Vorlesungsmitschrift

Höhere Algorithmik

gelesen von Prof. Dr. Günter Rote

Tobias Höppner

Wintersemester 2014/2015

Inhaltsverzeichnis

1 Einführung (Vorlesung 1 am 17.10.)

1.1 Organisatorisches

Mitschrift wird von Studenten erstellt.

Korrekturfarbe für Gummipunkte: Grün!

Voraussetzungen

- O-Notation
- Turing-Maschine
- Sortieralgorithmen
- Schubfachprinzip
- Gauß-Nummer
- Harmonische Reihe

1.2 Kuchen teilen

Problem: Ein Kuchen soll unter zwei Personen aufgeteilt werden.
Zwei Lösungsideen:
- perfektes Teilen
- einer teilt den Kuchen und der andere sucht sich eine Hälfte aus.

Was passiert, wenn jemand die Teile des Kuchens unterschiedlich bewertet? (z.B. Kirsche auf einer Seite, viel Sahne auf der anderen Seite)
Perfektes teilen bedeutet, dass jemand *für sich* perfekt teilt. (nach seinem Maßstab)
Ziel: Fairness Jeder will $\frac{1}{n}$ des Kuchens nach ihrem Maßstab. ($n = \#$Personen)

1.2.1 1. Algorithmus (für 2 Personen)

1. Erste teilt
2. Zweite sucht aus

Der Algorithmus ist toll, aber es gibt zu viele Schritte. Daher wollen wir den Algorithmus verbessern.
Ziel: möglichst wenige Schritte.

1.2.2 2. Algorithmus (für 3 Personen)

Anton, Berta und Clara:

1. Anton teilt $\frac{1}{3}|\frac{2}{3}$
2. Berta teilt $\frac{\frac{2}{3}}{2}|\frac{\frac{2}{3}}{2}$
3. Clara sucht aus.
4. Anton sucht aus.

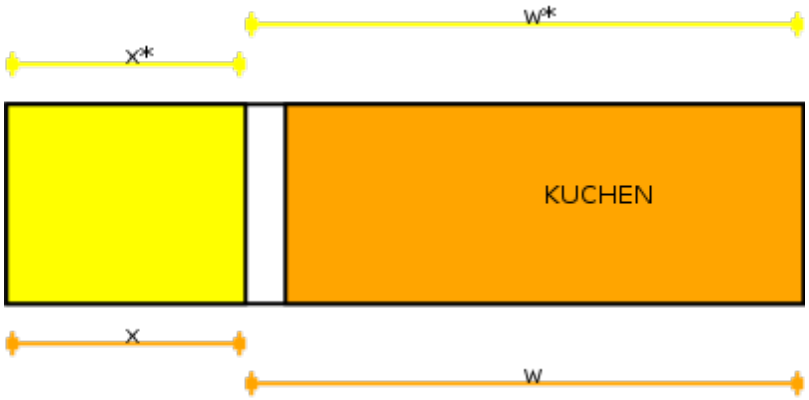
   Fall 1: Clara nimmt eines der rechten Stücke $\Rightarrow$ Anton nimmt linkes Stück.

Fall 2: Clara nimmt linkes Stück.
Schubfachprinzip: eines der rechten Stücke ist mindestens $\frac{1}{3}$

5. Berta):

1.2.3 3. Teilen und Trimmen

1. Anton teilt:



2. Berta:

Fall 1: Berta denkt $x \leq \frac{1}{3}$

Fall 2: Berta denkt $x > \frac{1}{3} \Rightarrow$ Trimmen

3. Clara darf sich entscheiden:

Fall 1: will x^* dann Algorithmus 1. für den Rest

Fall 2: will x^* nicht.
$\Rightarrow w^* \geq \frac{2}{3}$ für Clara und Anton

1.2.4 4. Teilen mit bewegtem Messer

Man nimmt ein Messer und jede Person sagt einfach Stop, wenn die *perfekte Wahl* für die Person getroffen ist.
#Schritte $= n - 1$

1.2.5 5. Simuliertes bewegtes Messer

- Jeder macht bei $\frac{1}{n}$ eine Markierung
- der/die Linkeste bekommt das Stück
  #Schritte $= n + (n-1) + ...3 + 2 = \theta(n^2)$ (Gauß-Nummer)

1.2.6 6. Simuliertes Messer + Zufall

Wie 5., aber

1. Reihenfolge zufällig
2. nur neue Linkeste Markierung werden gemacht

3. $T(n) = \#$erwartete Markierungen
   $= \underbrace{\frac{1}{n}}_{\text{Erwartete Anzahl der letzten Markierung}} + \underbrace{T(n-1)}_{\text{Erwartete Anzahl von Markierungen aller Anderen.}}$
4. $T(n) = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} = \theta(\log n)$ (harmonische Reihe)
5. Gesamtlaufzeit $\leq n * O(\log n) = O(n * \log n)$

1.2.7 7. Divide & Conquer

n Personen
n Markierungen bei $\frac{k}{n}$
#Schritte im Worst Case $T(n) = n + 2$

1.2.8 8. Divde & Conquer + Zufall

(erwartete) Laufzeit pro Teilen $\theta(\log n)$ also insgesamt $\theta(n)$

2 Einführung Teil 2 (Vorlesung 2 am 20.10.)

2.1 Ziele der Vorlesung

- Algorithmen nach den wichtigsten Entwurfsprinzipien entwerfen:
  - Devide and Conquer
  - dynamisches Programmieren
  - bound and bound
  - greedy-Algorithmen
- Algorithmen mit Analysetechniken analysieren im Bezug auf Laufzeit und Speicherbedarf (Stromverbrauch)
  - randomisierte Analyse
  - amortisierte randomisierte Analyse
  - Rekursionsgleichungen
- Vergleich und Beurteilung von Algorithmen nach Einsatzzweck
- Theorie der NP Vollständigkeit verstehen und einfache Vollständigkeitsbeweise führen

(Stromverbrauch ist zunehmend wichtig, aber nicht Teil der Vorlesung. Allgemein sind Algorithmen mit weniger Laufzeit besser.)

2.2 Rechnermodelle

2.2.1 Turing-Maschine

Eine Turing-Maschine ist ein theoretisches Modell. Es handelt sich um ein unendliches Band mit Symbolen aus einem endlichen Alphabet mit endlichem Zustandsraum. In jedem Schritt wird ein Symbol gelesen, das Band entsprechend der Eingabe beschrieben und der Zustand verändert. Prinzipiell ist alles mit einer Turing-Maschine berechenbar, jedoch teilweise sehr umständlich, weil immer nur ein Symbol gelesen werden kann.

2.2.2 Registermaschine (RAM - random access machine)

RAM ist auch als random access memory als Arbeitsspeicher bekannt

Eine RAM funktioniert nach einem ähnlichen Prinzip wie moderne Rechner arbeiten. Es gibt eine potentiell unendliche (unbeschränkte) Anzahl von Registern `R0, R1, R2, ...` wobei jedes Register eine ganze Zahl enthalten kann. Die Programmiersprache ist ähnlich wie Assembler.

1. Befehle

Zuweisung `R4 = R17`

Rechenbefehl `R1 = R2 + R3`

```
R1 = R2 - R3
R1 = R2 * R3
R1 = R2 / R3
```

Operanden der Befehle

1. Register `R17`
2. direkte Operanden (Zahlen) `250`
3. indirekte Adressen: `(R1)`

den Inhalt des Registers, dessen Nummer in Register R1 steht.

2. Sprünge

```
GOTO x
IF R_i = 0 THEN GOTO x

GZ R1, label ;if R1 is greater 0, goto label
```

Es sind nur die drei Vergleichsoperationen GLZ: < 0 , GGZ: > 0 , GZ: = 0 erlaubt!

x ist eine Sprungmarke im Programm.

```
loop:
  \\ some commands
  GOTO loop
```

3. HALT

Ein Programm endet immer mit **HALT**

Ein- und Ausgabe

Eingabe: R0 = n= die Länge der Eingabe R1, R2, ... Rn. Alle andere Zellen sind auf 0 initialisiert.
Ausgabe steht am Ende im Speicher!

2.2.3 Berechnung der Laufzeit

a) Einheitskostenmaß (EKM)

Jede Operation dauert eine Zeiteinheit.
unfair, weil es Operationen gibt, die offensichtlich komplizierter sind.

b) logarithmisches Kostenmaß (LKM)

Laufzeit = Summe der Längen aller vorkommenden Adressen und Operanden.

$$l(x) = \lfloor \log_2 \max\{|x|, 1\} \rfloor + 1$$

$$\begin{aligned} \texttt{R2} &= (\texttt{R0}) + 250 \\ \text{...Kosten} &= l(2) + l(0) + \underbrace{l(\texttt{R0})}_{\text{Adresse}} + \underbrace{l((\texttt{R0}))}_{\text{Operand}} + \underbrace{l(250)}_{\text{Operanden}} \end{aligned}$$

Im EKM kann man schwindeln:
Operationen auf langen Daten können in einem Schritt erledigt werden.
Andererseits ist das EKM näher an einem tatsächlichen Prozessor. Sofern die Operanden in ein Wort eines konventionellen Speichers (64 Bit) passen.
Abschätzung: LKM $\leq O$(EKM . l(längster vorkommender Operand oder Adresse))
Wenn die größten vorkommenden Zahlen nicht zu groß sind, dann ist das EKM realistisch.
LKM ist fairer, wenn es um sehr unterschiedliche Operanden geht (verschieden lang)

2.3 Laufzeit eines Algorithmus

Man muss den möglichen Eingaben eine Länge zuordnen.
x.. Eingabe $L(x)$
Bsp. n Zahlen $x_1, x_2, ..., x_n$ sortieren: $L = n$

Bsp. Multiplikation von langen Zahlen x, y: $L = \#$ Bits in der Eingabe.
Bsp. Lösen eines linearen Gleichungssystems: $Ax = bA \in \mathbb{Z}^{n \times x} b \in \mathbb{Z}^n x \in \mathbb{Q}^n$
Länge der Eingabe: n^2
Gauß-Elimination $O(n^3)$ Zeit, erfordert Rechnen mit rationalen Zahlen.
Man kann Zeigen, dass die Länge der Zähler und Nenner in den Zwischenergebnissen höchstens n-Mal ($\leq n$) ist, wenn man Brüche immer kürzt.
Laufzeit im LKM: $O(n^4, l(\text{größte Eingabezahl}))$

Tendenziell kompliziertes Beispiel, um zu illustrieren, dass LKM nicht immer leicht zu berechnen ist.

Was ist die Laufzeit eines Algorithmus?

$T(x) =$ Laufzeit des Algorithmus bei Eingabe$x$

$$(Analyse im schlimmsten Fall). T(n) = \max\{T(x) | L(x) = n\}$$

Andere Möglichkeiten

Analyse im Durchschnitt, Erwartungswert der Laufzeit
Benötigt eine Wahrscheinlichkeitsverteilung auf der Menge der Eingaben.