Wie erzeugt man große Primzahlen ?

Ac 8.7.2024

Es gibt eine Reihe von Formeln, welche zu Primzahlen führen können, d.h. es muss geprüft werden, ob es sich tatsächlich um eine Primzahl handelt !

Zunächst wird eine Methode vorgestellt, die auf **bereits bekannte** Primzahlen zurückgreift :

Kennt man die ersten k Primzahlen p_i (i = 1 bis k), so lässt sich eine neue Zahl n auf folgende Weise erzeugen:

$n = p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k + 1$

n ist entweder **prim**
oder
n besitzt **mindestens zwei Primfaktoren**, die von den bekannten k Primzahlen verschieden sind.

Man findet also auf jeden Fall mindestens eine Primzahl oberhalb von p_k !

Beispiele:

n = 2+1 = 3 prim
n = 2·3+1 = 7 prim
n = 2·3·5+1 = 31 prim
n = 2·3·5·7+1 = 211 prim
n = 2·3·5·7·11+1 = 2311 prim
n = 2·3·5·7·11·13+1 = 30031 = 59·509 alle Faktoren prim
n = 2·3·5·7·11·13·17+1 = 510511 = 19·97·277 alle Faktoren prim
n = 2·3·5·7·11·13·17·19+1 = 9699691 = 347·27953 alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23+1 = 223092871 = 317·703763 alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23·29+1 = 6469693231 = 331·571·34231 alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23·29·31+1 = 200560490131 prim
n = 2·3·5·7·11·13·17·19·23·29·31·37+1 = 7420738134811 = 181·60611·676421 alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23·29·31·37·41+1 = 304250263527211 = 61·450451·11072701 alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23·29·31·37·41·43+1 = 13082761331670031 = 167·78339888213593
alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23·29·31·37·41·43·47+1 = 614889782588491411 = 953·46727·13808181181
alle Faktoren prim
n = 2·3·5·7·11·13·17·19·23·29·31·37·41·43·47·53+1 = 32589158477190044731 =
73·139·173·18564761860301 alle Faktoren prim

Die auf diese Weise erzeugte Zahl n ist prim für k = 2, 3, 5, 7, 11, 31, 379, 1019, 1021, ...

Weitere Formeln:

n! - 1 ist prim für n = 3, 4, 6, 7, 12, 14, 30, 32, 33, 38, 94, 166, ...

n! + 1 ist prim für n = 1, 2, 3, 11, 27, 37, 41, 73, 77, 116, 154, …

kgV(1, … , n) + 1 liefert die Primzahlen 2, 3, 7, 13, 61, 421, 2521, 232792561, …

Drei sehr bekannte Generatoren gehen zurück auf die bedeutenden Mathematiker Leonhard **Euler** (Schweiz) und Pierre **de Fermat** (Frankreich) sowie auf den franz. Mönch Marin **Mersenne** :

Einige **„Euler-Zahlen“** ; das sind Zahlen der Form $E_n = n^2 + n + 41\,;\quad n \in \mathbb{N}$

Die ersten 40 Euler-Zahlen E_n sind Primzahlen.

$0^2 + 0 + 41 = 41$ (prim)
$1^2 + 1 + 41 = 43$ (prim)
$2^2 + 2 + 41 = 47$ (prim)
$3^2 + 3 + 41 = 53$ (prim)
...
...
$38^2 + 38 + 41 = 1523$ (prim)
$39^2 + 39 + 41 = 1601$ (prim)

$40^2 + 40 + 41 = 1681 = 41 \cdot 41$
$41^2 + 41 + 41 = 1763 = 41 \cdot 43$
...
$100^2 + 100 + 41 = 10141 =$ (prim)
...
$100000^2 + 100000 + 41 = 10000100041 = 163 \cdot 199 \cdot 308293$
...
$100000000^2 + 100000000 + 41 = 10000000100000041 =$ (prim)

Eine weitere von Euler angegebene Formel ist **$n^2 + n + 17$** ,
die 16 aufeinander folgende Primzahlen (n = 0 bis n = 15) erzeugt !
17, 19, 23, 29, 37, 47, 59, 73, 89, 107, 127, 149, 173, 199, 227, 257,
$289 = 17^2$, … erst die Zahl 289 (n = 16) ist zerlegbar !

Eine ähnliche (mithilfe von Computern gefundene) Formel ist: **$n^2 - 79n + 1601$** ,
die 80 aufeinander folgende Primzahlen (n = 0 bis n = 79) erzeugt !
1601, 1523, 1447, 1373, 1301, 1231, 1163, 1097, 1033, 971, 911, 853, 797, 743,
691, 641, 593, 547, 503, 461, 421, 383, 347, 313, 281, 251, 223, 197, 173, 151,
131, 113, 97, 83, 71, 61, 53, 47, 43, 41, 41, 43, 47, 53, 61, 71, 83, 97, 113,
131, 151, 173, 197, 223, 251, 281, 313, 347, 383, 421, 461, 503, 547, 593, 641,
691, 743, 797, 853, 911, 971, 1033, 1097, 1163, 1231, 1301, 1373, 1447, 1523,
1601, $1681=41^2$, ... erst die Zahl 1681 (n = 80) ist zerlegbar !

Auch **$n^2 - 9n + 61$** liefert sehr viele Primzahlen; unter den ersten 1000 Zahlen finden sich mehr als 50% Primzahlen !

Allgemeine Formel: **$n^2 + a \cdot n + b$**

Einiges über Fermat-Zahlen, das sind Zahlen der Form $F_n = 2^{2^n} + 1;\ n \in \mathbb{N}$

Pierre de Fermat (1601-1665) vermutete, dass diese Zahlen immer Primzahlen seien, jedoch zeigte der schweizer Mathematiker Leonhard Euler bereits 1732 , dass **F5 zerlegbar** ist.

$F_0 = 2^{2^0} + 1 = 2^1 + 1 = 3$ (prim)

$F_1 = 2^{2^1} + 1 = 2^2 + 1 = 5$ (prim)

$F_2 = 2^{2^2} + 1 = 2^4 + 1 = 17$ (prim)

$F_3 = 2^{2^3} + 1 = 2^8 + 1 = 257$ (prim)

$F_4 = 2^{2^4} + 1 = 2^{16} + 1 = 65537$ (prim ; größte bekannte Fermatsche Primzahl !)

$F_5 = 2^{2^5} + 1 = 2^{32} + 1 = 4294967297 = 641 \cdot 6700417$ von Leonhard Euler (1732) zerlegt !

$F_6 = 2^{2^6} + 1 = 2^{64} + 1 = 18446744073709551617 = 274177 \cdot 67280421310721$ von Landry (1880) zerlegt !

$F_7 = 2^{2^7} + 1 = 2^{128} + 1 = 340282366920938463463374607431768211457 =$

$59649589127497217 \cdot 5704689200685129054721$ von Morrison & Brillhart (1970) zerlegt !

$F_8 = 2^{2^8} + 1 = 2^{256} + 1 = 115792089237316195423570985008687907853269984665640564039457584007913129639937 = 1238926361552897 \cdot$

$93461639715357977769163558199606896584051237541638188580280321$

von Brent & Pollard (1980) zerlegt !

$F_9 = 2^{2^9} + 1 = 2^{512} + 1 = 13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031858186486050853753882811946569946433649006084097 = 2424833 \cdot 7455602825647884208337395736200454918783366342657 \cdot$

$74164006262753080152478714190193747405994078109751902390582131614441575950470500809281871169394073$7 von Lenstra & Manasse (1990) zerlegt !

Für Fermatzahlen gilt folgende Rekursionsformel: $F_n = \prod_{k=1}^{n-1} F_k + 2$

Beispiele: $F_3 = 257$ und $F_0 \cdot F_1 \cdot F_2 + 2 = 3 \cdot 5 \cdot 17 + 2 = 255 + 2 = 257$

$F_4 = 65537$ und $F_0 \cdot F_1 \cdot F_2 \cdot F_3 + 2 = 3 \cdot 5 \cdot 17 \cdot 257 + 2 = 65535 + 2 = 65537$

Interessant ist: Jede Fermat-Zahl F_n mit $n > 1$ endet mit der dezimalen Ziffer 7 .
Die beiden letzten Ziffern sind dabei 17, 37, 57 oder 97 .

Bisher unbewiesene Vermutung: $F_k = 2^{2^k} + 1;\quad k \in \mathbb{N}$ ist für $k > 4$ stets zerlegbar !

Weitere vollständige Zerlegungen (F10, F11) siehe Tabelle weiter unten !

Für F_k mit $k > 11$ gibt es derzeit (07/2024) noch keine vollständigen Zerlegungen !!
Bekannt ist jedoch, dass F_5 bis F_{32} zerlegbar sind !

Die Primzahleigenschaft von F_n lässt sich u.a. mit dem Pépin-Test überprüfen:

Dieser Test gilt ausschließlich für Fermatzahlen !

Pépin-Test : F_n mit n > 0 ist genau dann Primzahl, wenn gilt: 3^((F_n – 1) / 2) mod F_n = -1

Die Formel lässt sich vereinfachen, wenn man für F_n den Term 2^(2^n)+1 einsetzt. Dann ist nämlich F_n – 1 = 2^(2^n) und (F_n – 1) / 2 = 2^(2^n – 1). Fn lässt sich dann schreiben als 2^(2^n – 1) * 2 + 1 .

Algorithmus Pépin :

```
Fnm1h = 2^(2^n - 1)
  Fn = Fnm1h * 2 + 1
  Erg = 3^Fnm1h mod Fn
  Falls Erg = -1 oder Erg = Fn - 1
        dann ist Fn prim
        sonst ist Fn zusammengesetzt
```

Beispiele:

n = 2: Fnm1h = 8 F_2 = 17 Erg = 3^8 mod 17 = 16 = F_2 - 1 $\Rightarrow$ F_2 ist Primzahl !

n = 3: Fnm1h = 128 F_3 = 257 Erg = 3^128 mod 257 = 256 = F_3 - 1 $\Rightarrow$ F_3 ist Primzahl !

n = 5: *Fnm1h* = 2147483648 F_5 = 4294967297 3^2147483648 mod 4294967297 = 10324303 ≠ -1

$\Rightarrow$ F_5 ist zusammengesetzt !

Da die Zahlen sehr schnell riesengroß werden, sollte man unbedingt mit PowerMod rechnen.

Die **Anzahl der Ziffern** von F_n lässt sich übrigens schätzen durch $\lfloor 2^n \cdot \lg(2) \rfloor + 1$

Nach dieser Formel hat z.B. F_{20} 315653 Ziffern; in Wirklichkeit sind es 315686 ! .

Java-Methode (Pépin) (Primalität von Fermatzahlen prüfen) :

```
public static boolean istPrimPepin(int n) {
   // Pépin-Test für Fermat-Primzahlen.
   // Testet, ob 3^(2^(2^n-1)) mod F(n) = -1  ( bzw. = F(n) - 1 )
   int zweiHochNm1 = (1 << n) - 1;
   BigInteger Fnm1h =  BigInteger.ONE.shiftLeft(zweiHochNm1);
   BigInteger Fn = Fnm1h.shiftLeft(1).add(BigInteger.ONE);
   BigInteger Erg = BigInteger.valueOf(3).modPow(Fnm1h, Fn);
   if (Erg.equals(Fn.subtract(BigInteger.ONE)) || Erg.equals(BigInteger.ONE.negate()))
         return true;
   return false;
}
```

Der Nachweis der Zerlegbarkeit von F_{15} dauert bei meinem Computer mit dem Pépin-Test 19s und mit dem Miller-Rabin-Test 38s .

Bei F_{17} dauert der Pépin-Test bereits 1156s (ca. 19 min) !

JAVA-Methode Fermatzahl („schnell“) erzeugen :

```
public static BigInteger fermatZahl(int k) {
   // berechnet Fk = 2^(2^k)+1 ;   k >=0  und  k <= 30 (wegen BitShift  1 << k)
   int zweiHochK = (1 << k);
   return BigInteger.ONE.shiftLeft(zweiHochK).flipBit(0);  // 2^(2^k)+1 (Fermat)
}
```

Für die Berechnung von $F_{25} = 2^{33554432} + 1$ benötigt mein Computer bereits 13,5 s !

Einiges über Mersenne-Zahlen: $M_n = 2^n - 1\ ;\ n \in \mathbb{N}^*$

Z.B. $M_1 = 2^1 - 1 = 1$ $\quad M_2 = 2^2 - 1 = 3$ $\quad$ usw.

Mersenne-Zahlen haben die Binärdarstellung 111 ... 111 , wie man sich leicht verdeutlichen kann.
Mersenne-Zahlen M_n können nur dann Primzahlen sein, wenn auch n prim ist, aber nicht jede Mersenne-Zahl mit n = prim ist eine Primzahl (z.B. ist M_{11} keine Primzahl).
Bisher (07-2024) sind erst **51 Mersenne-Primzahlen** bekannt !

Mit dem **"Lucas-Lehmer-Test"** können Mersenne-Zahlen auf Primalität getestet werden .
Dieser Test dient zum Testen der Primalität von ungeraden Mersenne-Zahlen ab M_3 .
Voraussetzung: In der Darstellung von $M_n = 2^n - 1$ sei n ungerade und prim.
Die Folge s(k) sei definiert durch $s(1) = 4$; $s(k+1) = s(k)^2 - 2$.
Dann gilt: $M_n = 2^n - 1$ ist genau dann prim, wenn $s(n-1)$ durch M_n teilbar ist .
Andere Formulierung: $s(1) = 4$ und $s(k+1) = (s(k)^2 - 2) \bmod M_n \Rightarrow M_n$ prim $\Leftrightarrow s(n-1) = 0$

Beispiel 1: $M_7 = 2^7 - 1 = 127$. $\quad n = 7$
$s(1) = 4$
$s(2) = (4^2 - 2) \bmod 127 = 14 \bmod 127 = 14$
$s(3) = (14^2 - 2) \bmod 127 = 194 \bmod 127 = 67$
$s(4) = (67^2 - 2) \bmod 127 = 4487 \bmod 127 = 42$
$s(5) = (42^2 - 2) \bmod 127 = 1762 \bmod 127 = 111$
$s(6) = (111^2 - 2) \bmod 127 = 12319 \bmod 127 = 0$
Wegen $s(n-1) = s(6) = 0$ ist M_7 eine Primzahl !

Beispiel 2: $M_{11} = 2^{11} - 1 = 2047$. $\quad n = 11$
$s(1) = 4$
$s(2) = (4^2 - 2) \bmod 2047 = 14 \bmod 2047 = 14$
$s(3) = (14^2 - 2) \bmod 2047 = 194 \bmod 2047 = 194$
$s(4) = (194^2 - 2) \bmod 2047 = 37634 \bmod 2047 = 788$
$s(5) = (788^2 - 2) \bmod 2047 = 620942 \bmod 2047 = 701$
$s(6) = (701^2 - 2) \bmod 2047 = 491399 \bmod 2047 = 119$
$s(7) = (119^2 - 2) \bmod 2047 = 14159 \bmod 2047 = 1877$
$s(8) = (1877^2 - 2) \bmod 2047 = 3523127 \bmod 2047 = 240$
$s(9) = (240^2 - 2) \bmod 2047 = 57598 \bmod 2047 = 282$
$s(10) = (282^2 - 2) \bmod 2047 = 79522 \bmod 2047 = 1736$
Wegen $s(n-1) = s(10) = 1736 \neq 0$ ist M_{11} keine Primzahl !

Java-Methode "Lucas-Lehmer-Test" (Primalität von Mersennezahlen prüfen) :

```
public static boolean istPrimLucasLehmer(int n) {
   // Lucas-Lehmer-Test für Mersennesche Primzahlen.
   // Testet für ungerade(!) Zahlen n >= 3 (und n=2), ob 2^n-1 eine Primzahl ist
   if (n == 2 || n == 3) return true;
   if (n % 2 == 0 || n < 3) return false;
      // berechne Mersennezahl 2^n-1
   BigInteger mersenneBig = BigInteger.ONE.shiftLeft(n).subtract(BigInteger.ONE);
   BigInteger sBig = new BigInteger("4");
   BigInteger zweiBig = new BigInteger("2");
    for (int k = 2; k < n; k++)
        sBig = sBig.multiply(sBig).subtract(zweiBig).mod(mersenneBig);
    return sBig.equals(BigInteger.ZERO);
}
```

Im folgenden werden **die bisher bekannten 51 Mersenne-Primzahlen** aufgeführt.

1. $2^{2}-1$ = 3
2. $2^{3}-1$ = 7
3. $2^{5}-1$ = 31
4. $2^{7}-1$ = 127
5. $2^{13}-1$ = 8191
6. $2^{17}-1$ = 131071
7. $2^{19}-1$ = 524287
8. $2^{31}-1$ = 2147483647 (Nachweis durch Euler) 10 Stellen
9. $2^{61}-1$ = 2305843009213693951 19 Stellen
10. $2^{89}-1$ = 618970019642690137449562111
11. $2^{107}-1$ = 162259276829213363391578010288127
12. $2^{127}-1$ = 170141183460469231731687303715884105727 (Lucas) 39 Stellen
13. $2^{521}-1$ =
6864797660130609714981900799081393217269435300143305409394463459185543183397656052122559640661454554977296311391480858037121987999716643812574028291115057151 157 Stellen
14. $2^{607}-1$ =
5311379928167670986895882065524686273295931177270319231994441382004035598608522427391625022652292856688893294862465010153465793376527072394095199787665873519438312708353932190317281 27 183 Stellen
15. $2^{1279}-1$ =
1040793219466439908192524032736408553861526224726670480531911235040360805967336029801223944173232418484242161395428100779138356624832346490813990660567732076292412950938922034577318334966158355047295942054768981121169367714754847886696250138443826029173234888531116082853841658502825560466622483118909188018470682222031405210266984354887329580288780508697361869007147207105557031687290870 386 Stellen
16. $2^{2203}-1$ =
1475979915214180235084898622737381736312066145333169775147771216478570297878078949377407337049389289382748507531496480477281264838760259191814463365330269540496961201113430156902396093989090226259326935025281409614983499388222831448598601834318536230923772641390209490231836446899608210795482963763094236630945410832793769905399982457186322944729636418890623372171723742105636440368218459649632948538696905872650486914434637457507280441823676813517852099348660847172579408422316678097670224011990280170474894487426924742108823536808485072502240519452587542875349976558572670229633962575212637477897785501552646522609988869914013540483809865681250419497686697771007664 Stellen
17. $2^{2281}-1$ =
4460875571837584295711517064021018098862086324128599011119912199634046857928204733691125452690039890261532459311243167023957587056936793647909034974611470710652541933539381249782263079473124107988748690400702793284288103117548441080948782524948667609695869981289826458775960289791715369625030684296173317021847503245830091718321049160501576288866063721455017022259251252240768296054271735739648129952505694124807207384768552936816667128448311908776206067866638621902401185707368319018864792258104147140789353865624979681787291276295949244119609613867139462798992750069549171397587960612238033935373810346664944029510520590479686932553886479304409251041868170096401717641331724181328363516877 687 Stellen
18. $2^{3217}-1$ =
2591170860132026277762467679224415309418188875531254273039749231618740192665863620862012095168004834065506952417331941774416895092388070174103777095975120423130666240829163535179523111861548622656045476911275958487756105687579311910177114088262521538490358304011850721164247474618230314713983402292880745456779079410372882358207058923510684338829868886166586502809276920803396058693087905004095037098759021190183719916209940025689351131365488297391126567973032419865172501164127035097054277734779723498216764434466683831193225400996489940517902416240565190544836908096160616257430423617218633394158524264312087372665919620617535357488928945996291951830826218608534009379328394202618665861425032514507730962742353768229386494071277008460771242118230808041392980870575047138252645714483793711250320818261265666490842516994539518877896136502484057393785945994443352311882801236604062624686092121503499375847822922371443396288584859382157388212323936870461606773629093150719 969 Stellen

19. $2^{4253}-1$ = 1907970075... 1281 Stellen
20. $2^{4423}-1$ = 2855425422... 1332 Stellen
21. $2^{9689}-1$ = 4782202788... 2917 Stellen
22. $2^{9941}-1$ = 3460882824... 2993 Stellen
23. $2^{11213}-1$ = 2814112013... 3376 Stellen
24. $2^{19937}-1$ = 4315424797... 6002 Stellen
25. $2^{21701}-1$ = 4486791661... 6533 Stellen
26. $2^{23209}-1$ = 4028741157... 6987 Stellen
27. $2^{44497}-1$ = 8545098243... 13395 Stellen
28. $2^{86243}-1$ = 5369279955... 25962 Stellen
29. $2^{110503}-1$ = 5219283133... 33265 Stellen
30. $2^{132049}-1$ = 5127402762... 39751 Stellen
31. $2^{216091}-1$ = 7460931030... 65050 Stellen
32. $2^{756839}-1$ = 1741359068... 227832 Stellen
33. $2^{859433}-1$ = 1294981256... 258716 Stellen
34. $2^{1257787}-1$ = 4122457736... 378632 Stellen
35. $2^{1398269}-1$ = 8147175644... 420921 Stellen
36. $2^{2976221}-1$ = 6233400762... 895932 Stellen
37. $2^{3021377}-1$ = 1274116830... 909526 Stellen
38. $2^{6972593}-1$ = 4370757441... 2098960 Stellen
39. $2^{13466917}-1$ = 9249477380... 4053946 Stellen
40. $2^{20996011}-1$ = 1259768954... 6320430 Stellen
41. $2^{24036583}-1$ = 7235733 Stellen
42. $2^{25964951}-1$ = 7816230 Stellen
43. $2^{30402457}-1$ = 9152052 Stellen
44. $2^{32582657}-1$ = 9808358 Stellen
45. $2^{37156667}-1$ = 11185272 Stellen
46. $2^{42643801}-1$ = 12837064 Stellen ; 2009 gefunden
47. $2^{43112609}-1$ = 12978189 Stellen ; 2008 gefunden
48. $2^{57885161}-1$ = 17425170 Stellen ; 2013 gefunden
49. $2^{74207281}-1$ = 22338618 Stellen ; 2016 gefunden
50. $2^{77232917}-1$ = 23249425 Stellen ; 12-2017 gefunden (Jon Pace)
51. $2^{82589933}-1$ = 1488944457... 24862048 Stellen ; 2018 gefunden (Patrick Laroche)

größte bisher (02/2022) bekannte Primzahl (Stand: 12/2018 !)

JAVA-Methode Mersennezahl („schnell“) erzeugen :

```
public static BigInteger mersenneZahl(int k) {
    // berechnet Mk = 2^k-1 ; k > 0  und  k <= 2147483646 = Integer.MAX_VALUE - 1   !!!
    return BigInteger.ONE.shiftLeft(k).subtract(BigInteger.ONE);  // 2^k-1 (Mersenne)
}
```

Anmerkungen:

1) Die **Berechnung** der 43. Mersenneprimzahl $2^{30402457}-1$ dauert auf meinem Rechner bereits **12,5s**.
Die Berechnung der 51. Mersenneprimzahl sogar **54s** !

2) Zur Überprüfung der **Primalität** der 27. Mersenne-Primzahl mit dem **Lucas-Lehmer-Test** ,
programmiert mit Java17, benötigt mein Rechner **29s** (Ryzen 9 5900X ; Windows 11).
Bei der 28. Mersenne-Primzahl sind es bereits **166s** .
Der **Miller-Rabin-Test** dauert noch sehr viel länger (27. Mersenne-Primzahl: **174s**) !

Mersenne-Fermat-Zahlen :

"Mersenne-Fermat-Zahlen" haben die Form $MF(p, n) = \frac{2^{p^n} - 1}{2^{p^{n-1}} - 1}$

mit p = prim und n = natürliche Zahl .

Für **n = 1** ergeben sich **Mersenne-Zahlen** $2^p - 1$, für **p = 2** **Fermat-Zahlen** !

Beweis für p = 2:

(2^(2^n) – 1) / (2^(2^(n-1)) – 1) = (2^(2^n) – 1) / (2^(2^n/2)) – 1) = (2^(2^n) – 1) / (2^(2^n))^(1/2)) – 1) = (2^(2^n) – 1) / (√(2^(2^n)) – 1) = √(2^(2^n)) + 1 = 2^(2^(n-1)) + 1

Betrachten wir einige Beispiele :

```
MF(3,2) = (2^9-1)/(2^3-1) = 511/7 = 73  (prim)
MF(3,3) = (2^27-1)/(2^9-1) = 134217727/511 = 262657 (prim)
MF(3,4) = (2^81-1)/(2^27-1) = 241785163922925834941235 1/134217727 = 18014398643699713 =
2593·71119·97685839 (also zerlegbar)
```

Neben MF(3,2) und MF(3,3) sind MF(7,2) und MF(59,2) die einzigen bekannten primen Mersenne-Fermat-Zahlen für p > 2 !

Für p = 2 betrachte man die 4 bekannten primen Fermat-Zahlen F2, F3, F4, F5 !

RepUnit - Zahlen R_n

RepUnits sind Zahlen, die ausschließlich die Ziffer 1 (dezimal) enthalten !
R_n kann nur dann prim sein, wenn n prim ist (notwendig, nicht hinreichend !)
Gemessen an den zerlegbaren RepUnit-Zahlen gibt es nur wenige prime (s.u.),
man vermutet dennoch, dass es unendlich viele prime RepUnit-Zahlen gibt.

Beispiele:

R2 = 11 prim;
R3 = 111 = 3·37
R7 = 1111111 = 239·4649
R11 = 11111111111 = 21649·513239
R17 = 11111111111111111 = 2071723·5363222357
R19 = 1111111111111111111 prim;
R23 = 11111111111111111111111 prim;

R53 = 11111111111111111111111111111111111111111111111111111 =
107·1659431·1325815267337711173·4719885879949142566020007l yafu: 0,3s

R71 = 11111111111111111111111111111111111111111111111111111111111111111 =
24157314239362767357695743904 9·45994811347886846310221728895223034301839 yafu: 14s msi: 16s

R97 = 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111 = 12004721·84603573139691923376721153789909716 9·
10939984685537053754033926684207011910766229658034803 9 yafu: 425 s msi: 843 s

R114 = 1111...11 = 3· 7· 11· 13· 37· 21319· 1458973· 10749631· 909090909090909091·
1111111111111111111· 39311230223051293779765 19· 753201806271328462547977919407
yafu: 1 s msi: 22 s

R131 = 1111...11 = 80173·109517·141811693·446790173·7370364319027·15594845538029429933·
731772397003105767769 3·13175835106511615120521 3·18022206228783402545124708 1
yafu: 1,2s msi: 855s

R200 = 1111...11 = 11·41·73·101·137·251·271·401·1201·1601·3541·5051·9091·21401·25601·27961·
60101·1676321·7019801·5964848081·182521213001·14103673319201·78875943472201·1680588011350901·
12969441902905775055138577118456427449907570094765675782153729152719680 1
yafu: 0,8s msi: 336s brent: 138s

R431 = 1111...11 = 3449· 42362129· 7567197163·
10049662295682655156769587841862238326712165785450544362338167983483011083703916156849256124357671
42097226396821567424584912627196816645970883845618740958081203197221906694509297877794482680071 4
27651194893957926742116010296174904421933279054622701855491680458598498382162071236343774999377 27
62119223771166832044277782443582309364525727387690981501489126090503307810449887713171707038823 89
8180962558635376339757 (zerlegbar; 410 Zeichen) **bisher noch nicht zerlegt !**

Des weiteren sind auch **R317** und **R1031** prim !

Wahrscheinlich prim sind:
R49081 (09-1999; Dubner), **R86453** (10-2000; Baxter), **R109297** (03-2007; Bourdelais-Dubner), **R270343** (07-2007; Voznyy-Budnyy), **R5794777** (04-2021; Batalov-Proper), **R8177207** (05-2021; Batalov-Proper).

Repunit-Zahlen haben die Form $R_n = \frac{10^n - 1}{9}$, da $9 \cdot R_n = 10^n - 1$ gilt !

Beispiel: R_5 (= 11111) = $(10^5 - 1) / 9 = 99999 / 9 = 11111$

Verallgemeinerte („generalized“) RepUnits zur Basis b :

"Generalized" (verallgemeinerte) Repunit-Zahlen zur Basis b haben die Form $R_n(b) = \frac{b^n - 1}{b - 1}$

Für **b = 10** ergibt sich die normale obige Formel für **Repunits**.
Für **b = 2** erhalten wir $2^n - 1$, also die Definition der **Mersenne-Zahlen** !

Betrachten wir einige Beispiele für b = 12: **R_n (12) = (12^n – 1) / 11** :

```
R1(12) = 1
R2(12) = 13          prim !
R3(12) = 157         prim !
R4(12) = 1885 = 5· 13· 29
R5(12) = 22621     prim !
R6(12) = 271453 = 7· 13· 19· 157
...
R19(12) = 29043636306420266077    prim !
...
R50(12) = 827312559091104524847962297758415120447922963932416 93 =
13· 1951· 19141· 22621· 60601· 73951· 303551· 438472201· 12629757106815551
brent: 0,3 s
...
R97(12) = 435700623537534460534556100566797400050569661118420894078389027832099599815
930778113305073283279681915 81    prim !
...
R100(12) = 752890677472859547803712941779425907287384374412631304967456240093868108481
928384579791166633507290350521 25 =
5· 5· 5· 13· 29· 101· 1201· 1951· 19141· 22621· 60601· 73951· 303551· 85403261·
438472201· 700936801· 2334798291701· 14807687049800501· 12629757106815551   brent: 14s
...
R107(12) = 269774342001974285278694638169555674601473577261535176943833382434453840404
476293460132377629078395944401301624943 7 =
126047 · 8368183883 · 255762526541912675998574100663845947967020971943416941716598897
073179438991591162215340597323148093 7       brent: 0,9s
...
R109(12) = 388475052482842970801320278964160171426121951256610654799120070705613530183
244586258259062378587289015993787433991894 1   prim !
```

Desweiteren sind prim: $R_{317}(12)$ $R_{353}(12)$ $R_{701}(12)$ $R_{9739}(12)$ * ...

Die mit Sternchen versehene ist eine „Quasiprimzahl“ , deren Primalität noch nicht nachgewiesen werden konnte !

Def.: m heißt quasiprim zur Basis b, wenn $b^{m-1} \bmod m = 1$ gilt

Beispiel: 341 (= 11 · 31) ist quasiprim zur Basis 2, denn $2^{340} \bmod 341 = 1$

Anmerkung: Im Internet existieren Tabellen zur Zerlegung von Repunit-Zahlen (auch generalisierten) !

Links: https://stdkmd.net/nrr/repunit/
https://homes.cerias.purdue.edu/~ssw/cun/xtend/crombie
https://homes.cerias.purdue.edu/~ssw/cun/index.html