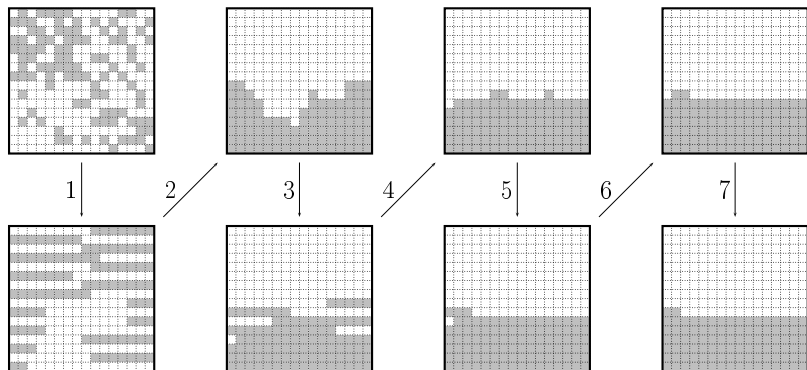












nach vorne

## Lemma

Algorithmus Shearsort leistet das Gewünschte und benötigt  $O(\sqrt{n} \log n)$  Schritte.

## Lemma

Algorithmus Shearsort leistet das Gewünschte und benötigt  $O(\sqrt{n} \log n)$  Schritte.

### **Gleich:**

Es ist *nicht* selbstverständlich, dass der Algorithmus funktioniert!

## Beweis

Das 0-1-Sortierlemma ist anwendbar.

Betrachte eine Konfiguration nach einer geraden Anzahl von Phasen:

[zur Erinnerung klicke man hier](#)

Das 0-1-Sortierlemma ist anwendbar.

Betrachte eine Konfiguration nach einer geraden Anzahl von Phasen:

zur Erinnerung klicke man hier

- ▶ ganz oben einige *reine* 0-Zeilen
- ▶ dann einige *gemischte* Zeilen
- ▶ ganz unten einige *reine* 1-Zeilen

**Zeige:**

Das 0-1-Sortierlemma ist anwendbar.

Betrachte eine Konfiguration nach einer geraden Anzahl von Phasen:

zur Erinnerung klicke man hier

- ▶ ganz oben einige *reine* 0-Zeilen
- ▶ dann einige *gemischte* Zeilen
- ▶ ganz unten einige *reine* 1-Zeilen

**Zeige:** Durch zwei aufeinanderfolgende Phasen wird die Anzahl gemischter Zeilen mindestens halbiert.

## Korrektheit von Shearsort (1)

Betrachte zwei aufeinanderfolgende Zeilen.

Nach Zeilensortierphase eine der folgenden Situationen  
(oder Spiegelbild davon):

$$\begin{array}{c|c|c} 0 \cdots 01 \cdots 11 \cdots 1 & 0 \cdots 01 \cdots 1 & 0 \cdots 00 \cdots 01 \cdots 1 \\ 1 \cdots 11 \cdots 10 \cdots 0 & 1 \cdots 10 \cdots 0 & 1 \cdots 10 \cdots 00 \cdots 0 \end{array}$$

Anschließend erfolgt eine Spaltensortierung:

$$\begin{array}{c|c|c} 0 \cdots 01 \cdots 10 \cdots 0 & 0 \cdots 00 \cdots 0 & 0 \cdots 00 \cdots 00 \cdots 0 \\ 1 \cdots 11 \cdots 11 \cdots 1 & 1 \cdots 11 \cdots 1 & 1 \cdots 10 \cdots 01 \cdots 1 \end{array}$$

Die Anzahl  $x$  der gemischten Zeilen sinkt um mindestens  $\frac{x-1}{2}$ .



## Korrektheit von Shearsort (2)

- ▶ anfangs maximal  $\sqrt{n}$  gemischte Zeilen.

## Korrektheit von Shearsort (2)

- ▶ anfangs maximal  $\sqrt{n}$  gemischte Zeilen.
- ▶ nach  $\lceil \log \sqrt{n} \rceil$  Zeilen- und Spaltenphasen

## Korrektheit von Shearsort (2)

- ▶ anfangs maximal  $\sqrt{n}$  gemischte Zeilen.
- ▶ nach  $\lceil \log \sqrt{n} \rceil$  Zeilen- und Spaltenphasen
  - ▶ höchstens eine gemischte Zeile übrig,

## Korrektheit von Shearsort (2)

- ▶ anfangs maximal  $\sqrt{n}$  gemischte Zeilen.
- ▶ nach  $\lceil \log \sqrt{n} \rceil$  Zeilen- und Spaltenphasen
  - ▶ höchstens eine gemischte Zeile übrig,
  - ▶ die in der letzten Zeilenphase sortiert wird.

## Korrektheit von Shearsort (2)

- ▶ anfangs maximal  $\sqrt{n}$  gemischte Zeilen.
- ▶ nach  $\lceil \log \sqrt{n} \rceil$  Zeilen- und Spaltenphasen
  - ▶ höchstens eine gemischte Zeile übrig,
  - ▶ die in der letzten Zeilenphase sortiert wird.
- ▶ Jede Phase kann in  $\Theta(\sqrt{n})$  Schritten erledigt werden.

## Korrektheit von Shearsort (2)

- ▶ anfangs maximal  $\sqrt{n}$  gemischte Zeilen.
- ▶ nach  $\lceil \log \sqrt{n} \rceil$  Zeilen- und Spaltenphasen
  - ▶ höchstens eine gemischte Zeile übrig,
  - ▶ die in der letzten Zeilenphase sortiert wird.
- ▶ Jede Phase kann in  $\Theta(\sqrt{n})$  Schritten erledigt werden.
- ▶ Insgesamt Zeitbedarf also  $\Theta(\sqrt{n} \log \sqrt{n}) = \Theta(\sqrt{n} \log n)$ .

---

# Ist die Korrektheit von Shearsort nicht trivial?

Ist die Korrektheit von Shearsort nicht trivial?

**Nein!**

**Nein!**

**Nein!**

**Nein!**

**Nein!**

**Nein!**



## Ist die Korrektheit von Shearsort nicht trivial?

**Nein!    Nein!    Nein!    Nein!    Nein!    Nein!**

- ▶ Ändere Shearsort so ab, dass  
*in allen Zeilen in die gleiche Richtung* aufsteigend sortiert wird.
- ▶ Was passiert (nicht)?

## Ist die Korrektheit von Shearsort nicht trivial?

**Nein!    Nein!    Nein!    Nein!    Nein!    Nein!**

- ▶ Ändere Shearsort so ab, dass  
*in allen Zeilen in die gleiche Richtung* aufsteigend sortiert wird.
- ▶ Was passiert (nicht)?
- ▶ Man erhält *nicht* einen Algorithmus, der so sortiert, dass die  
Konkatenation aller Zeilen ein sortiertes Feld ergibt.
- ▶ Gegenbeispiel:

|   |   |
|---|---|
| 1 | 8 |
| 2 | 9 |

# Überblick

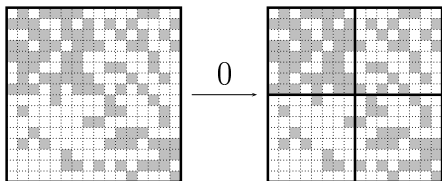
- Problemdefinition
- Shearsort
- Algorithmus von Schnorr und Shamir

## Algorithmus (von Schnorr und Shamir)

- ▶ Anzahl der zu sortierenden Elemente:  $n = k^8$
- ▶ quadratische Muster der Größe  $m \times m = \sqrt{n} \times \sqrt{n} = k^4 \times k^4$
- ▶ Algorithmus in neun Phasen

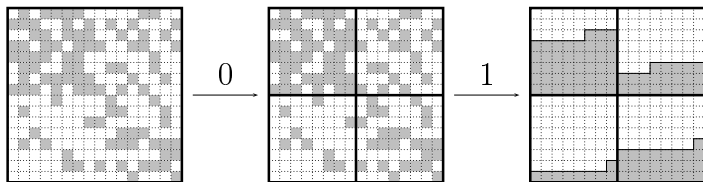
## Phase 0

Teile das Muster in  $k \times k$  *Blöcke* der Größe  $k^3 \times k^3$ . ( $k = \sqrt[4]{m} = \sqrt[8]{n}$ )



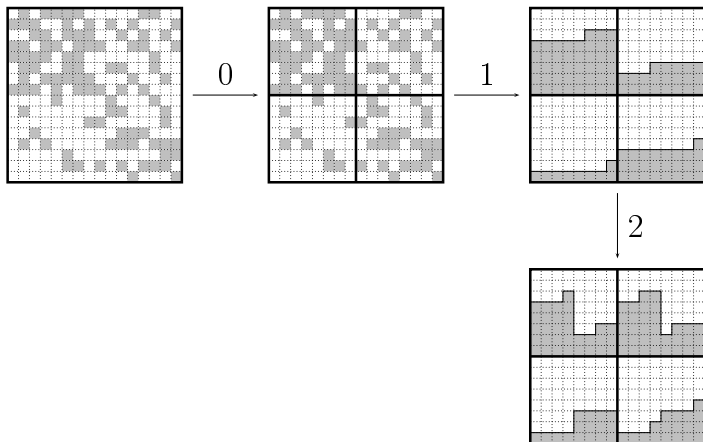
## Phase 1

Sortiere jeden Block in Schlangenlinienform.



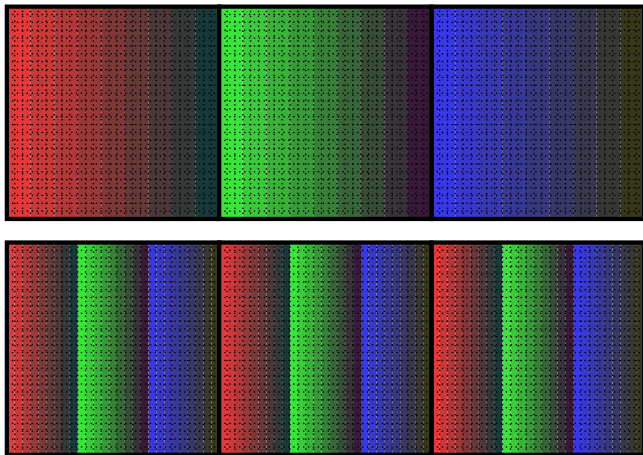
## Phase 2

Verteile die vollen Spalten jeder Spalte von Blöcken gleichmäßig auf alle Spalten von Blöcken.



## Spaltenumsortierung

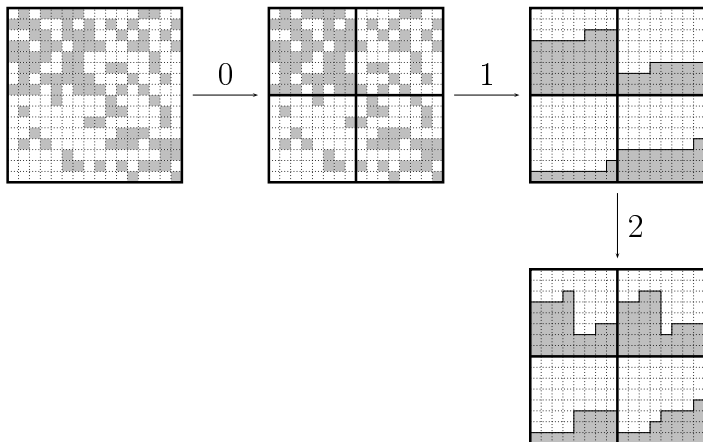
für eine volle Zeile von Blöcken im Fall  $k = 3$





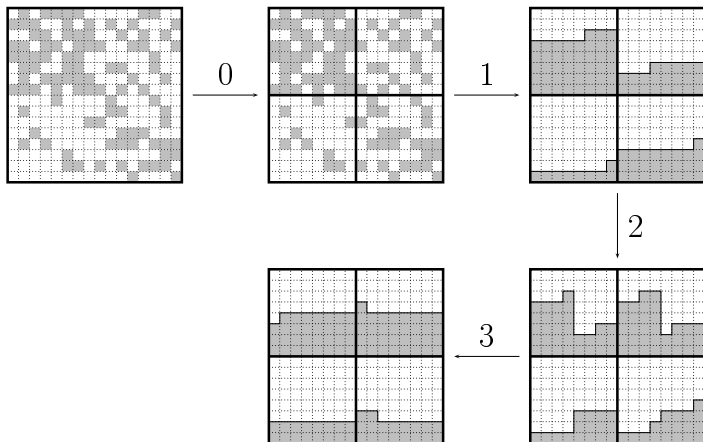
## Phase 2

Verteile die vollen Spalten jeder Spalte von Blöcken gleichmäßig auf alle Spalten von Blöcken.



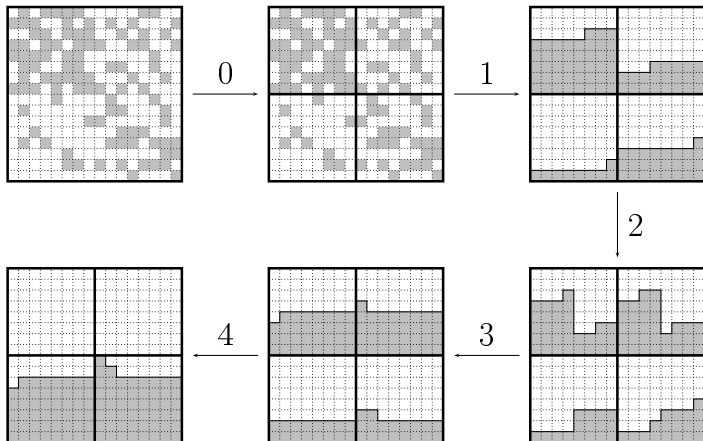
## Phase 3

Sortiere jeden Block in Schlangenlinienform.



## Phase 4

Sortiere jede volle Spalte (kleine Werte nach oben).



## Phase 5

Sortiere in jeder Spalte von Blöcken die Blöcke  $1 + 2, 3 + 4, \dots$  jeweils gemeinsam in Schlangenlinienform.

## Phase 6

Sortiere in jeder Spalte von Blöcken die Blöcke  $2 + 3, 4 + 5, \dots$  jeweils gemeinsam in Schlangenlinienform.

## Phase 7

Sortiere jede volle Zeile (in ungeraden kleine Werte nach links, in geraden nach rechts)

## Phase 8

Führe auf der „gesamten Schlangenlinie“  $2k^3$   
„Odd-Even-Transposition-Sort-Schritte“ durch.

## Satz

Der Algorithmus von Schnorr und Shamir sortiert quadratische  $\sqrt{n} \times \sqrt{n}$ -Muster in  $\Theta(\sqrt{n})$  Schritten.

Hier sollte endgültig klar sein, wie nützlich Knuths 0-1-Sortierlemma ist!



## Bewei(s)

**Korrektheit:** Das 0-1-Sortierlemma ist anwendbar.

Es gilt nach

- Phase 1:** In jedem Block gibt es höchstens eine gemischte Zeile, d.h. zwei Spalten des gleichen Blockes unterscheiden sich um höchstens eine 1.
- Phase 2:** Da jeder Block aus jedem anderen (der gleichen Zeile von Blöcken) mindestens eine Spalte bekommen hat, unterscheiden sich zwei Blöcke der gleichen Zeile von Blöcken in der Anzahl Einsen um höchstens  $k$ .
- Phase 3:** Da  $k < k^3$  ist, gibt es in jeder Zeile von Blöcken höchstens zwei gemischte volle Zeilen.
- Phase 4:** In jeder Spalte von Blöcken gibt es höchstens  $k$  gemischte Zeilen.

## Beweis (2)

### Korrektheit:

**Phasen 5+6:** Jede Spalte von Blöcken in Schlangenlinienform.  
Da in Phasen 3–6 kein Austausch zwischen verschiedenen Spalten von Blöcken, gilt wie schon nach Phase 2:  
Je zwei ganze Spalten von Blöcken unterscheiden sich in der Anzahl Einsen um höchstens  $k^2$ .  
Wegen  $k^2 < k^3$  gibt es folglich nach Phase 6 insgesamt nur noch höchstens zwei gemischte volle Zeilen.

## Beweis (3)

### Korrektheit:

**Phase 7:** nur noch höchstens zwei gemischte volle Zeilen: die eine enthält lauter Nullen und am „höheren“ Ende höchstens  $k^2 \cdot k = k^3$  Einsen, die andere lauter Einsen und am „niedrigeren“ Ende höchstens  $k^2 \cdot k = k^3$  Nullen. Es bedarf also  $\leq k^3 + k^3$  Sortierschritten entlang der großen Schlangenlinie um den Rest zu sortieren. Daher gilt nach

**Phase 8:** Alles ist sortiert.

## Beweis (4)

**Zeitbedarf** bei Verwendung von Shearsort für die Blöcke:

$$k^4 + k^3 \cdot \log k^3 + k^4 + k^3 \cdot \log k^3 + k^4 \\ + k^3 \cdot \log k^3 + k^3 \cdot \log k^3 + k^4 + k^3 \in \Theta(\sqrt{n})$$

da  $k^4 = \sqrt{n}$ .

## Beweis (5)

### Implementierungsdetails

- ▶ Umschaltung zwischen Phasen: FSSP
- ▶ Umverteilung der vollen Spalten in Phase 2: Übungsaufgabe  
man mache das hinreichend schnell, i. e. in Zeit  $O(k^4)$

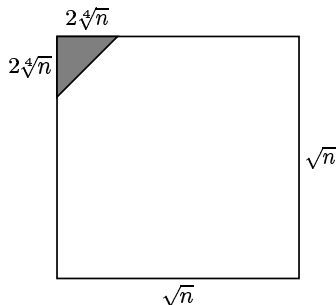
## Lemma

Jeder Algorithmus für das zweidimensionale Sortieren in Schlangenlinienform benötigt mindestens  $3\sqrt{n} - o(\sqrt{n})$  Schritte, sofern man beliebig viele verschiedene Werte sortieren können will.

## Beweis (1)

Betrachte folgende Aufteilung des Quadrates:

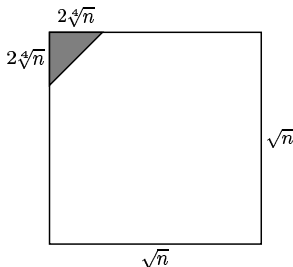
- ▶ graues Dreieck in linker oberer Ecke mit Kathetenlänge  $2\sqrt[4]{n}$ ; umfasst (diskret)  $2\sqrt{n} + \sqrt[4]{n}$ , also mehr als  $2\sqrt{n}$  Zellen
- ▶ Jeder Weg von einer grauen Zelle zur unteren rechten Ecke hat Länge  $\geq 2\sqrt{n} - 2\sqrt[4]{n}$  (bei  $H_1^1$ -Nachbarschaft)
- ▶ der Rest umfasst  $n - 2\sqrt{n} - \sqrt[4]{n}$  Zellen



## Beweis (2)

Betrachte anfängliche Werteverteilung

- ▶ im grauen Dreieck nur die Werte 0 und  $n$   
welcher wie oft, überlegen wir uns später noch ...
- ▶ im Rest jeder der Werte 1 bis  $n - 2\sqrt{n} - \sqrt[4]{n}$  genau einmal
- ▶ Wert  $x$ : der, der sich nach  $2\sqrt{n} - 2\sqrt[4]{n} - 3$  Schritten im rechten unteren Eck befindet
- ▶ Wo stand  $x$  am Anfang?

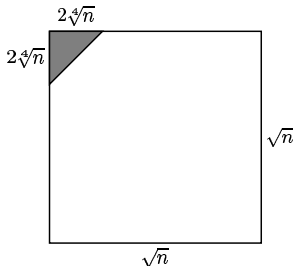




## Beweis (2)

Betrachte anfängliche Werteverteilung

- ▶ im grauen Dreieck nur die Werte 0 und  $n$   
welcher wie oft, überlegen wir uns später noch ...
- ▶ im Rest jeder der Werte 1 bis  $n - 2\sqrt{n} - \sqrt[4]{n}$  genau einmal
- ▶ Wert  $x$ : der, der sich nach  $2\sqrt{n} - 2\sqrt[4]{n} - 3$  Schritten im rechten unteren Eck befindet
- ▶ Wo stand  $x$  am Anfang? im Weißen, nicht nahe dem Grauen



## Beweis (3)

- ▶ Es sei  $C(m)$  die Nummer der Spalte, in der sich  $x$  *am Ende der Sortierung* befinden wird, wenn im grauen Dreieck zu Beginn genau  $m$  mal der Wert  $n$  vorkommt.
- ▶ Erhöhung von  $m$  bewirkt Änderung der Spaltennummer um 1 da  $0 < x < n$ . (Richtung?!)
- ▶ Dreiecksgröße ist  $2\sqrt{n}$ , also gibt es ein  $m'$ , so dass  $C(m') = 1$ .
- ▶ Für diese Wanderung werden noch weitere  $\sqrt{n} - 1$  Schritte benötigt.



## Zusammenfassung

- ▶ In eindimensionalen Zellularautomaten kann man  $n$  Datenelemente in Zeit  $O(n)$  sortieren.
- ▶ Im Zweidimensionalen ist die Schlangenlinie eine sinnvolle Sortierreihenfolge, weil in der Sortierung benachbarte Datenelemente am Ende auch in benachbarten Zellen liegen sollen.
- ▶ Sortieren von  $n = \sqrt{n} \times \sqrt{n}$  Elementen in Schlangenlinienform ist in Zeit  $O(\sqrt{n})$  möglich.
- ▶ Dabei ist  $3\sqrt{n} - o(\sqrt{n})$  untere Zeitschranke, sofern man beliebig viele verschiedene Werte sortieren können will.