Euklidischer Algorithmus

Proseminar Informatik “Algorithms Unplugged”
Wintersemester 2014/15

Institut für Theoretische Informatik
Leibniz Universität Hannover

Bharat Ahuja

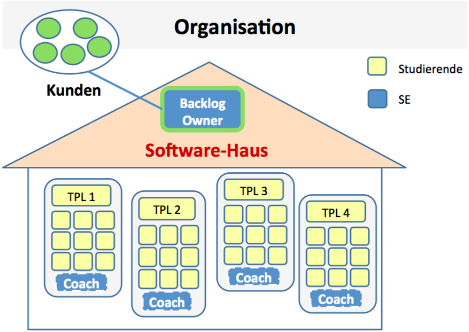
1. Dezember 2014

Überblick

- Motivation
- Vorstellung der Algorithmen
- Laufzeitanalyse
- Zusammenfassung

Problem 1: Software Projekt

Bestimmung der Anzahl von Teams



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

Software Projekt

Die Problemstellung

Software Projekt

Die Problemstellung

- 60 Inf., 24 Math., 24 Tech. Inf.

Software Projekt

Die Problemstellung

- 60 Inf., 24 Math., 24 Tech. Inf.
- Gleichmäßige Verteilung der Studiengänge in jedem Team.

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.

Software Projekt

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

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

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.

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

Software Projekt
Lösungsanalyse

Es gibt 12 Teams mit jeweils –

- 9 Studenten
- 5 Informatik Studenten

Software Projekt

Lösungsanalyse

Es gibt 12 Teams mit jeweils –

- 9 Studenten
- 5 Informatik Studenten
- 2 Mathematik Studenten

Software Projekt

Lösungsanalyse

Es gibt 12 Teams mit jeweils –

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

Analog: Computerspiele

Missionsdauer

Analog: Computerspiele

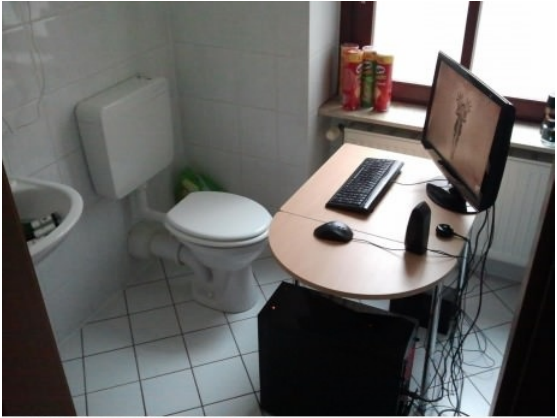
Missionsdauer

20 Stunden/50 Stunden pro Woche.

Analog: Computerspiele

Missionsdauer

20 Stunden/50 Stunden pro Woche.



Problem 2: Mit dem Teufel umgehen

Möglichst wenig Kontakt

Problem 2: Mit dem Teufel umgehen

Möglichst wenig Kontakt

Quelle: Office Space

Problem 2: Mit dem Teufel umgehen

Möglichst wenig Kontakt

Quelle: Office Space

- Chef macht alle 4 Stunden Pause.

Problem 2: Mit dem Teufel umgehen

Möglichst wenig Kontakt

Quelle: Office Space

- Chef macht alle 4 Stunden Pause.
- Das eigene Büro alle 3 Stunden verlassen.

Teilerfremdheit von Zahlen

Teilerfremdheit von Zahlen

- Teilerfremdheit durch Vergleich der Primfaktoren (NP).

Teilerfremdheit von Zahlen

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

Von der Problemstellung abstrahieren

Von der Problemstellung abstrahieren

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

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

Euklidischer Algorithmus

Historische Entwicklung

- ca. 300 *v.Chr.* in *Buch VII – Die Elemente* vorgestellt.

Euklidischer Algorithmus

Historische Entwicklung

- ca. 300 *v.Chr.* in *Buch VII – Die Elemente* vorgestellt.
- Wahrscheinlich nicht selbst erfunden.

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

LangsamEuklid: Gegeben $a, b \in \mathbb{N}$

LangsamEuklid

<u>LangsamEuklid</u>: Gegeben $a, b \in \mathbb{N}$

while $a \neq b$ **do**

LangsamEuklid

LangsamEuklid: Gegeben $a, b \in \mathbb{N}$

while $a \neq b$ **do**
    Falls a größer ist als b, $a \leftarrow a - b$
    Falls b größer ist als a, $b \leftarrow b - a$

LangsamEuklid

LangsamEuklid: Gegeben $a, b \in \mathbb{N}$

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

LangsamEuklid

<u>LangsamEuklid</u>: Gegeben $a, b \in \mathbb{N}$

```
while a ≠ b do
    Falls a größer ist als b, a ← a − b
    Falls b größer ist als a, b ← b − a
end while
Gib den gemeinsamen Wert der Zahlen aus
```

LangsamEuklid

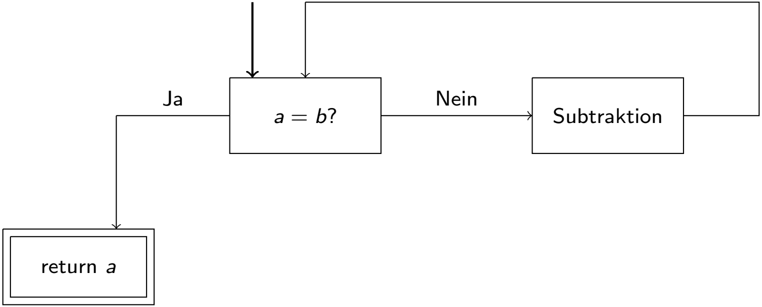


Figure: Flussdiagramm LangsamEuklid

LangsamEuklid

Beispiel

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

Iteration Nr.	a	b
	24	9

LangsamEuklid

Beispiel

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

Iteration Nr.	a	b
	24	9
1	15	9

LangsamEuklid
Beispiel

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

Iteration Nr.	a	b
	24	9
1	15	9
2	6	9

LangsamEuklid

Beispiel

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

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

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

LangsamEuklid

Endlichkeit

- Die Zahlen bleiben positiv und ganzzahlig beim Subtraktionsschritt.

LangsamEuklid

Endlichkeit

- Die Zahlen bleiben positiv und ganzzahlig beim Subtraktionsschritt.
- Eine Variable wird in jeder Iteration um mindestens 1 verringert.

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

LangsamEuklid

Korrektheit

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

$\square$

LangsamEuklid

Verbesserungsvorschläge

LangsamEuklid

Verbesserungsvorschläge

- $(1069, 2) \rightarrow (1067, 2) \rightarrow \ldots (3, 2) \rightarrow (1, 2) \rightarrow (1, 1)$

LangsamEuklid

Verbesserungsvorschläge

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

Euklid

Euklid: Gegeben $a, b \in \mathbb{N}$

EUKLID

<u>EUKLID</u>: Gegeben $a, b \in \mathbb{N}$

1: **while** $b > 0$ **do**

Euklid

Euklid: Gegeben $a, b \in \mathbb{N}$

while $b > 0$ **do**

  berechne q, r mit $a = q \cdot b + r$, wobei $0 \leq r < b$

EUKLID

<u>EUKLID</u>: Gegeben $a, b \in \mathbb{N}$

1: **while** $b > 0$ **do**

2: berechne q, r mit $a = q \cdot b + r$, wobei $0 \leq r < b$

3: $a \leftarrow b, b \leftarrow r$

Euklid

Euklid: Gegeben $a, b \in \mathbb{N}$

while $b > 0$ **do**
    berechne q, r mit $a = q \cdot b + r$, wobei $0 \leq r < b$
    $a \leftarrow b, b \leftarrow r$
end while
return a

EUKLID

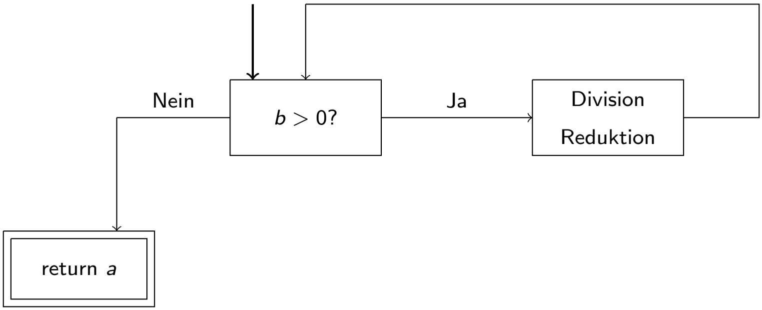


Figure: Flussdiagramm EUKLID

Euklid

Beispiel

Euklid

Beispiel

Iteration Nr.	a	b
	1069	2

Euklid

Beispiel

Iteration Nr.	a	b
	1069	2
1	2	1

Euklid

Beispiel

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

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

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

Laufzeitanalyse

EUKLID VS. LANGSAMEUKLID

Sei ohne Einschränkung $a > b$.

Laufzeitanalyse

Euklid vs. LangsamEuklid

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

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

Laufzeitanalyse

Euklid vs. LangsamEuklid

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

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

Außerdem gilt –

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

Laufzeitanalyse

EUKLID VS. LANGSAMEUKLID

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

- $a_2, b_2 < \frac{a}{2}$

Laufzeitanalyse

Euklid vs. LangsamEuklid

- $a_2, b_2 < \dfrac{a}{2}$
- Nach $2 \cdot k$ Iterationen sind die Variablen höchstens so groß wie $\dfrac{a}{2^k}$.

Laufzeitanalyse

Euklid vs. LangsamEuklid

- $a_2, b_2 < \frac{a}{2}$
- Nach $2 \cdot k$ Iterationen sind die Variablen höchstens so groß wie $\frac{a}{2^k}$.
- $k \leq \log_2 a$

Laufzeitanalyse

EUKLID VS. LANGSAMEUKLID

- $a_2, b_2 < \dfrac{a}{2}$
- Nach $2 \cdot k$ Iterationen sind die Variablen höchstens so groß wie $\dfrac{a}{2^k}$.
- $k \leq \log_2 a$
- $2 \cdot \log_2 a$ besser als $a + b$

Worst-Case

Euklid

Worst-Case

Euklid

Betrachte die Fibonacci-Zahlen –

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

Definition

Definition

- Sei $T(a, b)$ die Anzahl der Durchläufe von $\textsc{Euklid}(a, b)$.

Definition

- Sei $T(a, b)$ die Anzahl der Durchläufe von $\textsc{Euklid}(a, b)$.
- z.B. $T(1069, 2) = 3$

Durchschnittslaufzeit

Euklid

Durchschnittslaufzeit

Euklid

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

Durchschnittslaufzeit

Euklid

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

Sei T_b diese durchschnittliche Anzahl.

T_b berechnen

$T(3,5) = 4$	$T(718,5) = 4$
$3 = 0 \cdot 5 + 3$	$718 = 143 \cdot 5 + 3$
$5 = 1 \cdot 3 + 2$	$5 = 1 \cdot 3 + 2$
$3 = 1 \cdot 2 + 1$	$3 = 1 \cdot 2 + 1$
$2 = 1 \cdot 1 + 0$	$2 = 1 \cdot 1 + 0$

T_b berechnen

$T(3,5) = 4$	$T(718,5) = 4$
$3 = 0 \cdot 5 + 3$	$718 = 143 \cdot 5 + 3$
$5 = 1 \cdot 3 + 2$	$5 = 1 \cdot 3 + 2$
$3 = 1 \cdot 2 + 1$	$3 = 1 \cdot 2 + 1$
$2 = 1 \cdot 1 + 0$	$2 = 1 \cdot 1 + 0$

Nach der ersten Iteration ist immer nur der Divisionsrest relevant.

T_b berechnen

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

Durchschnittslaufzeit

EUKLID : Beispiel T_5

Durchschnittslaufzeit

EUKLID : Beispiel T_5

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

Durchschnittslaufzeit

Euklid : Beispiel T_5

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

- $T(1,5) = 2$

$$1 = 0 \cdot 5 + 1 \tag{1}$$
$$5 = 5 \cdot 1 + 0 \tag{2}$$

Durchschnittslaufzeit

EUKLID : Beispiel T_5

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

- $T(1,5) = 2$

$$1 = 0 \cdot 5 + 1 \tag{1}$$
$$5 = 5 \cdot 1 + 0 \tag{2}$$

- $T(2,5) = 3$

$$2 = 0 \cdot 5 + 2 \tag{1}$$
$$5 = 2 \cdot 2 + 1 \tag{2}$$
$$2 = 2 \cdot 1 + 0 \tag{3}$$

Durchschnittslaufzeit

EUKLID : Beispiel T_5

Analog-

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

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

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.

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.

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

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.

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

- $T_b \approx 1 + \frac{1}{b}(T_1 + T_2 \cdots + T_b)$

Durchschnittslaufzeit

EUKLID : Abschätzung von T_b

$$\Rightarrow T_b \approx H_b = \ln b + O(1)$$

Durchschnittslaufzeit

EUKLID : Abschätzung von T_b

$$\Rightarrow T_b \approx H_b = \ln b + O(1)$$

$T_{100} = 4.56$	$H_{100} \approx 5.18$
$T_{1000} = 6.422$	$H_{1000} \approx 7.49$
$T_{10^6} = 12.2$	$H_{10^6} \approx 14.4$
$T_{10^8} = 16.07$	$H_{10^8} \approx 18.99$
$T_{10^9} = 18.01$	$H_{10^9} \approx 21.3$

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.

Anhang

Beweis:
$T_b \approx S_b$, wobei

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

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

Anhang

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

$$
\therefore S_n = H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}
$$

$$
\Rightarrow T_b \approx \ln b + O(1)
$$