Teiler, ggT, kgV, Primzahlen und so weiter

Prof. Dr. Dörte Haftendorn Mathematik mit MuPAD 4.02, (ex. in 3.11 Sept. 05) Feb.07
http://haftendorn.uni-lueneburg.de **www.mathematik-verstehen.de**

##

Gliederung:

1. ggt, kgv, Vielfachsummen und erweiterter Euklidischer Algorithmus
####################
2. Primzahlen, Primfaktorzerlegung ####################
3. Teiler, Vielfache ####################
4. Menge Z*(n) der zu n teilerfemden (relativ primen) Zahlen
####################

----------eigene Zahlentheorie Ergänzungen----------

```
delete PACKAGEPATH:endml:=strmatch(NOTEBOOKPATH,"mathe-lehramt", Index)[2]:
gesamtpackpfad:=substring(NOTEBOOKPATH,1..endml+1).pathname("computer","mupad", "package
PACKAGEPATH:=gesamtpackpfad,PACKAGEPATH://Tipps zu Packages siehe unten auf der Seite.
```

```
package("zahltheo", Forced):zahltheo::init():export(zahlt
```

----------eigene Zahlentheorie Ergänzungen----------

1. ggt, kgv, Vielfachsummen und erweiterter Euklidischer Algorithmus

```
gcd(12,18); ggt(12,18); // (greatest common divisor)
```

$$6$$

$$6$$

```
lcm(12,18);//kgV(a,b)  // (lowest common mnultiple), kgV noch nicht installier
```

$$36$$

```
hold(gcd(12,18)*lcm(12,18))=hold(12*18);//beispiel für einen allg. Satz.
     gcd(12,18)*lcm(12,18) =12*18;
```

$$\mathrm{gcd}(12,\ 18)\cdot \mathrm{lcm}(12,\ 18) = 12\cdot 18$$

$$216 = 216$$

In der Kryptographie spielt die Vielfachsummen-Darstellung VSD eine besondere Rolle.
Dabei soll der größte gemeinsame Teiler von a und b als Summe von Vielfachen von a und b geschrieben werden.

ggt(a,b) = s * a + t * b

Man kann die VSD auch als Linearkombination von a und b ansehen.
Man erhält sie mit dem erweiterten Eukidischen Algorithmus:

```
igcdex(160,111); ggte(160,111) //Erweiterter Euklidischer Algori
```

$$1,\ 34,\ -49$$

$$[1,\ 34,\ -49]$$

```
[1, 34, -49]
```

Die Funktion igcd(a,b) gibt den ggt, s,t als Folge aus,
Die Funktion ggte(a,b) gibt den ggt, s,t als Liste aus,
ggte ist ebenso wie ggt und ggtex in dem eigenen package zahltheo vorhanden.
Es gibt Extraseiten, die die Programmierung genau zeigen.

```
ggtex(160,111);
 160=1*111+49 und es ist  VSD  49= 1*160+ (-1)*111
 111=2*49+13 und es ist  VSD  13= -2*160+ (3)*111
 49=3*13+10 und es ist  VSD  10= 7*160+ (-10)*111
 13=1*10+3 und es ist  VSD  3= -9*160+ (13)*111
 10=3*3+1 und es ist  VSD  1= 34*160+ (-49)*111
ggT(160,111)= 1       VSD  1= 34*160+ (-49)*111
 [1, 34, -49]
```

Probe

```
34*160 + (-49)*111
 1
```

2. Primzahlen, Primfaktorzerlegung

##################################

```
isprime(100003)
 TRUE
isprime(100007)
 FALSE
```

Primfaktorzerlegung

```
ifactor(100007)
 97 · 1031
ifactor(250348)
 2^2 · 7 · 8941
```

Probe

```
hold(2^2*7*8941)=2^2*7*8941
 2^2 · 7 · 8941 = 250348
```

Erzeugung von Primzahllisten

```
primzahlen:=[]:
if isprime(i)then primzahlen:=primzahlen.[i] end_if  $ i=2..700
pz:=nops(primzahlen);
 125
matrix([[primzahlen[i*10+k] $ k=1..10] $ i=0..((pz div
```

```
matrix([[primzahlen[i*10+k] $ k=1..10] $ i=0..((pz div
10)-1)])
```

$$\begin{pmatrix} 2 & 3 & 5 & 7 & 11 & 13 & 17 & 19 & 23 & 29 \\ 31 & 37 & 41 & 43 & 47 & 53 & 59 & 61 & 67 & 71 \\ 73 & 79 & 83 & 89 & 97 & 101 & 103 & 107 & 109 & 113 \\ 127 & 131 & 137 & 139 & 149 & 151 & 157 & 163 & 167 & 173 \\ 179 & 181 & 191 & 193 & 197 & 199 & 211 & 223 & 227 & 229 \\ 233 & 239 & 241 & 251 & 257 & 263 & 269 & 271 & 277 & 281 \\ 283 & 293 & 307 & 311 & 313 & 317 & 331 & 337 & 347 & 349 \\ 353 & 359 & 367 & 373 & 379 & 383 & 389 & 397 & 401 & 409 \\ 419 & 421 & 431 & 433 & 439 & 443 & 449 & 457 & 461 & 463 \\ 467 & 479 & 487 & 491 & 499 & 503 & 509 & 521 & 523 & 541 \\ 547 & 557 & 563 & 569 & 571 & 577 & 587 & 593 & 599 & 601 \\ 607 & 613 & 617 & 619 & 631 & 641 & 643 & 647 & 653 & 659 \end{pmatrix}$$

.

```
primzahlen:=[]:
if isprime(i)then primzahlen:=primzahlen.[i] end_if
            $ i=2000..2400:
pz:=nops(primzahlen);
matrix([[primzahlen[i*10+k] $ k=1..10] $ i=0..((pz div 10)-1)])
```

54

$$\begin{pmatrix} 2003 & 2011 & 2017 & 2027 & 2029 & 2039 & 2053 & 2063 & 2069 & 2081 \\ 2083 & 2087 & 2089 & 2099 & 2111 & 2113 & 2129 & 2131 & 2137 & 2141 \\ 2143 & 2153 & 2161 & 2179 & 2203 & 2207 & 2213 & 2221 & 2237 & 2239 \\ 2243 & 2251 & 2267 & 2269 & 2273 & 2281 & 2287 & 2293 & 2297 & 2309 \\ 2311 & 2333 & 2339 & 2341 & 2347 & 2351 & 2357 & 2371 & 2377 & 2381 \end{pmatrix}$$

```
numlib::proveprime(377777777777777777777777777)
```

FALSE

Dies Funktion `proveprime` arbeitet sicherer als isprime, siehe MuPAD-Hilfe

```
nextprime(377777777777777777777777777)
```

377777777777777777777777799

```
numlib::prevprime(377777777777777777777777777)
```

377777777777777777777777759

```
numlib::primedivisors(377777777777777777777777777)
```

[37, 1518976307, 672177055241600303]

```
ifactor(377777777777777777777777777)
```

$37 \cdot 1518976307 \cdot 672177055241600303$

3. Teiler, Vielfache

##

Teilermenge

```
numlib::divisors(15); teiler(15);
```

$$[1, 3, 5, 15]$$

$$[1, 15, 3, 5]$$

Bei der eigenen Prozedur teiler (verwendet auch im TI92/Voyage) kann man die
Teilerpaare sehen.

```
teiler(24)
```

$$[1, 24, 2, 12, 3, 8, 4, 6]$$

```
numlib::divisors(24)
```

$$[1, 2, 3, 4, 6, 8, 12, 24]$$

```
numlib::primedivisors(24)
```

$$[2, 3]$$

Hier sind nur die Primteiler ausgegeben.
so kann man die Exponenten der Primfaktoren sehen.

```
ifactor(24);
factor(24)
```

$$2^3 \cdot 3$$

$$2^3 \cdot 3$$

Vielfachenfolge

```
i*7 $ i=1..15
```

$$7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98, 105$$

etwas edler:

```
[i,i*7] $ i=1..10
```

$$[1, 7], [2, 14], [3, 21], [4, 28], [5, 35], [6, 42], [7, 49], [8, 56], [9, 63], [10, 70]$$

noch edler das Einmaleins

```
delete a:
matrix([[i*a=i*k $ k=5..10] $i=1..10])
```

$$\begin{pmatrix} a=5 & a=6 & a=7 & a=8 & a=9 & a=10 \\ 2\cdot a=10 & 2\cdot a=12 & 2\cdot a=14 & 2\cdot a=16 & 2\cdot a=18 & 2\cdot a=20 \\ 3\cdot a=15 & 3\cdot a=18 & 3\cdot a=21 & 3\cdot a=24 & 3\cdot a=27 & 3\cdot a=30 \\ 4\cdot a=20 & 4\cdot a=24 & 4\cdot a=28 & 4\cdot a=32 & 4\cdot a=36 & 4\cdot a=40 \\ 5\cdot a=25 & 5\cdot a=30 & 5\cdot a=35 & 5\cdot a=40 & 5\cdot a=45 & 5\cdot a=50 \\ 6\cdot a=30 & 6\cdot a=36 & 6\cdot a=42 & 6\cdot a=48 & 6\cdot a=54 & 6\cdot a=60 \\ 7\cdot a=35 & 7\cdot a=42 & 7\cdot a=49 & 7\cdot a=56 & 7\cdot a=63 & 7\cdot a=70 \\ 8\cdot a=40 & 8\cdot a=48 & 8\cdot a=56 & 8\cdot a=64 & 8\cdot a=72 & 8\cdot a=80 \\ 9\cdot a=45 & 9\cdot a=54 & 9\cdot a=63 & 9\cdot a=72 & 9\cdot a=81 & 9\cdot a=90 \\ 10\cdot a=50 & 10\cdot a=60 & 10\cdot a=70 & 10\cdot a=80 & 10\cdot a=90 & 10\cdot a=100 \end{pmatrix}$$

Die Teilerfremden:

Die Teilerfremden:
Zwei Zahlen a und b heißen teilerfremd, wenn ihr ggt(a,b)=1 ist.

```
liste:=[]:   n:=27:
            for i from 1 to n  do
               if gcd(n,i)=1 then liste:=liste.[i]
               end_if:
            end_for:
liste
```

$$[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26]$$

Diese Befehleszeilen sammeln einfach alle Teilerfremden auf. Sie sind in der Funktion
zstern(n) zusammengefasst, da man diese Menge stets Z*(n) nennt.

```
zstern(27);
```

$$[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26]$$

Die Anzahl der Teilerfremden von n heißt Eulersches Phi(n).

Diese Anzahl erhält man leicht durch die Elemente-Zählfunktion
nops(objekt)
oder durch die vorhandene Funktion `numlib::phi(n);`

```
numlib::phi(27);
nops(zstern(27));
```

$$18$$

$$18$$

Weiteres: Hilfe, Inhalte, Zahlentheorie

Tipps zu Packages:

```
gesamtpackpfad
```

$$\text{"D:\mathe–lehramt\computer\mupad\packages\"}$$

Unter Ansicht->Optionen->Kern können Sie in der Zeile Packages diesen String (ohne die "") eintragen,
bzw. den Pfad auf Ihrem Computer,(interaktiv dort das Entsprechende auswählen).
Nach dem erneuten Öffnen von MuPAD behält das
System diesen Pfad und Sie können den obigen winzig gedruckten Teil Auskommentieren /*....*/ deaktivieren.