Euklidischer Algorithmus

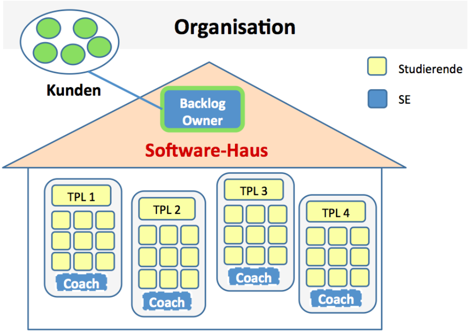
Proseminar Informatik “Algorithms Unplugged”
Wintersemester 2014/15
Institut für Theoretische Informatik
Leibniz Universität Hannover

Bharat Ahuja

1. Dezember 2014

Problem 1: Software Projekt

Bestimmung der Anzahl von Teams



Quelle: SE-Webseite (Stand 21. Oktober 2014)

Software Projekt

Die Problemstellung

- 60 Inf., 24 Math., 24 Tech. Inf.
- Gleichmäßige Verteilung der Studiengänge in jedem Team.
- Möglichst große Anzahl von Teams.

Software Projekt

Sei $n \in \mathbb{N}$ die Anzahl der Teams.

$$\Rightarrow n|60,\ n|24 \text{ und } n|24 \tag{i}$$

Wir suchen nach der größten Zahl n, die (i) erfüllt.

$$\Rightarrow n = ggT(60, 24, 24) = 12 \text{ Teams}$$

Software Projekt
Lösungsanalyse

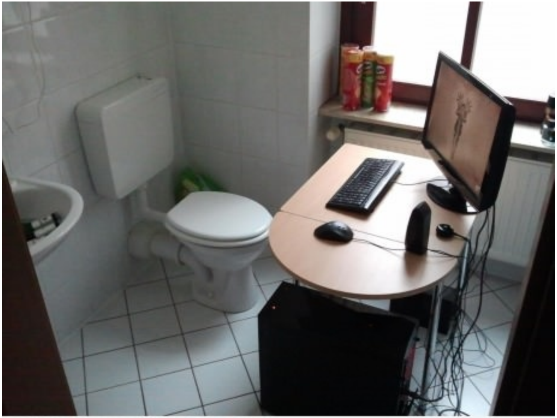
Es gibt 12 Teams mit jeweils –

- 9 Studenten
- 5 Informatik Studenten
- 2 Mathematik Studenten
- 2 Technische-Informatik Studenten

Analog: Computerspiele

Missionsdauer

20 Stunden/50 Stunden pro Woche.



Quelle: Google

Problem 2: Mit dem Teufel umgehen

Möglichst wenig Kontakt

Quelle: Google

- Chef macht alle 4 Stunden Pause.
- Das eigene Büro alle 5 Stunden verlassen.

Teilerfremdheit von Zahlen

- Teilerfremdheit durch Vergleich der Primfaktoren.
- Schnell Zahlen auf Teilerfremdheit prüfen, ohne zu faktorisieren.

Von der Problemstellung abstrahieren

Wie kann man den g.g.T. schnell bestimmen?

- Euklidischer Algorithmus
- Binary GCD Algorithm von Stein
- GCD Algorithm von Lehmer

Euklidischer Algorithmus

Historische Entwicklung

- ca. 300 *v.Chr.* in *Buch VII – Die Elemente* vorgestellt.
- Wahrscheinlich nicht selbst erfunden.
- Als geometrischen Algorithmus vorgestellt. Stäbe zerlegt.

Vorstellung der Algorithmen

- LangsamEuklid
- Euklid

LangsamEuklid

<u>LangsamEuklid</u>

while $a \neq b$ **do**
    Falls a größer ist als $b, a \leftarrow a - b$
    Falls b größer ist als $a, b \leftarrow b - a$
end while
Gib den gemeinsamen Wert der Zahlen aus

LangsamEuklid

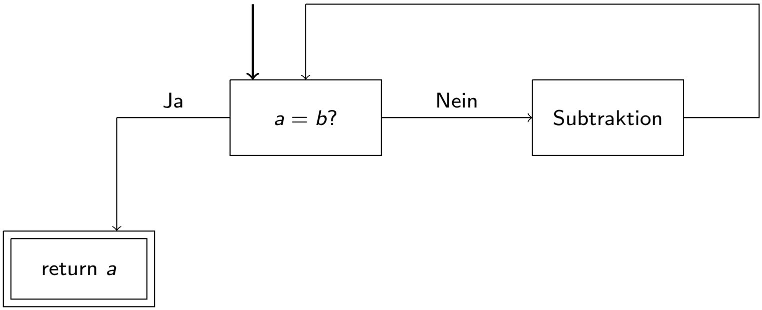


Figure: Flussdiagramm LangsamEuklid

LangsamEuklid

Beispiel

Beispiel: $a = 24, b = 9$:

Iteration Nr.	a	b
	24	9
1	15	9
2	6	9
3	6	3
4	**3**	**3**

$\therefore$ der $g.g.T.$ von 24 und 9 ist 3.

LangsamEuklid

Endlichkeit

- Die Zahlen bleiben positiv und ganzzahlig beim Subtraktionsschritt.
- Eine Variable wird in jeder Iteration um mindestens 1 verringert.
- D.h. der Algorithmus terminiert nach maximal $a + b$ Durchläufen.

LangsamEuklid

Korrektheit

- $g.g.T.(a, b) = g.g.T.(a - b, b)$
- $g.g.T.(a, a) = a$

$\square$

LangsamEuklid

Verbesserungsvorschläge

- $(1069, 2) \rightarrow (1067, 2) \rightarrow \ldots (3, 2) \rightarrow (1, 2) \rightarrow (1, 1)$
- In den meisten Iterationen braucht man nicht $a > b$ überprüfen.

Euklid

<u>Euklid</u>

```
if a < b: vertausche a und b.
while b > 0 do
    berechne q, r mit a = q · b + r, wobei 0 ≤ r < b
    a ← b, b ← r
end while
return a
```

Euklid

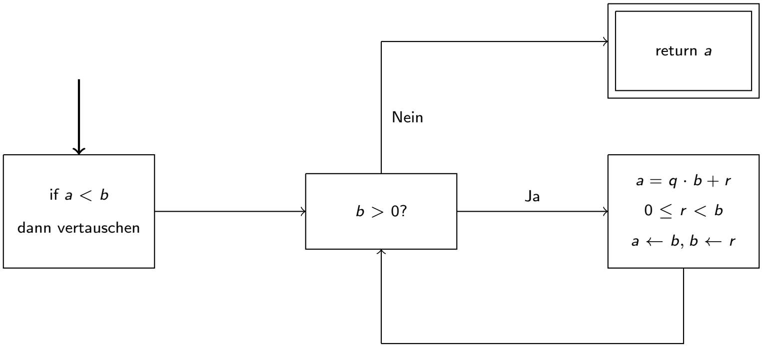


Figure: Flussdiagramm Euklid

Euklid

Beispiel

Iteration Nr.	a	b
	1069	2
1	2	1
2	1	**0**

$\therefore$ der *g.g.T.* von 1069 und 2 ist 1.

Sei ohne Einschränkung $a > b$.
Im ersten Durchlauf gilt –

$$a = q \cdot b + r, \text{ mit } r < b \tag{1}$$

Außerdem gilt –

$$a \geq b + r \tag{2}$$

Aus $(1), (2)$ folgt

$$r < \frac{a}{2}$$

Laufzeitanalyse

Euklid vs. LangsamEuklid

- Nach 2 Iterationen sind die Variablen a, b höchstens halb so groß wie der Anfangswert von a.
- Nach $2 \cdot k$ Iterationen sind die Variablen höchstens so groß wie $\frac{a}{2^k}$.
- $k \leq \log_2 a$
- $2 \cdot \log_2 a$ besser als $a + b$

Worst-Case

Euklid

Betrachte die Fibonacci-Zahlen –

$$\begin{aligned} f_{n+1} &= 1 \cdot f_n + f_{n-1} \\ f_n &= 1 \cdot f_{n-1} + f_{n-2} \\ &\vdots \\ f_2 &= 1 \cdot f_1 + 0 \\ f_1 &= 1 \end{aligned}$$

Durchschnittslaufzeit

Euklid

Wenn b feststeht und a über alle natürlichen Zahlen variiert, wieviele Durchläufe gibt es im Schnitt?

> *Sei T_b die durchschnittliche Anzahl an Iterationen des euklidischen Algorithmus über alle $a \in \mathbb{N}$, wenn der Parameter b konstant ist.*

T_b berechnen

Nach der ersten Iteration ist immer nur der Divisionsrest relevant.

$$\therefore T_b = \frac{1}{b} \sum_{0<k\leq b} T(k, b)$$

wobei $T(a, b)$ die Anzahl der Iterationen bei Eingaben a und b ist.

Durchschnittslaufzeit

Euklid : Beispiel T_5

$$T_5 = \frac{1}{5} \cdot (T(1,5) + T(2,5) \cdots + T(5,5))$$

- $T(1,5) = T(6,5) = \cdots = 2$

$$1 = 0 \cdot 5 + 1 \quad (1)$$
$$5 = 5 \cdot 1 + 0 \quad (2)$$

- $T(2,5) = T(7,5) = \cdots = 3$

$$2 = 0 \cdot 5 + 2 \quad (1)$$
$$5 = 2 \cdot 2 + 1 \quad (2)$$
$$2 = 2 \cdot 1 + 0 \quad (3)$$

Durchschnittslaufzeit

EUKLID : Beispiel T_5

Analog-

- $T(3,5) = T(8,5) = \cdots = 4$
- $T(4,5) = T(9,5) = \cdots = 3$
- $T(5,5) = T(10,5) = \cdots = 1$

$$\Rightarrow T_5 = \frac{2+3+4+3+1}{5} = 2.6 \text{ Iterationen}$$

Durchschnittslaufzeit

EUKLID : Abschätzung von T_b

Für große $b \in \mathbb{N}$ gilt

- Nach der ersten Iteration von $T(k, b)$ bleiben ungefähr T_k Iterationen.

$$\begin{aligned} k &= 0 \cdot b + k \\ b &= q_2 \cdot k + r_2 \\ &\vdots \end{aligned}$$

- $T_b \approx 1 + \frac{1}{b}(T_0 + T_1 \cdots + T_{b-1})$

Durchschnittslaufzeit

Euklid

D.h. $T_b \approx S_b$, wobei

$$S_0 := 0, S_n := 1 + \frac{1}{n}(S_0 + S_1 \cdots + S_{n-1})$$

$$\Rightarrow S_{n+1} = 1 + \frac{1}{n+1}(S_0 + S_1 \cdots + S_{n-1} + S_n)$$

Durchschnittslaufzeit

Euklid

$$= 1 + \frac{1}{n+1}(n \cdot (S_n - 1) + S_n)$$
$$\Rightarrow S_{n+1} = S_n + \frac{1}{n+1}$$

$$\therefore S_n = H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$$
$$\Rightarrow T_b \approx \ln b + O(1) \text{ (pessimistisch)}$$

Weiterführende Literatur

- *The Art of Computer Programming: Band 1* Kap. 1.1
- *The Art of Computer Programming: Band 2* Kap. 4.5.2

Zusammenfassung

- Euklidischer Algorithmus berechnet den $g.g.T$ zweier Zahlen.
- Teilerfremdheit mit Primfaktorzerlegung ineffizient.
- Laufzeit von LangsamEuklid abhängig von Zahlen.
- Laufzeit von Euklid abhängig von der Anzahl der Ziffern der Zahlen.
- Nach der 1. Iteration ist nur der Divisionsrest relevant.
- Fibonacci-Zahlen bilden das Worst-Case Szenario.