

### Mersenne und vollkommen

**Mersenne-Zahlen und Vollkommene Zahlen** Haftendorn Jan 2012

$ms(n) := 2^n - 1$  • Fertigkeit **Mersenne-Zahlen** sind also die Vorgänger der Zweierpotenzen.

Info: Wikipedia: Mersenne Zahlen, Mersenne Primzahlen, Vollkommene Zahlen

Einige wenige Mersenne-Zahlen sind Primzahlen, also "**Mersenne-Primzahlen**".

**Vollkommene Zahlen** sind die, die gleich der Summe ihrer echten Teiler sind.

Es gilt: Die zu Mersenne-Primzahlen gebildeten Dreieckszahlen sind Vollkommene Zahlen. Andere gerade Vollkommene Zahlen gibt es nicht. Ungerade Vollkommene Zahlen kennt man nicht unter  $10^{50}$ .

$v(n) := 2^{n-1} \cdot (2^n - 1)$  • Fertigkeit Eigentlich ist der Term  $\frac{1}{2} \cdot (2^n - 1) \cdot (2^n - 1) + \frac{2^n - (2^n - 1)}{2}$ .

Liste aller ersten Dreieckszahlen  $seq\left(\frac{i(i+1)}{2}, i, 1, 10\right) \cdot \{1, 3, 6, 10, 15, 21, 28, 36, 45, 55\}$

$seq\left(\text{ifFn}(\text{isPrime}(i), i, "kp"), j, 3, 30, 2\right) \cdot \{3, 5, 7, "kp", 11, 13, "kp", 17, 19, "kp", 23, "kp", "kp", 29\}$

Der Student Christoph Pigge hat zum Programmierenlernen in Excel die Erzeugung der Vollkommenen Zahlen durch Betrachtung der Teilersummen gewählt. Das ist in dem anderen "Problem" dieser Datei auf gleiche Weise am TI Nspire verfolgt.

1.1

### Mersenne und vollkommen

**mersprim** 3/9 **limers**

Define **mersprim**(liste) =

Func

Local i, li, z

li := {}

For i, 1, dim(liste)

z := mid(liste, i, 1)

If isPrime(z) Then

li := augment(li, {z})

EndIf

EndFor

Return li

EndFunc

**mer1 := mersprim(limers) • {3, 7, 31, 127, 8191}**

Dies sind also die ersten Mersenne-Primzahlen.

Als Zahl Herausgreifen: **limers[5] • 63**

**limers2 := seq(2<sup>i-1</sup> · i, 1, 5, 21)**

**• {32767, 65535, 131071, 262143, 524287, 1048575, 2097151}**

**mer2 := mersprim(limers2) • {131071, 524287}**

das sind zwei weitere Mersenne-Primzahlen

Die nächste Mersenne-Primzahl ist

**mp31 := 2<sup>31</sup> - 1 • 2147483647 isPrime(mp31) • true**

Das sind immerhin schon mehr als 2 Milliarden.

Übrigens kann Excel nur bis etwa 1,5 Milliarden "long integer"-Zahlen darstellen. Danach muss Excel eine Exponentialschreibweise wählen. Das CAS vom TI Nspire kann da wesentlich mehr.

1.3

### Teilersummen und vollkommen

**Teiler und Teilersummen, Vollkommene, defiziente und abundante Zahlen.**

Hier werden, ebenso wie in der Exceldatei von Christoph Pigge, erst alle Teiler einer Zahl bestimmt. Das ist hier als eigene Funktionen herausgenommen.

**teiler(24) • {1, 2, 3, 4, 6, 8, 12, 24}** Dabei ist allerdings wegen der mathematisch korrekten Definition auch die Zahl selbst ein Teiler von sich. Die anderen Teiler heißen "echte Teiler".

Eine Zahl heißt vollkommen, wenn sie gleich der Summe ihrer echten Teiler ist.

Mit der Funktion **teilsun(24) • [36 "abundant"]** wird die Summe der echten Teiler angegeben und eine Entscheidung gefällt. Die Zahl (hier die 24) ist abundantist.

Englisch abundant heißt reich, reichlich versehen mit. 24 hat also eine reichliche Teilersumme.

**teiler(25) • {1, 5, 25}** und **teilsun(25) • [6 "defizient"]** 25 ist defizient, hat einen Defizit in der Teilersumme.

**teiler(28) • {1, 2, 4, 7, 14, 28}** **teilsun(28) • [28 "vollkommen"]** 28 ist eine vollkommene Zahl.  $1 + 2 + 4 + 7 + 14 = 28$

Zur Programmierung ist anzumerken:

**a := ifFn(4 < 5, "erfüllt", "sonst") • erfüllt** ist die if-Verzweigung als Funktion **a • erfüllt**

2.1

### Mersenne und vollkommen

**primliste** 8/8 **Primzahlen**

Define **primliste**(m, n)

Func

Local i, li

li := {}

For i, m, n

If isPrime(i) Then

li := augment(li, {i})

EndIf

EndFor

Return li

EndFunc

**li50 := primliste(1, 50)**

**• {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47}**

**li107 := primliste(51, 127)**

**• {53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127}**

Bis 127 vollständige Primzahl Listen

Noch einige:

**li266 := primliste(245, 266) • {251, 257, 263}**

**Mersenne-Zahlen**

**limers := seq(2<sup>i-1</sup> · i, 2, 14)**

**• {3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047, 4095, 8191, 16383}**

Nur einige davon sind Primzahlen (siehe nächstes Programm)

**mersenneprim := mersprim(limers) • {3, 7, 31, 127, 8191}**

Dies sind die ersten 5 Mersenne-Primzahlen vollständig.

1.2

### Mersenne und vollkommen

Erzeugung von Vollkommenen Zahlen als Dreieckszahlen der Mersenne-Primzahlen:

**mp := augment(mer1, mer2) • {3, 7, 31, 127, 8191, 131071, 524287}**

**seq(mp[i] · (mp[i] + 1) / 2, i, 1, 6) • {6, 28, 496, 8128, 33550336, 8589869056}**

**seq(mp[i] · (mp[i] + 1) / 2, i, 7, 7) • {137438691328}**

Das sind die ersten 7 Vollkommenen Zahlen. Direkte Erzeugung:

**{v(2), v(3), v(5), v(7), v(13)} • {6, 28, 496, 8128, 33550336}**

**{v(17), v(19), v(31)} • {8589869056, 137438691328, 2305843008139952128}**

**ms(61) • 2305843009213693951**

**v(61) • 2658455991569831744654692615953842176**

**ms(89) • 618970019642690137449562111**

**v(89) • 19156194260823610729479378084303638130997321548169216**

**v(107) • 13164036458569648337239753460458722910223472318386943117783728128**

**ms(107) • 162259276829213363391578010288127**

Man kennt 47 vollkommene Zahlen die letzte ist v(43112609) (wiki 2011)

1.4

### Teilersummen und vollkommen

"teiler" erfolgreich gespeichert

Liste der Teiler einer Zahl

Define **teiler**(zahl) =

Func

Local te, i, li

li := {}

For i, 1, zahl

te := mod(zahl, i)

If te = 0 Then

li := augment(li, {i})

EndIf

EndFor

Return li

EndFunc

**teiler(48) • {1, 2, 3, 4, 6, 8, 12, 16, 24, 48}**

**teiler(28) • {1, 2, 4, 7, 14, 28}**

**teiler(27) • {1, 3, 9, 27}**

**teilsun(48) • [76 "abundant"]**

**teilsun(28) • [28 "vollkommen"]**

**teilsun(27) • [13 "defizient"]**

**isPrime(2<sup>3</sup> - 1) • true**

**teilsun(2<sup>2</sup> · (2<sup>3</sup> - 1)) • [28 "vollkommen"]**

**teilsun** 2/3

Define **teilsun**(zahl) =

Func

Local s

s := sum(teiler(zahl)) - zahl

Return ifFn(s < zahl, [s "defizient"], ifFn(s = zahl, [s "abundant"], [s "vollkommen"])

EndFunc

2.2