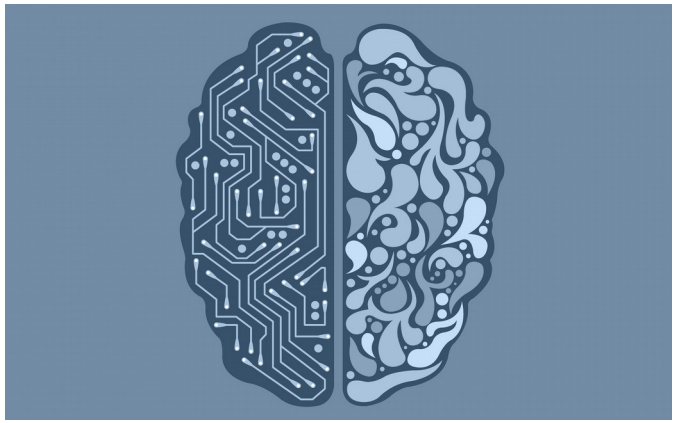
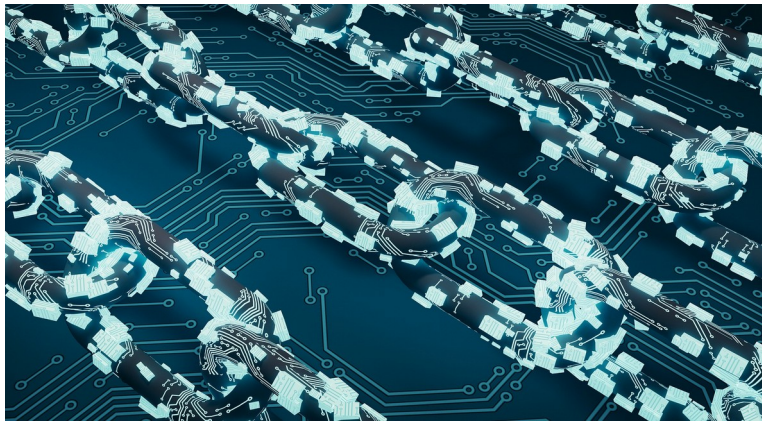
Algorithmen sind überall

Sebastian Forster

Zukunftstechnologien Mitgestalten!



Artificial Intelligence



Blockchain

Internet of Things

Data Science

Algo-was?

- *Programmieren* ist längst nicht alles
- Effiziente Berechnungsmethoden erfordern oft gute Ideen und mathematische Analyse
- Die Suche nach *Algorithmen* beginnt dort, wo mit reiner Programmierfertigkeit nichts mehr schneller gemacht werden kann
- Heute: Ein Algorithmus, den jeder kennt (zumindest im Prinzip)

Suche im Bücherregal der Bibliothek

Einfacher Algorithmus:

Überprüfe für jedes Buch, ob es das gesuchte ist

Einzelne Überprüfung: geringer Aufwand

Aber: in Summe dennoch hoher Aufwand

??

Life Hack #256: Binäre Suche

- Beobachtung: Bücher in der Bibliothek sind alphabetisch nach Autor/in sortiert!
- Idee: Wenn Buch von „Rowling“ gesucht wird und aktuell mit Buch von „King“ verglichen wird, dann können alle Bücher von „Aakhus“ bis „King“ ignoriert werden
- Abstrakt: Alphabetische Ordnung:

  $Buch_1 < Buch_2 < \ldots < Buch_n$

Beispiel für binäre Suche

Suche nach „Rowling“: Vergleiche mit Mitte

Beschreibung des Algorithmus

1. Überprüfe Buch aus der Mitte des Suchbereichs
2. Falls überprüftes Buch = gesuchtes Buch: fertig
3. Falls gesuchtes Buch < überprüftes Buch:
   Schränke Suchbereich auf alle „kleineren“ Bücher ein
4. Falls gesuchtes Buch > überprüftes Buch:
   Schränke Suchbereich auf alle „größeren“ Bücher ein
5. Gehe zu Schritt 1

Analyse der binären Suche

<u>Beobachtung:</u> Größe des Suchbereichs halbiert sich in jedem Schritt

- Anfangsgröße: n
- Nach einem Schritt: Größe n/2
- Nach zwei Schritten: Größe n/4
- Nach k Schritten: Größe $n/2^k$
- Nach $\log_2(n)$ Schritten: Größe 1
- Buch wird spätestens nach $\log_2(n)$ Schritten gefunden!

$\log_2(n)$ ist Umkehrung von 2^n

Signifikanz der binären Suche

- Durch binäre Suche kann Anzahl der Vergleiche von n auf $\log_2(n)$ verringert werden
- n = 100: ≤ 7 Vergleiche
- n = 10000: ≤ 14 Vergleiche
- n = 1000000: ≤ 20 Vergleiche
- Allgemein: Die Funktion $f(n) = \log_2(n)$ wächst viel langsamer als die Funktion $g(n) = n$
- Auch im Computer wird dieses Suchverfahren verwendet (→ Workshop *Hide and Seek*)

Ausführung im echten Leben

- Mitte wird nicht exakt getroffen
- Suchbereich verringert sich nur um ein Drittel
- Somit $f(n) = \log_{1.5}(n)$ viele Schritte
- Immer noch wesentlich besser als $g(n) = n$

Was ist ein Algorithmus?

Eindeutige Handlungsvorschrift mit endlicher Beschreibung

Für Computer:

- Berechnungsverfahren: Eingabe → Ausgabe
- Verfahren muss präzise beschrieben werden
- Algorithmus beinhaltet abstrakte Idee, die in einem Programm umgesetzt werden kann

Der älteste Algorithmus

<u>Euklidischer Algorithmus:</u>

Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen

Nach Euklid von Alexandria: ca. 300 vor unserer Zeitrechnung

Algorithmen im Smartphone

Beispiele:

- Identifizierung durch Fingerabruck
- Suche nach kürzesten Wegen
- Entwackeln von Selfies
- …

Expert/inn/en hier am Fachbereich!

Take-Home Message

- **Algorithmen sind Handlungsvorschriften**
  - Oft in Computerprogrammen umgesetzt
- **Effizienz von Algorithmen kann analysiert werden**
  - Beispiel: $\log_2(n)$ Vergleiche effizienter als n Vergleiche

Danke für die Aufmerksamkeit!

Sebastian Forster

www.cs.sbg.ac.at/~forster