

Algorithmen und Datenstrukturen in C/C++

Prof. Dr. rer. nat. Peter Gerwinski

17. März 2022

Vorab: Online-Werkzeuge

- **Mumble:** Seminarraum 2
Fragen: Mikrophon einschalten oder über den Chat
Umfragen: über den Chat
- **VNC:** Kanal 6, Passwort: `testcvh`
Eigenen Bildschirm freigeben: per VNC-Server oder Web-Interface
Kamerabild übertragen: Link zu Web-Interface auf Anfrage
- Allgemeine Informationen: <https://www.cvh-server.de/online-werkzeuge/>
- Notfall-Schnellzugang: <https://www.cvh-server.de/virtuelle-raeume/>
Seminarraum 2, VNC-Passwort: `testcvh`
- Bei Problemen: bitte notieren:
Art des Problems, genaue Uhrzeit, JavaScript-Fehlermeldungen (F12)
- GitLab: <https://gitlab.cvh-server.de/pgerwinski/ad>
Links auf die Datei klicken, nicht mittig auf den Kommentar.

Hardwarenahe Programmierung

<https://gitlab.cvh-server.de/pgerwinski/hp.git>

- 1 Einführung**
- 2 Einführung in C**
- 3 Bibliotheken**
- 4 Hardwarenahe Programmierung**
- 5 Algorithmen**
 - 5.1 Differentialgleichungen
 - 5.2 Rekursion
 - 5.3 Aufwandsabschätzungen
- 6 Objektorientierte Programmierung**
- 7 Datenstrukturen**
 - 7.1 Stack und FIFO
 - 7.2 Verkettete Listen
 - 7.3 Bäume

Hardwarenahe Programmierung

<https://gitlab.cvh-server.de/pgerwinski/hp.git>

- 1 Einführung
- 2 Einführung in C
- 3 Bibliotheken
- 4 Hardwarenahe Programmierung
- 5 **Algorithmen**
 - 5.1 Differentialgleichungen
 - 5.2 Rekursion
 - 5.3 Aufwandsabschätzungen
- 6 Objektorientierte Programmierung
- 7 **Datenstrukturen**
 - 7.1 Stack und FIFO
 - 7.2 Verkettete Listen
 - 7.3 Bäume

Algorithmen

Differentialgleichungen

- Pendel
- Basketball

Rekursion

- Türme von Hanoi

→ *Wie rechnet man das überhaupt aus?*

Aufwandsabschätzungen

- Selectionsort
- Bubblesort
- Quicksort

→ *Wie rechnet man das möglichst effizient aus?*

- ? möglichst schnell
- ? mit möglichst wenig Speicherplatzverbrauch
- ? unter Berücksichtigung gegebener Randbedingungen

Datenstrukturen

Stack und FIFO

- effizientes Anfügen und Entfernen vorne und hinten
- effizienter direkter Zugriff auf Elemente in der Mitte
- ineffizientes Einfügen und Entfernen in der Mitte

Verkettete Listen

- effizientes Einfügen und Entfernen in der Mitte
- ineffizienter direkter Zugriff auf Elemente in der Mitte

Bäume

- Kompromiß

→ *Wie speichert man das möglichst effizient?*

- ? möglichst schnell
- ? mit möglichst wenig Speicherplatzverbrauch
- ? unter Berücksichtigung gegebener Randbedingungen

Datenstrukturen

Structs und Objekte

- zusammengehörige Daten gemeinsam speichern

Stack und FIFO – Arrays

- effizientes Anfügen und Entfernen vorne und hinten
- effizienter direkter Zugriff auf Elemente in der Mitte
- ineffizientes Einfügen und Entfernen in der Mitte

Verkettete Listen – Structs mit Pointern

- effizientes Einfügen und Entfernen in der Mitte
- ineffizienter direkter Zugriff auf Elemente in der Mitte

Bäume – Structs mit Pointern

- Kompromiß

→ *Wie speichert man das möglichst effizient?*

- ? möglichst schnell
- ? mit möglichst wenig Speicherplatzverbrauch
- ? unter Berücksichtigung gegebener Randbedingungen

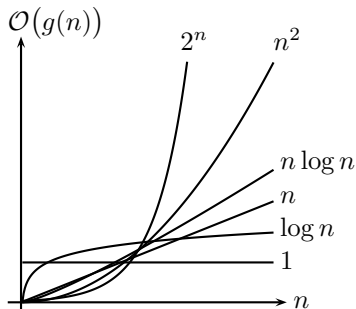
Aufwandsabschätzungen

- Türme von Hanoi: $\mathcal{O}(2^n)$

Für jede zusätzliche Scheibe
verdoppelt sich die Rechenzeit!

$$\rightarrow \frac{32,712 \text{ s} \cdot 2^{32}}{3600 \cdot 24 \cdot 365,25} \approx 4452 \text{ Jahre}$$

für 64 Scheiben



n : Eingabedaten

$g(n)$: Rechenzeit

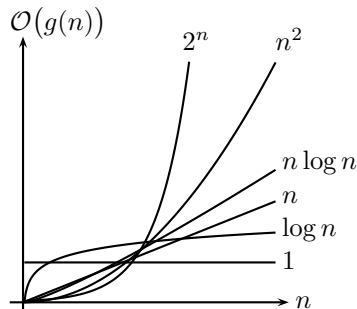
Aufwandsabschätzungen

- Türme von Hanoi: $\mathcal{O}(2^n)$
- Minimum suchen: $\mathcal{O}(n)$
- ... mit Schummeln: $\mathcal{O}(1)$
- Minimum an den Anfang tauschen, nächstes Minimum suchen:
→ Selectionsort: $\mathcal{O}(n^2)$
- Während Minimumsuche prüfen und abbrechen, falls schon sortiert
→ Bubblesort: $\mathcal{O}(n)$ bis $\mathcal{O}(n^2)$
- Rekursiv sortieren
→ Quicksort: $\mathcal{O}(n \log n)$ bis $\mathcal{O}(n^2)$

Faustregel:

Schachtelung der Schleifen zählen

x Schleifen → $\mathcal{O}(n^x)$

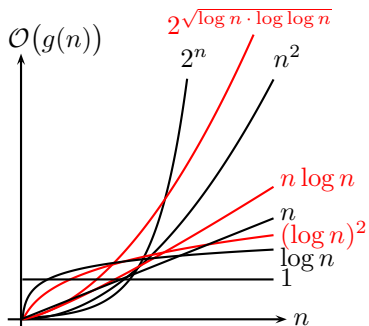


n : Eingabedaten

$g(n)$: Rechenzeit

Aufwandsabschätzungen

- Türme von Hanoi: $\mathcal{O}(2^n)$
- Minimum suchen: $\mathcal{O}(n)$
- ... mit Schummeln: $\mathcal{O}(1)$
- Minimum an den Anfang tauschen, nächstes Minimum suchen:
→ Selectionsort: $\mathcal{O}(n^2)$
- Während Minimumsuche prüfen und abbrechen, falls schon sortiert
→ Bubblesort: $\mathcal{O}(n)$ bis $\mathcal{O}(n^2)$
- Rekursiv sortieren
→ Quicksort: $\mathcal{O}(n \log n)$ bis $\mathcal{O}(n^2)$



n : Eingabedaten

$g(n)$: Rechenzeit

Faustregel:

Schachtelung der Schleifen zählen

x Schleifen → $\mathcal{O}(n^x)$

RSA: Schlüsselerzeugung (Berechnung von d): $\mathcal{O}((\log n)^2)$,

Ver- und Entschlüsselung (Exponentiation): $\mathcal{O}(n \log n)$,

Verschlüsselung brechen (Primfaktorzerlegung): $\mathcal{O}(2^{\sqrt{\log n \cdot \log \log n}})$

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 ...

$\mathcal{O}(n \log n)$ Rasterung array
struct Wegfindung CORDIC FFT
B-Baum Verschlüsseln pointer
verkettete Liste Datenbanken *Datenkompression*
digitale Signatur Hash-Tabelle
kryptographische Hash-Funktion Prüfsumme

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

2 ...

$O(n \log n)$

Rasterung

array

struct

Wegfindung

CORRIG

FFT

B-Baum

Verschlüsseln

pointer

verkettete Liste

Datenbanken

Datenkompression

digitale Signatur

Hash-Tabelle

Prüfsumme

kryptographische Hash-Funktion

Online-Werkzeuge für Home Office, Lehre
und Kommunikation mit Unterdrückten

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

1 Einführung

i Online-Werkzeuge für Home Office, Lehre
und Kommunikation mit Unterdrückten

2 Datenorganisation

3 Optimierung

4 Hardwarenahe Algorithmen

5 Datenkodierung

6 Numerik



Änderungen
vorbehalten

Übungsaufgabe: Verkettete Listen

- (a) Schreiben Sie eine Funktion, die eine verkettete Liste in die umgekehrte Reihenfolge bringt.
- (b) Schreiben Sie eine Funktion, die eine verkettete Liste in eine doppelt verkettete Liste umwandelt.
- (c) Wieviel Rechenzeit (Landau-Symbol) benötigen Ihre Funktionen?
- (d) Wieviel Speicherplatz (Landau-Symbol) benötigen Ihre Funktionen?

Algorithmen und Datenstrukturen in C/C++

<https://gitlab.cvh-server.de/pgerwinski/ad.git>

- 1 Einführung
 - i Gesellschaftliche Auswirkungen von Algorithmen und Datenstrukturen
- 2 Datenorganisation
- 3 Optimierung
- 4 Hardwarenahe Algorithmen
- 5 Datenkodierung
- 6 Numerik



Änderungen
vorbehalten