Einführung in die Informationsverarbeitung

Øyvind Eide

Woche 2
Datenstrukturen
Algorithmen

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

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

Datenstrukturen
Grundbegriffe

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

Datentyp

- Zahlen
- Bilder
- Zeichenketten
- Geburtstage
- Briefe

Operationen

- Addieren
- Komprimieren
- Vergleichen
- Abstand berechnen
- Beziehen

Datenstrukturen

- Zahl {Darstellung, Addieren, …}
- Bild {Darstellung, Komprimieren, …}
- Text {Darstellung, Vergleichen, … }
- Zeit {Darstellung, Abstand berechnen, … }
- Brief {Darstellung, Beziehen, …}

Datenstruktur =

{Datentyp, Legale Operationen }

“Datentyp” und “Datenstruktur” oft aber auch synonym!

Basisdatenstrukturen

- Boolean / Logischer Wert
- Integer
- [Rationale Zahlen]
- Realzahlen
- Zeichen
- Zeichenketten

Datenstrukturen und Hardware

Datenstrukturen geben Regeln wieder, wie ein bestimmter Speicherbereich interpretiert wird.

ASCII Zeichen 'a' = 97; 'A' = 65.

oder

'Pixel' 97 ist eineinhalb mal heller als 'Pixel' 65.

Datenstrukturen und Hardware

Festlegungen sind „willkürlich“.

Groß- / Klein v. Umlaute

Zeichen

a	097	A	65
b	098	B	66
c	099	C	67
d	100	D	68
e	101	E	69
f	102	F	70
g	103	G	71
h	104	H	72
i	105	I	73
j	106	J	74
k	107	K	75
…	…	…	…

Zeichen

a	01100001	A	01000001
b	01100010	B	01000010
c	01100011	C	01000011
d	01100100	D	01000100
e	01100101	E	01000101
f	01100110	F	01000110
g	01100111	G	01000111
h	01101000	H	01001000
i	01101001	I	01001001
j	01101010	J	01001010
k	01101011	K	01001011
…	…	…	…

Zeichen

a	01100001	A	01000001
b	01100010	B	01000010
c	01100011	C	01000011
d	01100100	D	01000100
e	01100101	E	01000101
f	01100110	F	01000110
g	01100111	G	01000111
h	01101000	H	01001000
i	01101001	I	01001001
j	01101010	J	01001010
k	01101011	K	01001011
…	…	…	…

Zeichen

Festlegungen sind „willkürlich“.

lower(x) = upper(x) | '00100000'

= schnellste verfügbare Operation des Rechners!

Merke

Darstellung von Datenstrukturen sind „willkürlich“.

… können den Aufwand für eine Anwendung aber entscheidend beeinflussen!

Datenstruktur „Zeiger“

Diagrammatische Darstellung:

A „zeigt auf“ B

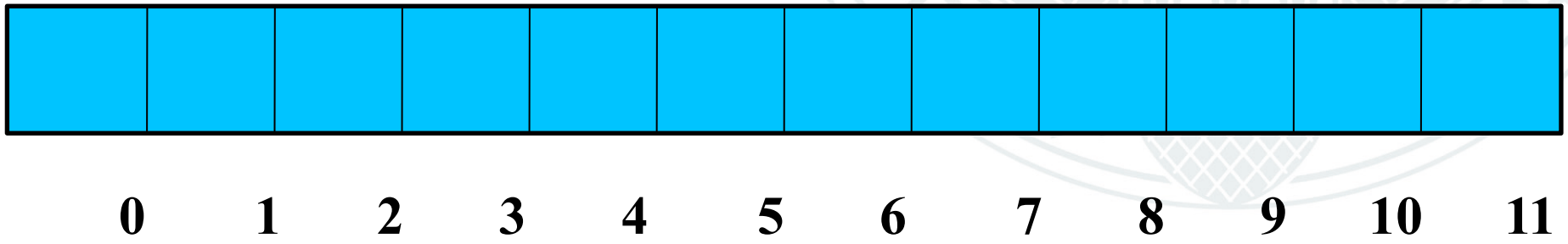
Datenstruktur „Zeiger“

Diagrammatische Darstellung:

„Zeiger“: Ein Speicherinhalt eines Rechners verweist auf einen anderen.

Datenstruktur im Speicher

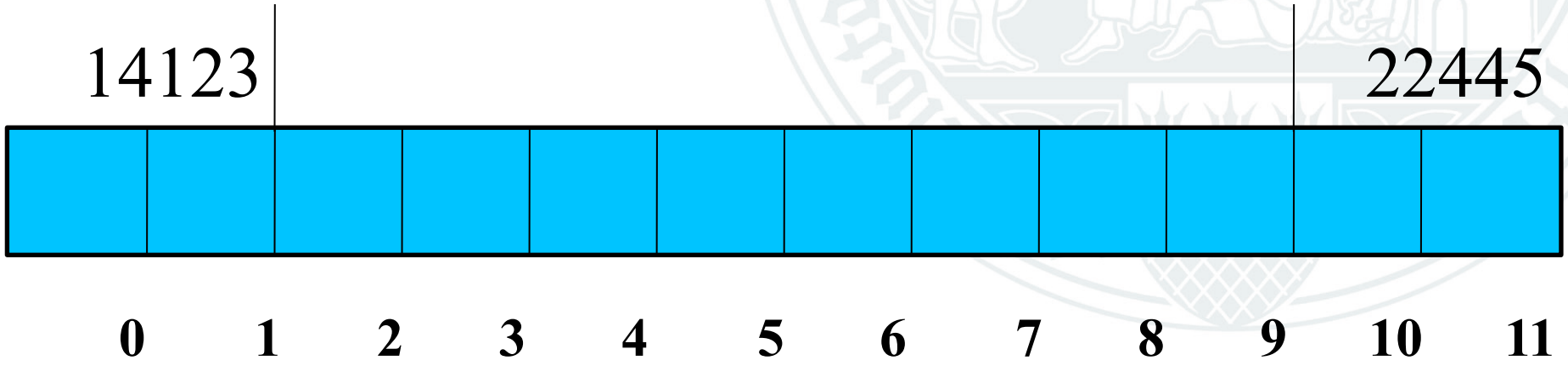
Speicher als „karierte Zeile“



Datenstruktur im Speicher

Zahl „14123“ in Bytes 0 bis 1

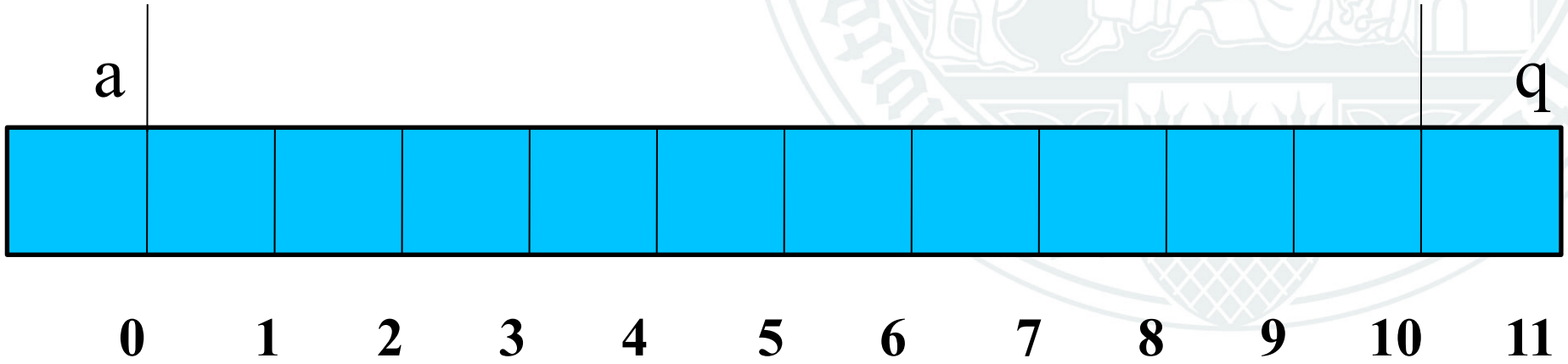
Zahl „22445“ in Bytes 10 bis 11



Datenstruktur im Speicher

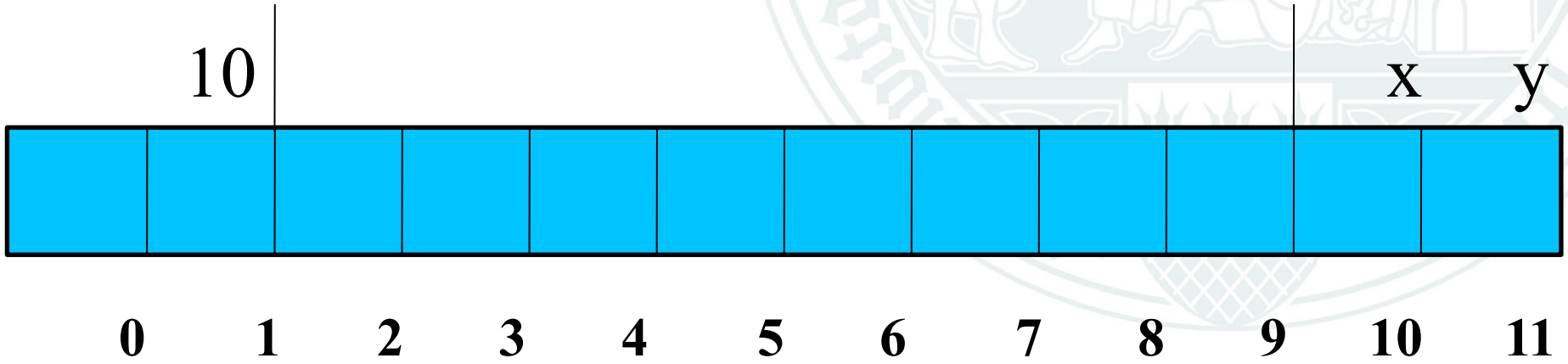
Zeichen „a“ in Byte 0

Zeichen „q“ in Byte 11



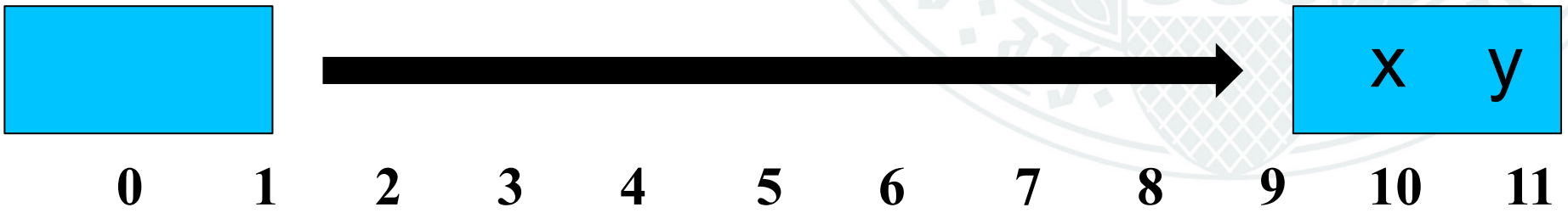
Datenstruktur im Speicher

Zeiger in Bytes 0 bis 1 verweist auf Speicherblock, enthaltend „xy“, beginnend in Byte 10



Zeiger graphisch

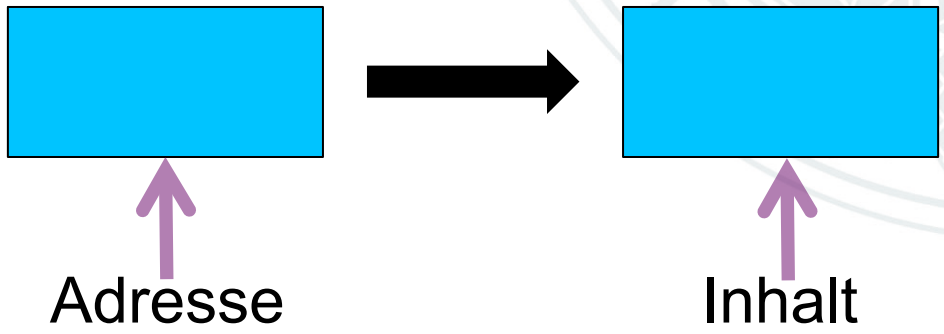
Zeiger in Bytes 0 bis 1 verweist auf Speicherblock, enthaltend „xy“, beginnend in Byte 10



Zeiger graphisch

Zeiger verweist von einem Datenblock auf einen anderen.

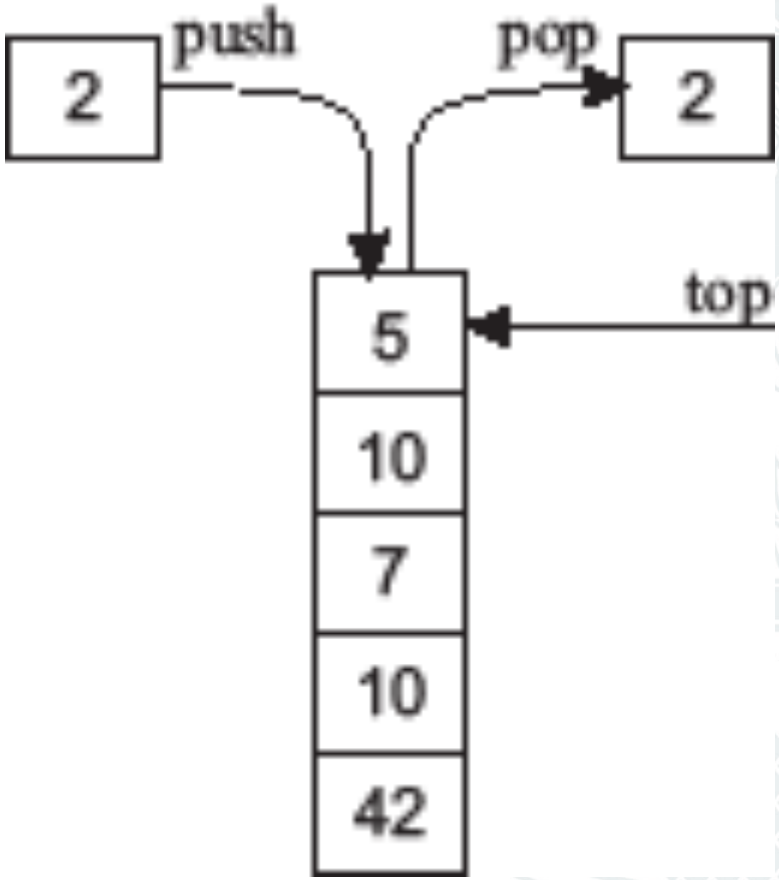
Inhalt im Datenblock: Kontext-basierte Interpretation



Technische Datenstrukturen

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

Stacks



Auch bekannt als: „LIFO“ – Last In, First Out

Start

Lies:

Atom 1

Verarbeite:

„Push to stack“

Lies:

Verarbeite:

Atom 1

„Lies weiter“

Lies:

Atom 2

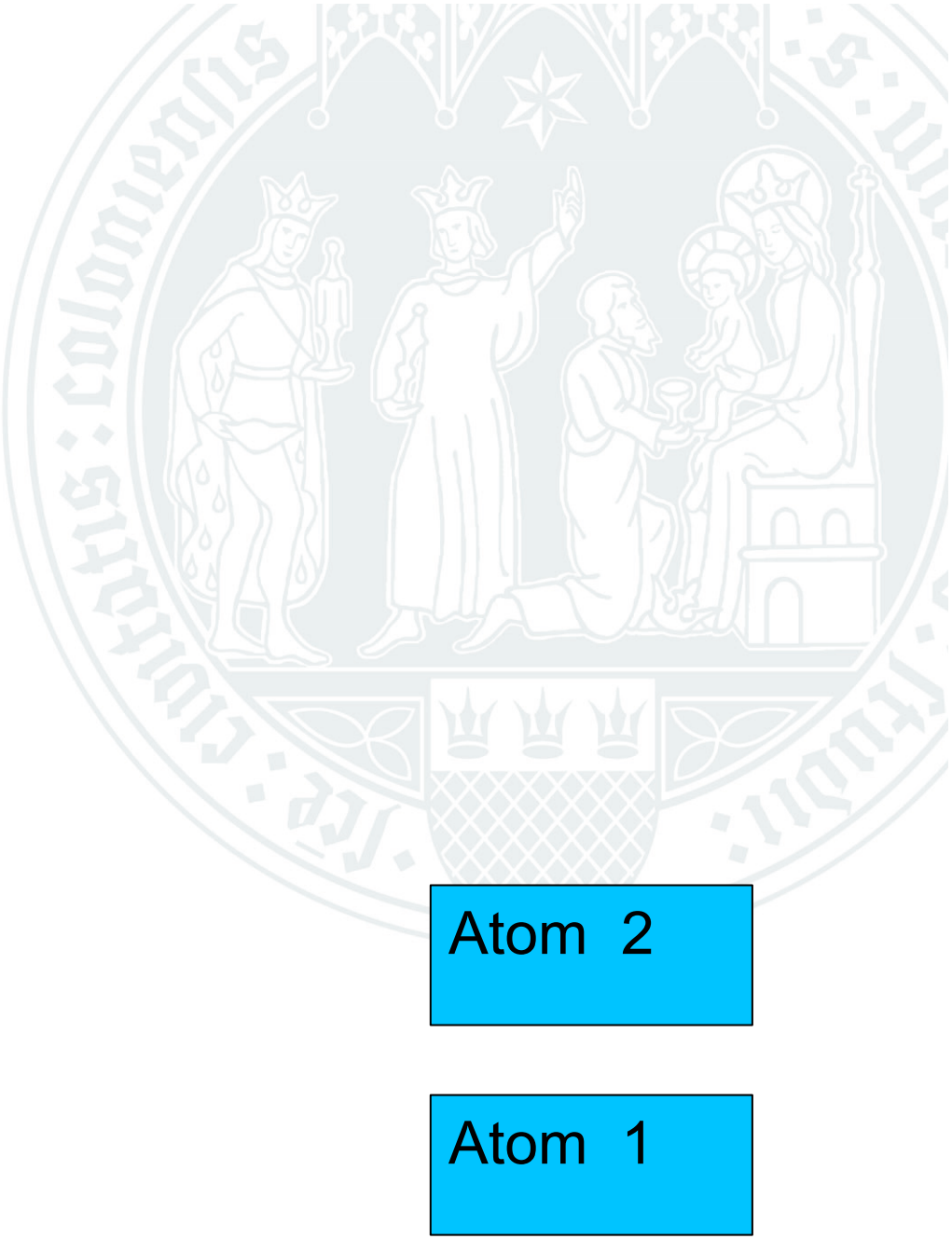
Verarbeite:

Atom 1

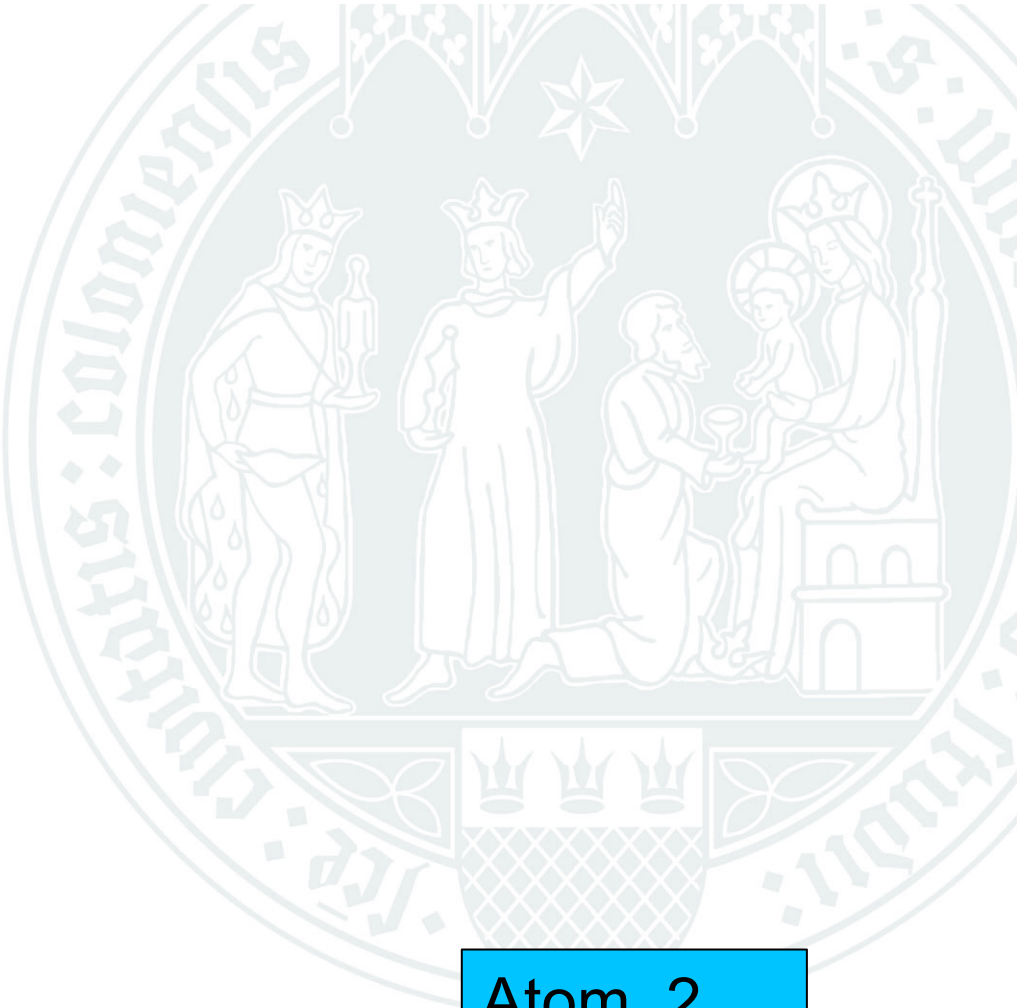
„Push to stack“

Lies:

Verarbeite:



„Lies weiter“



Lies:

Atom 3

Verarbeite:

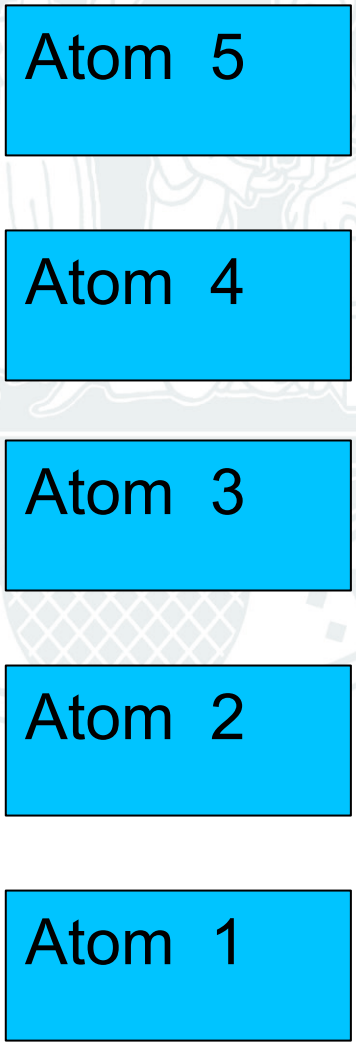
Atom 2

Atom 1

Schließlich

Lies:

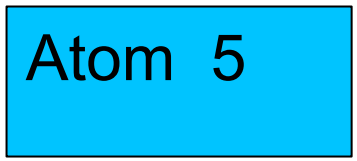
Verarbeite:

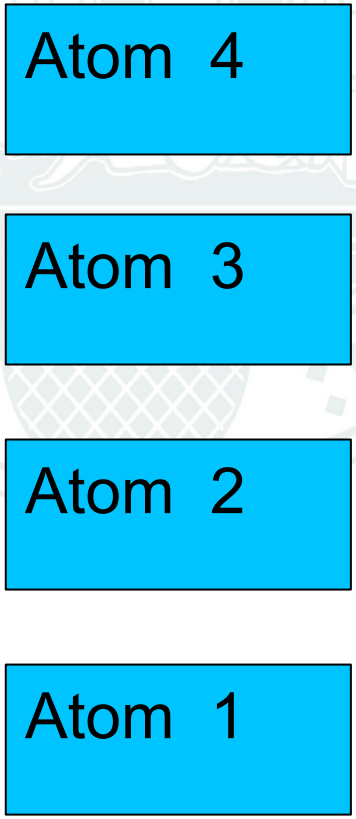


„Pop from stack“

Lies:

Verarbeite:

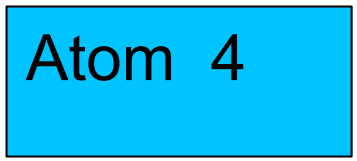


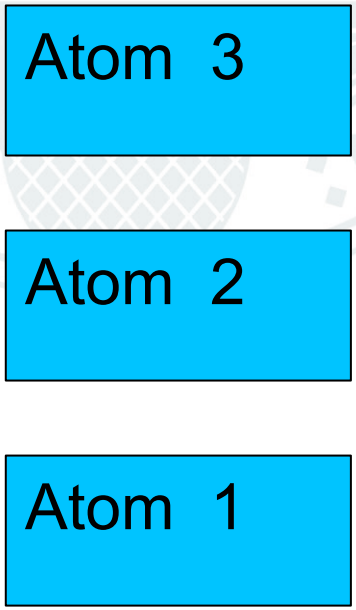


„Pop from stack“

Lies:

Verarbeite:

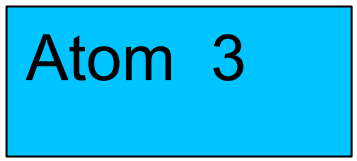


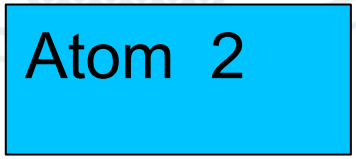


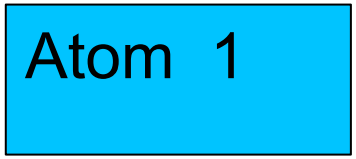
„Pop from stack“

Lies:

Verarbeite:







„Pop from stack“

Lies:

Verarbeite:

Atom 2

Atom 1

„Pop from stack“

Lies:

Verarbeite:

Atom 1

Queues

Auch bekannt als: „FIFO“ – First In, First Out

Start

Lies:

Atom 1

Verarbeite:

„Push to queue“

Lies:

Verarbeite:

Atom 1

„Lies weiter“

Lies:

Atom 2

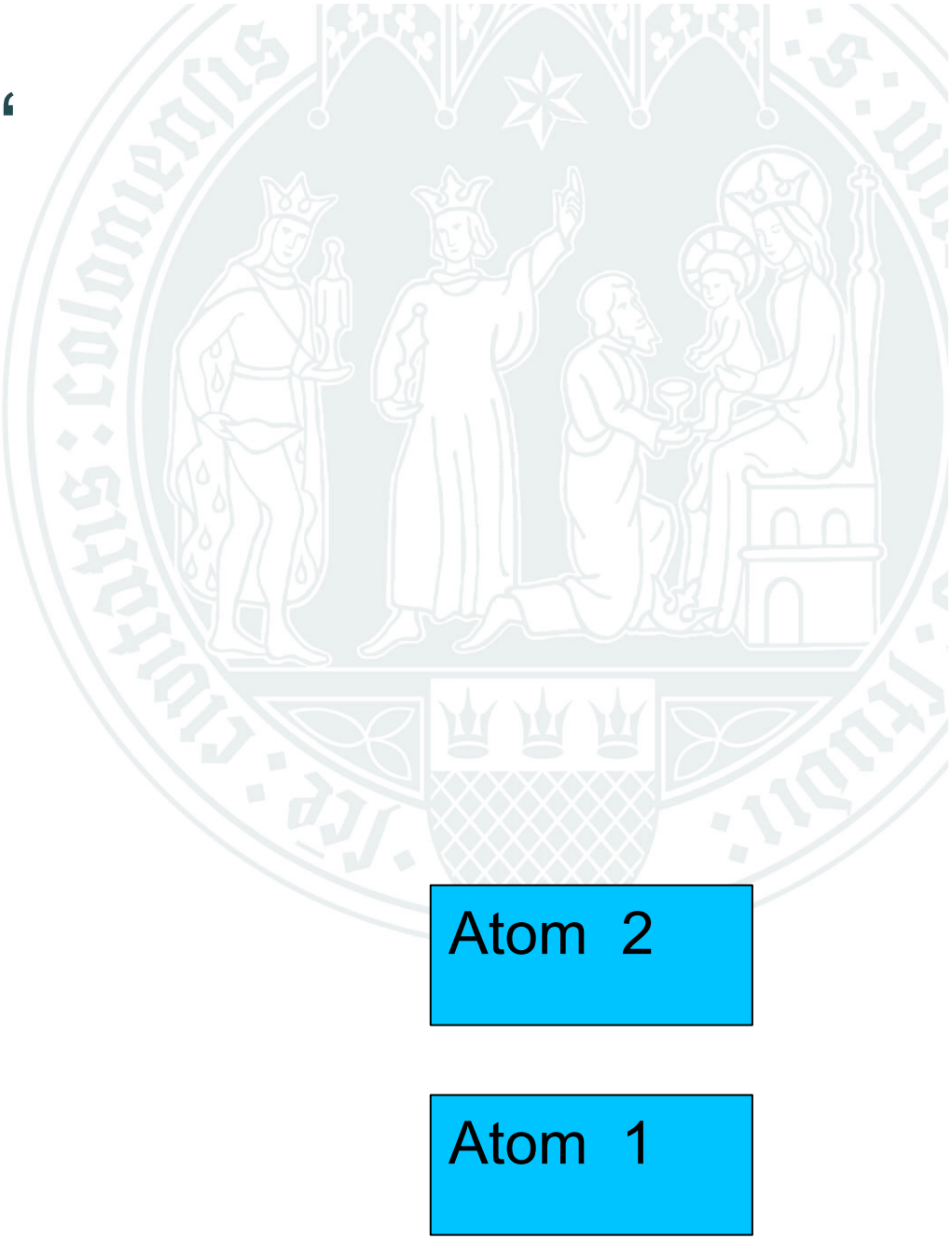
Verarbeite:

Atom 1

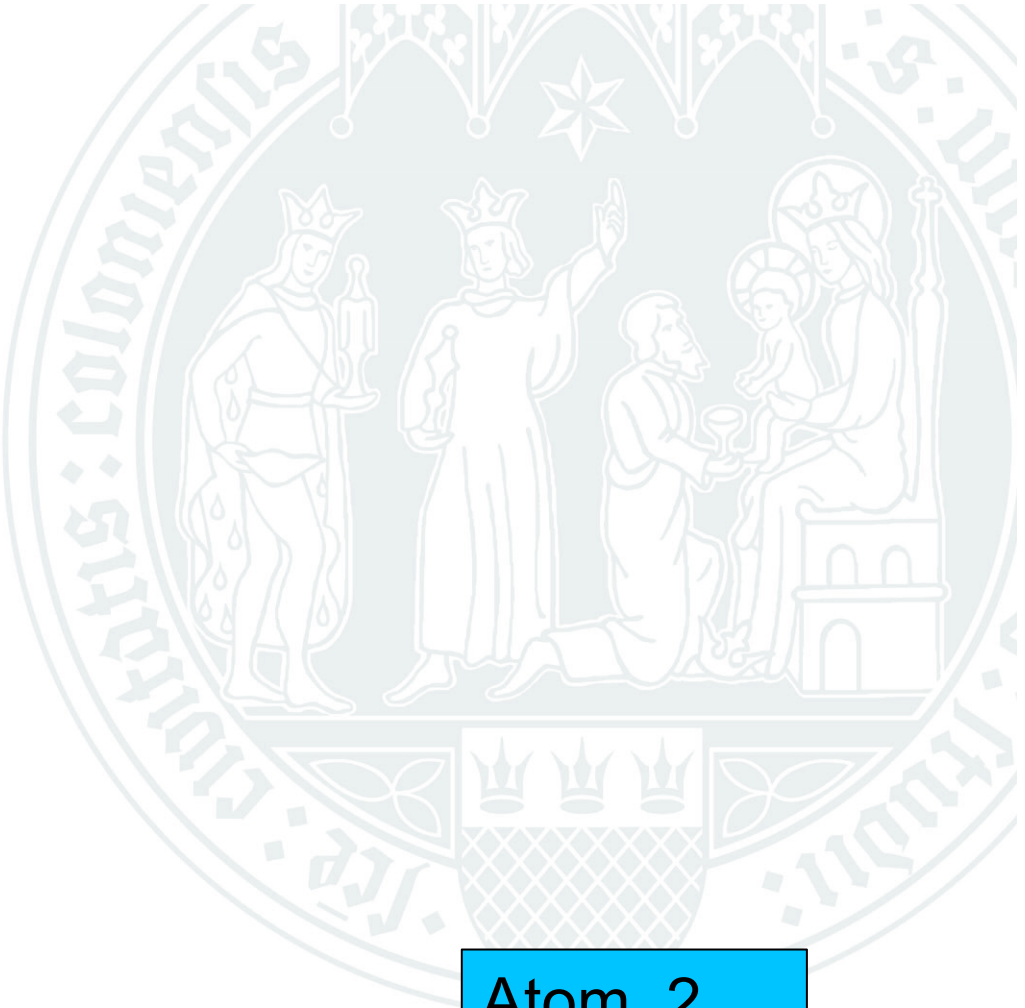
„Push to queue“

Lies:

Verarbeite:



„Lies weiter“



Lies:

Atom 3

Verarbeite:

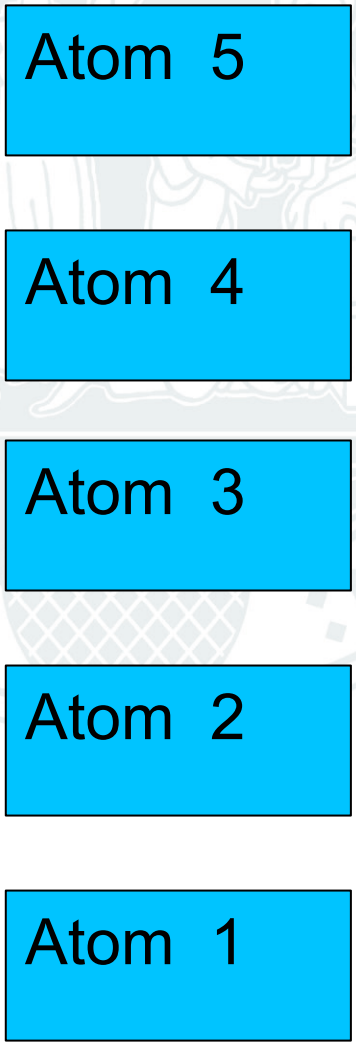
Atom 2

Atom 1

Schließlich

Lies:

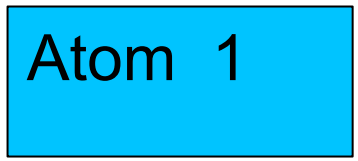
Verarbeite:

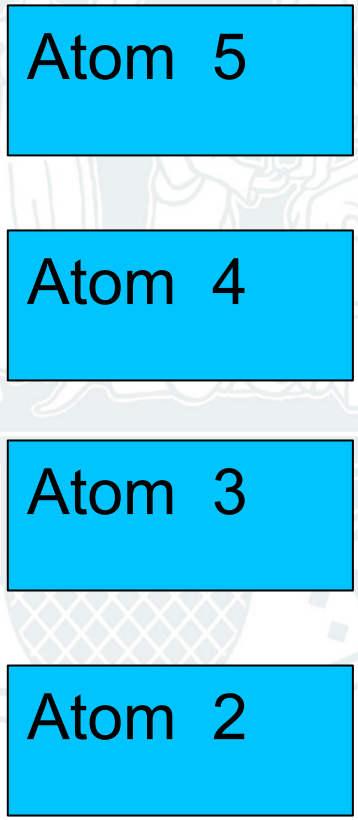


„Pop from queue”

Lies:

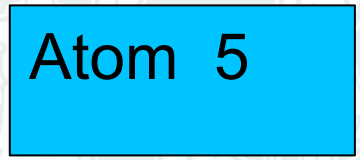
Verarbeite:

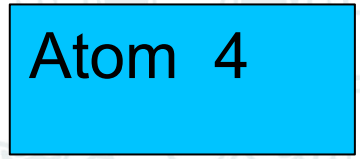


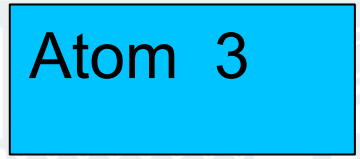


„Pop from queue”

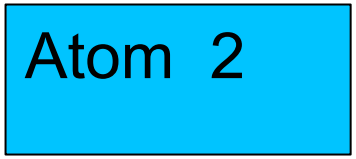
Lies:







Verarbeite:



„Pop from queue”

Lies:

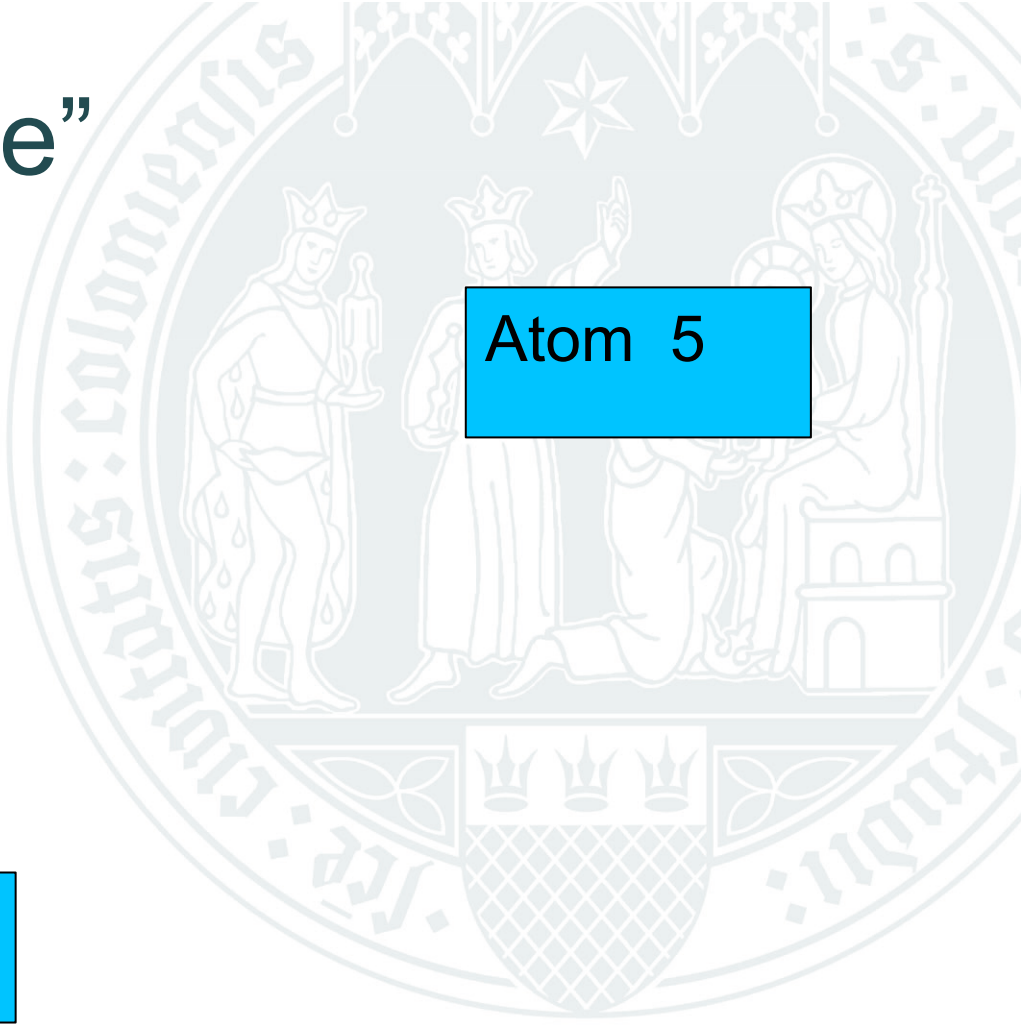
Atom 5

Atom 4

Verarbeite:

Atom 3

„Pop from queue”



Atom 5

Lies:

Verarbeite:

Atom 4

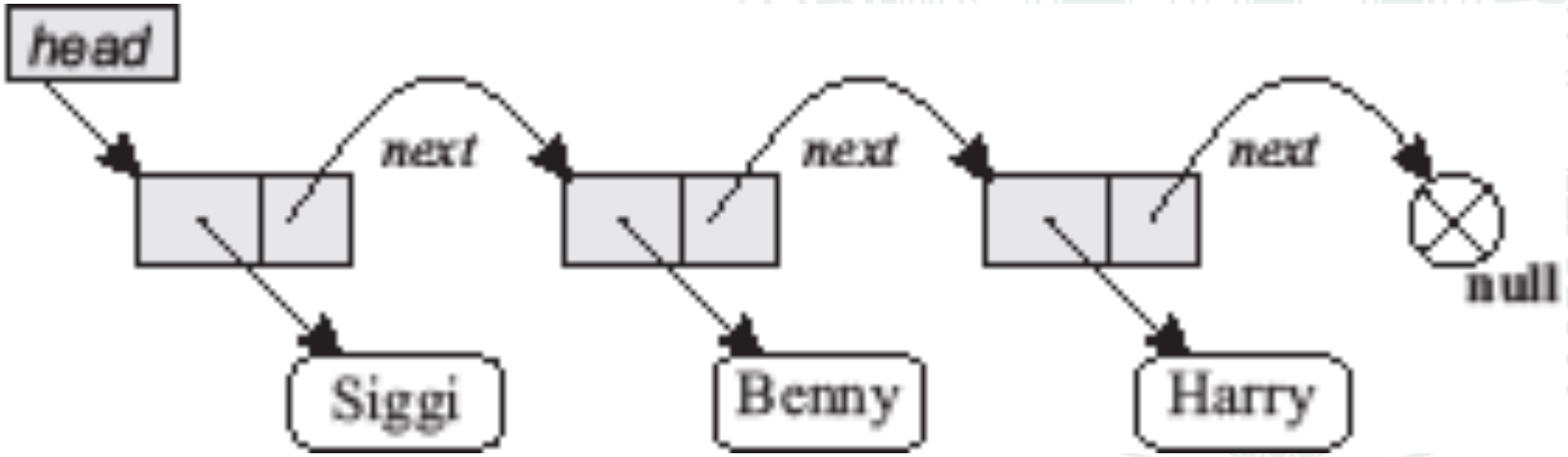
„Pop from queue”

Lies:

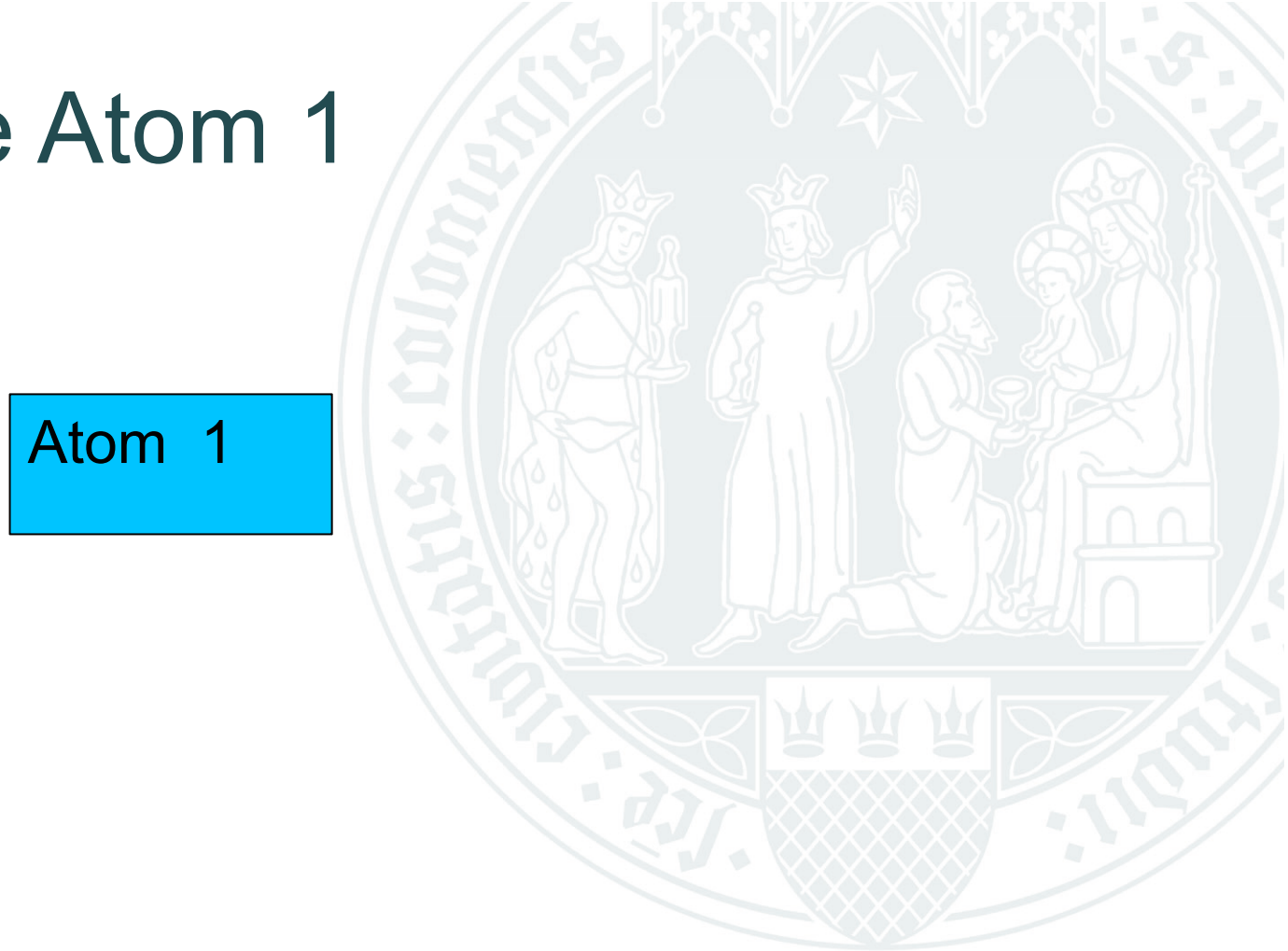
Verarbeite:

Atom 5

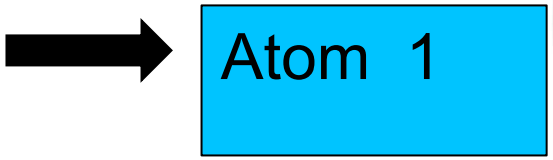
Einfach Verknüpfte Listen



Erzeuge Atom 1

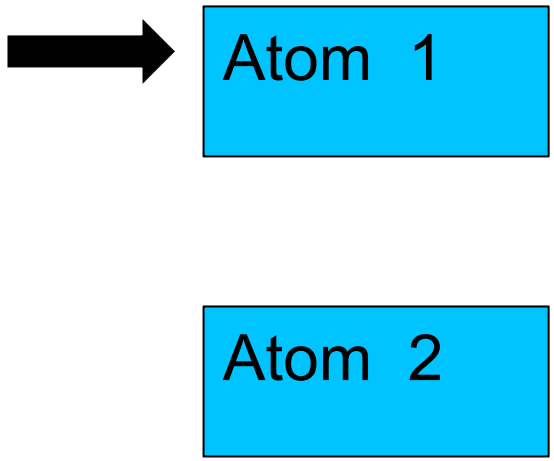


Mache Atom 1 zum Listenkopf

Kopf: 

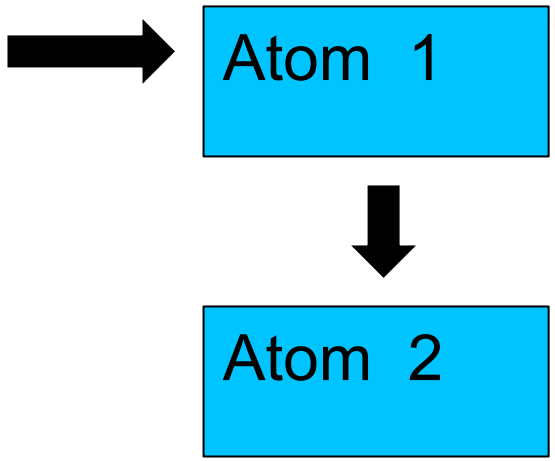
Erzeuge Atom 2

Kopf:



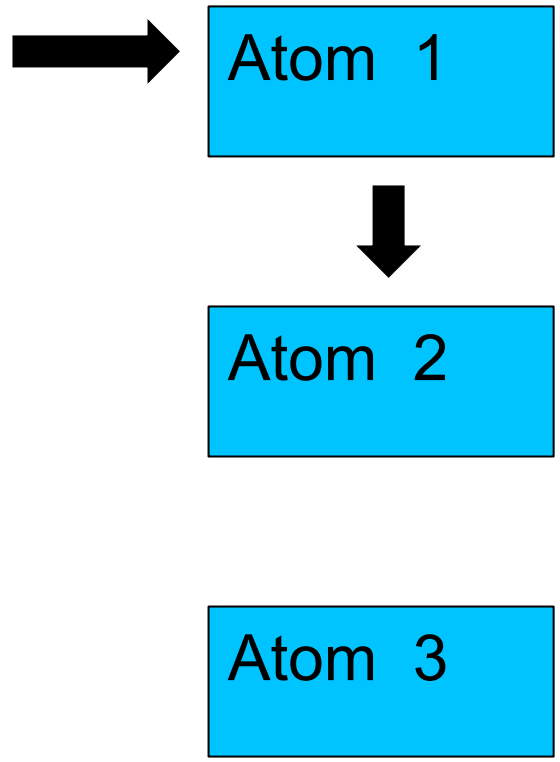
Verbinde Atom 2 mit Liste

Kopf:

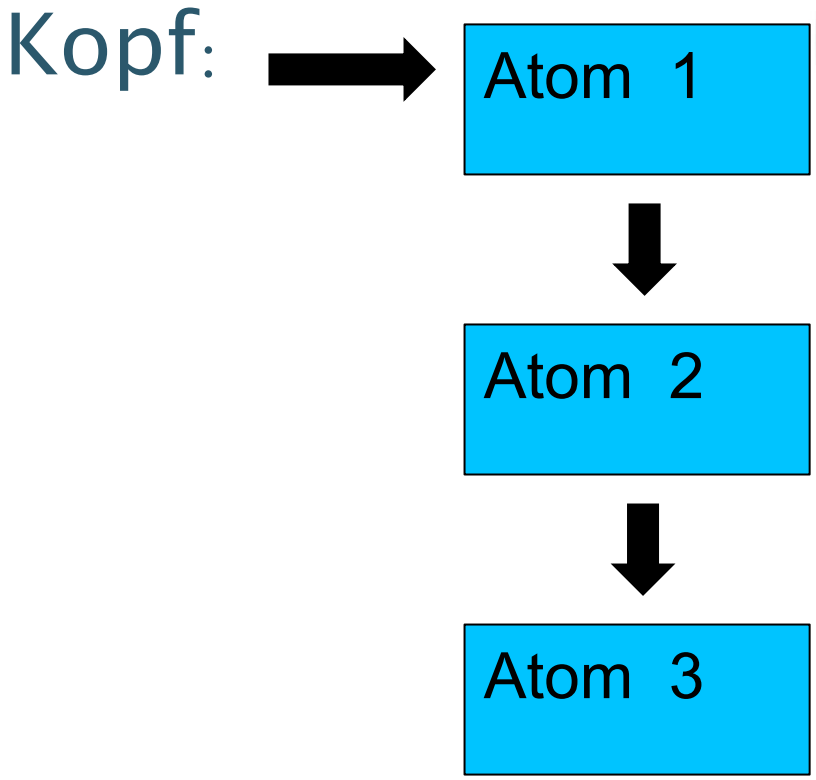


Erzeuge Atom 3

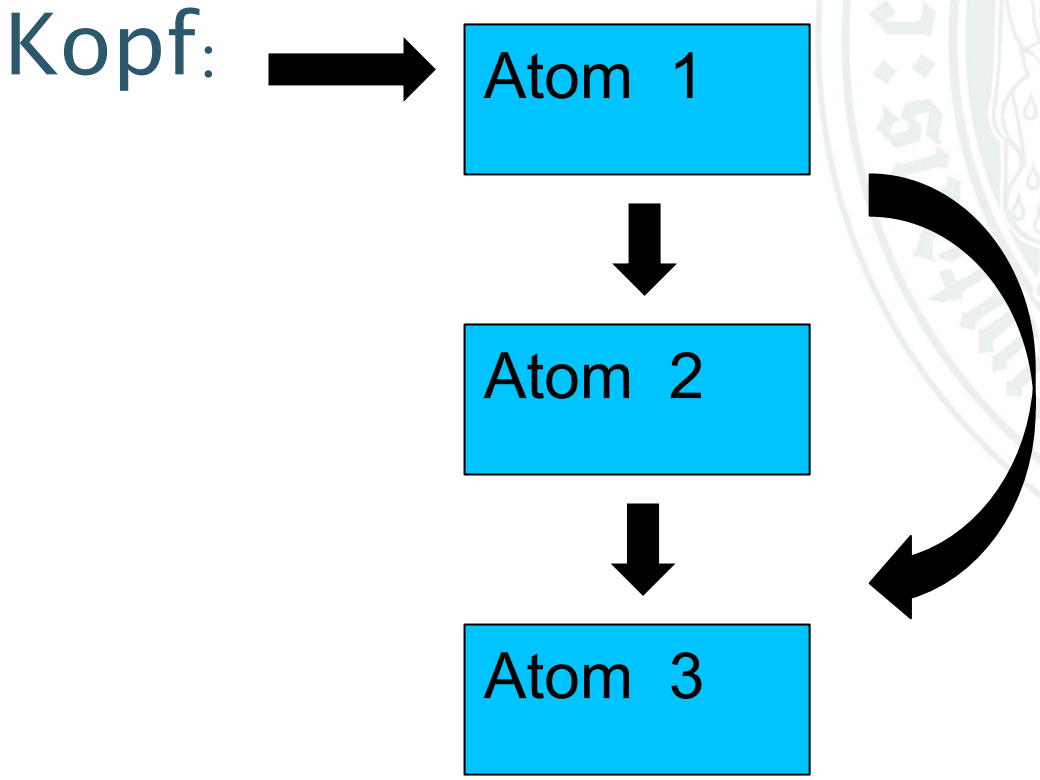
Kopf:



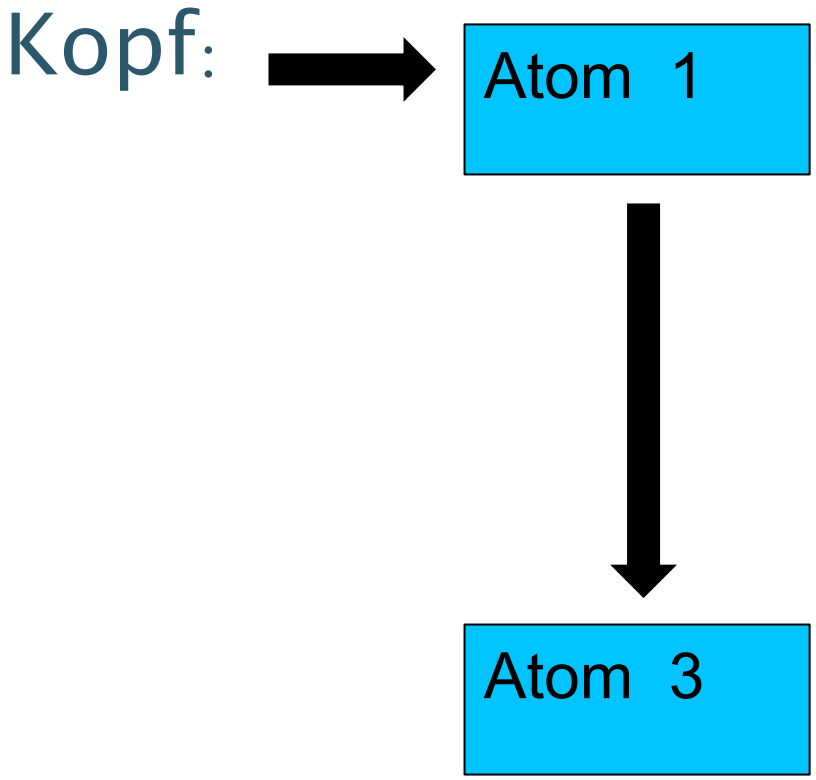
Verbinde Atom 3 mit Liste



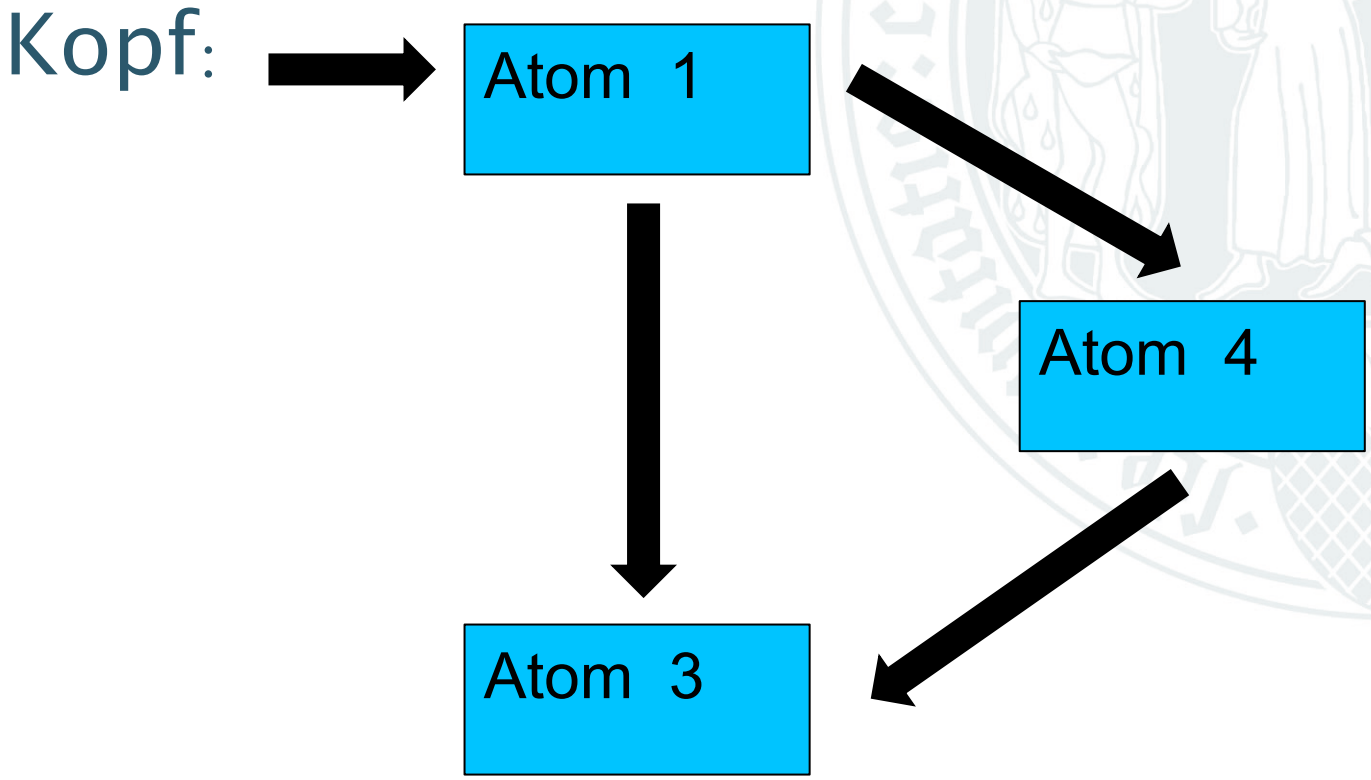
Lösche Atom 2



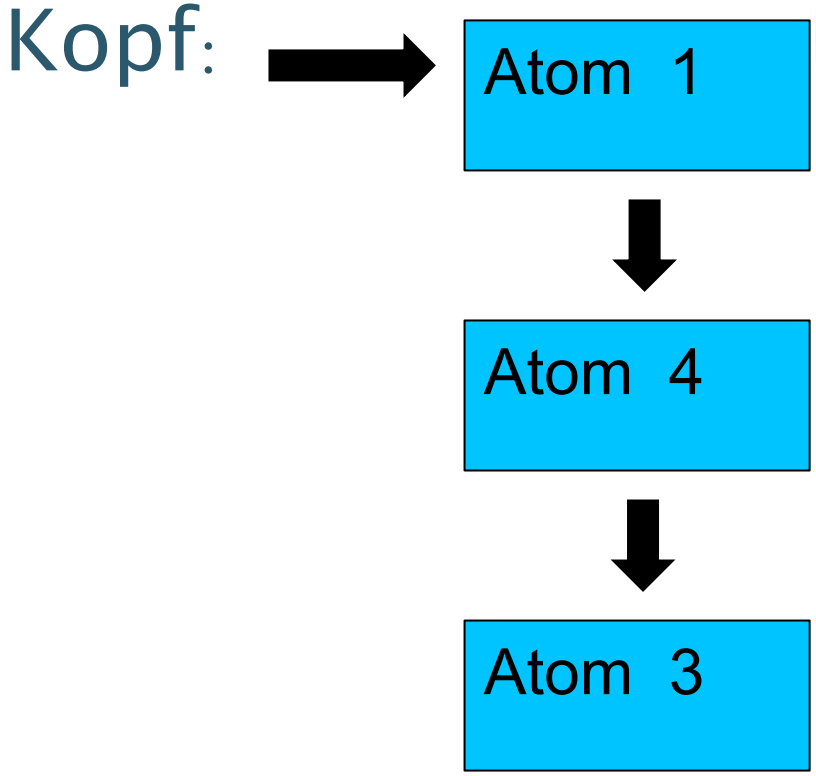
Lösche Atom 2



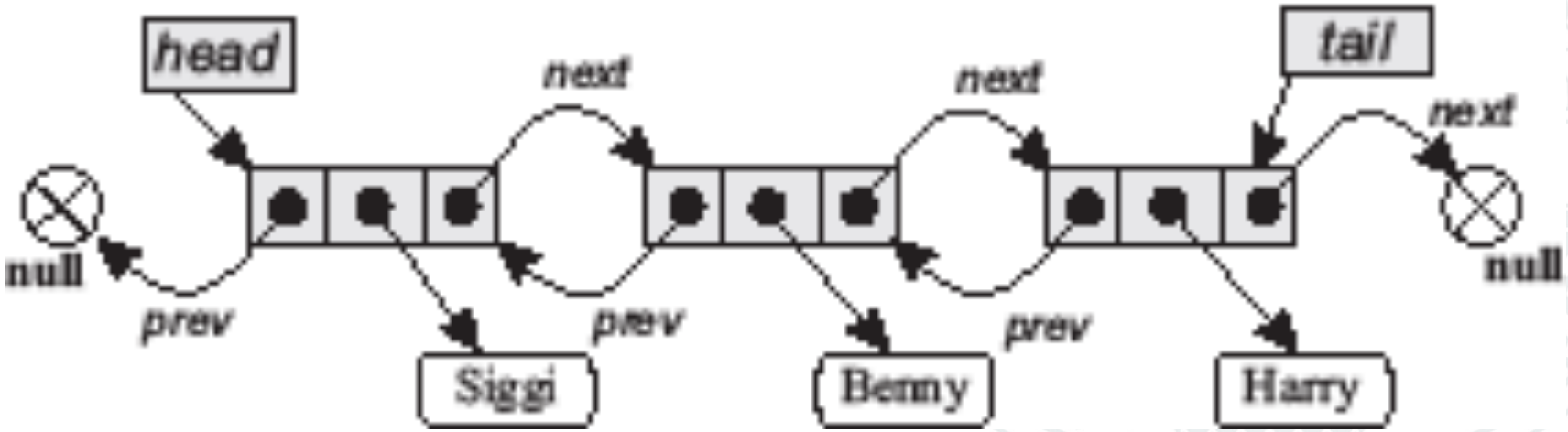
Füge Atom 4 ein



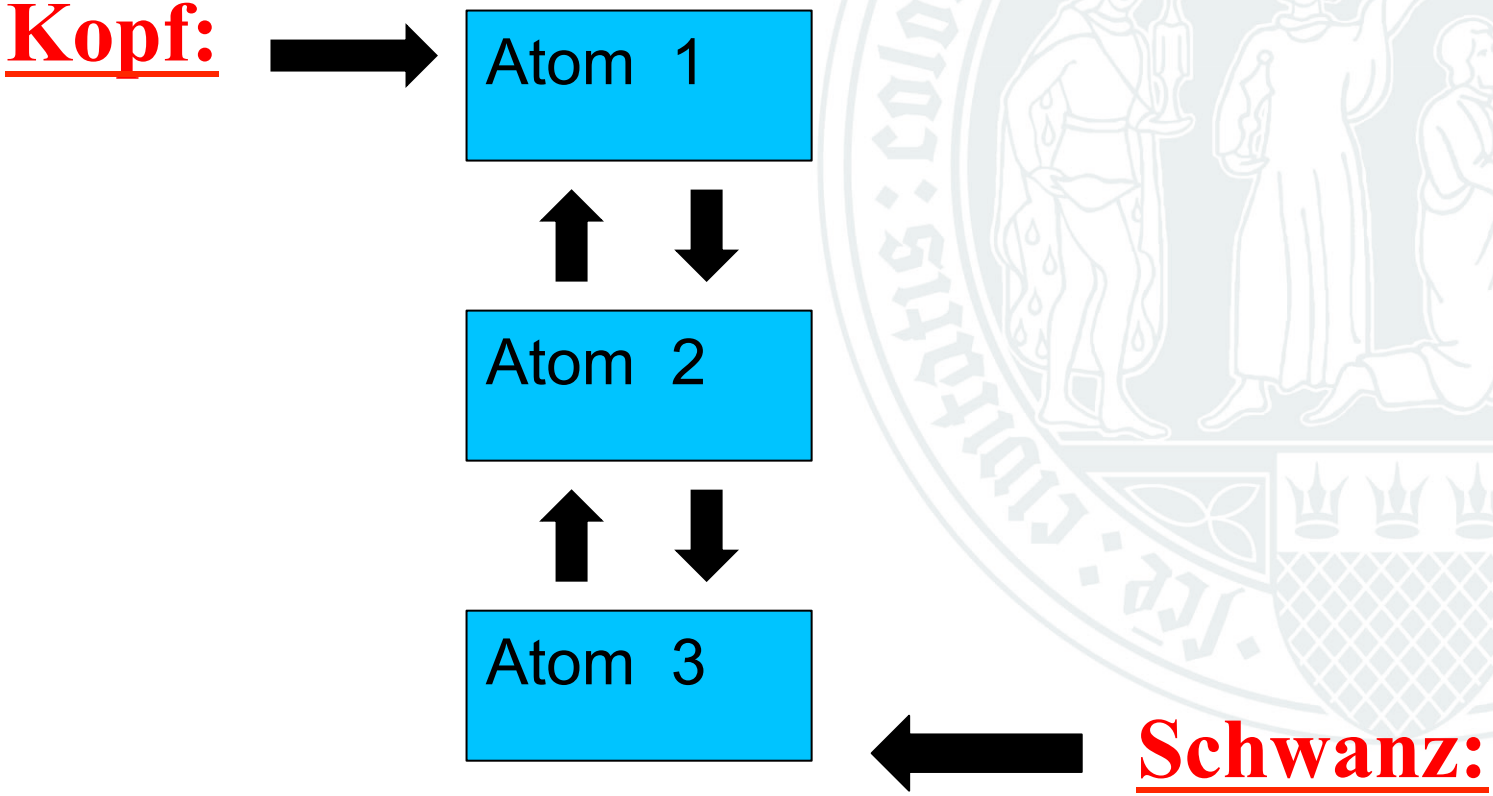
Füge Atom 4 ein



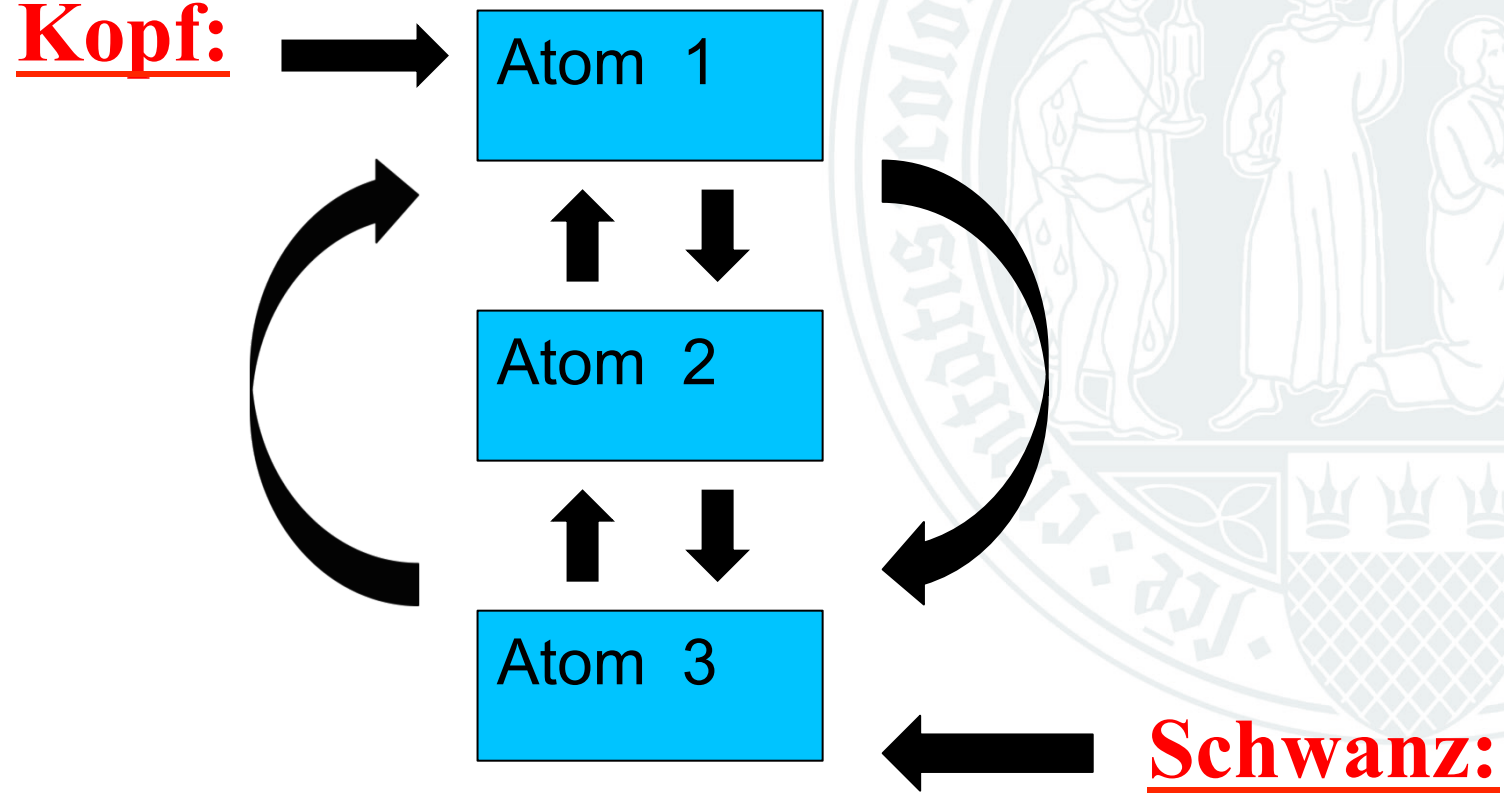
Doppelt Verknüpfte Listen



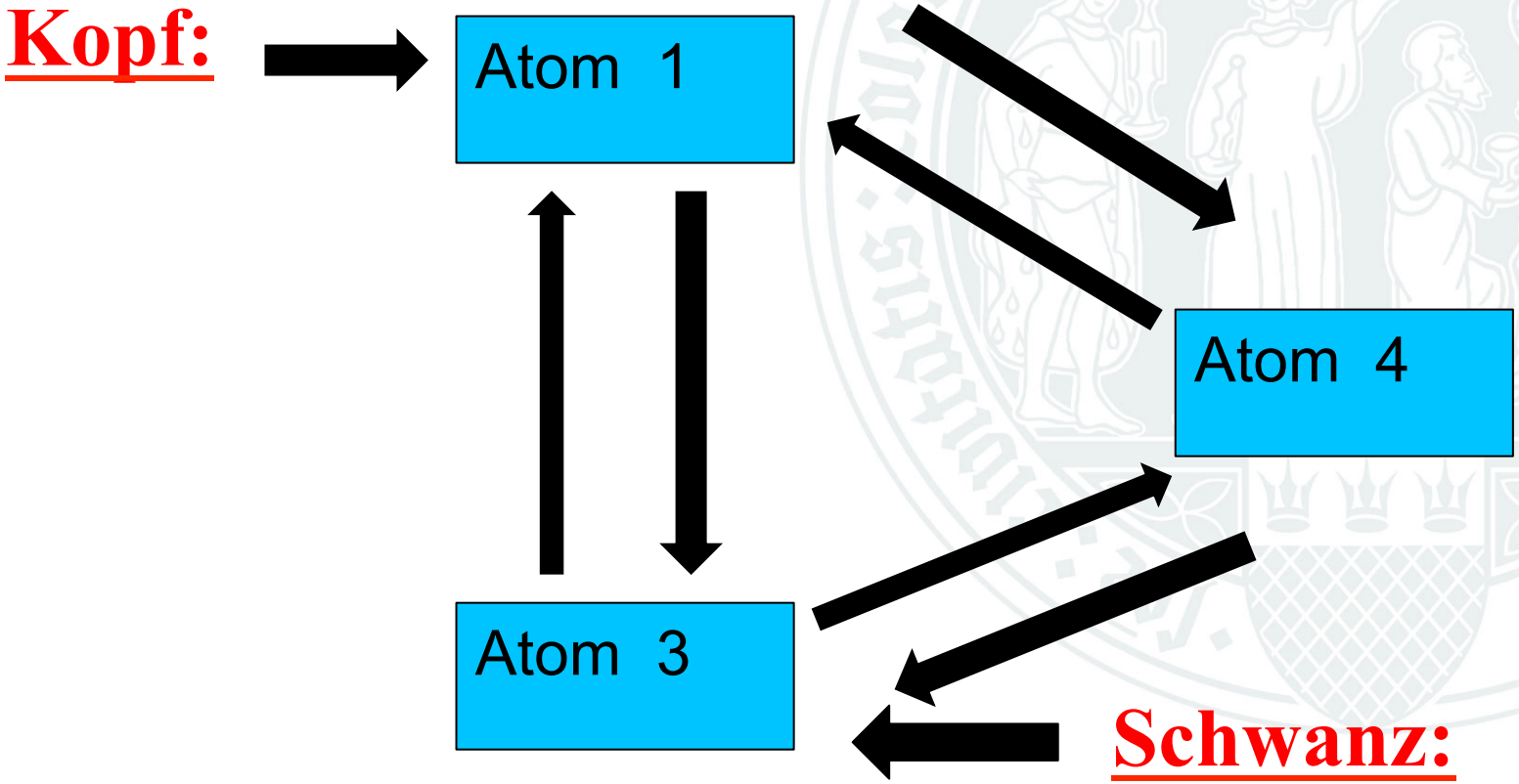
Doppelt Verknüpfte Listen



Löschen von Atom 2



Einfügen von Atom 4



Inhaltliche Datenstrukturen

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

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**

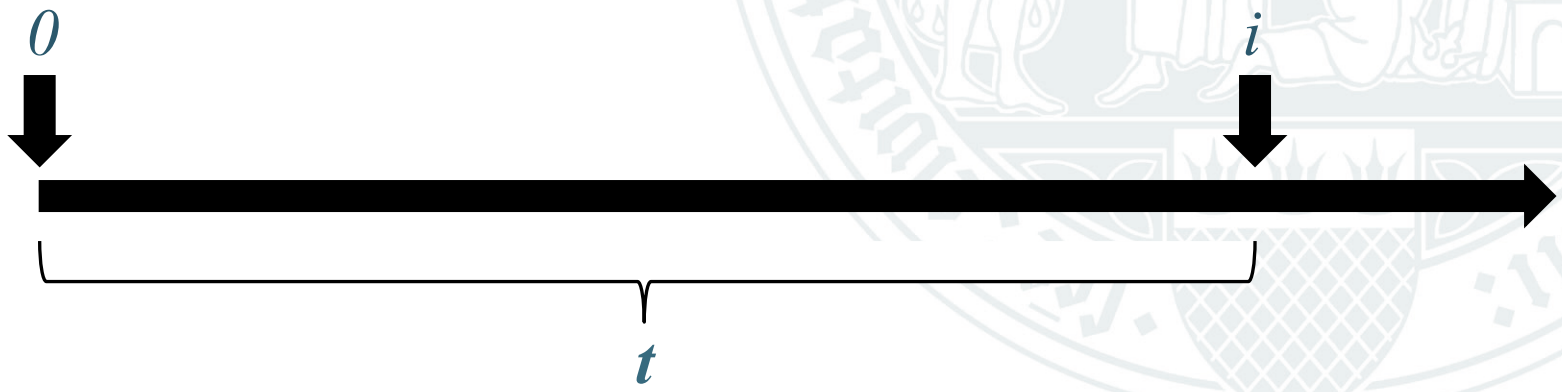
t-less(pri non jun 2007,non jun 2007) ===> **True**

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

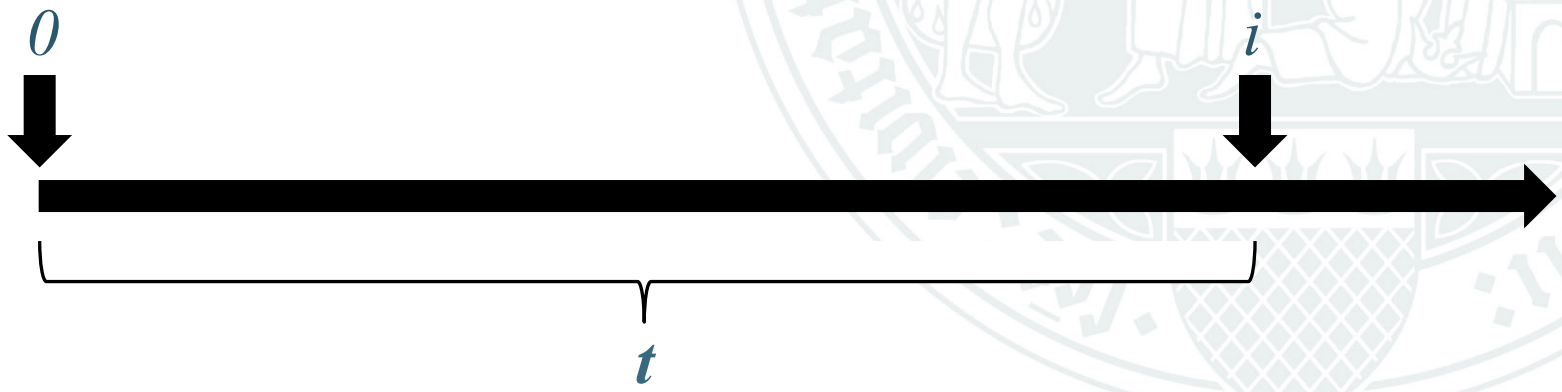
E Regel für "6 Tammuz 5763"
I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag i als Offset t vom Ursprung definiert ist.

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

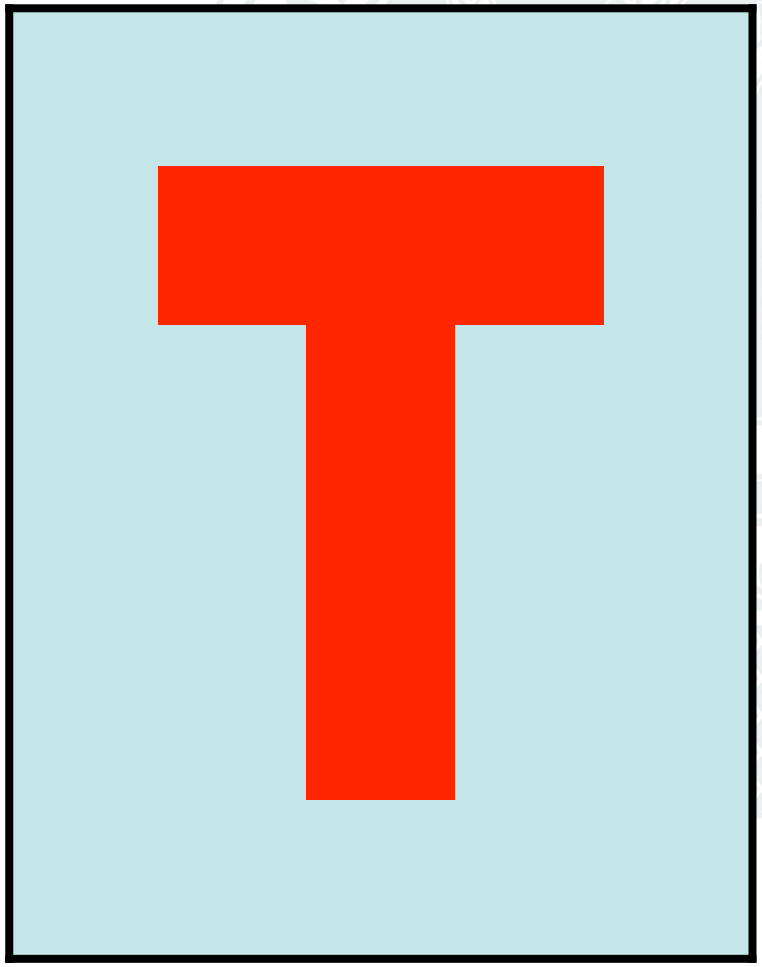
I Zeit ist ein Vektor von Tagen seit einem willkürlichen Tag 0, wobei ein beliebiger Tag *i* als Offset **t** vom Ursprung definiert ist.



Datenstruktur aus dem Softwarengineering

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

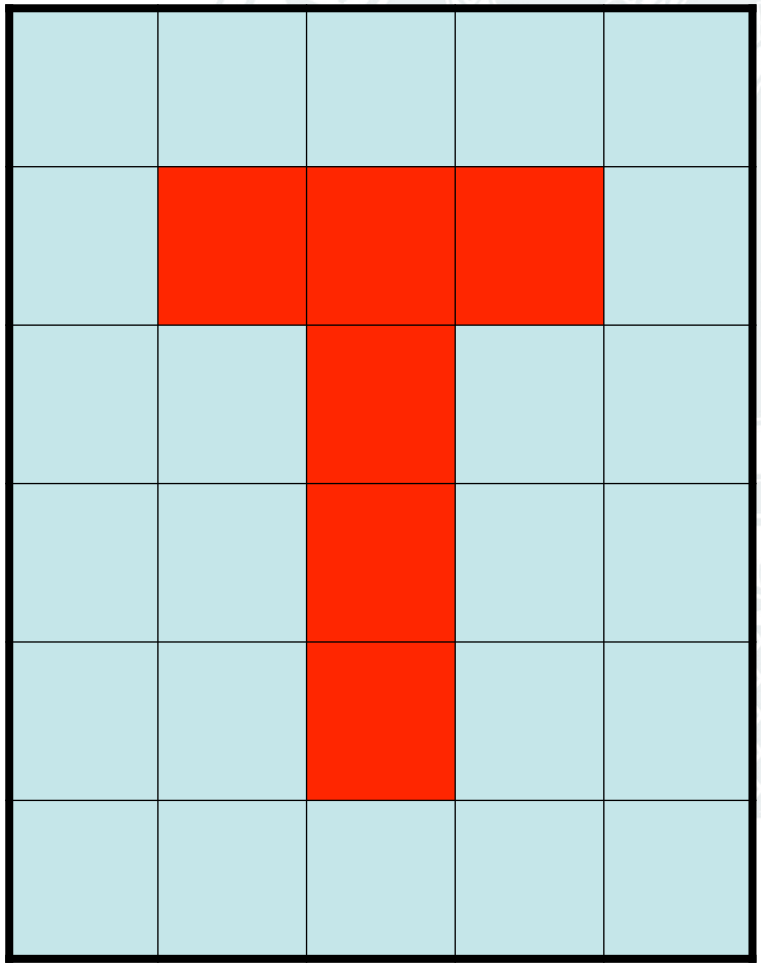
Ein Bild



Ein Bild

6 Zeilen

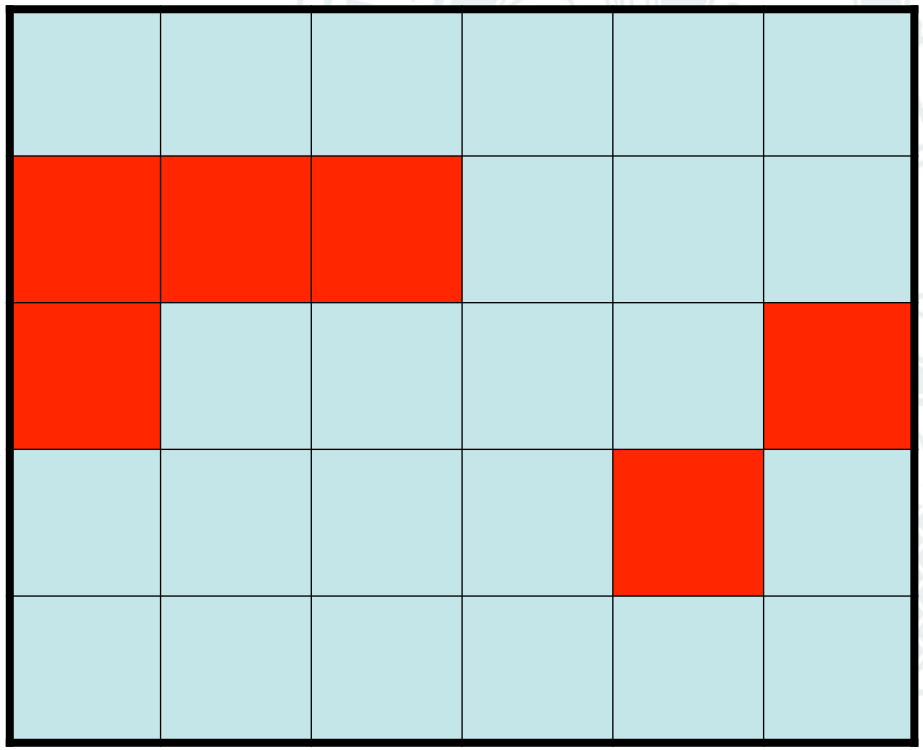
5 Spalten



Ein Bild

5 Zeilen

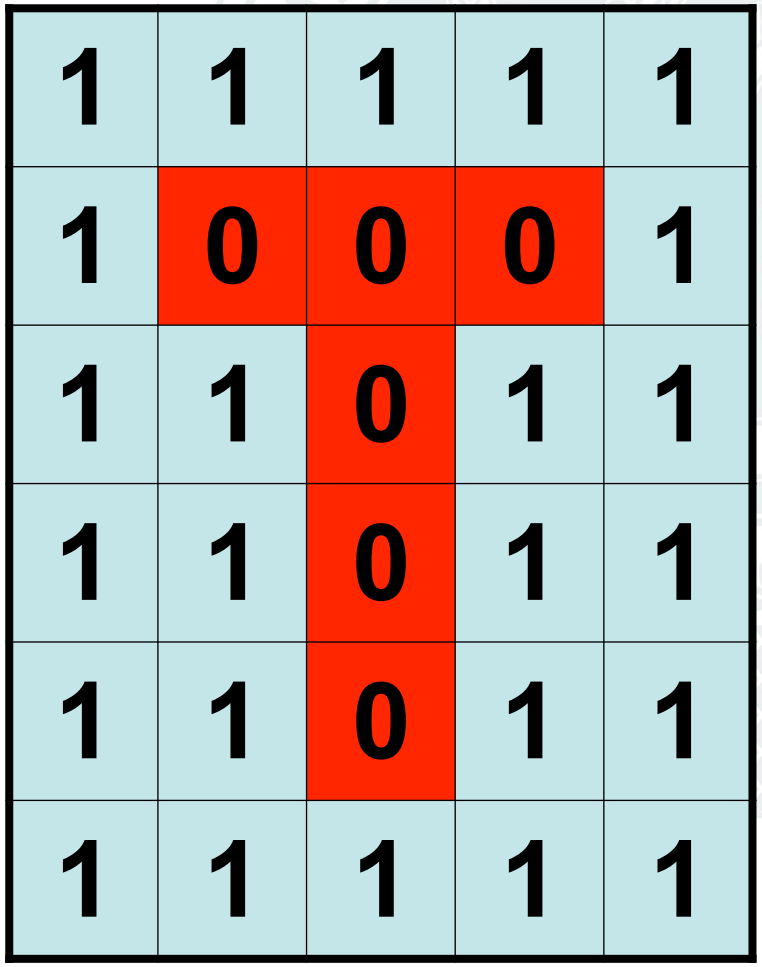
6 Spalten



Ein Bild

1 == grau

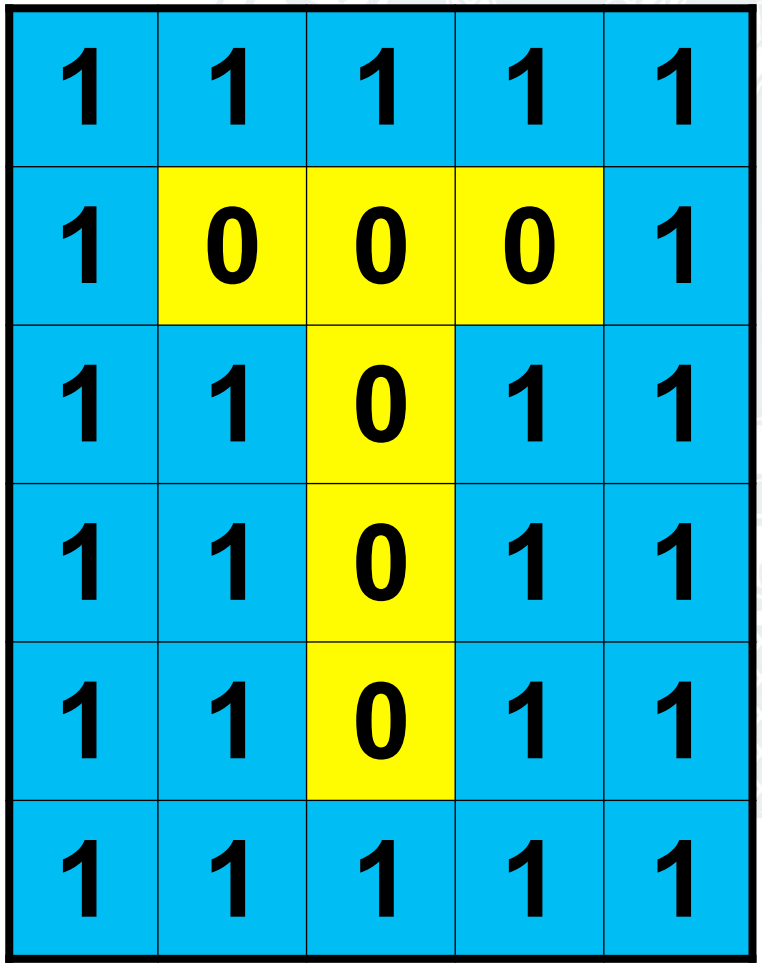
0 == rot



Ein Bild

1 == blau

0 == gelb

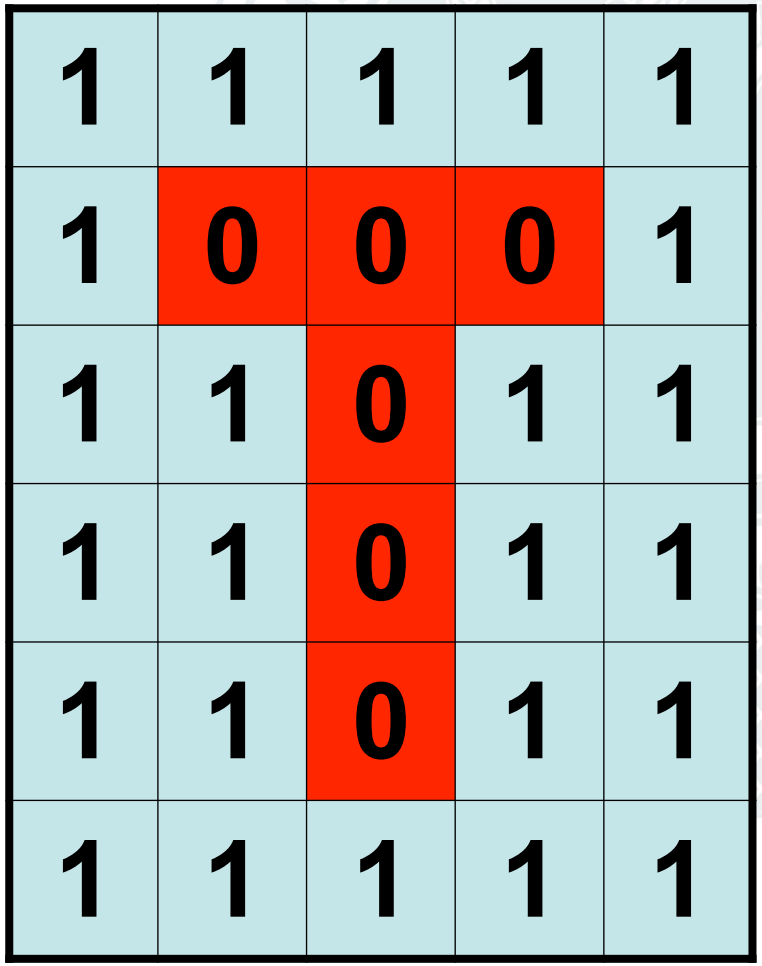


Ein Bild

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

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

Vector Format

Ein Bild

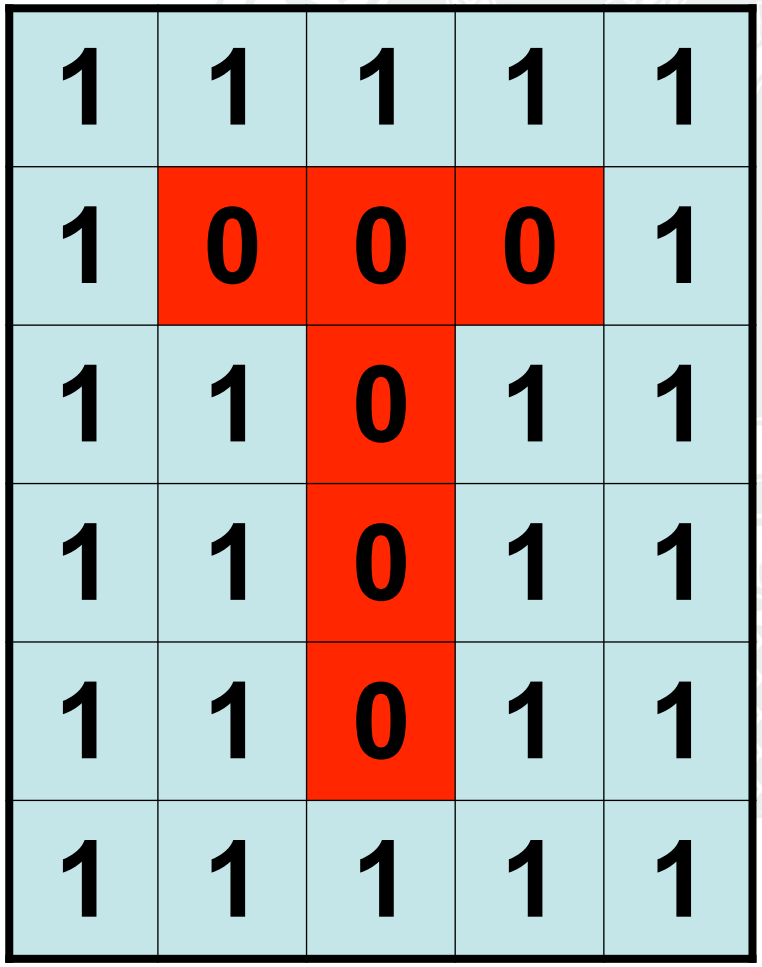
6 Zeilen

5 Spalten

1 == grau

0 == rot

Unkomprimiert



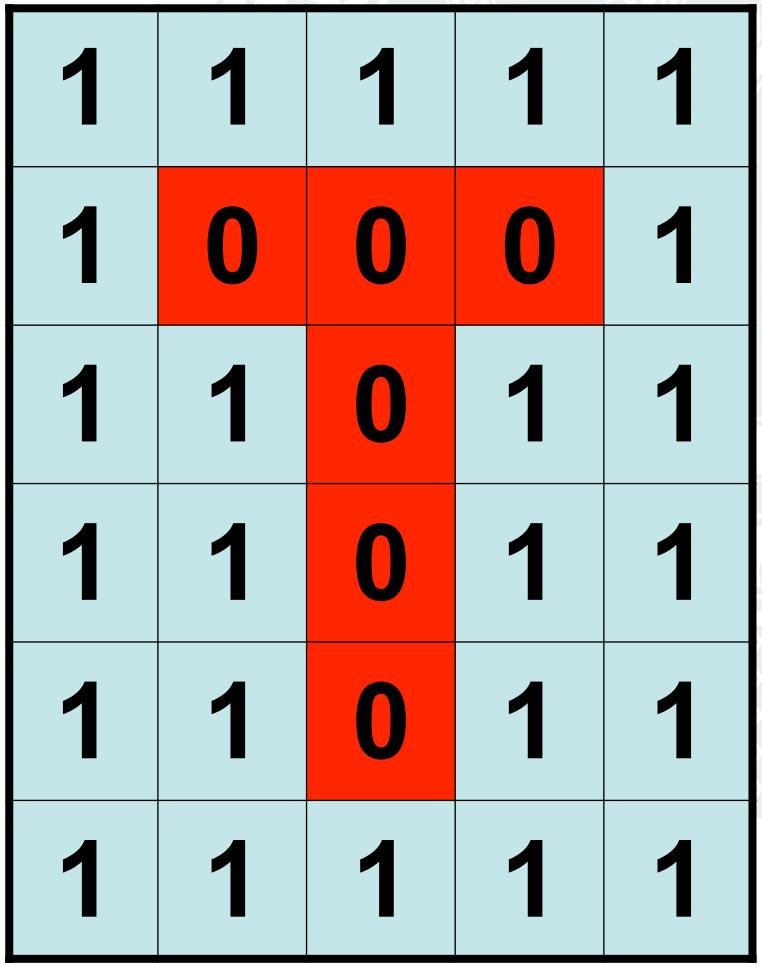
Ein Bild

Dimensionen

1 == gray

0 == red

Unkomprimiert

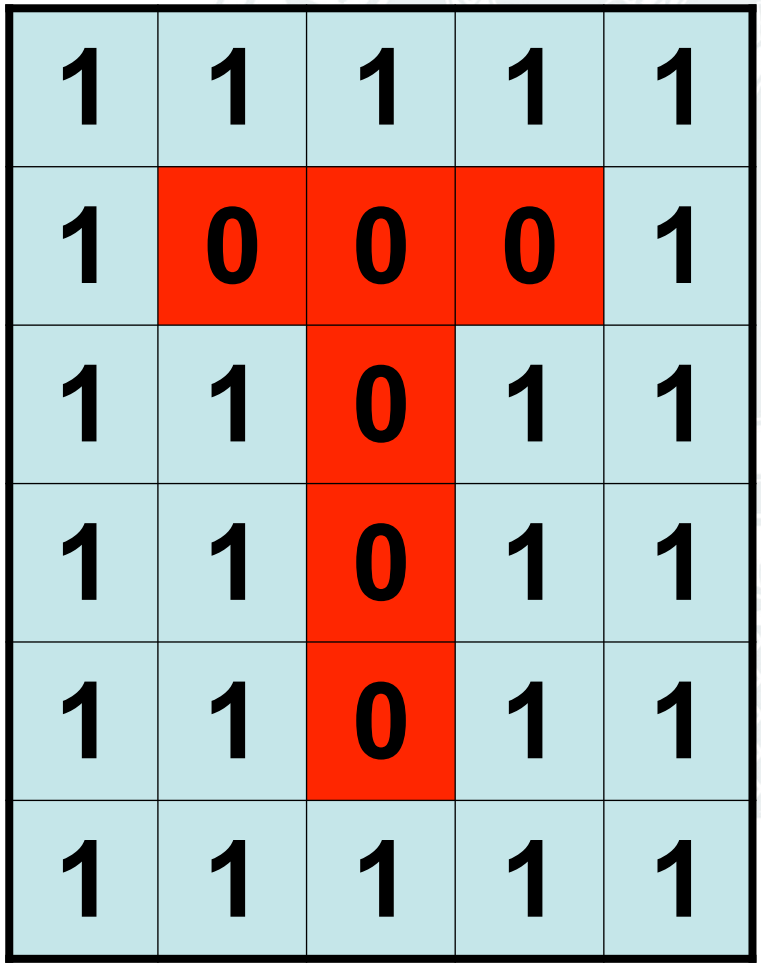


Ein Bild

Dimensionen

Photogrammetrische Interpretation

Unkomprimiert

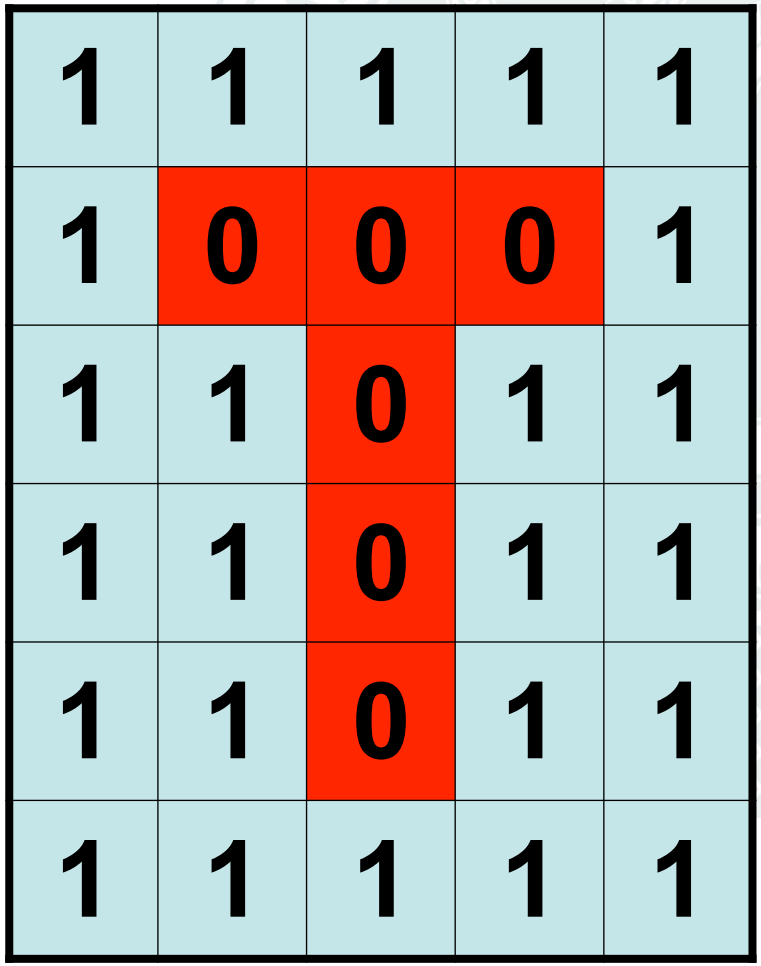


Ein Bild

Dimensionen

Photogrammetrische Interpretation

Kompressionstechnik

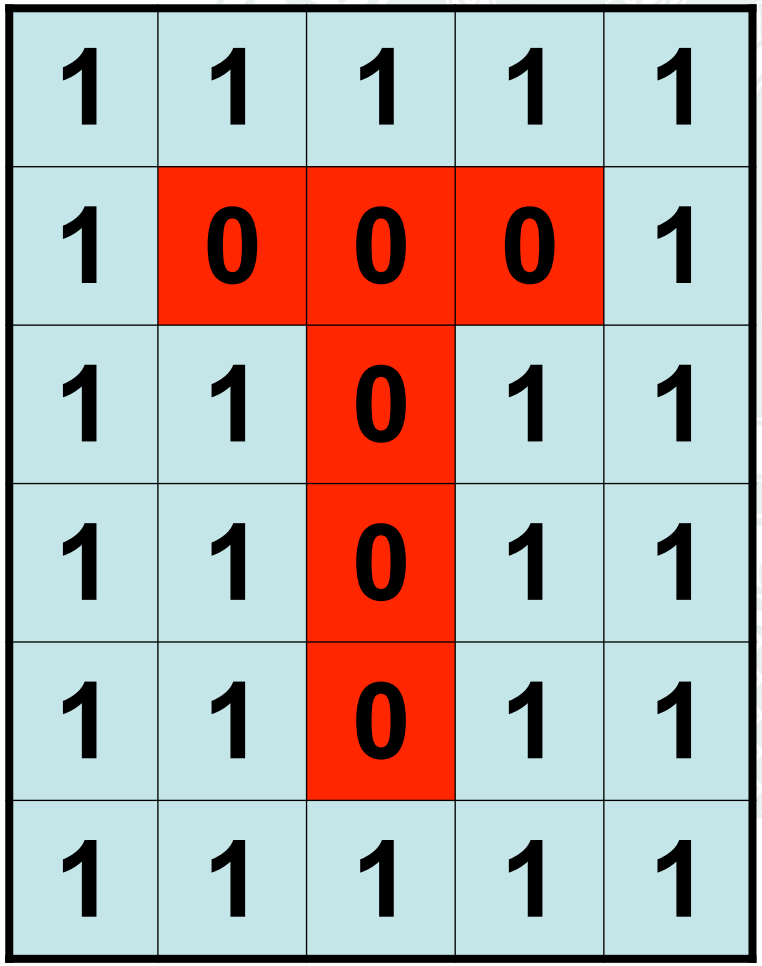


Ein Bild

<basic information>

<rendering information>

<storage information>



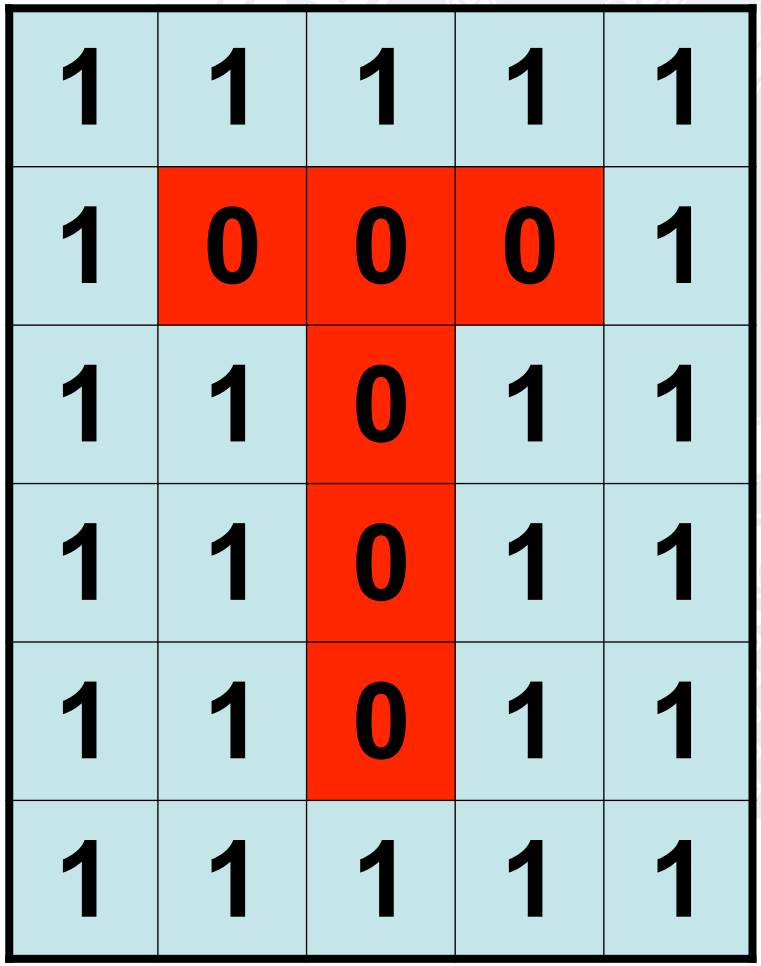
Ein Bild

<basic information>
(implizit / explizit)

<rendering information>
(implicit / explicit)

<storage information>
(implicit / explicit)

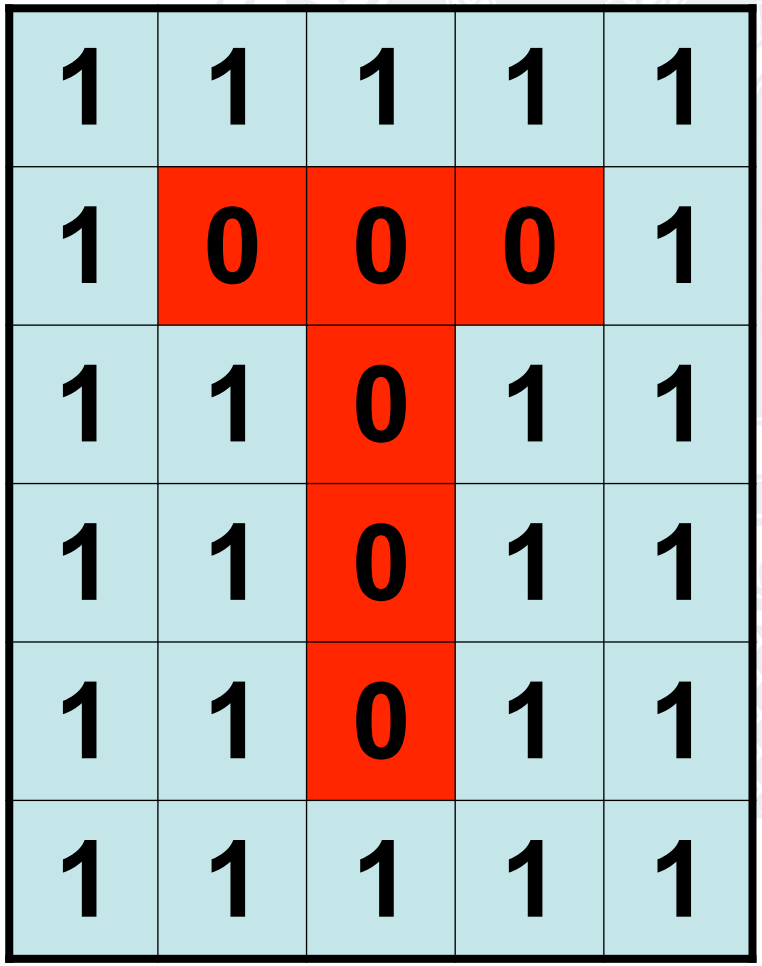
… und die Daten?



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

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

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

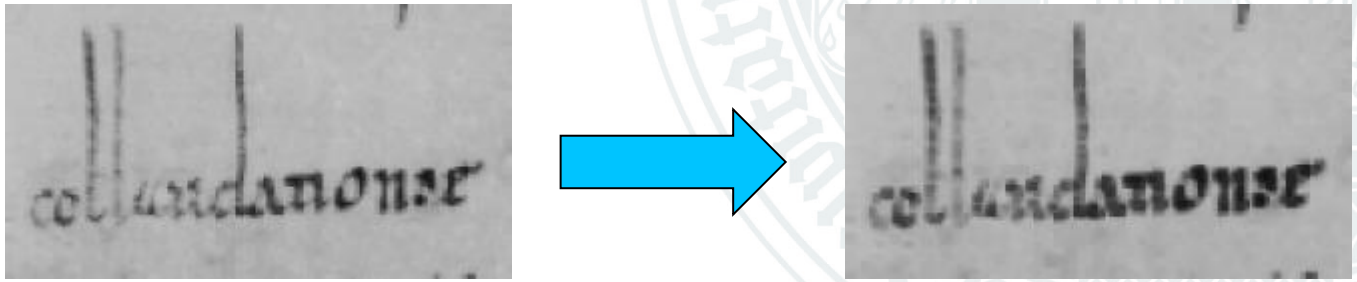
== 52 803.4 8-Stunden-Tage für einen Menschen

== Völlig lächerlich, dass das klappt

Einleitendes Beispiel (Selbstabbildende Information)

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

Minimal neighbour



Original Ergebnis

Minimal neighbour

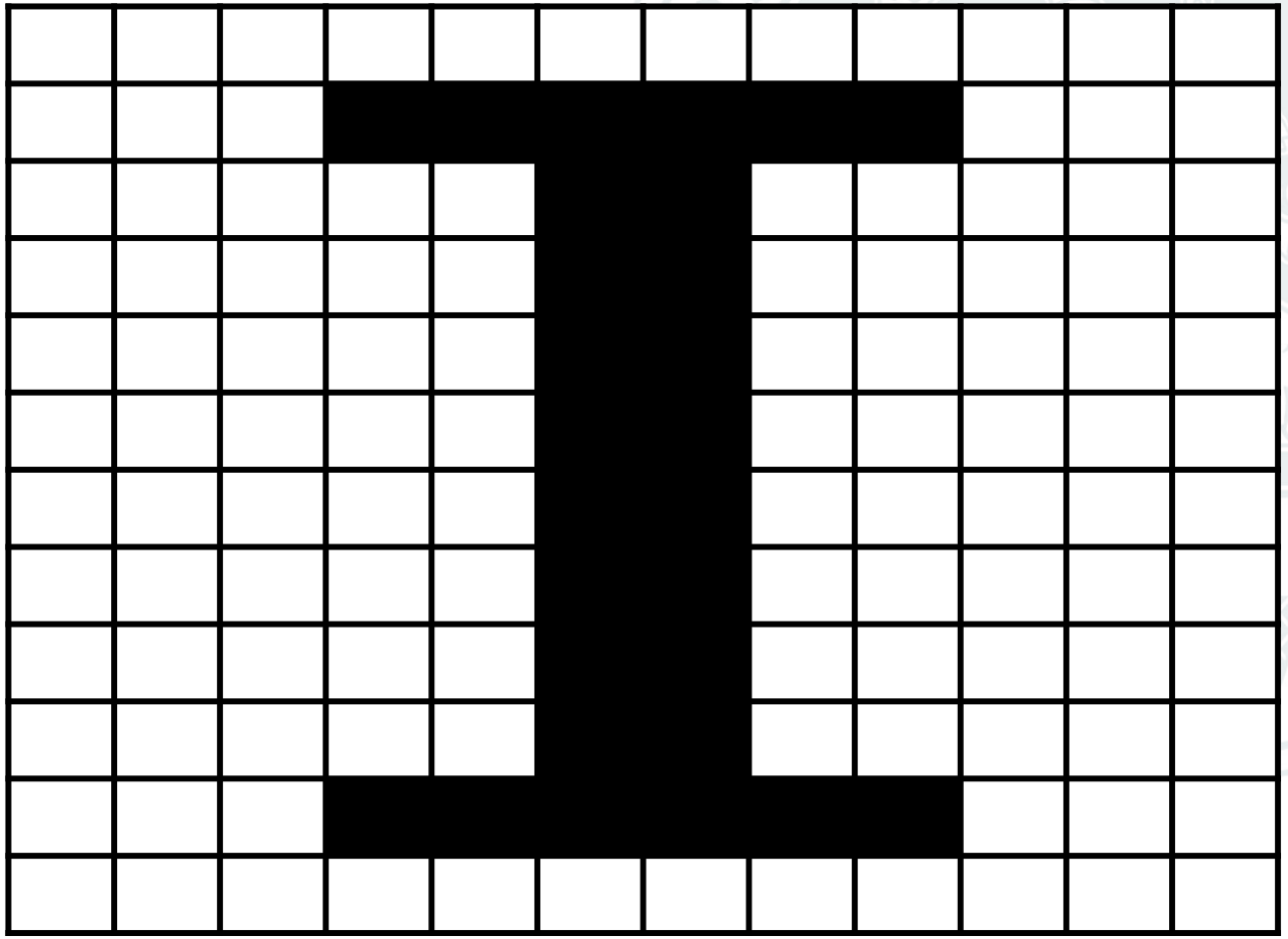
Ersetze in jeder Zeile

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

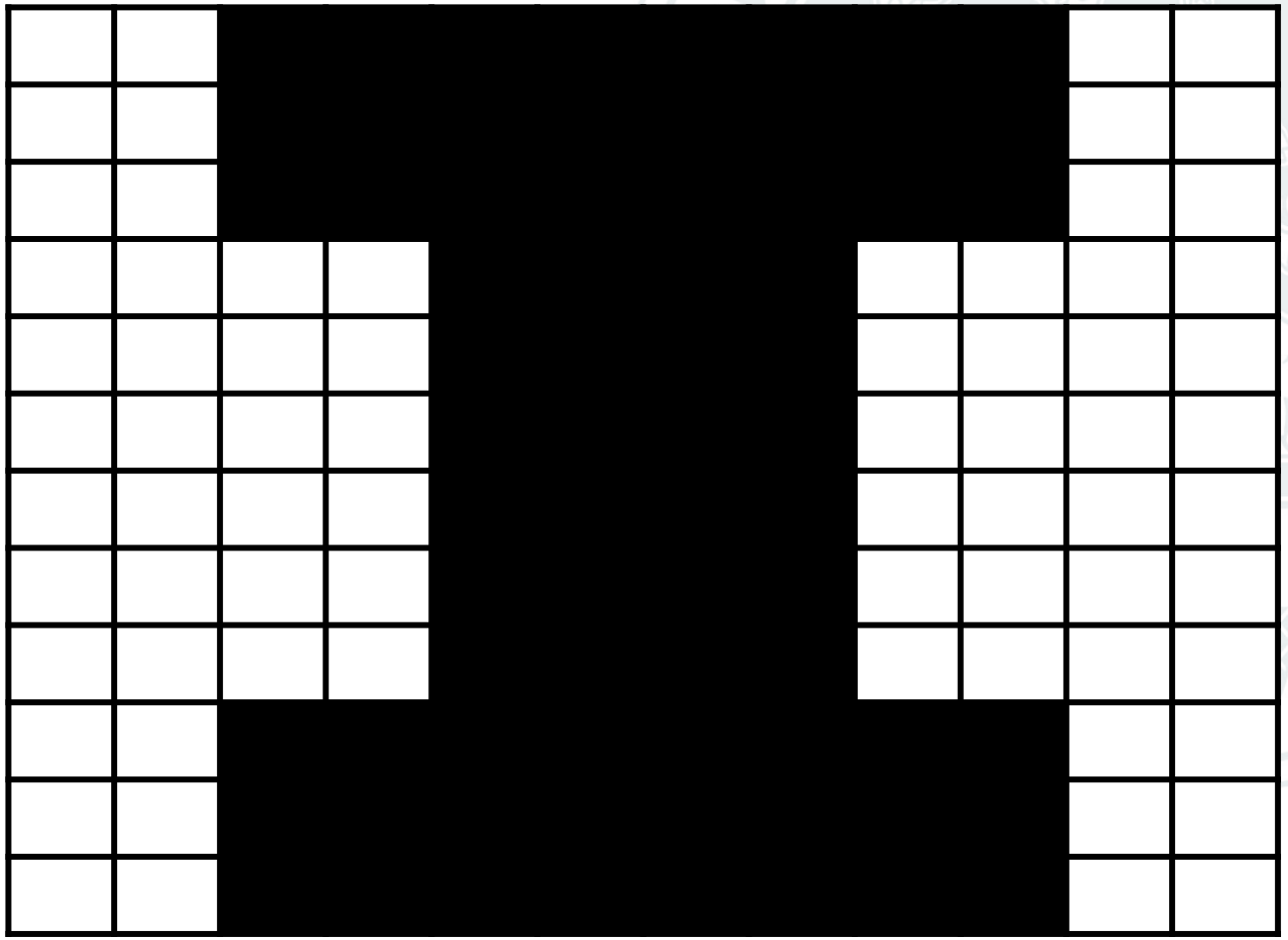
Minimal neighbour

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

Universität zu Köln
Digital Humanities – Historisch-Kulturwissenschaftliche Informationsverarbeitung
Prof. Dr. Eide

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.