Einführung in die Informatik

Vorkurs WiSe 24/25, Timo Skrobanek

Inhalt

- Was ist Informatik?
- Was ist ein Algorithmus?
- Landau-Notation
- Datenstrukturen
- Was ist ein Betriebssystem?
- Grundlagen der technischen Informatik
- Tipps für die Vorlesungen

Kurz zu den Quellen

Dieser Vortrag basiert auf vielen Quellen, die auf den jeweiligen Folien angegeben sind.

Teilweise sind auch Folien oder Ausschnitte von Prof. Schulz aus der Vorlesung "Algorithmen und Datenstrukturen 1" vorhanden.

Was ist die Informatik?

Definition 1.1

Die Wissenschaft der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen

Könnte ich die Bedeutung in 90min erklären, müsste keiner von uns 3+ Jahre studieren :D

Was ist die Informatik?

Technische Informatik
(Architekturen, ...)

IT-Sicherheit
(Kryptographie,...)

Software-Engineering
(Umsetzung von Projekten,...)

Robotik

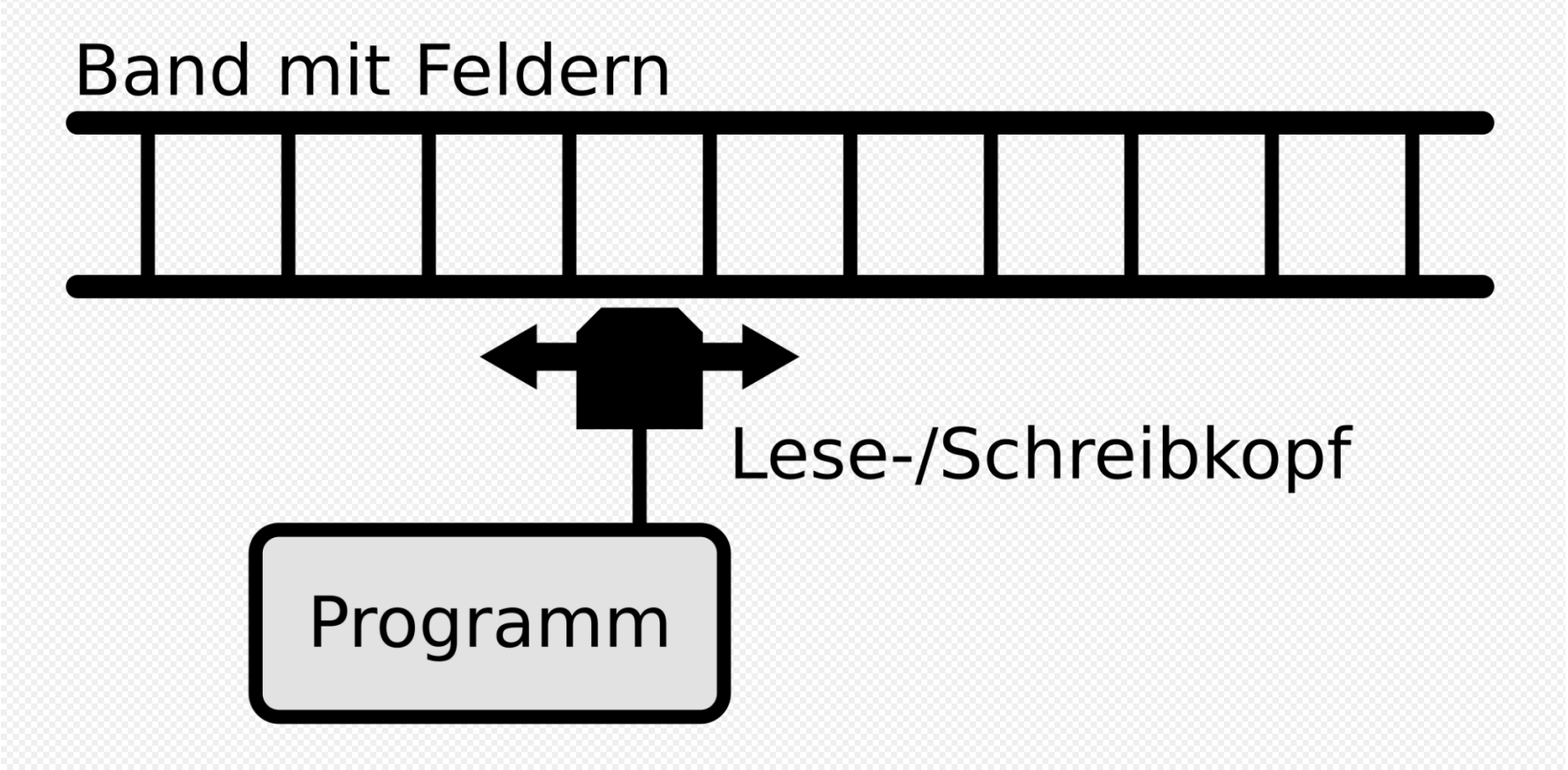
Theoretische Informatik
(Berechenbarkeitstheorie,...)

Computer Visualization
(OpenGL,...)

Und noch viiiiiiiiiele andere Bereiche...

Turingmaschine

Die Anfänge der Informatik



*https://de.wikipedia.org/wiki/Turingmaschine

Turingmaschine

Die Anfänge der Informatik

Vereinfacht: Alle Probleme, die auf einer Turingmaschine berechenbar sind, können mit heutigen Systemen dargestellt werden

*https://de.wikipedia.org/wiki/Turingmaschine

Was ist die Informatik?

Programmierung einer Messenger-App

Netzwerk zur Kommunikation (Routing, Sicherheit,...)

Schnittstelle zum Empfangen, Verarbeiten und Darstellen der Nachricht (Protokolle)

Was ist ein Algorithmus?

Ein Algorithmus ist eine Beschreibung zur Lösung eines spezifischen Problems anhand festgelegter Schritte

Kuchen backen:

1. Zutaten vorbereiten
2. Teig zubereiten
3. Ofen vorheizen
4. ...
5. Essen

Was ist ein Algorithmus?

Definition 2.1

Ein Algorithmus ist eine endliche Menge an elementaren Schritten zur Lösung eines Problems.

$\Rightarrow$ Es gibt Probleme die nicht/teilweise lösbar (Siehe P=NP)

Laufzeiten

- Wie schnell ist ein Algorithmus?
- Wie viel Speicherplatz braucht er?
- Wie viele Speicherbereiche muss er verwenden?

Landau-Notation

Definition 2.2

$$\mathrm{O}(f(n)) = \{g(n) : \exists c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$$

„höchstens“

$$\Omega(f(n)) = \{g(n) : \exists c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$$

„mindestens“

$$\Theta(f(n)) = \mathrm{O}(f(n)) \cap \Omega(f(n))$$

„genau“

$$o(f(n)) = \{g(n) : \forall c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$$

„weniger“

$$\omega(f(n)) = \{g(n) : \forall c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$$

„mehr“

=> Wir schauen uns nicht alles an!

*Professor Schulz - "Algorithmen und Datenstrukturen 1", SoSe 23

Landau-Notation

Definition 2.2

$$O(f(n)) = \{g(n) : \exists c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \leq c \cdot f(n)\}$$

„höchstens“

$$\Omega(f(n)) = \{g(n) : \exists c > 0 : \exists n_0 \in \mathbb{N}_+ : \forall n \geq n_0 : g(n) \geq c \cdot f(n)\}$$

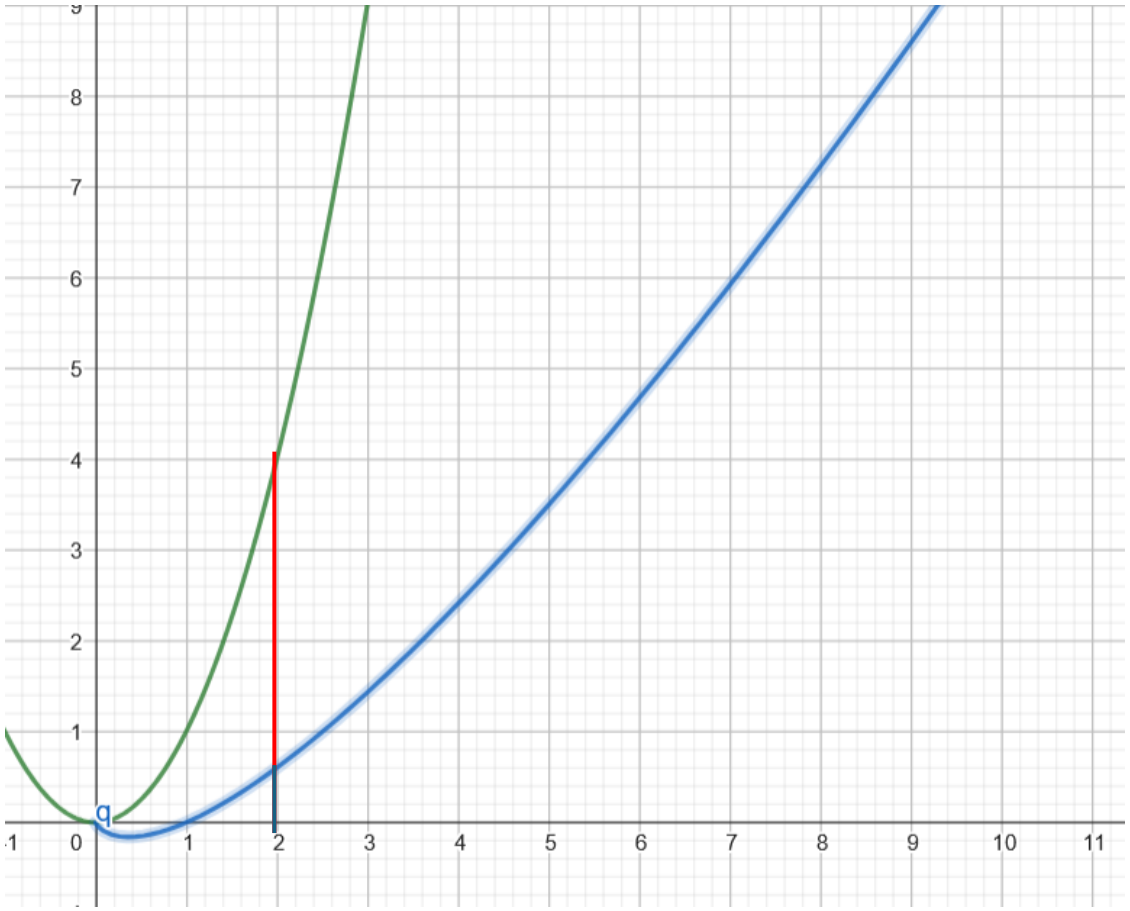
„mindestens“

Etwas einfacher:

O(f(n)) = „Ein Algorithmus hat höchstens f(n) Zeitaufwand“
Ω(f(n)) = „Ein Algorithmus hat mindestens f(n) Zeitaufwand“

*Professor Schulz - "Algorithmen und Datenstrukturen 1", SoSe 23

Etwas anschaulicher



Grün (Bsp Bubble sort) => $O(n^2)$

Blau (Bsp Quicksort) => O(n*logn)

Beispiel 2.3

Es gilt $n*\log n = O(n^2)$

Beweis der Laufzeit

Beispiel 2.4

Es gilt n*logn= $O(n^2)$

Beweis: Sei n∈**N** und c ∈**R** mit c > 0

O(n*logn) => g(n) ≤ c * n*logn

Also können wir folgendes nach der Landau-Notation aufstellen

n*logn = $O(n^2)$

n*logn ≤ $c*n^2$

$\frac{logn}{n} \leq c$

Somit existiert ein c ∈**R** mit c>0, wodurch die gezeigte Ungleichung gültig ist.

Beweis der Laufzeit

Beweise von Laufzeiten sind oft in Klausuren von „Algorithmen und Datenstrukturen 1“ zu sehen.
Somit würdet ihr schon jetzt nicht mehr mit 0 Punkten rausgehen!

Beispiel für Algorithmen

Definition 2.5

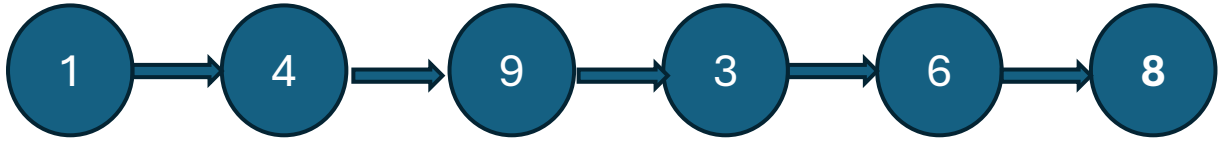
Ein Suchalgorithmus hat die Aufgabe, einen bestimmten Wert in einer Menge von Elementen zu finden.

Binary-Search, Linear-Search, Quick-Select

Linear-Search

A={1,4,9,3,6,8,10}

Aufgabe: Suche mithilfe von Binary-Search nach dem Element 8

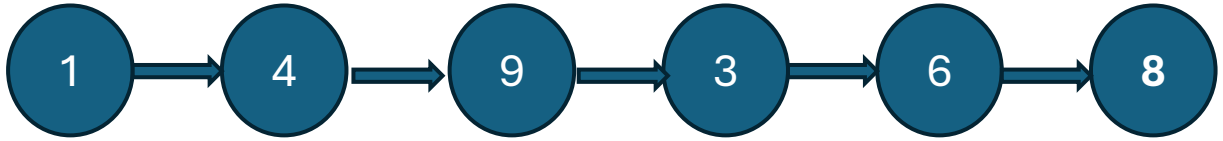


Also Linear-Search braucht hier 6 Schritte, um die 8 zu finden.

Online-Rechner für B-Search:
https://www.cs.usfca.edu/~galles/visualization/Search.html

Linear-Search

A={1,4,9,3,6,8,10}



Aufgabe: Suche mithilfe von Binary-Search nach dem Element 8

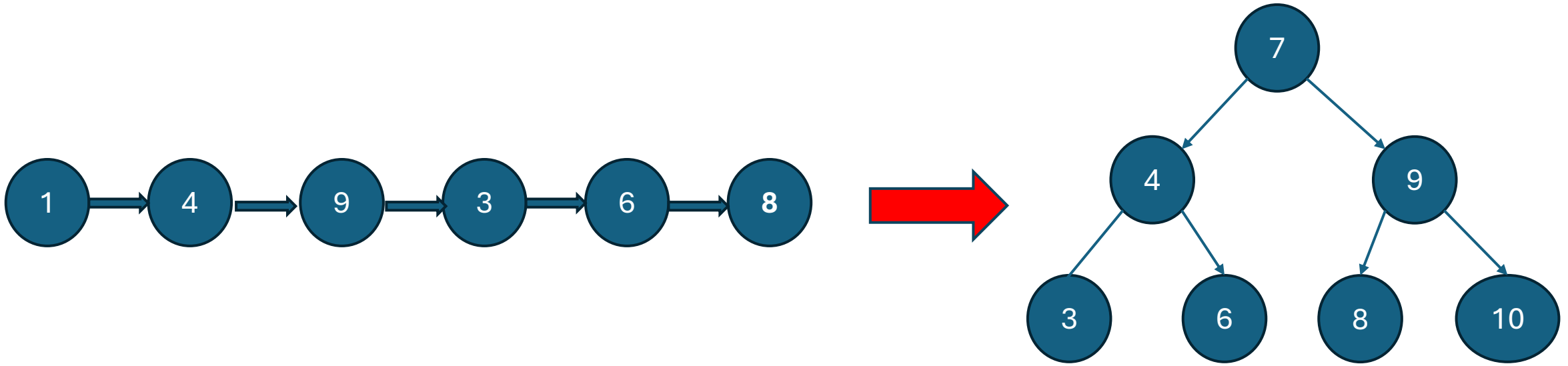
```
for i in range(10):
        if list[i]==gesucht:
                return list[i]
return NULL
```

Online-Rechner für B-Search:
https://www.cs.usfca.edu/~galles/visualization/Search.html

*Professor Schulz - "Algorithmen und Datenstrukturen 1", SoSe 23

Bäume

... gibt es nicht nur in der Natur



Binary-Search

A={1,4,9,3,6,8,10}

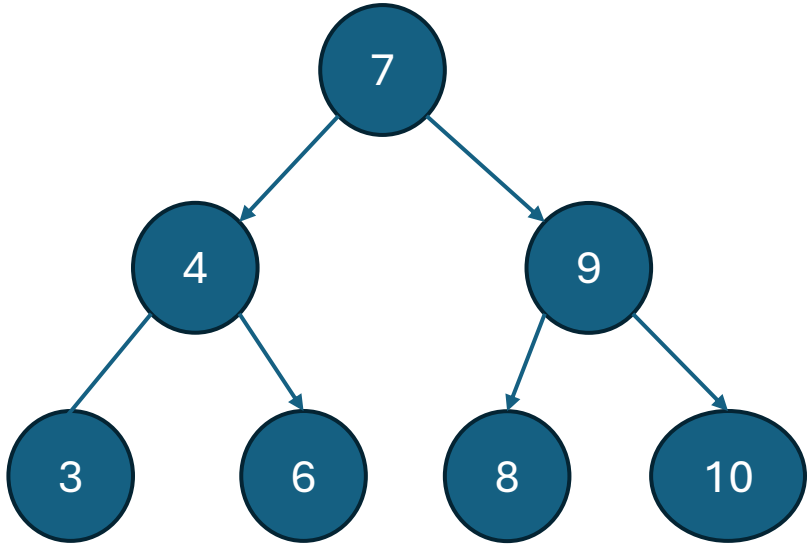
Aufgabe: Suche mithilfe von Binary-Search nach dem Element 8

Online-Rechner für B-Search:
https://www.cs.usfca.edu/~galles/visualization/Search.html

Binary-Search

- Binary Tree erstellen
- Suchen nach dem Element

Aufgabe: Suche mithilfe von Binary-Search nach dem Element 8



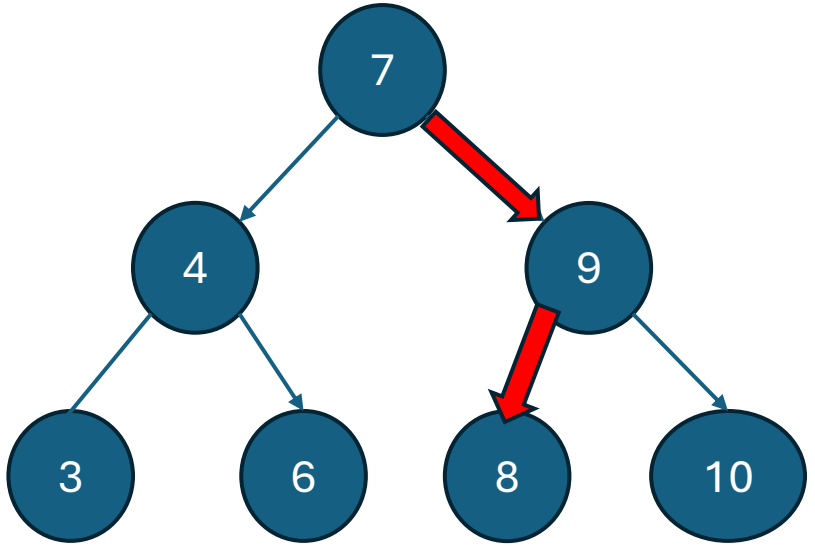
Online-Rechner für B-Search:
https://www.cs.usfca.edu/~galles/visualization/Search.html

Binary-Search

- Binary Tree erstellen
- Suchen nach dem Element

A={1,4,9,3,6,8,10}

Aufgabe: Suche mithilfe von Binary-Search nach dem Element 8

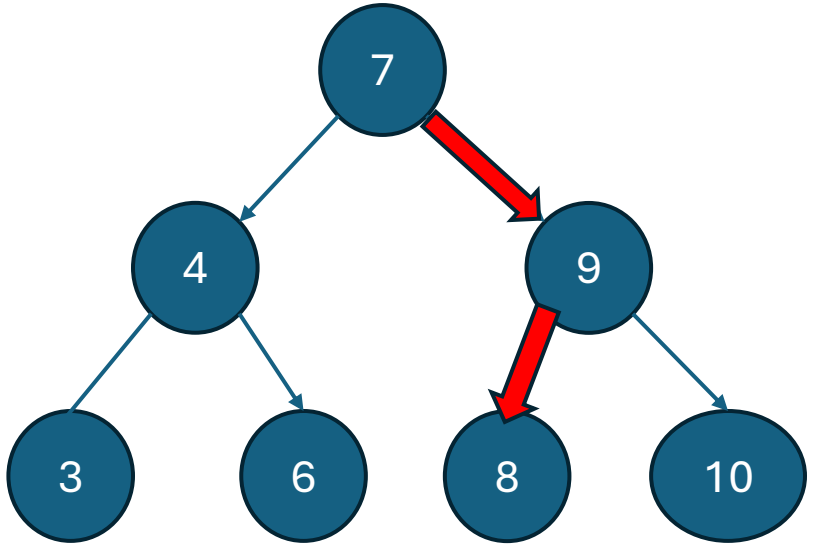


Binary-Search braucht hier 3 Schritte

Online-Rechner für B-Search:
https://www.cs.usfca.edu/~galles/visualization/Search.html

Binary-Search

```
min = 0
max = x-1
while min <= max:
        i=math.ceil((max+min)/2.0)
        if gesucht > list[i]:
                min = i+1
        elif gesucht < list[i]:
                max = i-1
        else:
                return list[i]
return NULL
```



*Professor Schulz - "Algorithmen und Datenstrukturen 1", SoSe 23

Beispiel für Algorithmen

Definition 2.6

Ein Sortieralgorithmus soll eine Menge von n Elementen nach vorgegebenen Kriterien in eine gewünschte Reihenfolge bringen.

Bubble-Sort, Insertion-Sort, Quicksort, Merge-Sort

=> Hier betrachten wir vergleichsbasierte Algorithmen. Diese können maximal O(n*logn) erreichen.

Insertion-Sort

Aufgabe: Sortiere die Liste A={2,5,1,4,9,7}

{2,5,1,4,9,7} | Erstes Element wird als „sortiert“ angesehen
{2,5,1,4,9,7} | 5 wird in die sortierte Liste eingefügt
{1,2,5,4,9,7} | 1 wird …
{1,2,4,5,9,7} | 4 wird…
{1,2,4,5,9,7} | …
{1,2,4,5,7,9} | Liste ist fertig sortiert

Insertion-Sort

Ein paar Zeilen code dazu...

```
Procedure insertionSort(a : Array [1..n] of Element)
    for i := 2 to n do
        invariant a[1] ≤ ··· ≤ a[i − 1]
        // move a[i] to the right place
        e:= a[i]
        if e < a[1] then                          // new minimum
            for j := i downto 2 do  a[j]:= a[j − 1]
            a[1]:= e
        else                                      // use a[1] as a sentinel
            for (j := i;  a[j − 1] > e;  j−− )  a[j]:= a[j − 1]
            a[j]:= e
```


*Professor Schulz - "Algorithmen und Datenstrukturen 1", SoSe 23

Datenstrukturen

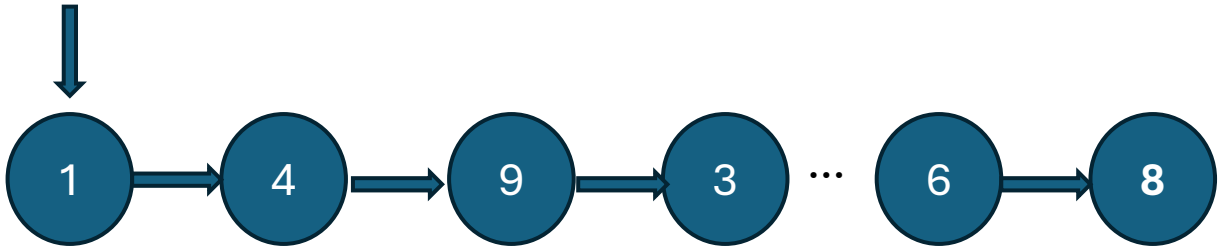
Definition 2.7

Eine Datenstruktur ist eine vordefinierte Struktur zum Speichern von Objekten, welche durch Methoden zur Verwaltung charakterisiert wird.

Heaps, Components, Arrays, Listen,...

Einfach verkettete Liste

First-Pointer



Eine Liste, deren Elemente durch Zeiger auf das jeweils nächste Element erreicht werden können.

Unterschied Liste und Array

Bemerkung 2.8

Listen haben variable Länge, da der letzte Pointer immer auf ein neues Element zeigen kann.

Unterschied Liste und Array

Bemerkung 2.9

Arrays haben eine feste Länge und sind durch einen vordefinierten Bereich im Speicher festgelegt.

Unterschied Liste und Array

Frage: Wenn Arrays feste Größen haben, warum verwenden wir sie überhaupt und nicht eine einfach/doppelt verkettete Liste?

Arrays

Vorteile	Nachteile
• Schnelle Zugriffszeiten O(1)	• Elemente einfügen ist schwierig • Größe ist fest

Liste

Vorteile	Nachteile
• Flexible Größe • Änderungen von Beziehungen einzelner Elemente ist einfach	• Zugriffe auf einzelne Elemente braucht lange O(n)

Stack und Queue

Definition 2.10

Ein Stack ist eine Datenstruktur, welche sich durch das LIFO-Prinzip auszeichnet.

Stack und Queue

=> Beispiel für Stack an der Tafel!

Stack und Queue

Definition 2.11

Eine Queue ist eine Datenstruktur, welche sich durch das FIFO-Prinzip auszeichnet.

Stack und Queue

=> Beispiel für Queue an der Tafel!

Verarbeitungsprinzipien

Definition 2.12

- FIFO = „First in first out“
- LIFO = „Last in first out“

Laufzeiten von Datenstrukturen

Operation	LIST	SLIST	UARRAY	CARRAY	explanation '*'
$[\cdot]$	n	n	1	1	
$\|\cdot\|$	1*	1*	1	1	not with inter-list SPLICE
FIRST	1	1	1	1	
LAST	1	1	1	1	
INSERT	1	1*	n	n	INSERTAFTER only
REMOVE	1	1*	n	n	REMOVEAFTER only
PUSHBACK	1	1	1*	1*	amortized
PUSHFRONT	1	1	n	1*	amortized
POPBACK	1	n	1*	1*	amortized
POPFRONT	1	1	n	1*	amortized
CONCAT	1	1	n	n	
SPLICE	1	1	n	n	
FINDNEXT,...	n	n	n*	n*	cache-efficient

=> Noch unwichtig für euch

Alle Operationen haben unterschiedliche Kosten.

Wie genau sich die Kosten zusammensetzen lernt ihr in „Algorithmen und Datenstrukturen 1“

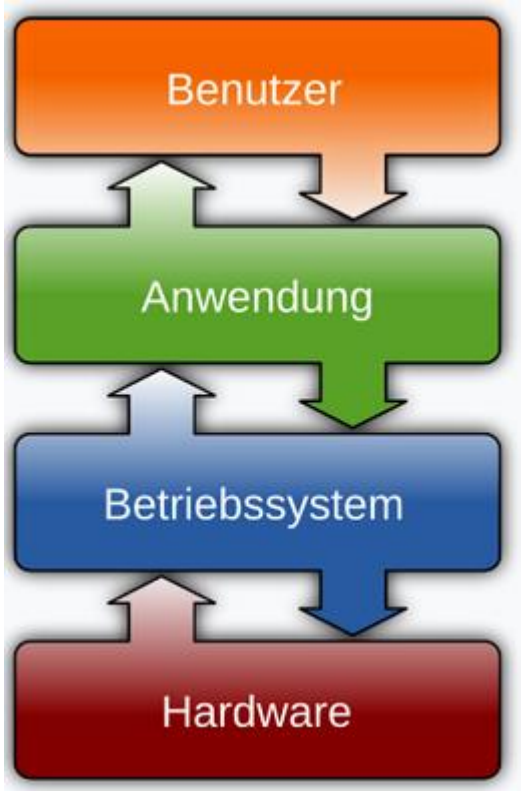
*Professor Schulz - "Algorithmen und Datenstrukturen 1", SoSe 23

Was ist ein Betriebssystem?

Definition 3.1

Ein Betriebssystem verwaltet Ressourcen und Rechte für den Benutzer.

Was ist ein Betriebssystem?



Peripheriegeräte, Prozessverwaltung, Rechteverwaltung(Admin,...)

<u>Aufgaben sind:</u>
- Abstraktion (Erweiterte Maschine)
- Ressourcenverwaltung
- Komplexität der Hardware "verstecken"

*https://de.wikipedia.org/wiki/Betriebssystem

Moore's Law

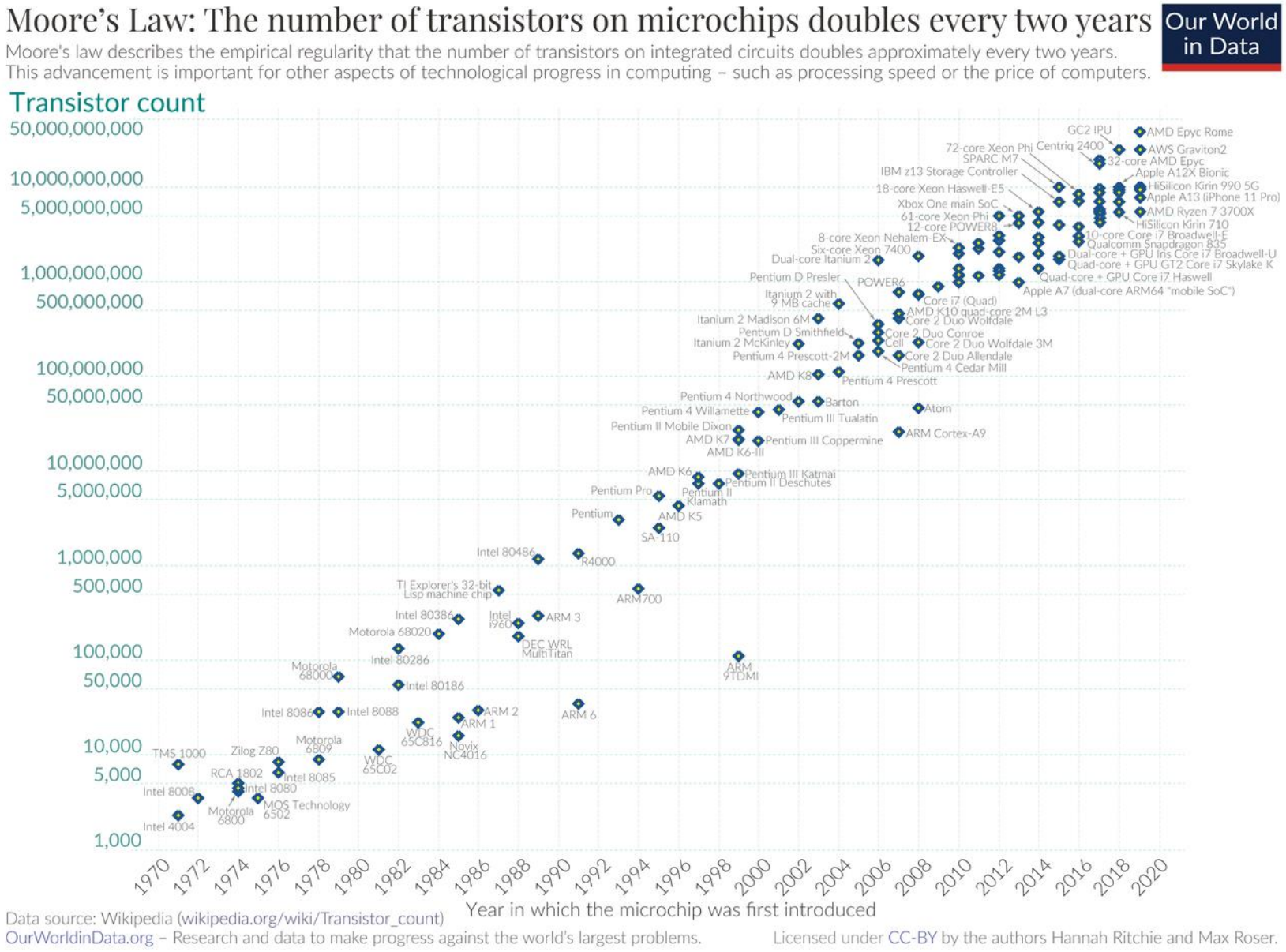
Definition 3.2

The density of transistors on a chip doubles every 18 months, for the same cost.

*https://en.wikipedia.org/wiki/Moore%27s_law

Moore's Law

War lange akkurat, langsam sieht man aber Abweichungen



*https://en.wikipedia.org/wiki/Moore%27s_law

Zahlensysteme

Hexadezimal(0-15) => 1,2,...,9,A,B,C,..
Dezimalsystem (0-9)
Oktalsystem(0-7)
Dualsystem(0-1)

=> Beispiel dazu an der Tafel

Zahlensysteme

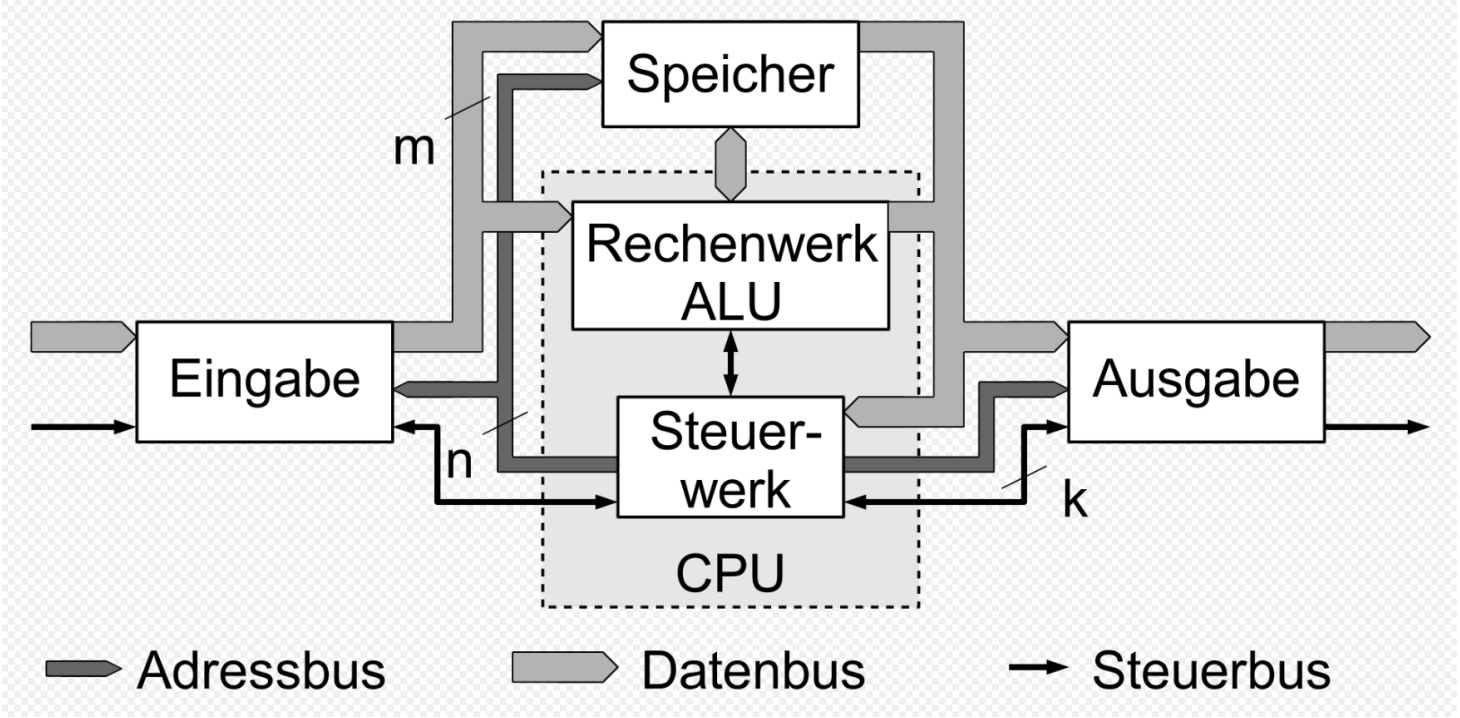
Bemerkung 3.3

Zahlensysteme sind nützlich um Daten mit verschiedener Repräsentationen darzustellen.

=> In "Einführung in die Technische Informatik" werdet ihr auch lernen, wie man zwischen verschiedenen Systemen umrechnet

Architekturen

Heutige Systeme basieren auf der "Von Neumann Architektur"



Stark vereinfacht!

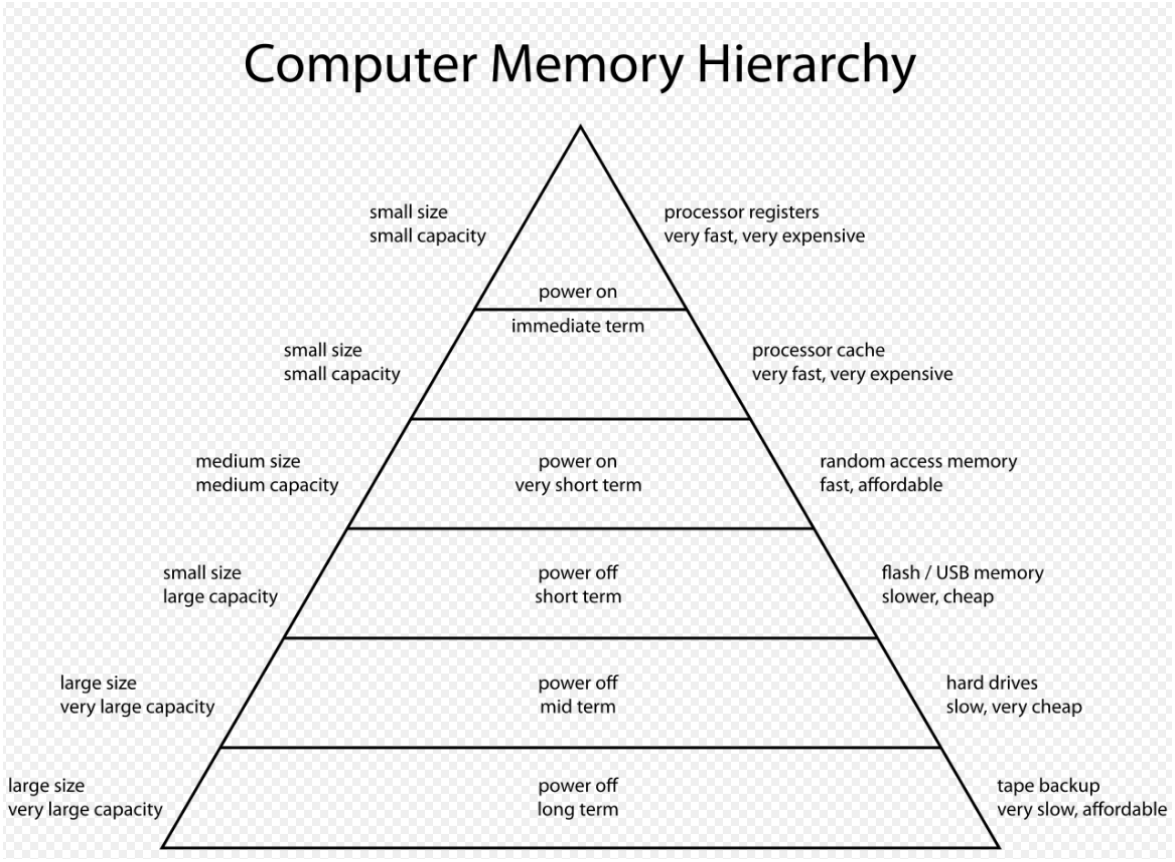
https://de.wikipedia.org/wiki/Von-Neumann-Architektur

Architekturen

Bemerkung 3.4

Architekturen sind Standards für die Entwicklung von Hardware und daraus folgender Software.

Speicherhierarchie



Kleiner Speicher => Schnell
Großer Speicher => Langsam

https://de.wikipedia.org/wiki/Speicherhierarchie

Programmiersprachen

Assembly

```
section     .text
global      _start

_start:

    mov     edx,len
    mov     ecx,msg
    mov     ebx,1
    mov     eax,4
    int     0x80

    mov     eax,1
    int     0x80

section     .data

msg     db  'Hello, world!',0xa
len     equ $ - msg
```

C/C++

```c
#include<stdio.h>
int main()
{
    float a,b,div;
    printf("Enter two numbers: ");
    scanf("%f%f",&a,&b);
    div=a/b;
    printf("The result is = %f",div);
    return 0;
}
```

Python,JScript,...

```python
import random

def get_random_color():
    colors = ['green', 'blue', 'red', 'yellow']
    random_color = random.choice(colors)
    return random color
```

Mit jeder Ebene kamen neue Konzepte: Prozedurale, funktionale, objektorientierte,... Programmierung

*https://de.wikipedia.org/wiki/Programmiersprache

Tipps für die Vorlesungen

- Beginnt frühzeitig mit dem Lernen!
- Nutzt die Tutorien um Fragen zu stellen
- Geht auch ab und zu in die Vorlesung

Viel Erfolg euch beim Studium :D