Informatik 12: Grenzen der Berechenbarkeit

Lehrplan 12.4

- experimentelle Abschätzung des Laufzeitaufwands typischer Algorithmen und die damit verbundenen Grenzen der praktischen Anwendbarkeit
- hoher Laufzeitaufwand als Schutz vor Entschlüsselung durch systematisches Ausprobieren aller Möglichkeiten (Brute-Force-Verfahren)
- prinzipielle Grenzen der Berechenbarkeit anhand von Plausibilitätsbetrachtungen zum Halteproblem

- Zeitrichtwert: 10 Unterrichtsstunden

Ausgangspunkt im Kurs

- Programme in Echtzeit oder getaktet
- Fehler ohne Bezug zur Laufzeit
- Bisher keine Konfrontation mit Laufzeitfragen
- Implementierungen liegt in der Regel das gleiche Modell zugrunde
  -> keine erkennbaren Laufzeitunterschiede

Ziel

- Unterschiedliche Datenstrukturen/Algorithmen können zu unterschiedlichem Laufzeitverhalten führen.
- Exemplarische Kenntnis typischer Laufzeitordnungen ($\log(N)$, N, $N \cdot \log(N)$, N^2, $\exp(N)$)

Sinnvolle Beispiele

- Suche in Listen versus Suche in Bäumen,
- Wegesuche in Graphen: Brute force gegen Dijkstra,
- Sortieralgorithmen: Auswählen und MergeSort.

Problematisierung

- Anwendungen zeigen, bei denen der Laufzeitaufwand bedeutsam ist
  - z.B. HLRB II, auf dem mehrere Monate die Aquarius-Simulation lief
  - Auf PC hätte die Ausführung mehrere Jahrzehnte benötigt!

Möglichkeiten zur Laufzeitmessung

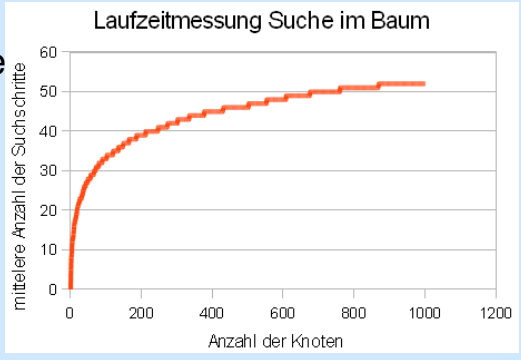
Methode 1: „Schritte zählen“

Probleme:

- Nur qualitative Aussagen
- Ausführungsdauer kann stark differieren (-> Benchmark-Messung)

Vorteile:

- Saubere Kurvenverläufe
- Tieferer Einstieg in Code



Möglichkeiten zur Laufzeitmessung

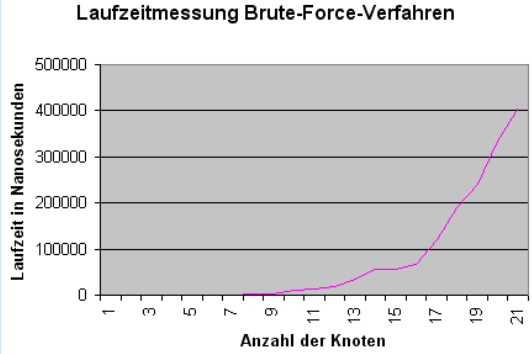
Methode 2: „Systemzeit messen“

Probleme:

- Messungenauigkeiten durch andere aktive Prozesse

Vorteil:

- Absolute Aussagen

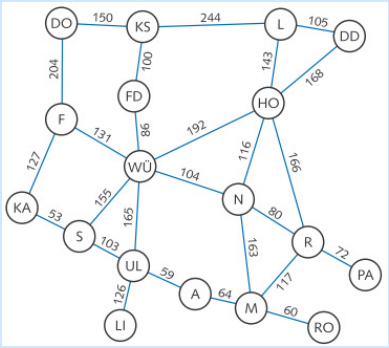


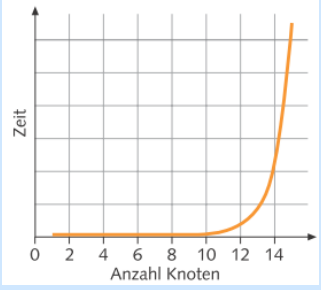
„Typische“ Algorithmen

LP: „Bei der Untersuchung des Laufzeitverhaltens typischer Algorithmen ...“

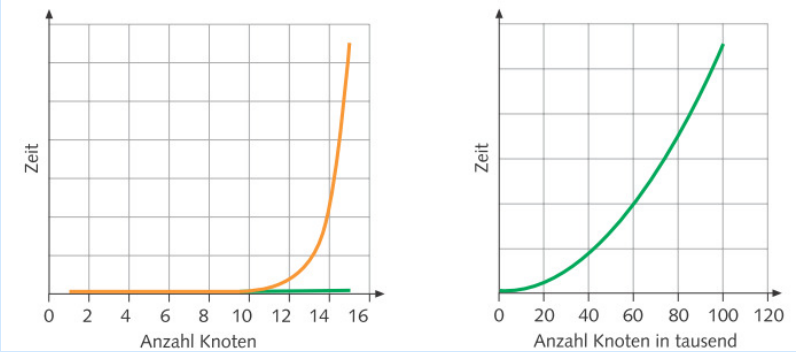
Beispiele aus dem Vorjahr:

- Suche in Liste und sortiertem Binärbaum
- Wegesuche im Graphen
  - Von jeder Kreuzung gehen mind. 3 Kanten aus => Abschätzung durch 2^n
  - Erklärung des Laufzeitgraphen für wachsendes n.
  - Evt. mit Dijkstra vergleichen





Brute Force vs. Dijkstra



Ausblick:

- Analyse des Codes
- Geeignete grafische Darstellung

Ersteres schult eher das Problembewußtsein!

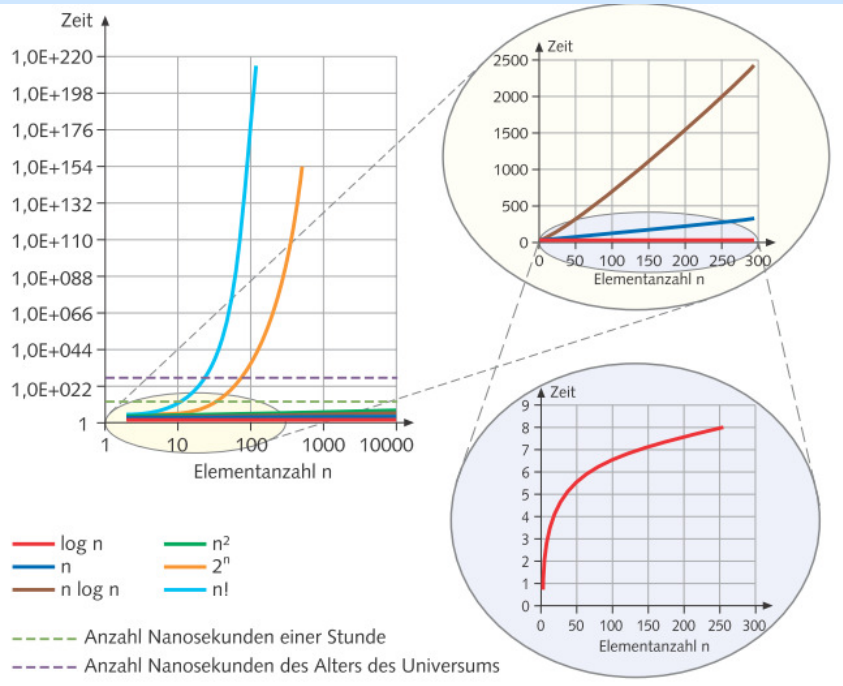
„Typische“ Algorithmen

Interessant, aber vorher nicht behandelt:
Sortieralgorithmen
z. B. MergeSort und InsertionSort

Zur Entwicklung eines Grundverständnisses
-> VisualSort von F.-J. Färber

Möglicher Ausblick: Best/Worst-Case-Laufzeit

Gegenüberstellung



Schutz vor Entschlüsselung

LP: hoher Laufzeitaufwand als Schutz vor Entschlüsselung durch systematisches Ausprobieren aller Möglichkeiten (Brute-Force-Verfahren)

Verschlüsselung „durch die Hintertür???

→ Passwort-Problematik

Passwort-Entschlüsselung

Zeichenschatz mit 26 Zeichen

Stellenzahl	mögliche Passwörter	Dauer in s	in min	in h	in d
1	26	0,000000065			
2	676	0,00000169			
3	17 576	0,00004394			
4	456 976	0,00114244			
5	11 881 376	0,02970344			
6	308 915 776	0,77228944	0,0129		
7	8 031 810 176	20,07	0,335		
8	$2{,}08827 \cdot 10^{11}$	522	8,70	0,145	
9	$5{,}4295 \cdot 10^{12}$	13573	226	3,77	0,157
10	$1{,}41167 \cdot 10^{14}$	352918	5881	98	4,08
11	$3{,}67034 \cdot 10^{15}$	9175861	152931	2549	106

11-stelliges Passwort

Zeichen	mögliche Passwörter	Dauer in Jahren
26 (einfaches Alphabet)	$3{,}67 \cdot 10^{15}$	0,291
52 (Groß- und Kleinbuchstaben)	$7{,}52 \cdot 10^{18}$	596
62 (zusätzlich Ziffern)	$5{,}20 \cdot 10^{19}$	4119
82 (zusätzlich 20 Sonderzeichen)	$1{,}13 \cdot 10^{21}$	89 519

Mögliche Ergänzungen

- Cäsar-Verschlüsselung -> Brechen über Häufigkeitsanalyse
- Auswirkungen des Moore‘schen Gesetzes
- Weitere Maßnahmen zum Passwortschutz
- Wörterbuchangriffe

Wörterbuchangriffe

Website RockYou: 32.000.000 User
Beliebteste Passwörter:
- 123456 (290.000)
- 12345 und 123456789 (je 80.000)
- password (60.000)
- iloveyou (50.000)

(Quelle: www.zdnet.de/i/news/201001/rockyou-v6.png)

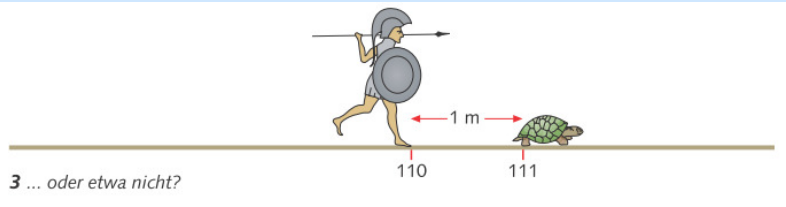
Lehrplan 12.4

Daneben gewinnen sie auch Einblicke in theoretische Grenzen der Berechenbarkeit, sodass sie die Einsatzmöglichkeiten automatischer Informationsverarbeitung realistischer einschätzen können.

Grenzen der Berechenbarkeit

- Hoffnung: Intelligente Computer, die sich z. B. selbst reparieren, Fehler erkennen
- Bisherige Grenzen:
  - Formale Beschreibbarkeit
    -> Was ist ein Fehler?
  - Zeitaufwand

Grenzen der Berechenbarkeit



3 ... oder etwa nicht?

Formal beschrieben durch...

Achill(vorsprung, v_achill, v_schildkroete)

```
wenn vorsprung == 0
dann
   return 0
sonst
   zeit = vorsprung/v_achill
   return Achill(zeit * v_schildkroete, v_achill, v_schildkroete)+ zeit
endewenn
```

... aber nicht berechenbar.

Halteproblem

Gibt es eine Methode

```
boolean TerminierungTesten (String methode,
                            String eingabe),
```

die für beliebige Werte von Methode und zugehöriger Eingabe (zugehörigen Parameterwerten) entscheiden kann, ob die gegebene Methode für diese Eingabe terminiert?

-> Widerspruchsbeweis (!)

Vielen Dank für Ihre Aufmerksamkeit!

Bildquellen: Informatik Oberstufe 2, Oldenbourg Schulbuchverlag