Einführung in die Informationsverarbeitung

Øyvind Eide

Woche 9
Datenstrukturen
Algorithmen

oeide@uni-koeln.de
http://idh.uni-koeln.de

Inhaltliche Datenstrukturen

Datentypen allgemein

<Datentyp> ::=
ein 3-Tupel (oder Tripel) **{ E, I, O }**
wobei
E ::= Externe Darstellung
I ::= Interne Darstellung
O ::= Menge auf I definierter Operationen

(Notation: " ::= " = "definiert als ")

Datentyp Zeit allgemein

E Regel für "4.6.2007"

I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag i als Offset t vom Ursprung definiert ist.

O

t-less(i,j) ===> **Boolean**

t-less(4.6.2007,5.6.2007) ===> **True**

t-subtract(i,j) ===> **Ganze Zahl**

t-subtract(5.6.2007,4.6.2007) ===> 1

Datentyp „Historische Zeit“ I

E Regel für "pri non jun 2007"

I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag i als Offset t vom Ursprung definiert ist.

O

t-less(i,j) ===> **Boolean**

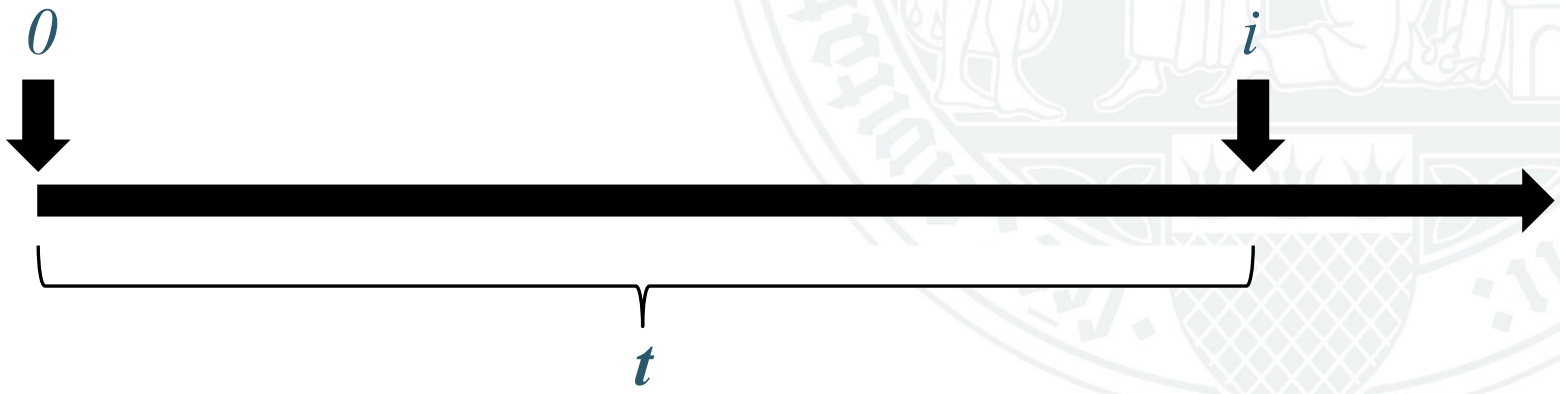
t-less(pri non jun 2007,non jun 2007) ===> **True**

t-subtract(i,j) ===> **Ganze Zahl**

t-subtract(non jun 2007,pri non jun 2007) ===> 1

Datentyp „Historische Zeit“ I

I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag *i* als Offset **t** vom Ursprung definiert ist.



Datentyp „Historische Zeit“ II

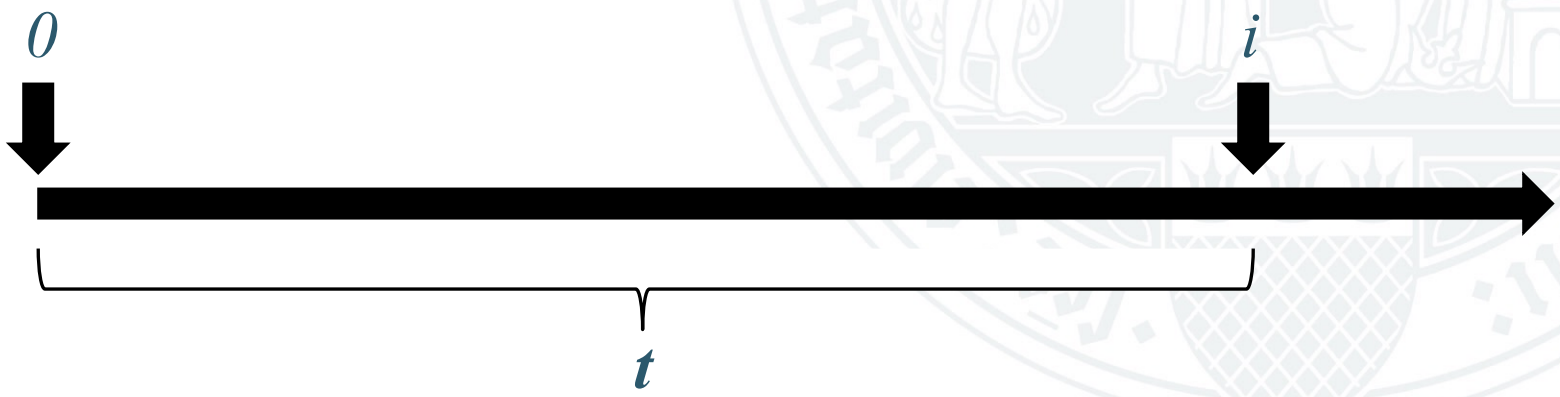
E Regel für "6 Tammuz 5763"
I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag i als Offset t vom Ursprung definiert ist.

O
t-less(i,j) ===> **Boolean**
t-less(6 Tammuz 5763,7 Tammuz 5763) ===> **True**

t-subtract(i,j) ===> **Ganze Zahl**
t-subtract(7 Tammuz 5763,6 Tammuz 5763) ===> 1

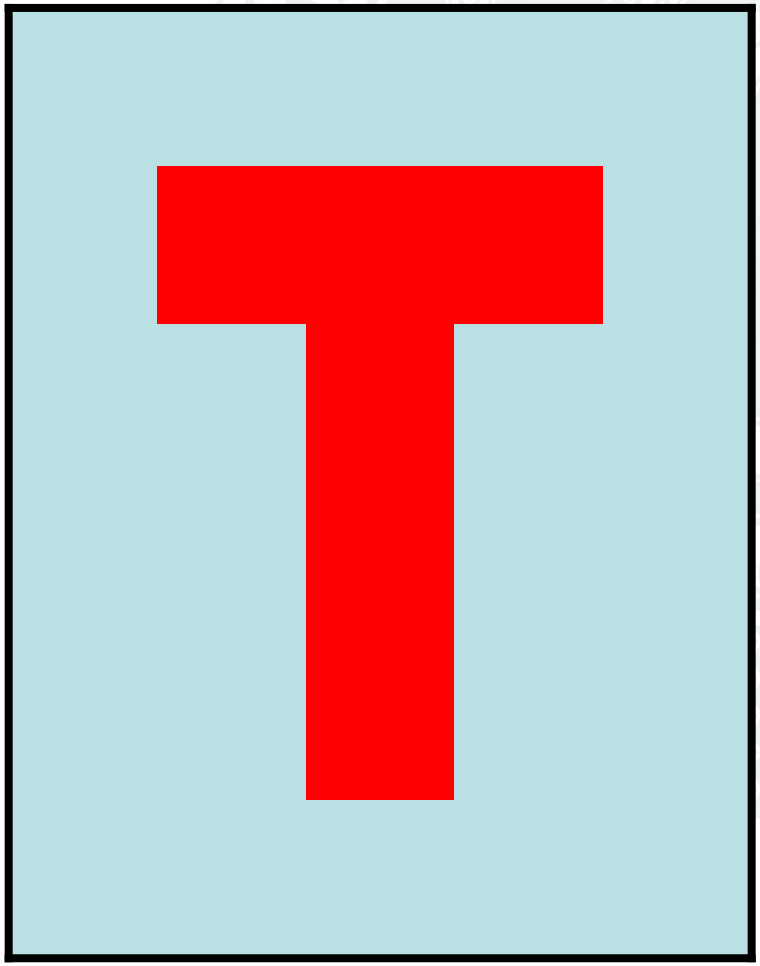
Datentyp „Historische Zeit“ II

I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag *i* als Offset **t** vom Ursprung definiert ist.



Datenstruktur aus dem Softwarengineering

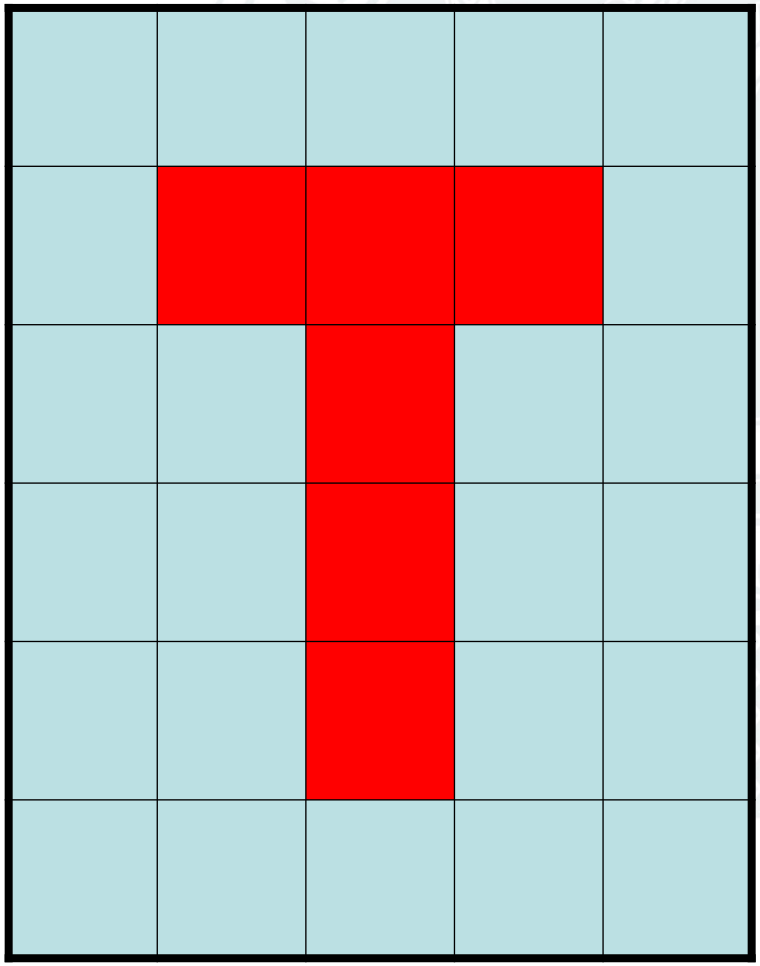
Ein Bild



Ein Bild

6 Zeilen

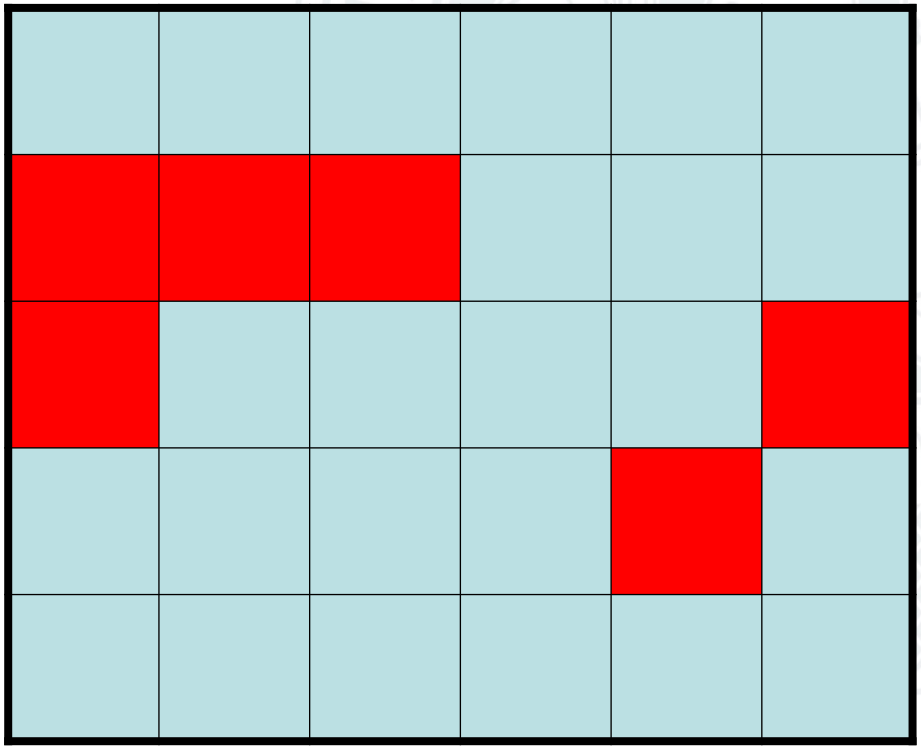
5 Spalten



Ein Bild

5 Zeilen

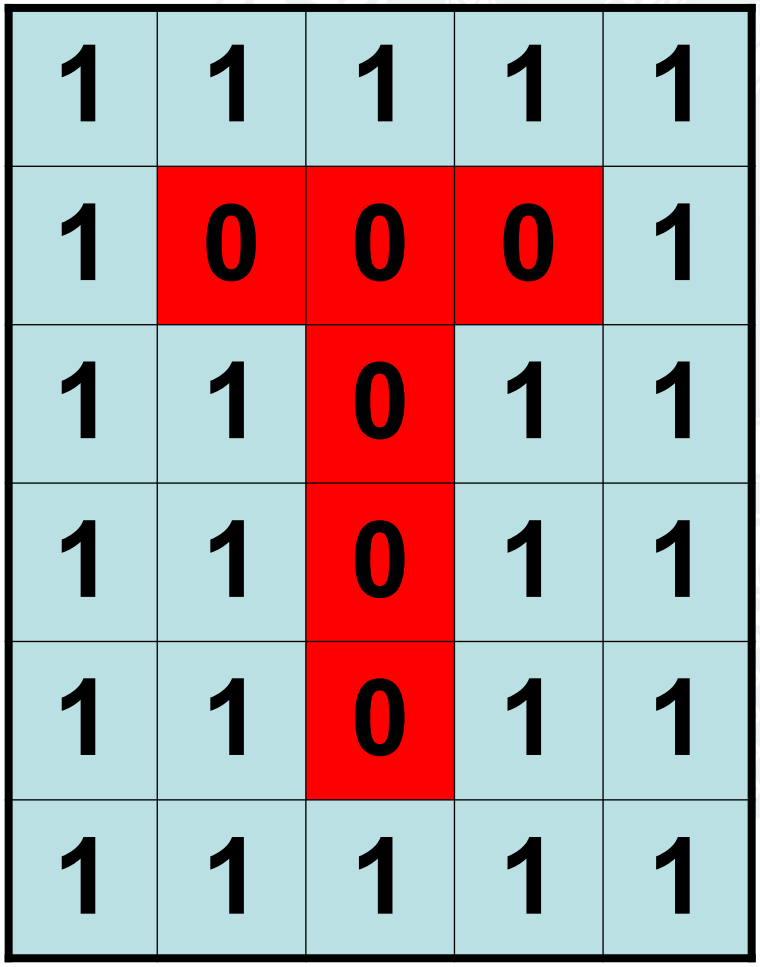
6 Spalten



Ein Bild

1 == grau

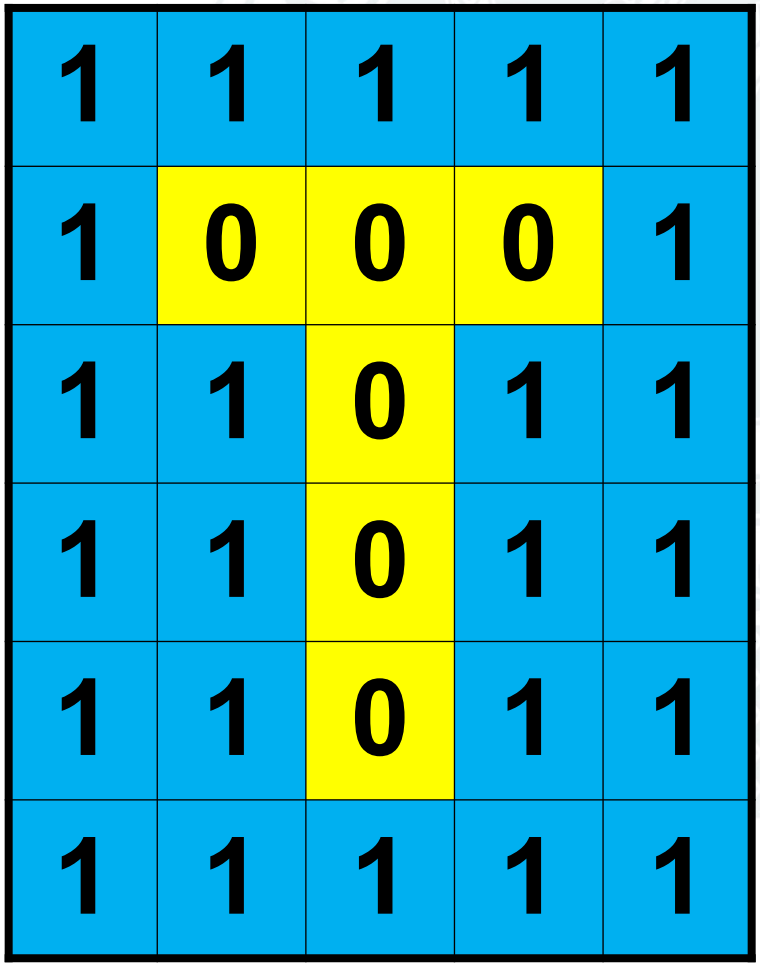
0 == rot



Ein Bild

1 == blau

0 == gelb

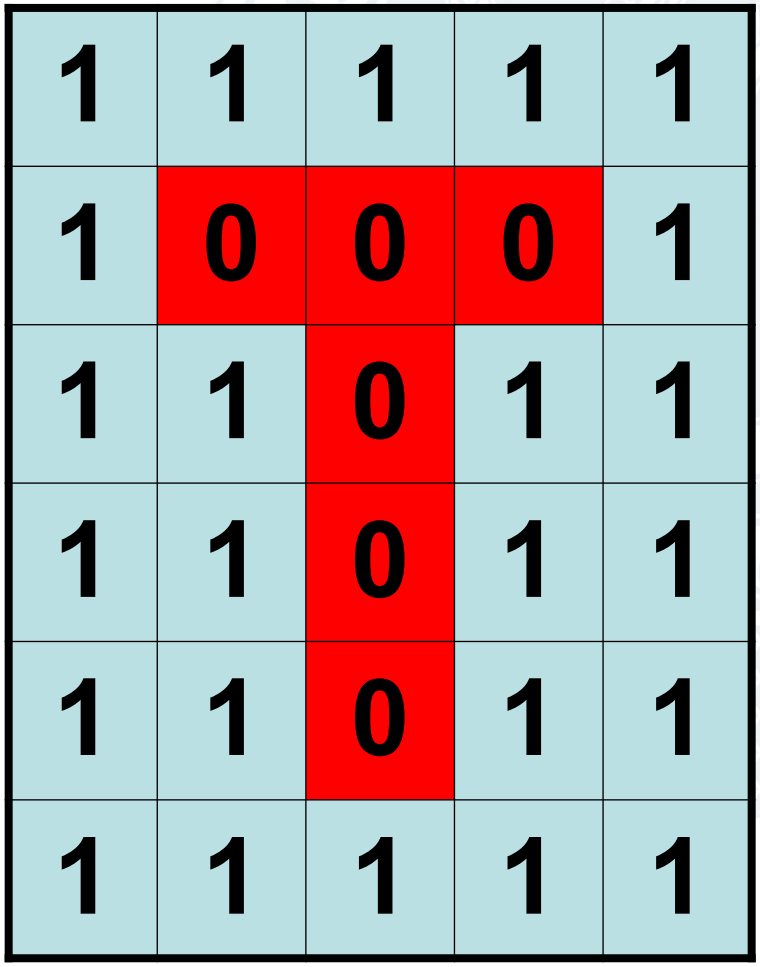


Ein Bild

Speicherung:

1,1,1,1,1,1,0,0,
0,1,1,1,0,1,1,1,
1,0,1,1,1,1,0,1,
1,1,1,1,1,1

Unkomprimiert



Ein Bild

Store:

6,1,3,0,3,1,1,0,
4,1,1,0,4,1,1,0,
7,1

(Compressed)

Run Length Encoded

1	1	1	1	1
1	0	0	0	1
1	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	1	1	1	1

Ein Bild

Speicherung:

SetSize: 5 by 6

SetBackgroundColor: Gray

SetForegroundColor: Red

SetLetterHeight: 4

MoveTo: 3,5

DrawLetter: T

Vector Format

1,1	2,1	3,1	4,1	5,1
1,2	2,2	3,2	4,2	5,2
1,3	2,3	3,3	4,3	5,3
1,4	2,4	3,4	4,4	5,4
1,5	2,5	3,5	4,5	5,5
1,6	2,6	3,6	4,6	5,6

Ein Bild

6 Zeilen

5 Spalten

1 == grau

0 == rot

Unkomprimiert

1	1	1	1	1
1	0	0	0	1
1	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	1	1	1	1

Ein Bild

Dimensionen

1 == gray
0 == red

Unkomprimiert

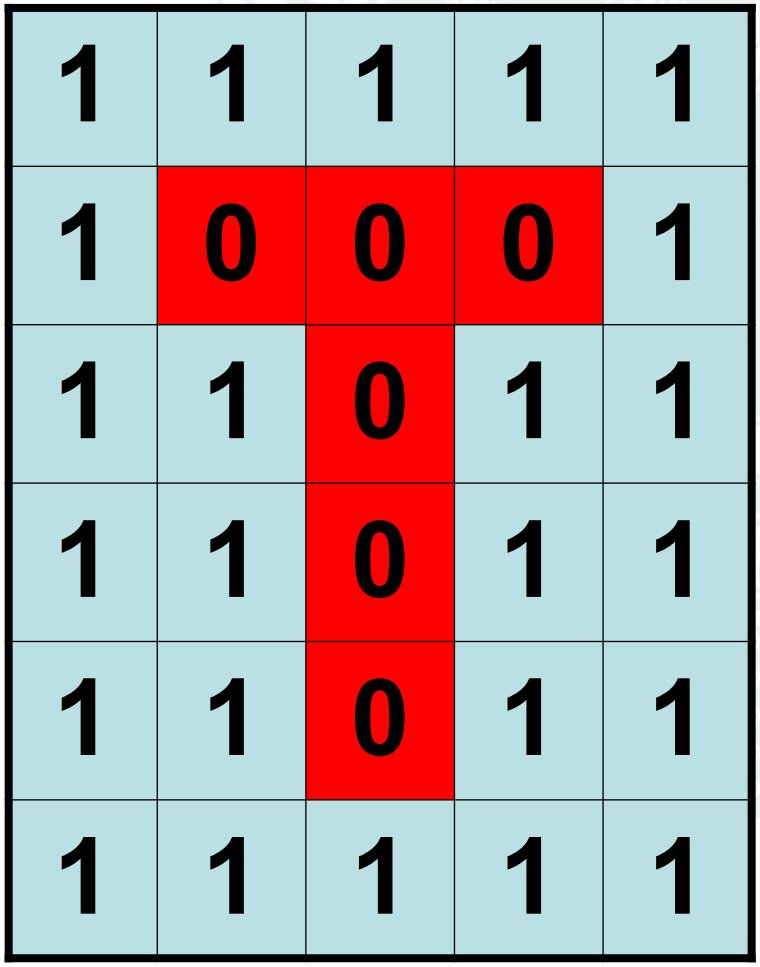
1	1	1	1	1
1	0	0	0	1
1	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	1	1	1	1

Ein Bild

Dimensionen

Photogrammetrische Interpretation

Unkomprimiert

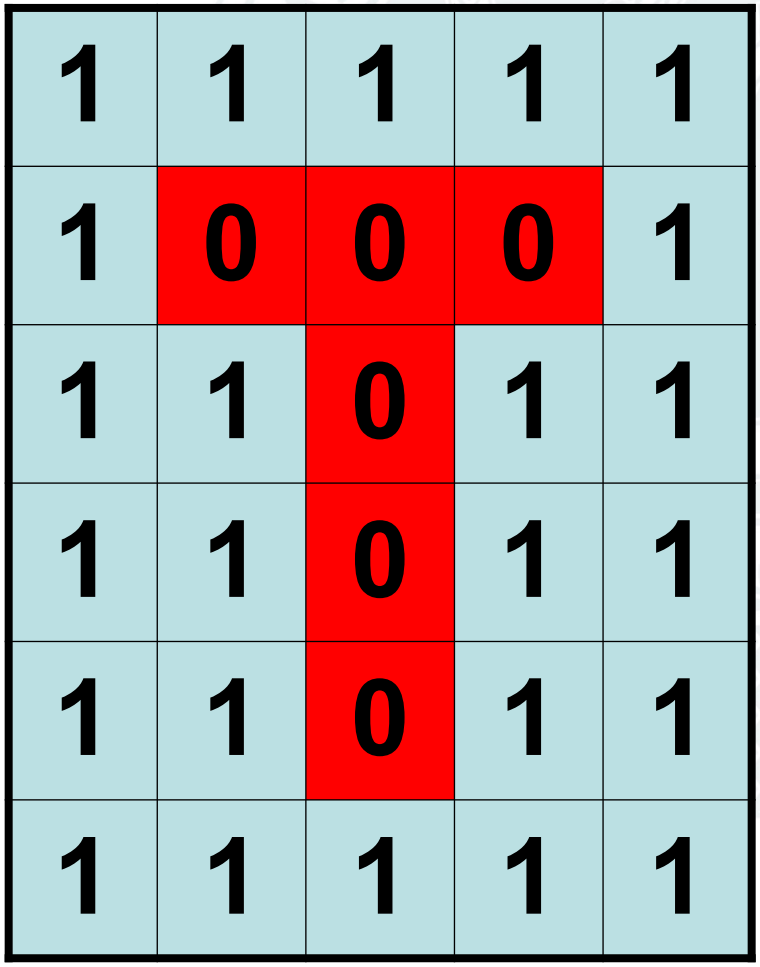


Ein Bild

Dimensionen

Photogrammetrische Interpretation

Kompressionstechnik

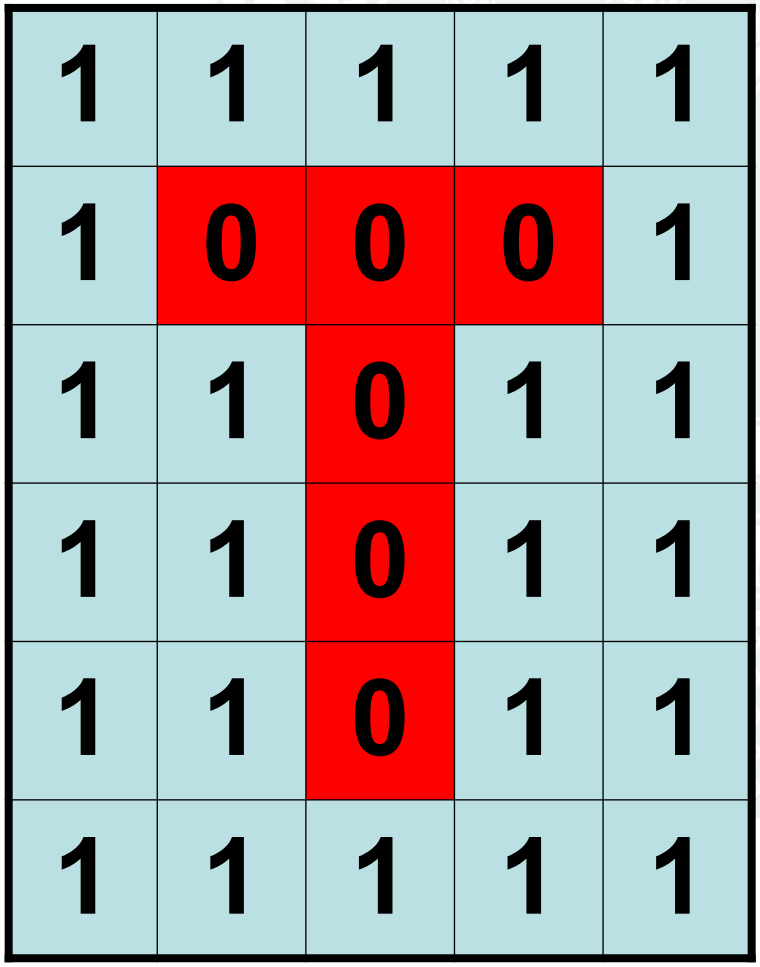


Ein Bild

<basic information>

<rendering information>

<storage information>



Ein Bild

<basic information>

(implizit / explizit)

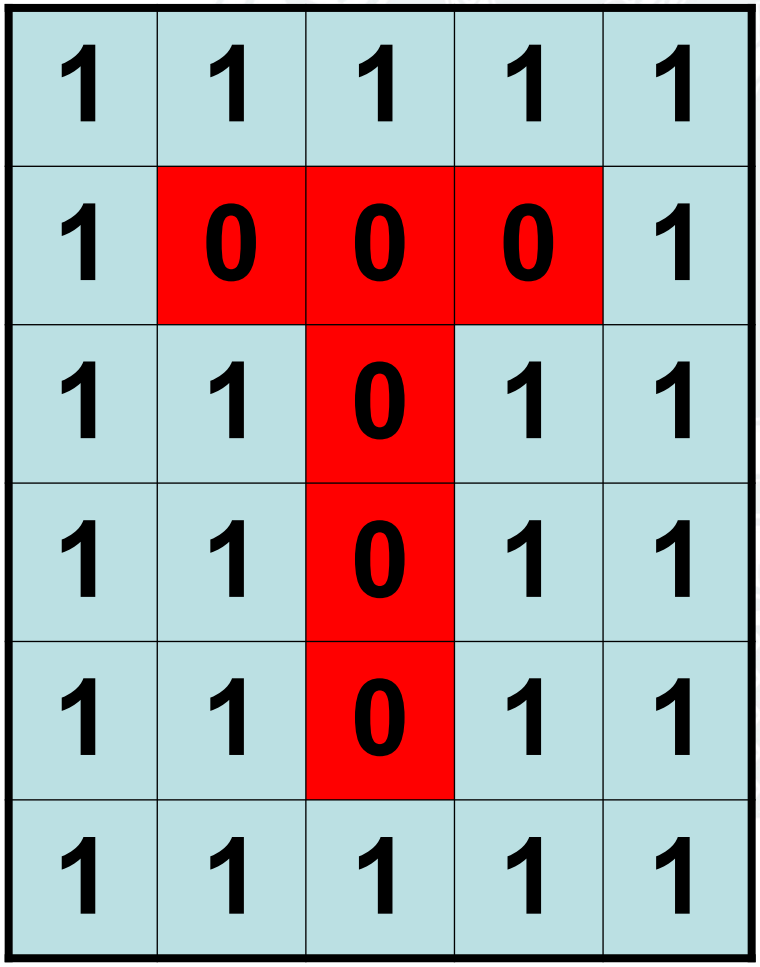
<rendering information>

(implicit / explicit)

<storage information>

(implicit / explicit)

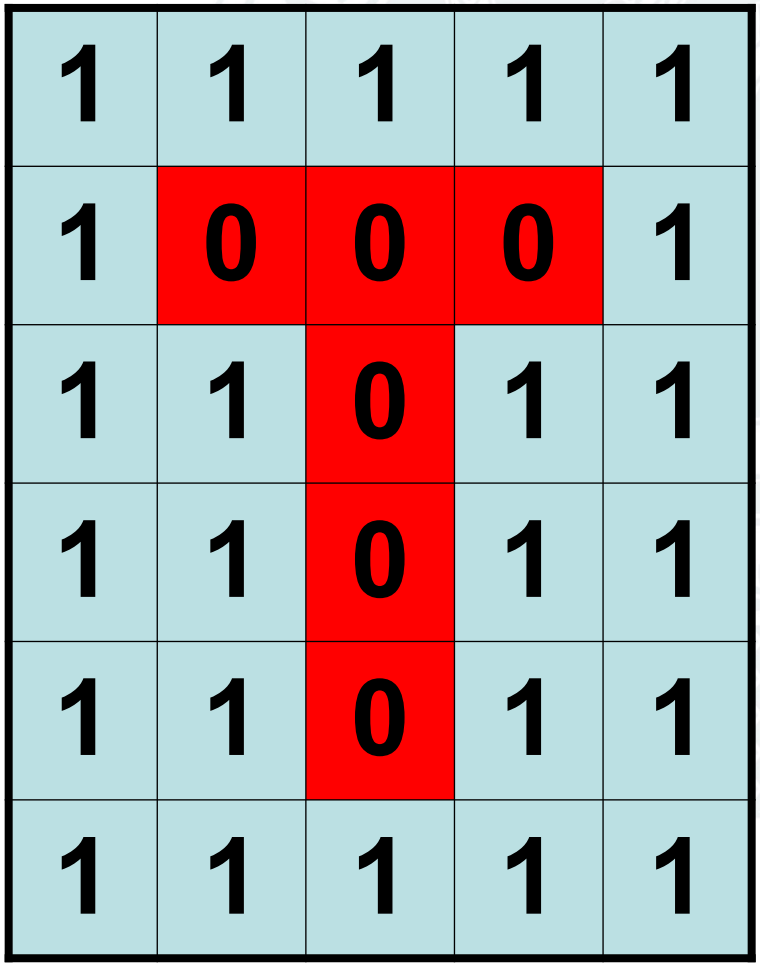
… und die Daten?



Ein Bild

Daten als Datenstrom

1,1,1,1,1,1,
0,0,0,1,1,1,
0,1,1,1,1,0,
1,1,1,1,0,1,
1,1,1,1,1,1,1



Ein Bild

Daten entweder als
Datenstrom
oder als
Verarbeitungsanweisungen

SetSize: 5 by 6
SetBackgroundColor: Ochre
SetForegroundColor: Red
SetLetterHeight: 4
MoveTo: 3,5
DrawLetter: T

1	1	1	1	1
1	0	0	0	1
1	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	1	1	1	1

Algorithmen

Warum automatisieren?

1 Million Objekte: eine Sekunde pro Stück.

== 16666.7 Minuten == 277.8 Stunden

== 11.57 Arbeitstage eines Computers

== 34.7 8-Stunden Tage eines Menschen

== 7 Arbeitswochen

Warum automatisieren?

1 Million Objekte: fünf Minuten pro Stück.

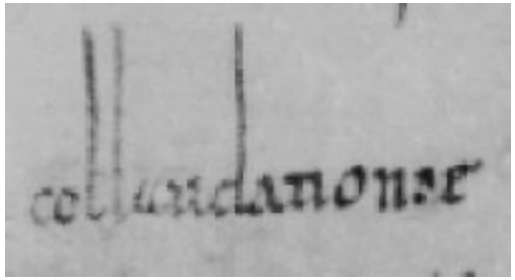
== 416 666.7 Stunden

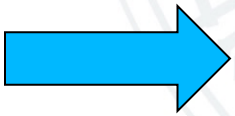
== 52 803.4 8-Stunden-Tage für einen Menschen

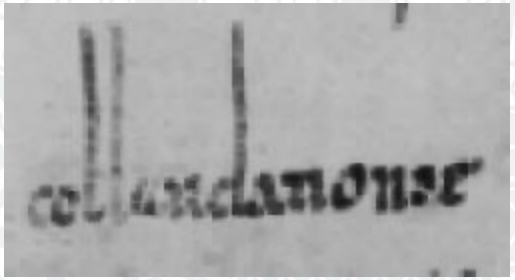
== Völlig lächerlich, dass das klappt

Einleitendes Beispiel (Selbstabbildende Information)

Minimal neighbour







Original

Ergebnis

Minimal neighbour

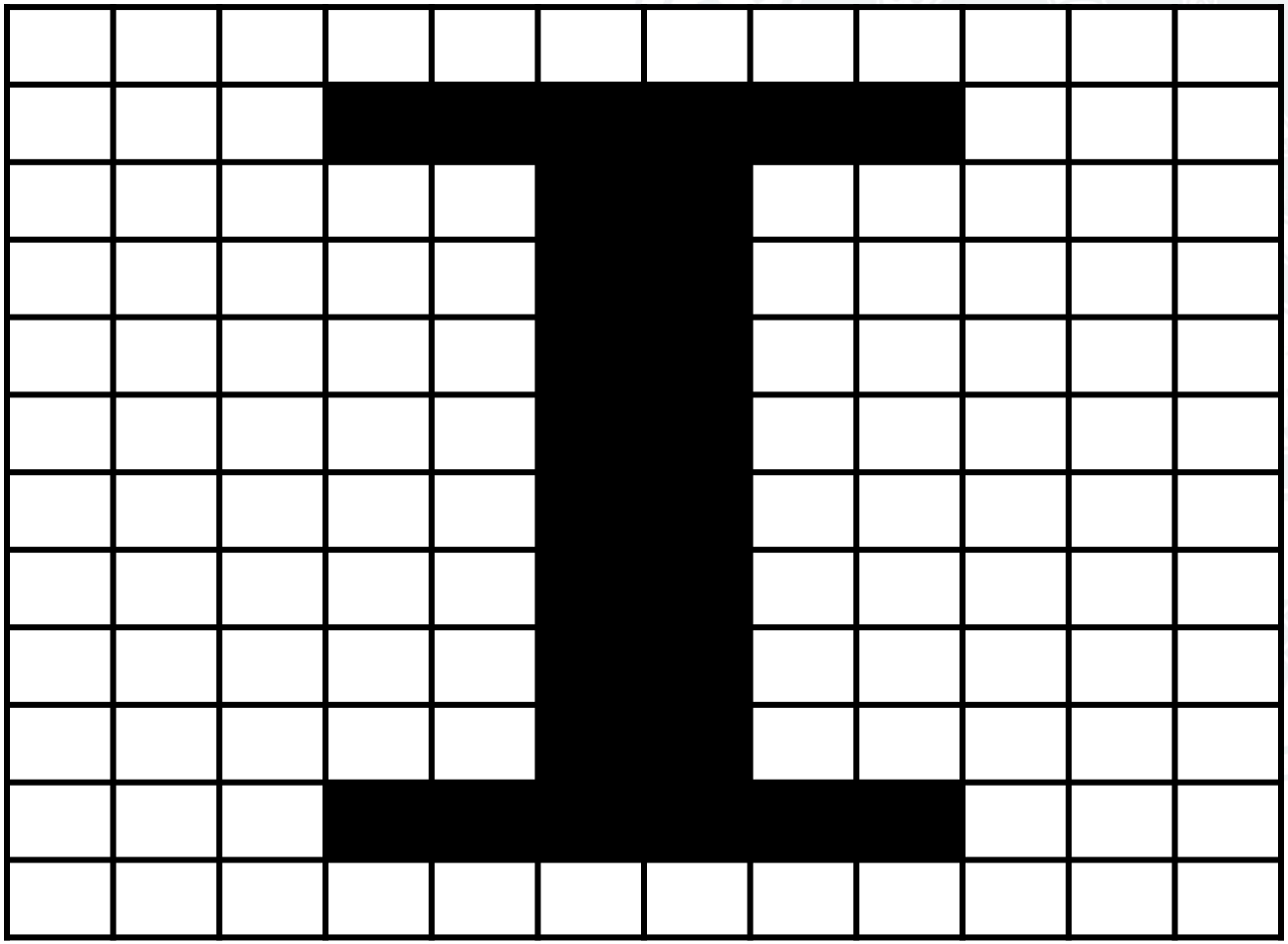
Ersetze in jeder Zeile

jedes Pixel

durch den niedrigsten Pixelwert der dieses

Pixels umschreibenden 3 x 3 Matrix.

Minimal neighbour



Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	50	50	50	50	50	50	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	250	250	50	50	250	250	250	250	250
250	250	250	50	50	50	50	50	50	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	50	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

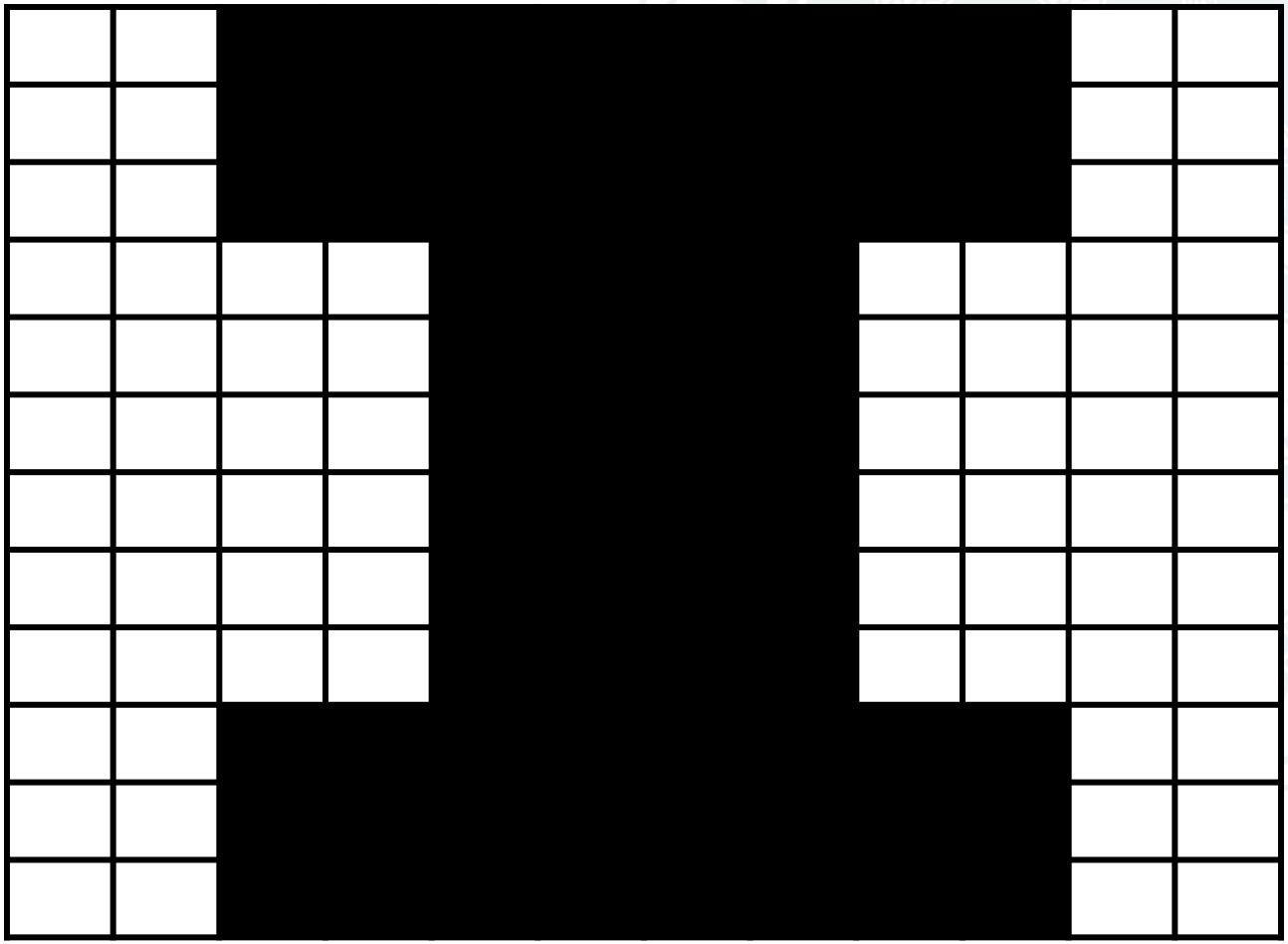
Minimal neighbour

250	250	250	250	250	250	250	250	250	250	250	250
250	250	250	50	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	250	250	**50**	**50**	250	250	250	250	250
250	250	250	**50**	**50**	**50**	**50**	**50**	**50**	250	250	250
250	250	250	250	250	250	250	250	250	250	250	250

Minimal neighbour

250	250	50	50	50	50	50	50	50	50	250	250
250	250	50	50	50	50	50	50	50	50	250	250
250	250	50	50	50	50	50	50	50	50	250	250
250	250	250	250	50	50	50	50	250	250	250	250
250	250	250	250	50	50	50	50	250	250	250	250
250	250	250	250	50	50	50	50	250	250	250	250
250	250	250	250	50	50	50	50	250	250	250	250
250	250	250	250	50	50	50	50	250	250	250	250
250	250	250	250	50	50	50	50	250	250	250	250
250	250	50	50	50	50	50	50	50	50	250	250
250	250	50	50	50	50	50	50	50	50	250	250
250	250	50	50	50	50	50	50	50	50	250	250

Minimal neighbour



Algorithmen: Allgemeine Eigenschaften

Algorithmen: Definition

Ein Algorithmus ist eine Funktion *f(Dein, Daus)*, die *Eingabedaten* D_{ein} in *Ausgabedaten* D_{aus} schrittweise transformiert und dabei bestimmte Bedingungen erfüllt.

Algorithmen: Eigenschaften

1. **Exaktheit.** Die Funktion f kann präzise auf formale Weise beschrieben werden.
2. **Finitheit.** Die Beschreibung von f ist endlich lang.
3. **Vollständigkeit.** Die Beschreibung von f umfasst alle vorkommenden Fälle.
4. **Effektivität.** Die Einzelschritte sind elementar und real ausführbar.
5. **Terminierung.** Die Funktion f hält nach endlich vielen Schritten an und liefert ein Resultat.
6. **Determinismus.** Die Funktion f liefert bei gleichen Eingabewerten stets das gleiche Ergebnis, wobei die Folge der Einzelschritte für jeden Eingabewert genau festgelegt ist.

Algorithmen: Laufzeit

1. linear.
2. logarithmisch.
3. exponentiell.

N=1	N=10	N=100	N=1000
1	10	100	1000
1	3	7	10
1	10^3	10^{30}	10^{300}

Algorithmen: Laufzeit

Beispiel: Sequentielles Suchen

Laufzeit: linear

1	Clio
2	Melpomene
3	Terpsichore
4	Thalia
5	Euterpe
6	Erato
7	Urania
8	Polyhymnia
9	Kalliope

Algorithmen: Laufzeit

Beispiel: Sequentielles Suchen

Suchzeit jedes Namens entspricht Rang in der Liste.

Durchschnittliche Suchzeit: n / 2.

Laufzeit steigt mit der zu durchsuchenden Anzahl

1	Clio
2	Melpomene
3	Terpsichore
4	Thalia
5	Euterpe
6	Erato
7	Urania
8	Polyhymnia
9	Kalliope

Algorithmen: Laufzeit

Beispiel: Binäres Suchen

Laufzeit: ?

1	Clio
2	Erato
3	Euterpe
4	Kalliope
5	Melpomene
6	Polyhymnia
7	Terpsichore
8	Thalia
9	Urania

Algorithmen: Laufzeit

Beispiel: Binäres Suchen – „Thalia“

„Melpomene“ gleich – größer – kleiner „Thalia“?

„Terpsichore“ gleich – größer – kleiner „Thalia“?

„Thalia“ gleich – größer – kleiner „Thalia“?

1	Clio
2	Erato
3	Euterpe
4	Kalliope
5	Melpomene
6	Polyhymnia
7	Terpsichore
8	Thalia
9	Urania

Algorithmen: Laufzeit

Beispiel: Binäres Suchen

Laufzeit steigt mit Logarithmus der zu durchsuchenden Anzahl.

1	Clio
2	Erato
3	Euterpe
4	Kalliope
5	Melpomene
6	Polyhymnia
7	Terpsichore
8	Thalia
9	Urania

Algorithmen: Sonderfälle

Nichtdeterministische Algorithmen: potentiell schneller - liefern u.U. keine Lösung

NP vollständige Algorithmen: Prinzipiell nicht möglich, irgendein NP-vollständiges Problem mit einem deterministischen Algorithmus in exponentieller Zeit zu lösen.

➢ Komplexitätstheorie.

Zeichenbasierte Algorithmen

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger
- Tegenberger
- Tekekenperger

Soundex

1 Das erste Zeichen jedes Namens wird beibehalten.

2 W und H werden ignoriert.

3 A, E, I, O, U und Y ergeben keinen Codewert, gelten jedoch als "Trenner" (s.Regel 5).

4 Die anderen Zeichen werden nach folgenden Regeln umgewandelt.

4.1	B, P, F, V	==>	1
4.2	C, G, J, K, Q, S, X, Z	==>	2
4.3	D, T	==>	3
4.4	L	==>	4
4.5	M, N	==>	5
4.6	R	==>	6

5 Ergeben zwei aufeinanderfolgende Zeichen denselben Code, wird er nur einmal gewertet. Sind sie durch einen "Trenner" (s. oben Regel 3) getrennt, wird er jedoch wiederholt.

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger
- Tegenberger
- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

T	Regel 1

- Tegenberger
- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

Tx	Regel 2

- Tegenberger
- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

T x	**Regel 3**

- Tegenberger

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

T 2	Regel 4,2

- Tegenberger

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

T 2x	**Regel 5**

- Tegenberger

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

T 2 x	**Regel 3**

- Tegenberger

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger

T 2 5	**Regel 4.5**

- Tegenberger

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**
- Tegenberger
- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**
- Tegenberger

T	**Regel 1**

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**

- Tegenberger

Tx	**Regel 3**

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**
- Tegenberger

T 2	**Regel 4.2**

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**

- Tegenberger

T 2x	**Regel 3**

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**

- Tegenberger

T 2 5	**Regel 4.5**

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**
- Tegenberger → **T251**

T 2 51	**Regel 4.1**

- Tekekenperger

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**
- Tegenberger → **T251**
- Tekekenperger

T 2 2	**Regel 4.2 / 5 / 3**

Soundex

Problem: Welche der drei folgenden Namen sind gleich?

- Theckenperger → **T251**
- Tegenberger → **T251**
- Tekekenperger → **T225**

Symbolische operationen

Towers of Hanoi

Situation in einem Tempel in Hanoi:

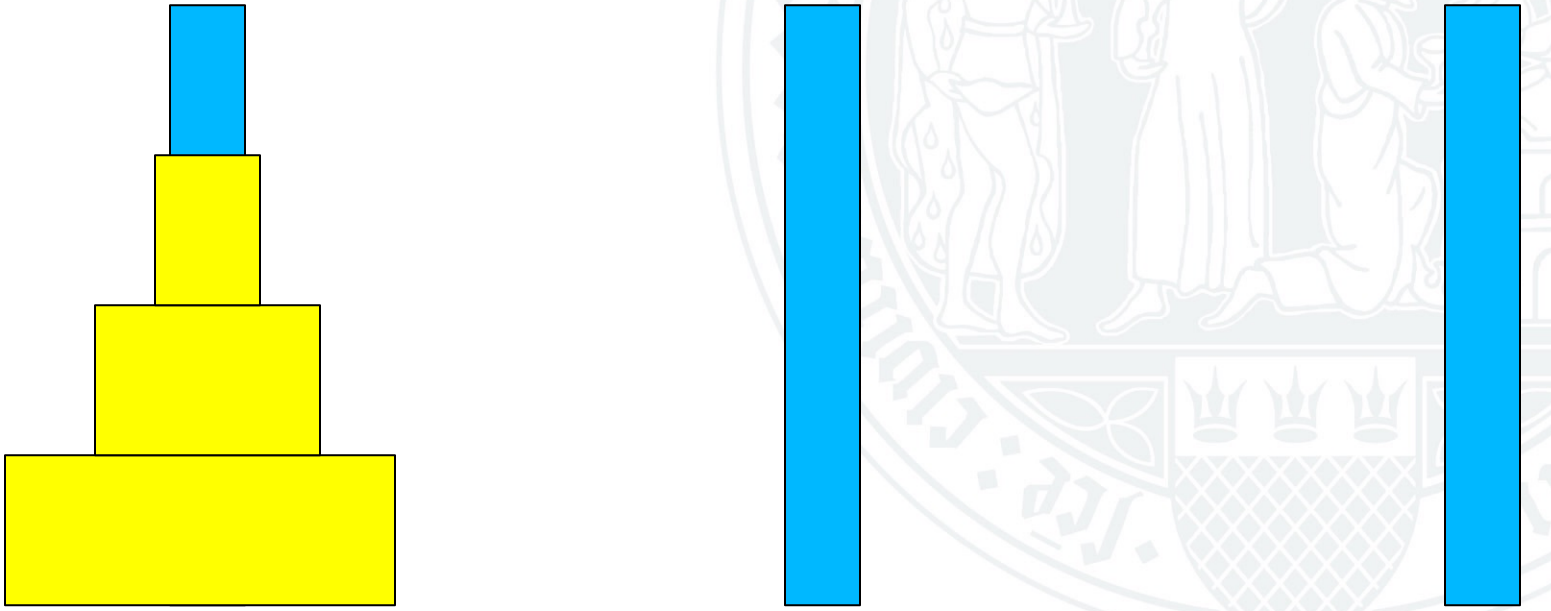
- Ein Turm von 100 Scheiben auf einer Spindel (S1).
- Eine leere Spindel (S2).
- Eine weitere leere Spindel (S3).

Transportiere S1 so nach S2 - wobei S3 als Zwischenlager verwendet werden darf - dass:

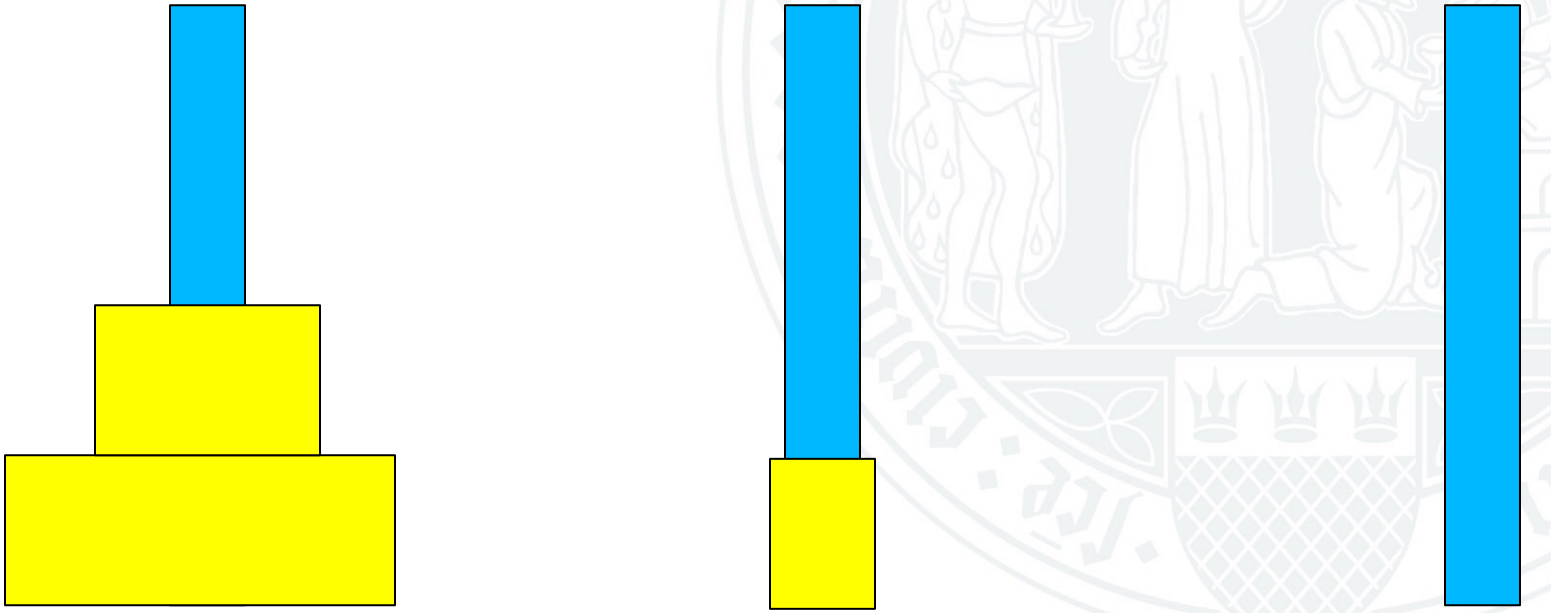
- Jeweils nur die oberste Scheibe von einem Turm genommen wird.
- Niemals eine größere Scheibe auf einer kleineren liegt.

Prophezeiung: Ist das erledigt, ist das Ende der Welt gekommen.

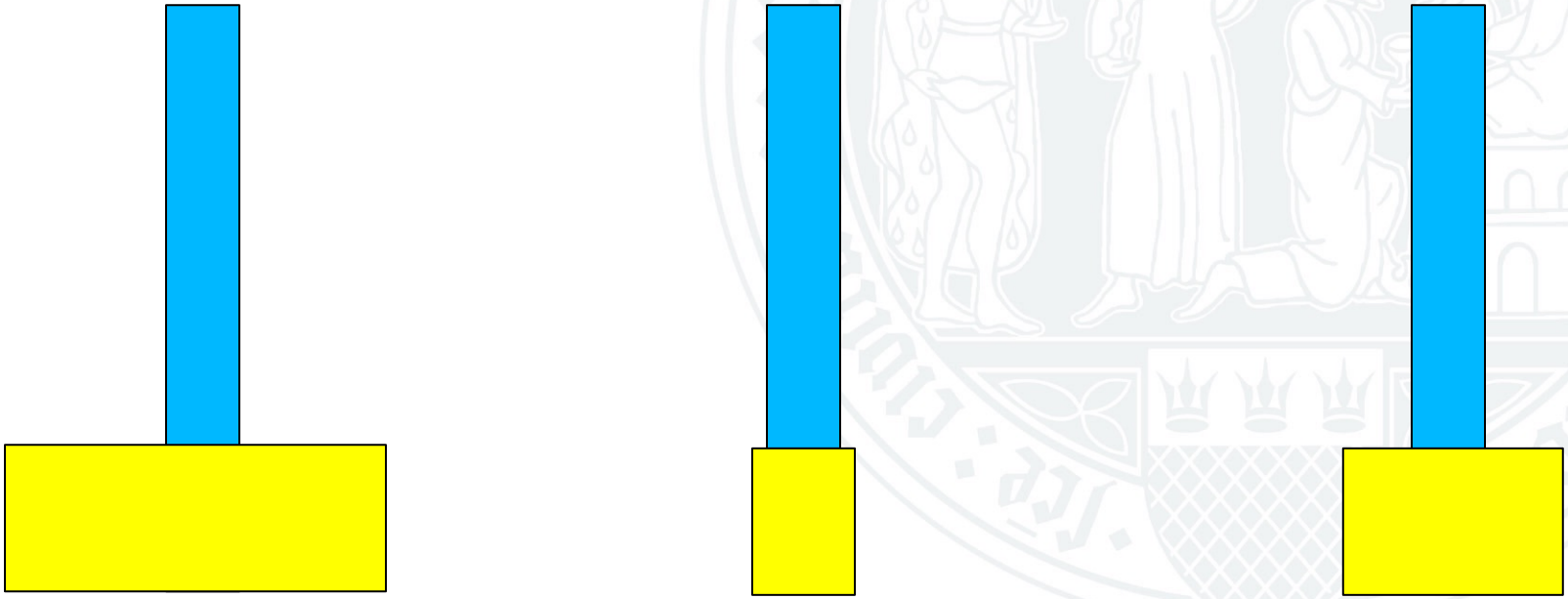
Towers of Hanoi



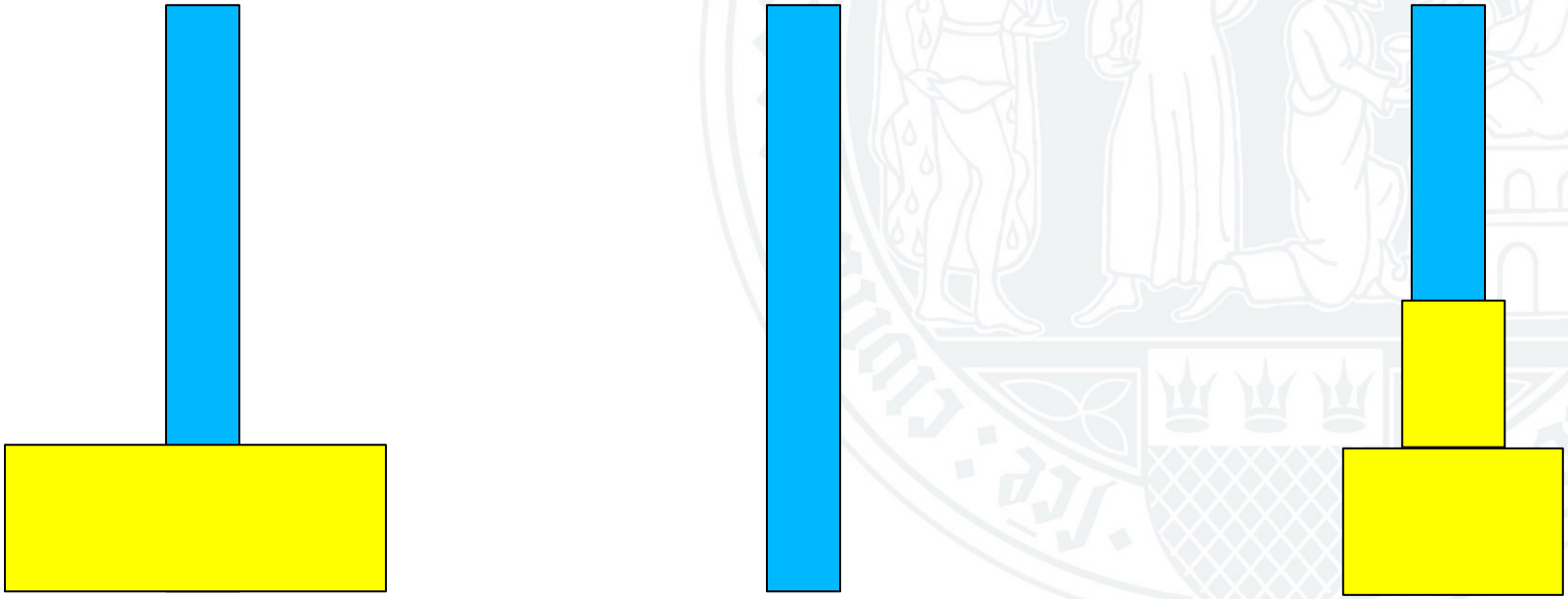
Towers of Hanoi



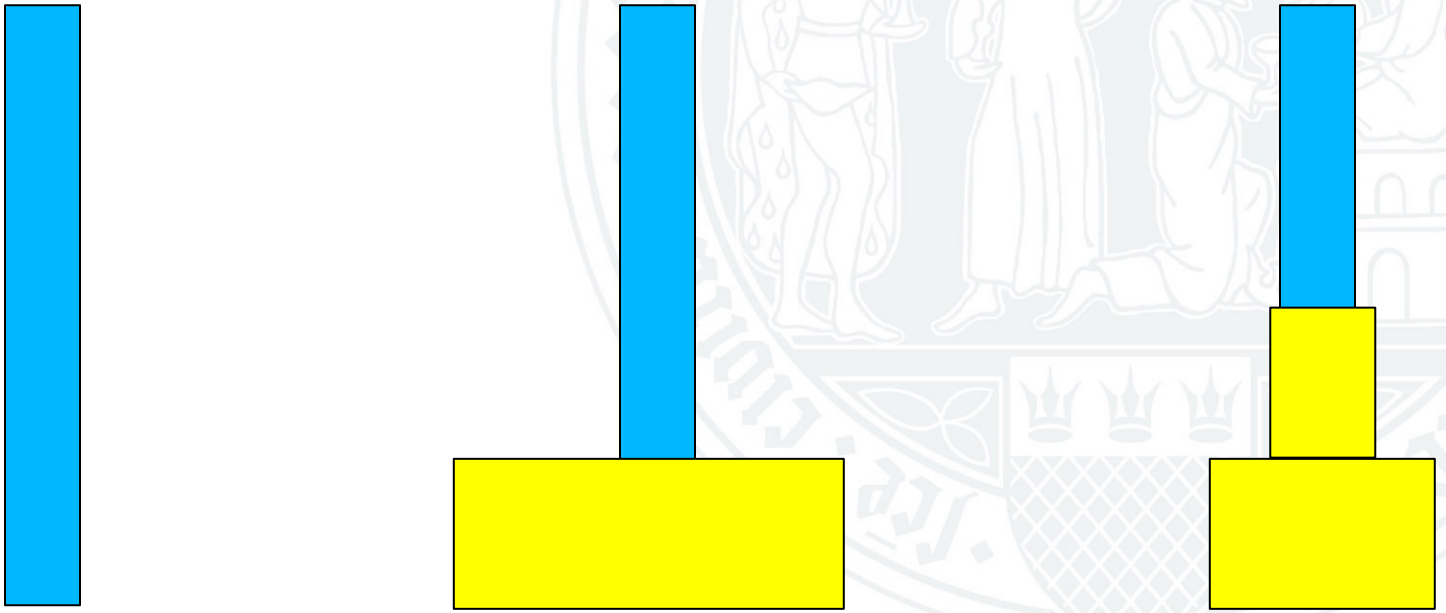
Towers of Hanoi



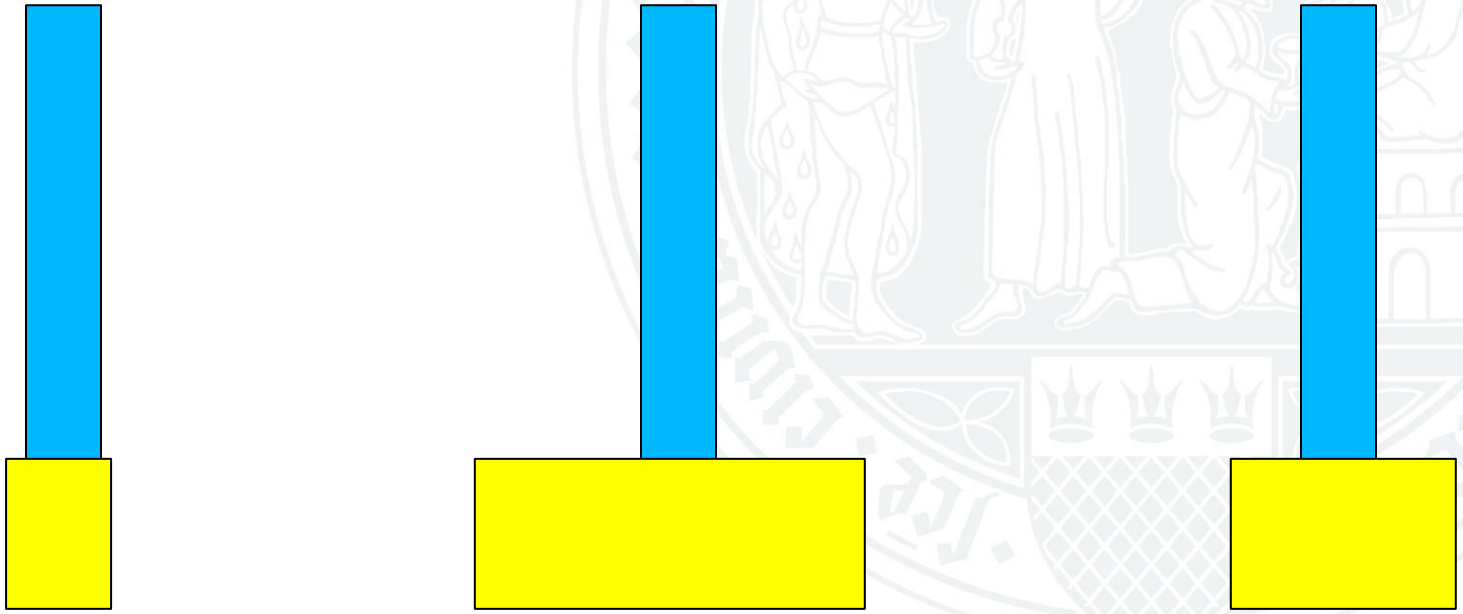
Towers of Hanoi



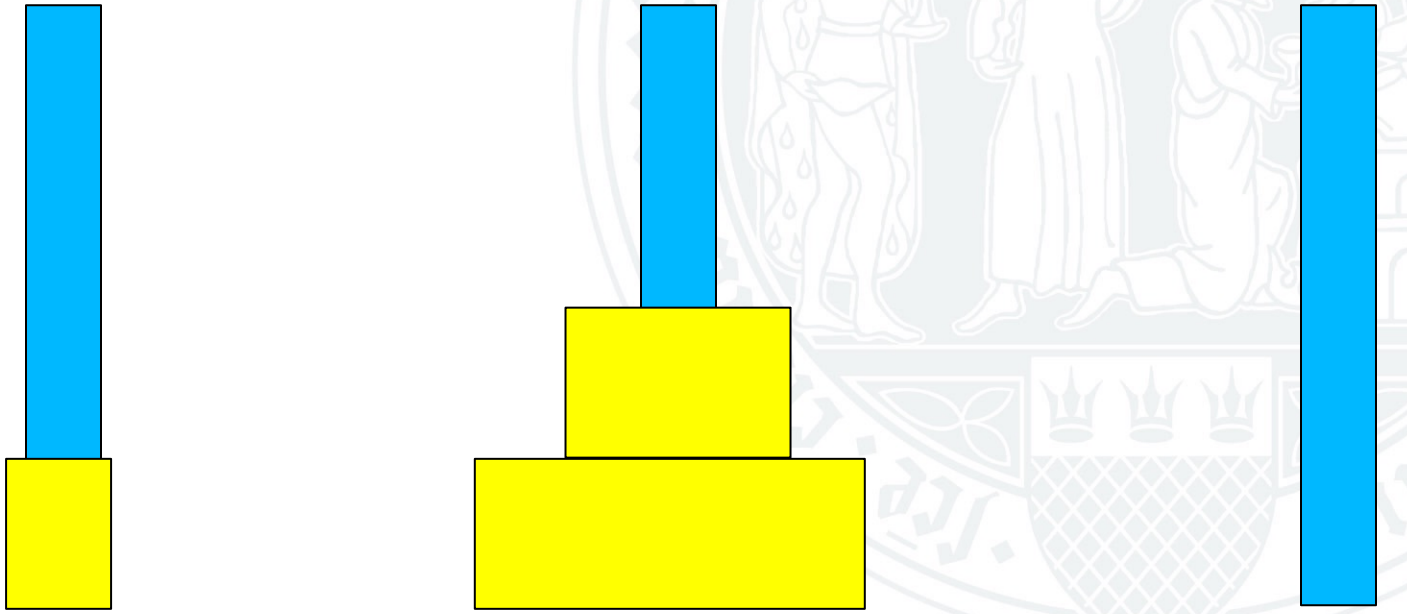
Towers of Hanoi



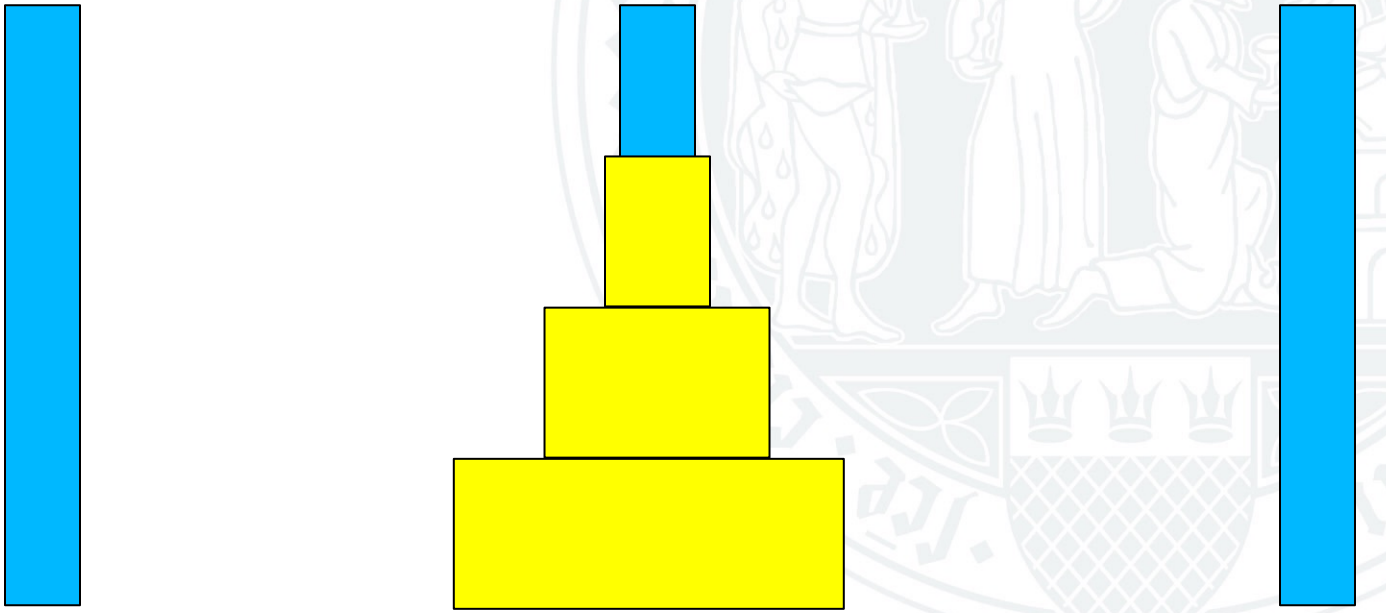
Towers of Hanoi



Towers of Hanoi



Towers of Hanoi



Towers of Hanoi

Lösung I

1. Finde jemand, der die obersten 99 Scheiben von S1 nach S3 transportiert.
2. Transportiere die unterste Scheibe von S1 nach S2.
3. Finde jemand, der die obersten 99 Scheiben von S3 nach S2 transportiert.

Towers of Hanoi

Lösung II

1. Besteht der zu transportierende Turm aus mehr als einer Scheibe, finde jemand, der einen Turm von n-1 Scheiben von S1 nach S3 transportiert. Nutze S2 als Zwischenablage.
2. Transportiere selbst die unterste Scheibe von S1 nach S2.
3. Besteht der zu transportierende Turm aus mehr als einer Scheibe, finde jemand, der einen Turm von n-1 Scheiben von S3 nach S2 transportiert. Nutze S1 als Zwischenablage.

Towers of Hanoi

Lösung III

```
function transport(int n, stack spindel1, stack spindel2,
                                stack spindel3) {
    if (n >1) transport(n-1,spindel1,spindel3,spindel2);
    schritt(spindel1,spindel2);
    if (n>1) transport(n-1,spindel3,spindel2,spindel1);
   }

function schritt(stack spindel1, stack spindel2) {
   spindel2.push(spindel1.pop());
  }
```

Towers of Hanoi

Fragen

1. Wie viele Mitarbeiter werden benötigt?

   n

2. Wieviele Transferschritte?

   2^n -1

3. Wie lange?

   2^{100}-1 Schritte == ca. 10^{30}

   1 Schritt == 1 Sekunde ==> ca. 10^{30} Sekunden == ca. 4 * 10^{22} Jahre