<u>Wie erzeugt man große Primzahlen ?</u>

Ac 17.11.2024

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 !

<u>Zunächst wird eine Methode vorgestellt, die auf **bereits bekannte** Primzahlen zurückgreift :</u>

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 <u>mindestens eine</u> Primzahl oberhalb von p_k !

<u>Beispiele:</u>

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

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

<u>Weitere Formeln:</u>

$n! - 1$ 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 $\boxed{E_n = n^2 + n + 41; \quad n \in \mathbb{N}}$

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 !

$F_8 = 2^{2^8} + 1 = 2^{256} + 1 = 11579208923731619542357098500868790785326998466564056403945758400791312963$
$9937 = 1238926361552897 \cdot$
$93461639715357977769163558199606896584051237541638188580280321$
von Brent & Pollard (1980) zerlegt !

$F_9 = 2^{2^9} + 1 = 2^{512} + 1 = 134078079299425970995740249982058461274793658205923933777235614437217640300$
$735469768018742981669034276900318581864860508537538828119465699464336490060$
$84097 = 2424833 \cdot 7455602825647884208337395736200454918783366342657 \cdot$
$741640062627530801524787141901937474059940781097519023905821316144415759504$
705008092818711693940737 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: $F_k = 2^{2^k} + 1;\ k \in \mathbb{N}$ 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 $\boxed{\lfloor 2^n \cdot \lg(2) \rfloor + 1}$

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

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

```java
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 :

```java
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: $\boxed{M_n = 2^n - 1\,;\ n \in \mathbb{N}^*}$

Z.B. $M_1 = 2^1 - 1 = 1$ $M_2 = 2^2 - 1 = 3$ 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 (11-2024) sind erst **52** **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: $M_7 = 2^7 - 1 = 127$. n = 7
s(1) = 4
s(2) = (4² - 2) mod 127 = 14 mod 127 = 14
s(3) = (14² - 2) mod 127 = 194 mod 127 = 67
s(4) = (67² - 2) mod 127 = 4487 mod 127 = 42
s(5) = (42² - 2) mod 127 = 1762 mod 127 = 111
s(6) = (111² - 2) mod 127 = 12319 mod 127 = 0
Wegen s(n -1) = s(6) = 0 ist M_7 eine Primzahl !

Beispiel 2: $M_{11} = 2^{11} - 1 = 2047$. n = 11
s(1) = 4
s(2) = (4² - 2) mod 2047 = 14 mod 2047 = 14
s(3) = (14² - 2) mod 2047 = 194 mod 2047 = 194
s(4) = (194² - 2) mod 2047 = 37634 mod 2047 = 788
s(5) = (788² - 2) mod 2047 = 620942 mod 2047 = 701
s(6) = (701² - 2) mod 2047 = 491399 mod 2047 = 119
s(7) = (119² - 2) mod 2047 = 14159 mod 2047 = 1877
s(8) = (1877² - 2) mod 2047 = 3523127 mod 2047 = 240
s(9) = (240² - 2) mod 2047 = 57598 mod 2047 = 282
s(10) = (282² - 2) mod 2047 = 79522 mod 2047 = 1736
Wegen s(n -1) = s(10) = 1736 ≠ 0 ist M_{11} keine Primzahl !

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

```java
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 52 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. $2^{521}-1$ = 686479766013060971498190079908139321726943530014330540939446345918554318
339765605212255964066145455497729631139148085803712198799971664381257402
8291115057151 157 Stellen

14. $2^{607}-1$ = 531137992816767098689588206552468627329593117727031923199444138200403559860
852242739162502265229285668889329486246501015346579337652707239409519978766
587351943831270835393219031728127 183 Stellen

15. $2^{1279}-1$ = 104079321946643990819252403273640855386152622472667048053191123504036080596
733602980122394417323241848424216139542810077913835662483234649081399066056
773207629241295093892203457731833496615835504729594205476898112116936771
475484788669625013844382602917323488853111608285384165850282556046662248318
909188018470682222031405210266984354887329580288780508697361869007147207
1055570316872908 7 386 Stellen

16. $2^{2203}-1$ = 1475979915214180235084898622737381736312066145333169775147771216478570297878
078949377407337049389289382748507531496480477281264838760259191814463365330
269540496961201113430156902396093989090226259326935025281409614983499388
222831448598601834318536230923772641390209490231836446899608210795482963763
094236630945410832793769905399982457186322944729636418890623372171723742105
636440368218459649632948538696905872650486914434637457507280441823676813
517852099348660847172579408422316678097670224011990280170474894487426924742
108823536808485072502240519452587542875349976558572670229633962575212637477
8977855015526465226099888699140135404838098656812504194976866977710 07
664 Stellen

17. $2^{2281}-1$ = 446087557183758429571151706402101809886208632412859901111991219963404685792
820473369112545269003989026153245931124316702395758705693679364790903497461
147071065254193353938124978226307947312410798874869040070279328428810311
754844108094878252494866760969586998128982645877596028979171536962503068429
617331702184750324583009171832104916050157628886606372145501702225925125224
076829605427173573964812995250569412480720738476855293681666712844831190
877620606786663862190240118570736831901886479225810414714078935386562497968
178729127629594924411960961386713946279899275006954917139758796061223803393
537381034666494402951052059047968693255388647930440925104186817009640171
764133172418132836351 687 Stellen

18. $2^{3217}-1$ =

259117086013202627776246767922441530941818887553125427303974923161874019266586362086
201209516800483406550695241733194177441689509238807017410377709597512042313066624082916
353517952311186154862265604547691127595848775610568757931191017711408826252153849035
830401185072116424747461823031471398340229288074545677907941037288235820705892351068433
882986888616658650280927692080339605869308790500409503709875902119018371991620994002
568935113136548829739112656797303241986517250116412703509705427773477972349821676443
446668383119322540099648994051790241624056519054483690809616061625743042361721863339415
852426431208737266591962061753535748892894599629195183082621860853400937932839420261
866586142503251450773096274235376822938649407127700846077124211823080804139298087057
504713825264571448379371125032081826126566649084251699453951887789613650248405739378594
599444335231188280123660406262468609212150349937584782292237144339628858485938215738
8212323936870461606773629093150 71 969 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)
52.	$2^{136279841}-1$ =	8816943275...	41024320 Stellen ; 2024 gefunden (Luke Durant)

JAVA-Methode Mersennezahl („schnell“) erzeugen :

```java
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** , die der 52. Mersenneprimzahl **99s** !

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 $MF(p, n) = \dfrac{2^{p^n} - 1}{2^{p^{n-1}} - 1}$

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) = 2417851639229258349412351/134217727 = 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 =
107·1659431·1325815267337711173·47198858799491425660200071 yafu: 0,3s

R71 = 11111111111111111111111111111111111111111111111111111111111111111111111 =
241573142393627673576957439049·45994811347886846310221728895223034301839 yafu: 14s msi: 16s

R97 = 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111 = 12004721·846035731396919233767211537899097169·
109399846855370537540339266842070119107662296580348039 yafu: 425 s msi: 843 s

R114 = 1111...11 = 3· 7· 11· 13· 37· 21319· 1458973· 10749631· 9090909090909091·
1111111111111111111· 3931123022305129377976519· 753201806271328462547977919407
yafu: 1 s msi: 22 s

R131 = 1111...11 = 80173·109517·141811693·446790173·7370364319027·1559484553802942993 3·
7317723970031057677693·131758351065116151205213·180222062287834025451247081
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·
129694419029057750551385771184564274499075700947656757821537291527196801
yafu: 0,8s msi: 336s brent: 138s

R431 = 1111...11 = 3449· 42362129· 7567197163·
1004966229568265515676958784186223832671216578545054436233816798348301108370391615684925612435767
1420972263968215674245849126271968166459708838456187409580812031972219066945092978777944826800714
2765119489395792674211601029617490442193327905462270185549168045859849838216207123634377499937727
6211922377116683204427778244358230936452572738769098150148912609050330781044988771317170703882389
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).

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

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) = 82731255909110452484796229775841512044792296393241693 =
13· 1951· 19141· 22621· 60601· 73951· 303551· 438472201· 12629757106815551
brent: 0,3 s
...
R97(12) = 435700623537534460534556100566797400050569661118420894078389027832099599815
93077811330507328327968191581   prim !
...
R100(12) = 752890677472859547803712941779425907287384374412631304967456240093868108481
92838457979116663350729035052125 =
5· 5· 5· 13· 29· 101· 1201· 1951· 19141· 22621· 60601· 73951· 303551· 85403261·
438472201· 700936801· 2334798291701· 14807687049800501· 12629757106815551   brent: 14s
...
R107(12) = 269774342001974285278694638169555674601473577261535176943833382434453840404
4762934601323776290783959444013016249437 =
126047 ·  8368183883 · 255762526541912675998574100663845947967020971943416941716598897
0731794389915911622153405973231480937       brent: 0,9s
...
R109(12) = 38847505248284297080132027896416017142612195125661065479912007070561353018
2445862582590623785872890159937874339918941   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} mod m = 1 gilt

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