WIKIPEDIA

Erweiterter euklidischer Algorithmus

Der **erweiterte euklidische Algorithmus** ist ein Algorithmus aus dem mathematischen Teilgebiet der Zahlentheorie. Er berechnet neben dem größten gemeinsamen Teiler $\operatorname{ggT}(a, b)$ zweier natürlicher Zahlen a und b noch zwei ganze Zahlen s und t, die die folgende Gleichung erfüllen:

$$\operatorname{ggT}(a, b) = s \cdot a + t \cdot b$$

Der Algorithmus ist eine Erweiterung des bereits in der Antike bekannten euklidischen Algorithmus, der nur den größten gemeinsamen Teiler berechnet.

Das Haupteinsatzgebiet des erweiterten euklidischen Algorithmus ist die Berechnung der inversen Elemente in ganzzahligen Restklassenringen, denn wenn der Algorithmus das Tripel $(d = \operatorname{ggT}(a, b), s, t)$ ermittelt, ist entweder $d = 1$ und damit $1 \equiv t \cdot b \pmod a$, t also das multiplikative Inverse von b modulo a, oder aber $d \neq 1$, was bedeutet, dass b modulo a kein Inverses hat. Dies ist die Grundlage für die Lösung von diophantischen Gleichungen oder allgemeiner von ganzzahligen linearen Gleichungssystemen. Ebenso ist die Bestimmung inverser Elemente eine Grundlage für den chinesischen Restsatz, welcher wiederum Grundlage des bedeutenden Tricks der kleinen Primzahlen in der berechenbaren Algebra ist. Dabei wird eine Aufgabe in mehreren endlichen Körpern gelöst und diese Teillösungen in immer größere Restklassenringe gehoben, bis sich eine ganzzahlige Lösung ablesen lässt. Der Algorithmus liefert zudem einen konstruktiven Beweis für das Lemma von Bézout, also $1 = s \cdot a + t \cdot b$.

Die am weitesten bekannte Version des euklidischen Algorithmus bezieht sich auf den Bereich der ganzen Zahlen. Jedoch kann er auf jeden Ring angewandt werden, in welchem eine Division mit kleinstem Rest durchgeführt werden kann. Solche Ringe werden euklidisch genannt, ein Beispiel ist der Polynomring in einer Variablen mit rationalen oder reellen Koeffizienten. In diesem kann immer ein eindeutig bestimmter Rest mit kleinstem Grad gefunden werden.

Inhaltsverzeichnis

Funktionsweise der iterativen Variante

Die iterative Variante des EEA berechnet aus den vorgegebenen a, b drei Zahlenfolgen r_i, s_i, t_i. Diese werden zunächst initialisiert:

$$r_0 = a, r_1 = b$$
$$s_0 = 1, s_1 = 0$$
$$t_0 = 0, t_1 = 1$$

Dann wird für $i = 1, 2, \ldots$ berechnet:

$$r_{i+1} = r_{i-1} - q_i \cdot r_i$$
$$s_{i+1} = s_{i-1} - q_i \cdot s_i$$
$$t_{i+1} = t_{i-1} - q_i \cdot t_i$$

Dabei ist q_i der Quotient (und r_{i+1} der Rest) der Division $r_{i-1} : r_i$.

Die Berechnung endet, sobald $r_{i+1} = 0$ auftritt. Dann ist r_i der ggT, und es gilt $r_i = a \cdot s_i + b \cdot t_i$.

Beispiel für die rekursive Variante

Zu der Vorgabe der Zahlen 99 und 78 produziert der einfache euklidische Algorithmus die Folge von Divisionen mit Rest:

$$\begin{aligned}
\underline{99} &= 1 \cdot \underline{78} + \underline{21} \\
\underline{78} &= 3 \cdot \underline{21} + \underline{15} \\
\underline{21} &= 1 \cdot \underline{15} + \underline{6} \\
\underline{15} &= 2 \cdot \underline{6} + \underline{3} \\
\underline{6} &= 2 \cdot \underline{3} + \underline{0}
\end{aligned}$$

3 ist ein Teiler von 6 und damit der gesuchte größte gemeinsame Teiler von 99 und 78. Nun kann man diese Gleichungen rückwärts lesen und den Rest jeweils als Differenz der beiden anderen Terme darstellen. Setzt man diese Restdarstellungen rekursiv ineinander ein, so ergeben sich verschiedene Darstellungen des letzten Restes 3:

$$\begin{aligned}
\underline{3} &= \underline{15} - 2 \cdot \underline{6} && \\
&= \underline{15} - 2 \cdot (\underline{21} - 1 \cdot \underline{15}) &&= 3 \cdot \underline{15} - 2 \cdot \underline{21} \\
&= 3 \cdot (\underline{78} - 3 \cdot \underline{21}) - 2 \cdot \underline{21} &&= 3 \cdot \underline{78} - 11 \cdot \underline{21} \\
&= 3 \cdot \underline{78} - 11 \cdot (\underline{99} - 1 \cdot \underline{78}) &&= 14 \cdot \underline{78} - 11 \cdot \underline{99}
\end{aligned}$$

Der größte gemeinsame Teiler ist so als ganzzahlige Linearkombination der beiden Ausgangszahlen 78 und 99 dargestellt.

In der eben dargestellten Berechnungsvorschrift muss man erst den letzten Schritt des einfachen euklidischen Algorithmus abwarten, bevor die Berechnung der gesuchten Koeffizienten beginnen kann. Man kann aber auch ebenso alle anderen Reste als ganzzahlige Linearkombination von 78 und 99 darstellen und die zugehörigen Koeffizienten in jedem Schritt des einfachen euklidischen Algorithmus mit bestimmen:

$$\begin{array}{rclcrclcl}
\underline{99} &=& 1 \cdot \underline{78} + \underline{21} & \Longleftrightarrow & \underline{21} &=& & & 1 \cdot \underline{99} - 1 \cdot \underline{78} \\
\underline{78} &=& 3 \cdot \underline{21} + \underline{15} & \Longleftrightarrow & \underline{15} &=& 1 \cdot \underline{78} - 3 \cdot \underline{21} &=& -3 \cdot \underline{99} + 4 \cdot \underline{78} \\
\underline{21} &=& 1 \cdot \underline{15} + \underline{6} & \Longleftrightarrow & \underline{6} &=& 1 \cdot \underline{21} - 1 \cdot \underline{15} &=& 4 \cdot \underline{99} - 5 \cdot \underline{78} \\
\underline{15} &=& 2 \cdot \underline{6} + \underline{3} & \Longleftrightarrow & \underline{3} &=& 1 \cdot \underline{15} - 2 \cdot \underline{6} &=& -11 \cdot \underline{99} + 14 \cdot \underline{78}
\end{array}$$

Tabellarische Darstellung

Rekursive Variante

Die Zwischenergebnisse beider Berechnungsmöglichkeiten lassen sich übersichtlich in Tabellen darstellen. Für die erste Variante, bei der die Folge der Divisionen mit Rest rückwärts aufgearbeitet wird, kann dies die folgende Gestalt annehmen:

a	b	q	s	t
99	78	1		
78	21	3		
21	15	1		
15	6	2		
6	3	2		
3	0			

a	b	q	s	t
99	78	1		
78	21	3		
21	15	1		
15	6	2		
6	3	2		
3	0		1	0

a	b	q	s	t
99	78	1	−11	14
78	21	3	3	−11
21	15	1	−2	3
15	6	2	1	−2
6	3	2	0	1
3	0		1	0

Dabei wird zuerst, wie in der linken Tabelle, der einfache euklidische Algorithmus ausgeführt. Die Division mit Rest hat dabei immer die Form $a = q \cdot b + r$ (anders gesagt, bei q handelt es sich um das Ergebnis der Ganzzahldivision von a durch b, mit Rest r), wobei q und r bestimmt werden. q wird in der Zeile vermerkt, das

Paar (b, r) wird an die Stelle des Paars (a, b) in der nächsten Zeile eingetragen. Dieser Schritt wird solange wiederholt, bis in der Spalte von b eine Null steht.

Bis zu diesem Punkt wurde der einfache euklidische Algorithmus ausgeführt, und in der linken unteren Ecke (Spalte a) kann der größte gemeinsame Teiler abgelesen werden. In unserem Fall die Drei. Nun beginnt die Berechnung der ganzzahligen Koeffizienten s und t. In jeder Zeile soll dabei $3 = s \cdot a + t \cdot b$ gelten. Dementsprechend wird in der letzten Zeile $s = 1$ eingetragen, denn $3 \cdot 1 = 3$. Da in der letzten Zeile der Spalte $b = 0$ steht, kann für t ein beliebiger Wert genommen werden, denn $0 \cdot t = 0$. Hier im Beispiel ist $t = 0$ gesetzt, es ergibt sich die mittlere Tabelle.

Nun arbeitet man sich von unten nach oben. Für das s nimmt man das t der darunterliegenden Zeile. Das t berechnet sich aus dem q der jeweiligen Zeile und dem s und t der darunterliegenden Zeile.

$$s = t_{\text{alt}}$$

bzw.

$$t = s_{\text{alt}} - q \cdot t_{\text{alt}}$$

Für die vorletzte Zeile ergibt sich so $s = 0$ und $t = 1 - 2 \cdot 0 = 1$, darüber dann $s = 1$ und $t = 0 - 2 \cdot 1 = -2$ usw.

Diesen Schritt wiederholen wir solange, bis die Tabelle ausgefüllt ist. Es ergibt sich die rechte Tabelle. Die Einträge für s und t in der ersten Zeile sind die gesuchten Werte. Der größte gemeinsame Teiler findet sich, wie schon erwähnt, in der unteren linken Ecke. Für das Beispiel gilt damit

$$3 = -11 \cdot 99 + 14 \cdot 78$$

Iterative Variante

Gegeben sind wieder die Werte 99 und 78:

Wie sich aus dem Beispiel ablesen lässt, hängt der aktuelle Rechenschritt von den Zwischenergebnissen der zwei vorhergehenden Rechenschritte ab. Dem kann Rechnung getragen werden, indem bei der Initialisierung eine Hilfszeile vorangestellt wird. Weiter werden, der Übersicht halber, Hilfsvariablen u und v mit einer eigenen Spalte eingefügt.

a	b	q	u	s	v	t
0	99	0	0	1	1	0
99	78	1	1	0	0	1
78	21	3	0	1	1	−1
21	15	1	1	−3	−1	4
15	6	2	−3	4	4	−5
6	3	2	4	−11	−5	14
3	0		−11	26	14	−33

Um die jeweils nächste Zeile zu bestimmen, werden folgende Operationen ausgeführt:

- Es wird die Division mit Rest ausgeführt, $a = q \cdot b + r$ und $a_{\text{neu}} = b$ sowie $b_{\text{neu}} = r$ gesetzt.
- Die neuen Werte der Hilfsvariablen werden aus der aktuellen Zeile übernommen, $u_{\text{neu}} = s$ und $v_{\text{neu}} = t$.

- Die neuen Koeffizienten ergeben sich durch $s_{\text{neu}} = u - q \cdot s$ und $t_{\text{neu}} = v - q \cdot t$

Durch diese Vorschrift gelten in jeder außer der ersten Zeile die Beziehungen

$$a = u \cdot a_{\text{orig}} + v \cdot b_{\text{orig}} \text{ und } b = s \cdot a_{\text{orig}} + t \cdot b_{\text{orig}},$$

hier mit den Ausgangswerten $a_{\text{orig}} = 99$ und $b_{\text{orig}} = 78$. Aus den letzten zwei Zeilen liest man daher ab, dass *3* der größte gemeinsame Teiler ist und $3 = -11 \cdot 99 + 14 \cdot 78$ gilt.

Ist man mit dieser Methode vertraut genug, so kann man in der Tabelle die Spalten a, u und v weglassen, da diese nur bereits weiter oben stehende Einträge wiederholen. Weitere Beispiele in dieser verknappten Form sind in den folgenden Tabellen dargestellt:

k.	b	q	s	t
−1.	a_{orig}=**122**	–	1	0
0.	b_{orig}=**22**	5	0	1
1.	12	1	1	−5
2.	10	1	−1	6
3.	**2**=ggT	5	**2**=s	**-11**=t
4.	~~0~~	–	–	–

k.	b	q	s	t
−1.	a_{orig}=**120**	–	1	0
0.	b_{orig}=**23**	5	0	1
1.	5	4	1	−5
2.	3	1	−4	21
3.	2	1	5	−26
4.	**1**=ggT	2	**-9**=s	**47**=t

Allgemeine mathematische Grundlage

Der euklidische Algorithmus erzeugt zu vorgegebenen ganzen Zahlen *a* und *b* (allgemein: Elementen eines euklidischen Rings) zwei Folgen: eine Folge $(q_k)_k$ von Quotienten und eine Folge $(r_k)_k$ von Resten, wobei $r_0 = a,\ r_1 = b$. In jedem Schritt $k = 1, 2, \ldots$ gilt dabei

$$r_{k-1} = q_k \cdot r_k + r_{k+1}, \, |r_{k+1}| < |r_k|.$$

Nach endlich vielen Schritten ergibt sich der Rest Null.

Wir gehen nun zu Restklassen modulo *b* über. Es ist trivial zu sehen, dass $r_0 = a = 1 \cdot a$ und $r_1 = b = 0 \cdot a + 1 \cdot b$ Vielfache von a nach Hinzufügen oder Abziehen von Vielfachen von b sind. Genauer sind die Restklassen $[r_k]_b$ Vielfache der Restklasse $[a]_b$ für $k = 0, 1$, und nach Rekursionsvorschrift auch für $k = 2, 3, \ldots$. Es kann also eine Folge von Multiplikatoren $(s_k)_k$ konstruiert werden, die mit $s_0 = 1$ und $s_1 = 0$ initialisiert ist, so dass

$$r_k \equiv s_k \cdot a \pmod{b}$$

gilt. Es ergibt sich die rekursive Beziehung

$$s_{k-1} \equiv q_k \cdot s_k + s_{k+1}.$$

Man kann diese Rekursion in folgende Abfolge von Schritten für den erweiterten euklidischen Algorithmus fassen:

- Erhalte a und b als Eingabe.
- Setze $k = 0$, $r_0 = a,\ r_1 = b$, $s_0 = 1$ und $s_1 = 0$.
- Wiederhole:
  - Erhöhe k um eins.
  - Bestimme den ganzzahligen Quotienten $q_k = r_{k-1} \ \div \ r_k$.

    - Setze $r_{k+1} = r_{k-1} - q_k \cdot r_k$ und $s_{k+1} = s_{k-1} - q_k \cdot s_k$.
- bis $r_{k+1} = 0$ gilt.
- Gib den Rest $r_k = \operatorname{ggT}(a, b)$ und die Zahl s_k mit $\operatorname{ggT}(a, b) \equiv s_k \cdot a \pmod b$ zurück.

Jeder Schritt enthält implizit auch einen Multiplikator $t_k = (r_k - s_k \cdot a) \div b$, wobei diese Division keinen Rest lässt. Die Folge $(t_k)_k$ kann auch explizit bestimmt werden, es gelten $t_0 = 0$, $t_1 = 1$ und

$$t_{k-1} \equiv q_k \cdot t_k + t_{k+1}.$$

Nach dem letzten Schritt ergibt sich nun $\operatorname{ggT}(a, b) = s_k \cdot a + t_k \cdot b$

Der Übersicht halber werden beim händischen Rechnen auch noch die Hilfsfolgen $(a_k = r_{k-1})_k$ und $(b_k = r_k)_k$ sowie $(u_k = s_{k-1})_k$ sowie $(v_k = t_{k-1})_k$ mitgeführt.

Algorithmus

Rekursive Variante

Für den erweiterten euklidischen Algorithmus existiert auch eine rekursive Variante, die durch den folgenden Pseudocode gegeben ist:[1]

```
a,b: zwei Zahlen für die der erweiterte euklidische Algorithmus durchgeführt wird

extended_euclid(a,b)
wenn b = 0
    dann return (a,1,0)
(d',s',t') ← extended_euclid(b, a mod b)
(d,s,t) ← (d',t',s' - (a div b)t')
return (d,s,t)
```

Gleichbedeutend ist folgende mathematische Funktionsdefinition mit Fallunterscheidung:

$$\text{extended\_euclid}(a, b) = \begin{cases} (a, 1, 0) & \text{wenn } b = 0 \\ (d', t', s' - t'(a \operatorname{div} b)) & \text{mit } (d', s', t') = \text{extended\_euclid}(b, a \bmod b) \end{cases}$$

Programmierung

Das folgende Programm in der Programmiersprache C++ zeigt die Implementierung der rekursiven Variante und der iterativen Variante. Die zwei Varianten werden jeweils in einer Funktion mit den Parametern *a* und *b* sowie *s* und *t* implementiert. Die Parameter *s* und *t* sind Zeiger auf die berechneten Zahlen. Bei der Ausführung des Programms wird die Hauptfunktion *main* verwendet, die die Eingabe der beiden Zahlen über die Konsole ermöglicht und dann das Ergebnis der beiden Varianten dort ausgibt.

```cpp
#include <iostream>
using namespace std;

int gcdExtendedRecursive(int a, int b, int* s, int* t)
{
    if (b == 0)
    {
        *s = 1;
        *t = 0;
        return a;
    }
    int s1, t1; // Deklaration der Variablen, die die Ergebnisse des rekursiven Aufrufs speichern
    int d = gcdExtendedRecursive(b, a % b, &s1, &t1); // Rekursiver Aufruf der Methode
    *s = t1;
    *t = s1 - (a / b) * t1;
```

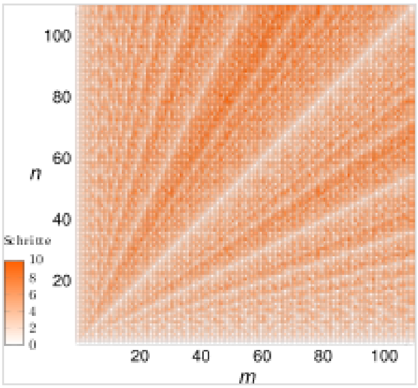
```
    return d;
}

int gcdExtendedIterative(int a, int b, int* s, int* t)
{
    *s = 1; // Initialisierung der Zeiger
    *t = 0;
    int u = 0; // Deklaration der lokalen Variablen
    int v = 1;
    while (b != 0)
    {
        int q = a / b;
        int b1 = b; // Variable zum Zwischenspeichern
        b = a - q * b;
        a = b1;
        int u1 = u; // Variable zum Zwischenspeichern
        u = *s - q * u;
        *s = u1;
        int v1 = v; // Variable zum Zwischenspeichern
        v = *t - q * v;
        *t = v1;
    }
    return a;
}

// Hauptfunktion die das Programm ausführt
int main()
{
    int a, b, s, t; // Deklaration der lokalen Variablen
    cout << "Erste Zahl: "; // Ausgabe auf der Konsole
    cin >> a; // Eingabe über die Konsole
    cout << "Zweite Zahl: "; // Ausgabe auf der Konsole
    cin >> b; // Eingabe über die Konsole
    int d = gcdExtendedRecursive(a, b, &s, &t); // Aufruf der rekursiven Funktion
    cout << "Groesster gemeinsamer Teiler: " << s << " * " << a << " + " << t << " * " << b << " = "
<< d << endl; // Ausgabe auf der Konsole
    d = gcdExtendedIterative(a, b, &s, &t); // Aufruf der iterativen Funktion
    cout << "Groesster gemeinsamer Teiler: " << s << " * " << a << " + " << t << " * " << b << " = "
<< d << endl; // Ausgabe auf der Konsole
}
```

Hinweise zur effizienten Computerimplementierung

Darstellung mittels Matrizen



Darstellung der Anzahl der Schleifendurchläufe für zwei Zahlen m und n, für die die einfache Implementierung des erweiterten euklidischen Algorithmus verwendet wurde.

Versieht man die Variablen des euklidischen Algorithmus mit Indizes für den Iterationsschritt, so wird im Schritt k die Division mit Rest $m_k = n_k \cdot q_k + r_k$ ausgeführt. Im Übergang zum nächsten Schritt wird $m_{k+1} = n_k$ und $n_{k+1} = r_k = m_k - q_k \cdot n_k$ gesetzt. Bildet man aus m und n einen Spaltenvektor, so hat der gesamte Schritt eine Darstellung mit Übergangsmatrix,

$$\begin{pmatrix} m_{k+1} \\ n_{k+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_k \end{pmatrix} \cdot \begin{pmatrix} m_k \\ n_k \end{pmatrix}.$$

Terminiert der Algorithmus nach L Schritten, so gilt $n_{L+1} = r_L = 0$ und daher $m_{L+1} = n_L = \operatorname{ggT}(a, b)$. Setzt man die Bildungsvorschriften der Spaltenvektoren ineinander ein, so ergibt sich die Verbindung zwischen dem ersten und dem letzten Spaltenvektor durch ein Matrizenprodukt,

$$\begin{pmatrix} \operatorname{ggT}(a,b) \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_L \end{pmatrix} \cdot \ldots \cdot \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix}.$$

Durch Multiplikation mit dem Zeilenvektor $(1, 0)$ wird die erste Zeile auf beiden Seiten extrahiert, somit gilt

$$\operatorname{ggT}(a,b) = \begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 \\ 1 & -q_L \end{pmatrix} \cdot \ldots \cdot \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix}.$$

Die verschiedenen Arten, das Matrixprodukt der letzten Identität auszurechnen, ergeben die verschiedenen Varianten des erweiterten euklidischen Algorithmus. In der klassischen Variante, in welcher die Divisionen mit Rest von der letzten beginnend ausgewertet werden, entspricht der Bildung der Matrixprodukte beginnend von links. Diese entspricht dem nachfolgenden rekursiven Algorithmus. Es wird $(s_1, t_1) = (1, 0)$ gesetzt und rekursiv

$$(s_{k+1}, t_{k+1}) = (s_k, t_k) \begin{pmatrix} 0 & 1 \\ 1 & -q_{L+1-k} \end{pmatrix} \text{ für } k = 1, \ldots, L$$

bestimmt. Am Ende gilt $\operatorname{ggT}(a,b) = s_{L+1}a + t_{L+1}b$. Es müssen aber zuerst alle Quotienten bestimmt werden, bevor der erste Rekursionsschritt ausgeführt werden kann.

Beginnt man die Produktbildung von rechts, so wird der Quotient der Division mit Rest in dem Augenblick benutzt, in dem er bestimmt wurde und kann danach vergessen werden. Dies entspricht dem am Anfang angegebenen Algorithmus, in welchem am Anfang

$$\begin{pmatrix} s_1 & t_1 \\ u_1 & v_1 \end{pmatrix} = I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ festgesetzt und } \begin{pmatrix} s_{k+1} & t_{k+1} \\ u_{k+1} & v_{k+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_k \end{pmatrix} \cdot \begin{pmatrix} s_k & t_k \\ u_k & v_k \end{pmatrix} \text{ für } k = 1, \ldots, L$$

iteriert wird. Insgesamt ergibt sich damit

$$\begin{pmatrix} s_{L+1} & t_{L+1} \\ u_{L+1} & v_{L+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q_L \end{pmatrix} \cdot \ldots \cdot \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix}.$$

Am Ende gilt $\operatorname{ggT}(a,b) = s_{L+1}a + t_{L+1}b$, wobei wegen der Beziehungen $s_{L+1} = u_L$ und $t_{L+1} = v_L$ der letzte Iterationsschritt auch weggelassen werden kann.

Siehe auch

- Euklidischer Algorithmus
- Steinscher Algorithmus

Weblinks

- Universität Ulm: "Elementare Zahlentheorie" [1] (https://www.uni-ulm.de/fileadmin/website_uni_ulm/mawi.inst.zawa/lehre/11elzahl/Skript.pdf)
- Iterative Variante in Java (Quellcode) (http://www.inf.fh-flensburg.de/lang/krypto/algo/euklid.htm)
- JavaScript-Rechner mit Berechnungsdetails und Zwischenschritten (http://arndt-bruenner.de/mathe/scripts/erweitertereuklid.htm)
- Alternative Darstellung der Berechnung (https://www.netzwolf.info/mathematik/erweiterter-euklidischer-algorithmus.html)
- Video: *Erweiterter Euklidischer Algorithmus Teil 1 (https://av.tib.eu/media/19885)*. Pädagogische Hochschule Heidelberg (PHHD) 2012, zur Verfügung gestellt von der Technischen Informationsbibliothek (TIB), doi:10.5446/19885 (https://doi.org/10.5446/19885).
- Video: *Erweiterter Euklidischer Algorithmus Teil 2 (https://av.tib.eu/media/19886)*. Pädagogische Hochschule Heidelberg (PHHD) 2012, zur Verfügung gestellt von der Technischen Informationsbibliothek (TIB), doi:10.5446/19886 (https://doi.org/10.5446/19886).
- Video: *Erweiterter Euklidischer Algorithmus Teil 3 (https://av.tib.eu/media/19887)*. Pädagogische Hochschule Heidelberg (PHHD) 2012, zur Verfügung gestellt von der Technischen Informationsbibliothek (TIB), doi:10.5446/19887 (https://doi.org/10.5446/19887).

- Hochschule Flensburg: Euklidischer Algorithmus (https://www.inf.hs-flensburg.de/lang/krypto/algo/euklid.htm)
- GeeksforGeeks: Euclidean algorithms (Basic and Extended) (https://www.geeksforgeeks.org/euclidean-algorithms-basic-and-extended/)

Einzelnachweise

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Introduction to Algorithms*. 2. Auflage. MIT Press, Cambridge MA 2001, ISBN 0-262-03293-7.

Abgerufen von „https://de.wikipedia.org/w/index.php?title=Erweiterter_euklidischer_Algorithmus&oldid=242026995“

Diese Seite wurde zuletzt am 9. Februar 2024 um 16:08 Uhr bearbeitet.

Der Text ist unter der Lizenz „Creative-Commons Namensnennung – Weitergabe unter gleichen Bedingungen“ verfügbar; Informationen zu den Urhebern und zum Lizenzstatus eingebundener Mediendateien (etwa Bilder oder Videos) können im Regelfall durch Anklicken dieser abgerufen werden. Möglicherweise unterliegen die Inhalte jeweils zusätzlichen Bedingungen. Durch die Nutzung dieser Website erklären Sie sich mit den Nutzungsbedingungen und der Datenschutzrichtlinie einverstanden.
Wikipedia® ist eine eingetragene Marke der Wikimedia Foundation Inc.