Számítógépes számelmélet

Járai Antal

Ezek a programok csak szemléltetésre szolgálnak

▶ 1. A prímek eloszlása, szitálás

▶ 2. Egyszerű faktorizálási módszerek

▶ 3. Egyszerű prímtesztelési módszerek

▶ 4. Lucas–sorozatok

▼ 5. Alkalmazások

```
> restart; with(numtheory);
```

[*GIgcd*, *bigomega*, *cfrac*, *cfracpol*, *cyclotomic*, *divisors*, *factorEQ*, *factorset*, *fermat*, *imagunit*, *index*, *integral_basis*, *invcfrac*, *invphi*, *issqrfree*, *jacobi*, *kronecker*, λ, *legendre*, *mcombine*, *mersenne*, *migcdex*, *minkowski*, *mipolys*, *mlog*, *mobius*, *mroot*, *msqrt*, *nearestp*, *nthconver*, *nthdenom*, *nthnumer*, *nthpow*, *order*, *pdexpand*, ϕ, π, *pprimroot*, *primroot*, *quadres*, *rootsunity*, *safeprime*, σ, *sq2factor*, *sum2sqr*, τ, *thue*] (5.1)

▶ 5.1. Fermat–számok.

▼ 5.2. Feladat.

```
> #
  # This procedure try to find factors i*2^(n+2)+1, 1<=i<=k
  # of the Fermat number 2^(2^n)+1.
  #

  fermatfactors:=proc(n::posint,k::posint) local i,f;
  f:=[];
  for i from 1+2^(n+2) by 2^(n+2) to k*2^(n+2)+1 do
    if 2&^(2^n)+1 mod i=0 then f:=[op(f),i] fi
  od; f end;
```

fermatfactors := **proc**(*n*::*posint*, *k*::*posint*) (5.2.1)

```
    local i, f;
    f := [ ];
    for i from 1 + 2^(n + 2) by 2^(n + 2) to k*2^(n + 2) + 1 do
        if mod(2 &^ (2^n) + 1, i) = 0 then
            f := [op(f), i]
        end if
    end do;
    f
end proc
```

```
> fermatfactors(5,10); fermatfactors(5,100000);
```

$$[641]$$

$$[641, 6700417] \tag{5.2.2}$$

5.3. Feladat.

```
> interface(verboseproc=2);
```

$$1 \tag{5.3.1}$$

```
> print(fermat);
```

```
proc(n::(Or(nonnegint, Not(constant))), _info)                    (5.3.2)
    option
    Copyright (c) 1992 by the University of Waterloo. All rights reserved.;
    local f, info;
    if nargs = 0 then
        return cat( `The primality character of the following  `,
        `Fermat numbers is known to Maple:`),
        sort(convert(known_fermat_numbers, 'list'))
    elif type(n,
    'name') then
        return 2^(2^n) + 1
    elif type(n, 'nonnegint') then
        if not member(n, known_fermat_numbers) then
            return `character unknown`
        elif n < 22 or _EnvTryHard = true then
            f := 2^(2^n) + 1
```

```
    else
        f:= `object too big`
    end if;
    if nargs = 2 then
        if n <= 4 then
            _info := `it is prime`
        elif n = 5 then
            _info := `it is completely factored `,
            (ifactor(640) + 1)*(ifactor(6700416) + 1)
        elif n = 6 then
            _info := `it is completely factored `,
            (ifactor(274176) + 1)*(ifactor(67280421310720) + 1)
        elif n = 7 then
            _info := `it is completely factored `,
            (ifactor(59649589127497216)
            + 1)*(ifactor(5704689200685129054720) + 1)
        elif n = 8 then
            _info := `it is completely factored `,
            (ifactor(1238926361552896)
            + 1)*((2)^11*(
            4563556626726463758259939365215180497268126833\
                0878021767715) + 1)
        elif n = 9 then
            _info := `it is completely factored `,
            (ifactor(2424832)
            + 1)*((
            364043106721088096110224401181662837831219059\
            7)*(2)^11 + 1)*(
            (
                362128936829849024182024971631805407255830\
                45952027296089151431452364050757065674223282\
                16\
                36569307)*(2)^11 + 1)
```

```
elif n = 10 then
    _info := `it is completely factored `,
    (ifactor(45592576) + 1)*(ifactor(6487031808)
    + 1)*((113764057256348108966419940016522905 1)
    *(2)^12 + 1)*(
    (1 /
    112900309264422855029854345952491376300349629005\
        30512583237632*f - 1 / 8192)*(2)^13 + 1)
elif n = 11 then
    _info := `it is completely factored `,
    (ifactor(319488) + 1)*(ifactor(974848)
    + 1)*((10253207784531279)*(2)^14
    + 1)*((434673084282938711)*(2)^13
    + 1)*((1 /
    152621456713776304010443922801743403340729872 18\
        65090998272*f - 1 / 8192)*(2)^13 + 1)
elif n = 14 then
    _info := `it is composite `
elif n = 20 then
    _info := `it is composite`
elif n = 22 then
    _info := `it is composite`
elif n = 24 then
    _info := `it is composite`
elif member(n,
known_fermat_numbers) then
    if assigned(fermat_tab[n]) then
        info := fermat_tab[n]
    else
        error "number recognized but no factor known"
    end if;
    _info := if(nops(info) = 1, `it has this prime factor `,
```

```
            `it has these prime factors `), op(info)
        else
            return `no complete factorization is known`
        end if
    end if;
    return f
else
    'procname(args)'
end if
end proc
```

5.4. Mersenne-számok.

```
> mersennes:=[2,3,5,7,13,17,19,31,61,89,107,127,521,607,1279,
  2203,2281,3217,4253,4423,9689,9941,11213,19937,21701,23209,
  44497,86243,110503,132049,216091,756839,859833,1257787,
  1398269,2976221,3021377,6972593,13466917,20996011,24036583,
  25964951,30402457,32582657,37156667,42643801,43112609];
```

$$mersennes := [2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091, 756839, 859833, 1257787, 1398269, 2976221, 3021377, 6972593, 13466917, 20996011, 24036583, 25964951, 30402457, 32582657, 37156667, 42643801, 43112609] \quad (5.4.1)$$

```
> map(p->p mod 4, mersennes);
```

$$[2, 3, 1, 3, 1, 1, 3, 3, 1, 1, 3, 3, 1, 3, 3, 3, 1, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 3, 3, 1, 3, 3, 1, 3, 1, 1, 1, 1, 1, 3, 3, 3, 1, 1, 3, 1, 1] \quad (5.4.2)$$

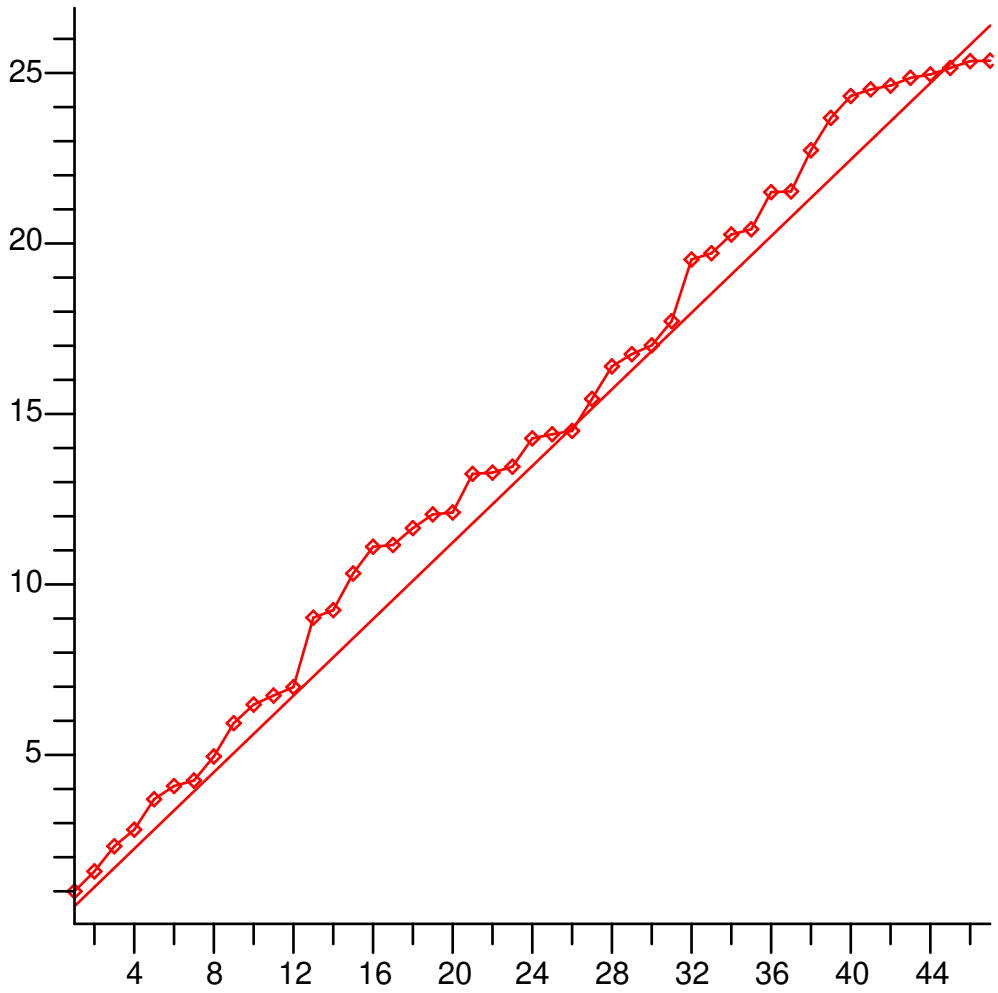
```
> logm:=evalf([[i,log[2.](mersennes[i])]$i=1..nops(mersennes)]
  );
```

$$logm := [[1., 1.000000000], [2., 1.584962501], [3., 2.321928094], [4., 2.807354922], [5., 3.700439718], [6., 4.087462841], [7., 4.247927514], [8., 4.954196310], [9., 5.930737338], [10., 6.475733432], [11., 6.741466986], [12., 6.988684687], [13., 9.025139563], [14., 9.245552707], [15., 10.32080055], [16., 11.10525378], [17., 11.15545073], [18., 11.65150022], [19., 12.05426514], [20., 12.11080953], [21., 13.24213206], [22., \quad (5.4.3)$$

```
13.27917527], [23., 13.45288470], [24., 14.28316072], [25.,
14.40547390], [26., 14.50239674], [27., 15.44142045], [28.,
16.39611974], [29., 16.75372601], [30., 17.01071384], [31.,
17.72127946], [32., 19.52962691], [33., 19.71369695], [34.,
20.26245621], [35., 20.41521050], [36., 21.50505022], [37.,
21.52677479], [38., 22.73326385], [39., 23.68291627], [40.,
24.32361193], [41., 24.51872849], [42., 24.63006217], [43.,
24.85768459], [44., 24.95760092], [45., 25.14711776], [46.,
25.34583271], [47., 25.36160653]]
```

```
> with(plots); plotm:=plot(logm):
  plotline:=plot(evalf(exp(-gamma))*x,x=1..nops(mersennes)):
  pointm:=plot(logm,style=POINT):
  display({plotline,plotm,pointm});
```

```
[Interactive, animate, animate3d, animatecurve, arrow, changecoords,
  complexplot, complexplot3d, conformal, conformal3d, contourplot,
  contourplot3d, coordplot, coordplot3d, cylinderplot, densityplot, display,
  display3d, fieldplot, fieldplot3d, gradplot, gradplot3d, graphplot3d,
  implicitplot, implicitplot3d, inequal, interactive, interactiveparams,
  listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d, loglogplot,
  logplot, matrixplot, multiple, odeplot, pareto, plotcompare, pointplot,
  pointplot3d, polarplot, polygonplot, polygonplot3d, polyhedra_supported,
  polyhedraplot, replot, rootlocus, semilogplot, setoptions, setoptions3d,
  spacecurve, sparsematrixplot, sphereplot, surfdata, textplot, textplot3d,
  tubeplot]
```



5.5. Feladat.

```
> #
  # This procedure do a pretest for the Mersenne numbers
  # M[n]=2^n-1. The exponents are taken from
  # the sequence L. The result is the sequence of sequences,
  # which contains all exponents n which is prime and the
  # prime divisors having the form 2*k*n+1 for k until K, where
  # k congruent to 0 or -n mod 4.
  #

  mersennepretest:=proc(L::list(posint),K::posint)
  local nL,D,M,n,k,d1,d2; nL:=[];
  for n in L do
    if isprime(n) then
      D:=[n]; d1:=modp(n,4); d2:=modp(-n,4);
      for k from d2 while k<=K do
        M:=2*k*n+1;
```

```
        if isprime(M) then
          if pmod(2&^n-1,M)=0 then D:=[op(D),M]; fi;
        fi;
        k:=k+d1; M:=2*k*n+1;
        if isprime(M) then
          if 2&^n-1 mod M=0 then D:=[op(D),M] fi;
        fi;
        k:=k+d2;
      od;
      nL:=[op(nL),D];
    fi;
  od; nL end;
```

```
mersennepretest := proc(L::(list(posint)), K::posint)                    (5.5.1)
  local nL, D, M, n, k,
  d1, d2;
  nL := [ ];
  for n in L do
    if isprime(n) then
      D := [n];
      d1 := modp(n, 4);
      d2 := modp(-n, 4);
      for k from d2 while k <= K do
        M := 2 * k * n + 1;
        if isprime(M) then
          if pmod(2 &^ n - 1, M) = 0 then
            D := [op(D), M]
          end if
        end if;
        k := k + d1;
        M := 2 * k * n + 1;
        if isprime(M) then
          if mod(2 &^ n - 1,
          M) = 0 then
            D := [op(D), M]
          end if
        end if;
```

```
            k := k + d2
        end do;
        nL := [op(nL), D]
    end if
end do;
nL
end proc
```

```
> L:=[i$i=20..1000];
```

$$L := [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, \quad (5.5.2)$$
$$39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57,$$
$$58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76,$$
$$77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,$$
$$96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110,$$
$$111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124,$$
$$125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138,$$
$$139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152,$$
$$153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166,$$
$$167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180,$$
$$181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,$$
$$195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,$$
$$209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,$$
$$223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236,$$
$$237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250,$$
$$251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264,$$
$$265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278,$$
$$279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292,$$
$$293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306,$$
$$307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320,$$
$$321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334,$$
$$335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348,$$
$$349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362,$$
$$363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,$$
$$377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,$$

391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838,

839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000]

```
> LL:=mersennepretest(L,10000);
```

$$LL := [[23], [29, 233, 1103], [31], [37], [41], [43], [47], [53], [59], [61], [67], [71, 228479], [73], [79], [83], [89], [97, 11447], [101], [103], [107], [109], [113], [127], [131], [137], [139], [149], [151, 165799], [157], [163], [167], [173], [179, 1433], [181], [191], [193], [197, 7487], [199], [211], [223], [227], [229, 1504073], [233], [239, 1913, 176383], [241], [251], [257], [263], [269], [271], [277, 1121297], [281, 80929], [283], [293], [307], [311], [313], [317], [331], [337, 2806537], [347], [349], [353], [359], [367], [373], [379], [383], [389], [397], [401], [409], [419], [421], [431, 3449], [433], [439], [443], [449, 1256303], [457], [461], [463], [467], [479], [487], [491, 7707719], [499], [503], [509], [521], [523], [541], [547], [557], [563], [569], [571, 27409], [577], [587], [593], [599], [601], [607], [613], [617], [619, 110183], [631], [641, 49999], [643], [647], [653], [659], [661], [673], [677], [683], [691], [701], [709], [719], [727], [733], [739], [743], [751], [757], [761, 6089], [769], [773], [787], [797], [809], [811], [821], [823], [827], [829, 72953], [839], [853], [857, 6857], [859], [863], [877, 1436527], [881], [883], [887], [907], [911], [919], [929], [937], [941, 7529], [947], [953], [967, 549257], [971], [977, 867577], [983], [991], [997]] \quad (5.5.3)$$

```
> nops(LL); select(x->evalb(nops(x)=1),LL); nops(%);
```

160

[[23], [31], [37], [41], [43], [47], [53], [59], [61], [67], [73], [79], [83], [89], [101], [103], [107], [109], [113], [127], [131], [137], [139], [149], [157], [163], [167], [173], [181], [191], [193], [199], [211], [223], [227], [233], [241], [251], [257], [263], [269], [271], [283], [293], [307], [311], [313], [317], [331], [347], [349], [353], [359], [367], [373], [379], [383], [389], [397], [401], [409], [419], [421], [433], [439], [443], [457], [461], [463], [467], [479], [487], [499], [503], [509], [521], [523], [541], [547], [557], [563], [569], [577], [587], [593], [599], [601], [607], [613], [617], [631], [643], [647], [653], [659], [661], [673], [677], [683], [691], [701], [709], [719], [727], [733], [739], [743], [751], [757], [769], [773], [787], [797], [809], [811], [821], [823], [827], [839], [853], [859], [863], [881], [883], [887], [907], [911], [919], [929], [937], [947], [953], [971], [983], [991], [997]]

136 (5.5.4)

5.6. Feladat.

```
> interface(verboseproc=2);
```

2 (5.6.1)

```
> print(mersenne);
```

proc(*n*::({*posint*, [*posint*]})) (5.6.2)

option

Copyright (c) 1992 by the University of Waterloo. All rights reserved.;

local *w*;

if *nargs* <> 1 **then**

error "wrong number of arguments"

end if;

w := [2, 3, 5, 7, 13, 17, 19, 31, 61, 89, 107, 127, 521, 607, 1279, 2203, 2281, 3217, 4253, 4423, 9689, 9941, 11213, 19937, 21701, 23209, 44497, 86243, 110503, 132049, 216091, 756839, 859433, 1257787, 1398269, 2976221, 3021377, 6972593, 13466917, 20996011, 24036583, 25964951, *NULL*];

```
    if type(n, ['posint']) then
        if nops(w) < n[1] then
            error "index too large"
        end if;
        2^w[n[1]] - 1
    else
        if member(n, w) then
            2^n - 1
        elif n < w[-1] or not isprime(n) then
            false
        else
            FAIL
        end if
    end if
end proc
```

5.7. h*2^m+-1 alakú prímek keresése.

```
> #
  # A pretest will be done for (h0+c*h)*b^n+d, where h runs
  # from 0 to H-1 and d in D. All prime divisors from P0 to
  # P1 will be tried. P0 must be odd, and b,c must have prime
  # factors smaller then P0. The values h, for which
  # (h0+c*h)*b^n+d not divisible with primes from P0 to P1
  # for all d in D are given back.
  #

  pretest:=proc(h0::nonnegint,H::posint,c::posint,b::posint,
  n::posint,
  D::set(integer),P0::posint,P1::posint)
  local T,h,p,d,hh,L,nn,bb,cc,bT,cT,i,j;
  T:=array(0..H-1);
  for h from 0 to H-1 do T[h]:=1 od;
  bT:=array(0..b-1);                    # table of inverses for
  b
  for i from 0 to b-1 do bT[i]:=0 od;
  for i from 0 to b-1 do
    for j from 0 to b-1 do
      if irem(i*j+1,b)=0 then bT[i]:=j fi
    od
```

```
  od;
  cT:=array(0..c-1);                    # table of inverses for
  c
  for i from 0 to c-1 do cT[i]:=0 od;
  for i from 0 to c-1 do
    for j from 0 to c-1 do
      if irem(i*j+1,c)=0 then cT[i]:=j fi
    od
  od;
  for p from P0 to P1 by 2 do
    if isprime(p) then
      nn:=modp(n,p-1);
      bb:=(bT[modp(p,b)]*p+1)/b;
      bb:=modp(bb&^nn,p);
      cc:=(cT[modp(p,c)]*p+1)/c;
      for d in D do
        hh:=modp((-d*bb-h0)*cc,p);
        for h from hh to H-1 by p do T[h]:=0; od
      od
    fi
  od;
  L:=[];
  for h from 0 to H-1 do
    if T[h]=1 then L:=[op(L),h] fi
  od; L end;
```

```
pretest := proc(h0::nonnegint, H::posint, c::posint, b::posint, n::posint,
  D::(set(integer)), P0::posint, P1::posint)
  local T, h, p, d, hh, L, nn, bb,
  cc, bT, cT, i, j;
  T := array(0..H - 1);
  for h from 0 to H - 1 do
      T[h] := 1
  end do;
  bT := array(0..b - 1);
  for i from 0 to b - 1 do
      bT[i] := 0
  end do;
  for i from 0 to b - 1 do
      for j from 0 to b - 1 do
          if irem(i*j + 1, b) = 0 then
              bT[i] := j
```

(5.7.1)

```
            end if
        end do
    end do;
    cT := array(0..c − 1);
    for i from 0 to c − 1 do
        cT[i] := 0
    end do;
    for i from 0 to c − 1 do
        for j from 0 to c − 1 do
            if irem(i * j + 1,
            c) = 0 then
                cT[i] := j
            end if
        end do
    end do;
    for p from P0 by 2 to P1 do
        if isprime(p) then
            nn := modp(n, p − 1);
            bb := (bT[modp(p, b)] * p + 1) / b;
            bb := modp(bb &^ nn, p);
            cc := (cT[modp(p, c)] * p + 1) / c;
            for d in D do
                hh := modp((−d * bb − h0) * cc, p);
                for h from hh by p to H − 1 do
                    T[h] := 0
                end do
            end do
        end if
    end do;
    L := [ ];
    for h from 0 to H − 1 do
        if T[h] = 1 then
```

```
            L := [op(L), h]
        end if
    end do;
    L
end proc
```

```
> #
  # This procedure do the Proth test for the number n=h*2^m+1.
  #
  proth:=proc(h::posint,m::posint) local a,b,n; n:=h*2^m+1;
  if h>=2^m then error "first parmeter is too large",h fi;
  if not type(h,odd) then error "first parameter must be odd",h
  fi;
  if m=1 or m=2 then return true fi;
  if m=3 then if h=5 then return true else return false fi fi;
  # 2 is not a quadratic residue mod n, we start with 3
  # (2*a|n)=(a|n), hence enough to try odd bases a
  for a from 3 by 2 do
    b:=2&^m mod a; b:=h*b+1 mod a;
    # by quadratic reciprocity, (a|n)=(n|a)=(b|a)
    b:=modp(2&^m,a); b:=modp(h*b+1,a);
    if jacobi(b,a)=0 then return false fi;
    if jacobi(b,a)=-1 then
      if modp(a&^((n-1)/2),n)=n-1 then return true else return
  false fi;
    fi;
  od end;
```

$$proth := \mathbf{proc}(h::posint,\ m::posint) \tag{5.7.2}$$

```
    local a, b, n;
    n := h*2^m+1;
    if 2^m <= h then
        error "first parmeter is too large", h
    end if;
    if not type(h, odd) then
        error "first parameter must be odd", h
    end if;
    if m = 1 or m = 2 then
        return true
    end if;
    if m = 3 then
```

```
        if h = 5 then
            return true
        else
            return false
        end if
    end if;
    for a from 3 by 2 do
        b := mod(2 &^ m, a);
        b := mod(h*b + 1, a);
        b := modp(2 &^ m, a);
        b := modp(h*b + 1, a);
        if numtheory:-jacobi(b, a) = 0 then
            return false
        end if;
        if numtheory:-jacobi(b, a) = -1 then
            if modp(a &^ (1 / 2*n - 1 / 2), n) = n - 1 then
                return true
            else
                return false
            end if
        end if
    end do
end proc
```

```
> #
  # After a pretest the Proth test is used to find primes
  # (h0+c*h)*2^n+1, where h runs from 0 to H-1. All prime
  # divisors from interval P will be tried during the pretest.
  #

  prothprimes:=proc(h0::posint,c::posint,H::posint,n::posint,
  P::range(posint))
  local h,L,LL;
  L:=pretest(h0,H,c,2,n,{1},op(1,P),op(2,P)); LL:=[];
  for h in L do
    if proth(h0+c*h,n) then LL:=[op(LL),h] fi
  od; LL end;
```

```
prothprimes := proc(h0::posint, c::posint, H::posint, n::posint,
```

(5.7.3)

```
    P::(range(posint)))
    local h, L, LL;
    L := pretest(h0, H, c, 2, n, {1}, op(1,
    P), op(2, P));
    LL := [];
    for h in L do
        if proth(h0 + c* h, n) then
            LL := [op(LL), h]
        end if
    end do;
    LL
end proc
```

```
> prothprimes(3,30,3000,2000,7..1000);
```

[518, 591, 861, 1294, 1550, 2879, 2921] (5.7.4)

5.8. Feladat.

5.9. Feladat.

```
> #
  # This procedure calculate the Lucas sequence terms
  # [V[n],V[n+1]]. If the parameter N also given
  # then the calculations are done mod N. Here V[n]=a^n+b^n,
  # where a,b are the roots of x^2-Px+Q=0.
  #

  lucasV:=proc(n,P,Q,N) local L,B,i,Qk;
  B:=convert(n,base,2);
  Qk:=1;
  if nargs=3 then L:=[2,P];
    for i from nops(B) to 1 by -1 do
      if B[i]=0 then
        L:=[L[1]^2-2*Qk,L[2]*L[1]-P*Qk]; Qk:=Qk^2;
      else
        L:=[L[2]*L[1]-P*Qk,L[2]^2-2*Q*Qk]; Qk:=Qk^2*Q;
      fi;
    od;
  else L:=[2 mod N,P mod N];
    for i from nops(B) to 1 by -1 do
      if B[i]=0 then
        L:=[L[1]&^2-2*Qk mod N,L[2]*L[1]-P*Qk mod N]; Qk:=Qk&^2
```

```
    mod N;
        else
          L:=[L[2]*L[1]-P*Qk mod N,L[2]&^2-2*Q*Qk mod N]; Qk:=
    Qk&^2*Q mod N;
        fi;
      od;
    fi; L end;
```

$$lucasV := \mathbf{proc}(n, P, Q, N) \tag{5.9.1}$$

$\mathbf{local}\ L, B, i, Qk;$

$B := convert(n, base, 2);$

$Qk := 1;$

$\mathbf{if}\ nargs = 3\ \mathbf{then}$

$L := [2, P];$

$\mathbf{for}\ i\ \mathbf{from}\ nops(B)\ \mathbf{by}\ -1\ \mathbf{to}\ 1\ \mathbf{do}$

$\mathbf{if}\ B[i] = 0\ \mathbf{then}$

$L := [L[1]\text{^}2 - 2 * Qk, L[2] * L[1] - P * Qk];$

$Qk := Qk\text{^}2$

$\mathbf{else}$

$L := [L[2] * L[1] - P * Qk,$

$L[2]\text{^}2 - 2 * Q * Qk];$

$Qk := Qk\text{^}2 * Q$

$\mathbf{end\ if}$

$\mathbf{end\ do}$

$\mathbf{else}$

$L := [mod(2, N), mod(P, N)];$

$\mathbf{for}\ i\ \mathbf{from}\ nops(B)\ \mathbf{by}\ -1\ \mathbf{to}\ 1\ \mathbf{do}$

$\mathbf{if}\ B[i] = 0\ \mathbf{then}$

$L := [mod(L[1]\ \&\text{^}\ 2 - 2 * Qk, N),$

$mod(L[2] * L[1] - P * Qk, N)];$

$Qk := mod(Qk\ \&\text{^}\ 2, N)$

$\mathbf{else}$

$L := [mod(L[2] * L[1] - P * Qk, N), mod(L[2]\ \&\text{^}\ 2 - 2 * Q *$

$Qk,$

$N)];$

$Qk := mod(Qk\ \&\text{^}\ 2 * Q, N)$

```
            end if
        end do
    end if;
    L
end proc
```

```
> #
  # This procedure calculate a^h+a^(-h) for a unit a=r+s*sqrt
  (D)
  # with norm 1. Here r,s rational numbers, D must be a square
  # free integer. All computations are done mod N.
  #

  rieselsetup:=proc(r,s,D,h,N) local P,a;
  a:=r+s*sqrt(D);
  P:=r+s*sqrt(D)+(r-s*sqrt(D))/(r^2-s^2*D);
  if not type(P,integer) then ERROR(a+1/a, `not an integer`)
  fi;
  lucasV(h,P,1,N)[1]; end;
```

$$rieselsetup := \mathbf{proc}(r, s, \mathrm{D}, h, N) \tag{5.9.2}$$

```
    local P, a;
    a := r + s*sqrt(D);
    P := r + s*sqrt(D) + (r - s*sqrt(D)) / (r^2 - s^2*D);
    if not type(P, integer) then
        ERROR(a + 1 / a, `not an integer`)
    end if;
    lucasV(h, P, 1, N)[1]
end proc
```

```
> #
  # This procedure use iteration v<-v^2-2 starting with v=a^h+
  a^-h
  # where the quadratic unit a=r+s*sqrt(D) is given for the
  # primality testing of h*2^n-1. Here n>1 and h<2^n is odd.
  #

  riesel:=proc(h,n,r,s,D) local N,v,i;
  if h>=2^n then error "first parameter is too large",h fi;
  if type(h,even) then error "first parameter have to be odd",h
  fi;
  if n<2 then error "second parameter is too small",h fi;
  N:=h*2^n-1;
  v:=rieselsetup(r,s,D,h,N);
```

```
   for i to n-2 do v:=v^2-2 mod N od;
   evalb(v=0); end;
```

riesel := **proc**(*h*, *n*, *r*, *s*, D) (5.9.3)

local *N*, *v*, *i*;

if 2^*n* <= *h* **then**

error "first parameter is too large", *h*

end if;

if *type*(*h*, *even*) **then**

error "first parameter have to be odd", *h*

end if;

if *n* < 2 **then**

error "second parameter is too small", *h*

end if;

N := *h* * 2^*n* − 1;

v := *rieselsetup*(*r*, *s*, D, *h*, *N*);

for *i* **to** *n* − 2 **do**

v := *mod*(*v*^2 − 2, *N*)

end do;

evalb(*v* = 0)

end proc

```
> #
  # After a pretest the Riesel test with D=5 and unit
  # (1+sqrt(5))^2/4 is used to find primes (h0+30*h)*2^n-1,
  # where h runs from 0 to H-1. Only the pairs [3,0],[9,0],
  # [23,0],[29,0],[7,1],[9,1],[19,1],[27,1],[11,2],[17,2],
  # [21,2],[27,2],[1,3],[3,3],[13,3],[21,3] for
  # [h0 mod 30,n mod 4] gives combinations for which the
  # test may used and N is not divisible by 2,3,5, hence
  # this procedure should be used only with these pairs.
  # All prime divisors until P will be tried during the
  # pretest.
  #

  rieselsqrt5:=proc(h0,H,n,P) local h,L,LL;
  L:=pretest(h0,H,30,2,n,{-1},7,P); LL:=[];
  for h in L do
    if riesel(h0+30*h,n,3/2,1/2,5) then LL:=[op(LL),h] fi
  od; LL end;
```

rieselsqrt5 := **proc**(*h0*, *H*, *n*, *P*) (5.9.4)

```
    local h, L, LL;
    L := pretest(h0, H, 30, 2, n,
    {-1}, 7, P);
    LL := [ ];
    for h in L do
        if riesel(h0 + 30 * h, n, 3 / 2, 1 / 2,
        5) then
            LL := [op(LL), h]
        end if
    end do;
    LL
end proc
```

```
> rieselsqrt5(3,3000,2000,1000);
```

$$[193, 713, 730, 1264, 1337, 1349, 1392, 1492, 1512, 2148, 2829, 2974] \quad (5.9.5)$$

5.10. Ikerprímek.

```
> #
  # This filter left only those h's from the list L in the
  # resulting list for which (h0+c*h)*2^n+d is base-2
  pseudoprime.
  #

  prob:=proc(h0::nonnegint,c::posint,n::posint,d::integer,
  L::list(nonnegint)) local LL,h,N; LL:=[];
  for h in L do
    N:=(h0+c*h)*2^n+d;
    if modp(3&^(N-1),N)=1 then LL:=[op(LL),h] fi
  od; LL end;
```

$$prob := \mathbf{proc}(h0::nonnegint, c::posint, n::posint, d::integer, L::(list(nonnegint))) \quad (5.10.1)$$

```
    local LL, h, N;
    LL := [ ];
    for h in L do
        N := (h0 + c * h) * 2^n + d;
        if modp(3 &^ (N - 1), N) = 1 then
            LL := [op(LL), h]
```

```
        end if
    end do;
    LL
end proc
```

```
> #
  # This filter left only those h's from the list L in the
  # resulting list for which (h0+c*h)*2^n+1 is prime. It use
  # the Proth test.
  #

  exactplus:=proc(h0,c,n,L) local LL,h,a,S;
  LL:=[];
  for h in L do
    if proth(h0+c*h,n) then LL:=[op(LL),h] fi
  od; LL end;
```

$$exactplus := \mathbf{proc}(h0, c, n, L) \tag{5.10.2}$$

```
    local LL, h, a, S;
    LL := [ ];
    for h in L do
        if proth(h0 + c* h, n) then
            LL := [op(LL), h]
        end if
    end do;
    LL
end proc
```

5.11. Feladat.

```
> #
  # This filter left only those h's from the list L in the
  # resulting list for which (h0+30*h)*2^n-1 is prime.
  #

  exactminus:=proc(h0,n,L) local LL,h;
  LL:=[];
  for h in L do
    if riesel(h0+30*h,n,3/2,1/2,5) then LL:=[op(LL),h] fi
  od; LL end;
```

$$exactminus := \mathbf{proc}(h0, n, L) \tag{5.11.1}$$

```
    local LL, h;
```

```
    LL:= [ ];
    for h in L do
        if riesel( h0 + 30 * h, n, 3 / 2, 1 / 2, 5 ) then
            LL:= [ op( LL ), h ]
        end if
    end do;
    LL
end proc
```

5.12. Feladat.

```
> #
  # After a pretest the probabilistic test for N-1,
  # the Riesel test with D=5 and unit (1+sqrt(5))^2/4 for N-1,
  # the probabilistic test for N+1, and the Proth test for N+1
  # is used to find twin primes N+-1 with N=(h0+30*h)*2^n,
  # where h runs from 0 to H-1. Only the pairs
  # [3,0],[9,1],[21,3],[27,2], for [h0 mod 30,n mod 4] gives
  # combinations for which the Riesel test may used and
  # N+-1 is not divisible by 2,3,5, hence this procedure
  # should be used only with these pairs. All prime divisors
  # until P will be tried during the pretest.
  #

  twin:=proc(h0,H,n,P) local LL;
  LL:=pretest(h0,H,30,2,n,{1,-1},7,P);
  print(nops(LL));
  LL:=prob(h0,30,n,-1,LL);
  print(nops(LL));
  LL:=exactminus(h0,n,LL);
  print(nops(LL));
  LL:=prob(h0,30,n,1,LL);
  print(nops(LL));
  LL:=exactplus(h0,30,n,LL)
  end;
```

(5.12.1)

```
twin:= proc( h0, H, n, P )
    local LL;
    LL:= pretest( h0, H, 30, 2, n, { -1, 1 },
    7, P );
    print( nops( LL ) );
    LL:= prob( h0, 30, n, -1, LL );
```

```
    print(nops(LL));
    LL:= exactminus(h0, n, LL);
    print(nops(LL));
    LL:= prob(h0, 30, n, 1, LL);
    print(nops(LL));
    LL:= exactplus(h0,
   30, n, LL)
end proc
```

```
> twin(3,20000,400,1000);
```

$$1739$$
$$72$$
$$72$$
$$1$$
$$[16411] \qquad (5.12.2)$$

```
> #
  # This is a test for measurement densities and times.
  # First a pretests is done for N+-1 where N=(h0+30*h)*2^n
  # and h runs from 0 to H-1. The pretest is repeated for
  # prime limit P=2^e where e is integer from the interval
  # eint. The time and the remaining number after the
  # pretest is reported, and p*ln(P)^2 also calculated,
  # where p is the relative frequency of numbers
  # passed the pretest. The result of the last pretest
  # became the input of a probabilistic test for N-1 and
  # the result of this will be the input of the Riesel test
  # with D=5 and unit (1+sqrt(5))^2/4 for N-1. The time of
  # the probabilistic test and the result of booth tests is
  # reported. The probabilistic test for N+1 and the
  # Proth test for N+1 also done. Only the pairs
  # [3,0],[9,1],[21,3],[27,2] for [h0 mod 30,n mod 4] gives
  # combinations for which the Riesel test
  # may used and N+-1 is not divisible
  # by 2,3,5, hence this procedure
  # should be used only with these pairs.
  #

  twintest:=proc(h0,H,n,eint) local e,p,P,L,t,N;
  print(`Twintest; h0=`,h0,`H=`,H,`n=`,n);
  N:=(h0+15*H)*2^n;
  print(`Pretest for N+-1`);
  for e from op(1,eint) to op(2,eint) do
    P:=2^e;
```

```
    t:=time();
    L:=pretest(h0,H,30,2,n,{1,-1},7,P);
    t:=time()-t;
    p:=evalf(nops(L)/H);
    print(`P=`,2^e,`p=`,p,`p*ln(P)^2=`,evalf((ln(P))^2*p),
  `time=`,t)
  od;
  print(`Test for N+-1`);
  t:=time();
  L:=prob(h0,30,n,-1,L);
  L:=exactminus(h0,n,L);
  L:=prob(h0,30,n,1,L);
  L:=exactplus(h0,30,n,L);
  t:=time()-t;
  p:=evalf(nops(L)/H);
  print(`p=`,p,`p*ln(N)^2=`,evalf((ln(N))^2*p),`time=`,t);
  L
  end;
```

$$twintest := \mathbf{proc}(h0, H, n, eint) \tag{5.12.3}$$

```
local e, p, P, L, t, N;
print( `Twintest; h0= `, h0, H=, H, n=, n);
N := (h0 + 15 * H) * 2 ^ n;
print( `Pretest for N+-1 ` );
for e from op(1, eint) to op(2, eint) do
    P := 2 ^ e;
    t := time( );
    L := pretest(h0, H, 30, 2, n, {-1, 1}, 7, P);
    t := time( ) - t;
    p := evalf(nops(L) / H);
    print(P=, 2 ^ e, p=, p,
    `p*ln(P)^2= `, evalf(ln(P) ^ 2 * p), time=, t)
end do;
print( `Test for N+-1 ` );
t := time( );
L := prob(h0, 30, n, -1, L);
L := exactminus(h0, n, L);
L := prob(h0, 30, n, 1, L);
L := exactplus(h0, 30, n, L);
```

```
    t := time() - t;
    p := evalf(nops(L) / H);
    print(p=, p, `p*ln(N)^2=`, evalf(ln(N)^2*p), time=, t);
    L
end proc
```

```
> twintest(3,20000,400,3..15);
```

Twintest; h0=, 3, H=, 20000, n=, 400

Pretest for N+-1

*P=, 8, p=, 0.7142500000, p*ln(P)^2=, 3.088472087, time=, 4.847*

*P=, 16, p=, 0.4944500000, p*ln(P)^2=, 3.800959884, time=, 2.407*

*P=, 32, p=, 0.3098500000, p*ln(P)^2=, 3.721709160, time=, 1.064*

*P=, 64, p=, 0.2279000000, p*ln(P)^2=, 3.941828708, time=, 0.684*

*P=, 128, p=, 0.1705500000, p*ln(P)^2=, 4.015121815, time=, 0.492*

*P=, 256, p=, 0.1332000000, p*ln(P)^2=, 4.095765854, time=, 0.448*

*P=, 512, p=, 0.1062500000, p*ln(P)^2=, 4.134898752, time=, 0.393*

*P=, 1024, p=, 0.08645000000, p*ln(P)^2=, 4.153516306, time=, 0.387*

*P=, 2048, p=, 0.07185000000, p*ln(P)^2=, 4.176986436, time=, 0.429*

*P=, 4096, p=, 0.05945000000, p*ln(P)^2=, 4.113062162, time=, 0.492*

*P=, 8192, p=, 0.05165000000, p*ln(P)^2=, 4.193802291, time=, 0.635*

*P=, 16384, p=, 0.04430000000, p*ln(P)^2=, 4.171677430, time=, 0.883*

*P=, 32768, p=, 0.03860000000, p*ln(P)^2=, 4.172734427, time=, 1.404*

Test for N+-1

*p=, 0.00005000000000, p*ln(N)^2=, 4.201243020, time=, 1.331*

$$[16411] \tag{5.12.4}$$

5.13. Sophie Germain prímek.

```
> #
  # A pretest will be done for Sophie Germain prime candidates,

  # to be more exact, for (h0+c*h)*b^n-1 and for (h0+c*h)*b^
  (n+1)-1,
  # where h runs from 0 to H-1. All prime divisors from P0 to
  P1
  # will be tried. P0 must be odd, and b,c must have prime
  factors
```

```
# smaller then P0. The values h0+c*h, for which (h0+c*h)*b^n
-1 and
# (h0+c*h)*b^(n+1)-1 are not divisible with primes from P0 to
P1
# are given back.
#

presophie:=proc(h0::nonnegint,H::posint,c::posint,b::posint,
n::posint,P0::posint,P1::posint)
local T,h,p,hh,L,nn,bb,bbb,cc,bT,cT,i,j;
T:=array(0..H-1);
for h from 0 to H-1 do T[h]:=1 od;
bT:=array(0..b-1);                    # table of inverses for
b
for i from 0 to b-1 do bT[i]:=0 od;
for i from 0 to b-1 do
  for j from 0 to b-1 do if irem(i*j+1,b)=0 then bT[i]:=j fi
od
od;
cT:=array(0..c-1);                    # table of inverses for
c
for i from 0 to c-1 do cT[i]:=0 od;
for i from 0 to c-1 do
  for j from 0 to c-1 do if irem(i*j+1,c)=0 then cT[i]:=j fi
od
od;
for p from P0 to P1 by 2 do
  if isprime(p) then
    cc:=(cT[p mod c]*p+1)/c;
    nn:=modp(n,p-1);
    bb:=(bT[modp(p,b)]*p+1)/b;
    bbb:=modp(bb&^nn,p);
    hh:=modp((bbb-h0)*cc,p);
    for h from hh to H-1 by p do T[h]:=0; od;
    bbb:=modp(bbb*bb,p);
    hh:=modp((bbb-h0)*cc,p);
    for h from hh to H-1 by p do T[h]:=0 od
  fi
od; L:=[];
for h from 0 to H-1 do
  if T[h]=1 then L:=[op(L),h] fi
od; L end;
```

$$presophie := \mathbf{proc}(h0::nonnegint, H::posint, c::posint, b::posint, n::posint, \quad (5.13.1)$$
$$P0::posint, P1::posint)$$
$$\mathbf{local}\ T, h, p, hh, L, nn, bb, bbb, cc, bT, cT, i, j;$$
$$T := array(0..H-1);$$

```
for h from 0 to H − 1 do
    T[h] := 1
end do;
bT := array(0..b − 1);
for i from 0 to b − 1 do
    bT[i] := 0
end do;
for i from 0 to b − 1 do
    for j from 0 to b − 1 do
        if irem(i*j + 1,
        b) = 0 then
            bT[i] := j
        end if
    end do
end do;
cT := array(0..c − 1);
for i from 0 to c − 1 do
    cT[i] := 0
end do;
for i from 0 to c − 1 do
    for j from 0 to c − 1 do
        if irem(i*j + 1,
        c) = 0 then
            cT[i] := j
        end if
    end do
end do;
for p from P0 by 2 to P1 do
    if isprime(p) then
        cc := (cT[mod(p, c)]*p + 1) / c;
        nn := modp(n, p − 1);
        bb := (bT[modp(p, b)]*p + 1) / b;
```

```
            bbb := modp(bb &^ nn,
           p);
            hh := modp((bbb - h0)*cc, p);
            for h from hh by p to H - 1 do
                T[h] := 0
            end do;
            bbb := modp(bbb*bb, p);
            hh := modp((bbb - h0)*cc, p);
            for h from hh by p to H - 1 do
                T[h] := 0
            end do
        end if
    end do;
    L := [];
    for h from 0 to H - 1 do
        if T[h] = 1 then
            L := [op(L), h]
        end if
    end do;
    L
end proc
> #
  # After a pretest the probabilistic test for N-1, the Riesel
  # test with D=5 and unit (1+sqrt(5))^2/4 for N-1, the
  # probabilistic test for 2*N-1, and the Riesel test for 2*N-1
  # with D=5 and unit (1+sqrt(5))^2/4 is used to find
  # Sophie Germain primes with N=(h0+30*h)*2^n, where h runs
  # from 0 to H-1. Only h0=9 and n mod 4=0 used; in this case
  # the Riesel test can be used and N-1, 2*N-1 is not divisible
  # by 2,3,5 hence this procedure should be used only with this
  # pair. All prime divisors until P will be tried during the
  # pretest.
  #

  sophiegermain:=proc(h0,H,n,P) local LL;
  LL:=presophie(h0,H,30,2,n,7,P);
  print(nops(LL));
  LL:=prob(h0,30,n,-1,LL);
```

```
   print(nops(LL));
   LL:=exactminus(h0,n,LL);
   print(nops(LL));
   LL:=prob(h0,30,n+1,-1,LL);
   print(nops(LL));
   LL:=exactminus(h0,n+1,LL);
   end;
```

$$sophiegermain := \mathbf{proc}(h0, H, n, P) \tag{5.13.2}$$

```
 local LL;
 LL:= presophie(h0, H, 30,
 2, n, 7, P);
 print(nops(LL));
 LL:= prob(h0, 30, n, -1, LL);
 print(nops(LL));
 LL:= exactminus(h0, n, LL);
 print(nops(LL));
 LL:= prob(h0, 30, n+1, -1, LL);
 print(nops(LL));
 LL:= exactminus(h0, n+1, LL)
end proc
```

```
> sophiegermain(9,20000,400,1000);
```

$$1737$$

$$69$$

$$69$$

$$2$$

$$[2803, 6289] \tag{5.13.3}$$

5.14. Ikerprím, amely Sophie Germain prím is.

```
> #
  # This part contains some special calculations
  # which are useful for search of primes N
  # for which N+2 and 2*N+1 are prime too.
  # We try to find p in the form N=h*2^e-1.
  # The test given by Riesel works with
  # discriminnant D if the there are integers
  # a,b,r such that (a+b*sqrt(D))^2/r is a unit
  # such that r=|a^2-b^2*D|, (D|N)=-1,
  # (r|D)(a^2-b^2*D)/r=-1. We are here interested
```

```
# for the choose D=13, a=3, b=1, r=4.
# The test is then applicable for those
# N, for which (13|N)=(N|13)=-1 and
# (13|2*N+1)=(2*N+1|13)=-1. This gives
# the condition h*7^e mod 13 is in {3,6,8}.
# In these cases N+2 mod 13 also not 0.
# We plan to use the modulus 30030=2*3*5*7*11*13.
# Because N, N+2 and 2*N+1 are certainly
# not divisible by 2, 3, 5, 7 and 11 if
# h divisible by 3, 5, 7 and 11, independently
# of the choose of n, we obtain a lot of cases
# in which the discriminant D=13 may be used,
# as follows:
#
# e mod 12 =0  : h mod 30030 = 5775,26565,10395
# e mod 12 =1  : h mod 30030 = 10395,5775,12705
# e mod 12 =2  : h mod 30030 = 12705,10395,28875
# e mod 12 =3  : h mod 30030 = 28875,12705,21945
# e mod 12 =4  : h mod 30030 = 21945,28875,3465
# e mod 12 =5  : h mod 30030 = 3465,21945,24255
# e mod 12 =6  : h mod 30030 = 24255,3465,19635
# e mod 12 =7  : h mod 30030 = 19635,24255,17325
# e mod 12 =8  : h mod 30030 = 17325,19635,1155
# e mod 12 =9  : h mod 30030 = 1155,17325,8085
# e mod 12 =10 : h mod 30030 = 8085,1155,26565
# e mod 12 =11 : h mod 30030 = 265775,8085,5775
#
#
# These where calculated by the following
# simple program:
#

calcsophietwinclasses:=proc() local n,L; L:=[];
for n from 0 to 11 do
  [n,chrem([1,0,0,0,0,3*7^n mod 13],[2,3,5,7,11,13]),
  chrem([1,0,0,0,0,6*7^n mod 13],[2,3,5,7,11,13]),
  chrem([1,0,0,0,0,8*7^n mod 13],[2,3,5,7,11,13])];
  L:=[op(L),%];
od; L end;
```

$$calcsophietwinclasses := \mathbf{proc}()\ \mathbf{local}\ n, L;\ L := [\,];\ \mathbf{for}\ n\ \mathbf{from}\ 0\ \mathbf{to}\ 11\ \mathbf{do}\ [n, chrem([1, 0, 0, 0, 0, mod(3*7\hat{}n, 13)], [2, 3, 5, 7, 11, 13]), chrem([1, 0, 0, 0, 0, mod(6*7\hat{}n, 13)], [2, 3, 5, 7, 11, 13]), \tag{5.14.1}$$

```
        chrem([1, 0, 0, 0, 0, mod(8*7^n, 13)], [2, 3, 5, 7, 11, 13])];
        L := [op(L), `%`]
    end do;
    L
end proc
```

```
> #
  # More generally, such calculations can be done
  # by the following program, which find all congruence classes
  # for h for a given discriminant D=5,13,17,29,53 and primes
  from
  # the list P, for which h*2^(n+e)+d<>0 mod p and all cases
  # where d=-1 are appropriate for the Riesel test by D.
  # List L contains the pairs [e,d].
  #

  classes:=proc(D::posint,n::integer,P::list(posint),L::list)
  local l,i,ll,r,p,LL,q,x;
  ll:=ilcm(phi(D),op(map(i->i-1,P)));
  r:=[i$i=1..D];
  for l in L do
    if l[2]=-1 then
      r:=remove(x->jacobi(D,x*2^(n+l[1])+l[2])<>-1,r);
    else
      r:=select(x->jacobi(D,x*2^(n+l[1])+l[2])<>0,r);
    fi;
  od; LL:=[[D,r]];
  for p in P do
    r:=[i$i=0..p-1];
    for l in L do
      r:=select(x->modp(x*2^(n+l[1])+l[2],p)<>0,r)
    od; LL:=[op(LL),[p,r]];
  od; ll,LL end;
```

$$classes := \mathbf{proc}(\mathrm{D}::posint,\ n::integer,\ P::(list(posint)),\ L::list) \qquad (5.14.2)$$

```
    local l, i, ll, r,
    p, LL, q, x;
    ll := ilcm(numtheory:-phi(D), op(map(proc(i)
        option operator, arrow;
        i - 1
    end proc, P)));
    r := [ `$`(i, i = 1 ..D)];
    for l in L do
```

```
        if l[2] = -1 then
            r := remove(proc(x)
                 option operator,
                arrow;
                 numtheory:-jacobi(D,
                x*2^(n+l[1])+l[2]) <> -1
            end proc, r)
        else
            r := select(proc(x)
                 option operator, arrow;
                 numtheory:-jacobi(D, x*2^(n+l[1])+l[2]) <> 0
            end proc, r)
        end if
    end do;
    LL := [[D, r]];
    for p in P do
        r := [ `$`(i, i = 0..p-1)];
        for l in L do
            r := select(proc(x)
                 option operator, arrow;
                 modp(x*2^(n+l[1])+l[2], p) <> 0
            end proc, r)
        end do;
        LL := [op(LL), [p, r]]
    end do;
    ll, LL
end proc
```

```
> classes(5,60000,[2,3,7,11,13],[[0,-1],[1,-1]]);
```

$$60, [[5, [4]], [2, [0, 1]], [3, [0]], [7, [0, 2, 3, 5, 6]], [11, [0, 2, 3, 4, 5, 7, 8, 9, 10]], [13, [0, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12]]] \quad (5.14.3)$$

```
> chrem([1,0,4],[2,3,5]);
```

$$9 \quad (5.14.4)$$

```
> classes(13,1983*128,[2,3,5,7,11],[[0,-1],[1,-1],[0,+1]]);
```

60, [[13, [3, 6, 8]], [2, [0, 1]], [3, [0]], [5, [0, 2]], [7, [0, 2, 3, 5]], [11, [0, 1, 3, 4, 5, 6, 7, 8]]] (5.14.5)

```
> chrem([1,0,0,0,0,3],[2,3,5,7,11,13]);
```

5775 (5.14.6)

```
> classes(13,1983*128,[2,3,5,7,11],[[0,-1],[1,+1],[0,+1]]);
```

60, [[13, [3, 7, 8, 9]], [2, [0, 1]], [3, [0]], [5, [0, 3]], [7, [0, 2, 4, 5]], [11, [0, 3, 4, 5, 6, 7, 8, 10]]] (5.14.7)

```
> chrem([1,0,0,0,0,9],[2,3,5,7,11,13]);
```

17325 (5.14.8)

```
> classes(13,21*2^14-128,[2,3,5,7,11],[[0,-1],[1,-1],[0,+1]]);
```

60, [[13, [1, 2, 7]], [2, [0, 1]], [3, [0]], [5, [0, 2]], [7, [0, 1, 5, 6]], [11, [0, 1, 2, 3, 4, 7, 9, 10]]] (5.14.9)

```
> chrem([0,0,0,0,0,1],[2,3,5,7,11,13]);
```

6930 (5.14.10)

```
> classes(13,247*128,[2,3,5,7,11,13],[[0,-1],[1,-1],[2,-1]]);
```

60, [[13, [9, 11]], [2, [0, 1]], [3, [0]], [5, [0, 2]], [7, [0, 3, 5, 6]], [11, [0, 1, 2, 3, 6, 7, 9, 10]], [13, [0, 1, 2, 5, 6, 7, 9, 10, 11, 12]]] (5.14.11)

```
> classes(13,273*128,[2,3,5,7,11,13],[[0,-1],[1,-1],[2,-1]]);
```

60, [[13, [3, 8]], [2, [0, 1]], [3, [0]], [5, [0, 2]], [7, [0, 3, 5, 6]], [11, [0, 1, 2, 3, 4, 6, 7, 8]], [13, [0, 2, 3, 4, 5, 6, 8, 9, 11, 12]]] (5.14.12)

```
> 273*128;
```

34944 (5.14.13)

```
> chrem([1,0,0,0,0,3],[2,3,5,7,11,13]);
```

5775 (5.14.14)

```
> ee:=273; log[10.](2.)*128*ee; (ee/247)^4.;
```

$ee := 273$

10519.19217

1.492322803 (5.14.15)

```
> a:=expand((3+1*sqrt(13))^2/4);
```

$$a := \frac{11}{2} + \frac{3}{2}\sqrt{13} \quad (5.14.16)$$

```
> riesel(5110664609396115,34944,11/2,3/2,13);
```

true (5.14.17)

```
> riesel(5110664609396115,34945,11/2,3/2,13);
```

true (5.14.18)

```
> riesel(5110664609396115,34946,11/2,3/2,13);
```

true (5.14.19)

```
> 2; log[2.](%); 2*3; log[2.](%); 2*3*5;log[2.](%); 2*3*5*7;
  log[2.](%);
  2*3*5*7*11; log[2.](%); 2*3*5*7*11*13; log[2.](%);
  2*3*5*7*11*13*17; log[2.](%); 2*3*5*7*11*13*17*19; log[2.](%)
  ;
  2*3*5*7*11*13*17*19*23; log[2.](%); 2*3*5*7*11*13*17*19*23*
  29; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37*41; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37*41*47; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37*41*47*53; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37*41*47*53*59; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37*41*47*53*59*61; log[2.](%);
  2*3*5*7*11*13*17*19*23*29*31*37*41*47*53*59*61*67; log[2.](%)
  ;
```

2

1.

6

2.584962501

30

4.906890596

210

7.714245518

2310

11.17367714

30030

14.87411685

510510

18.96157970

9699690

23.20950721

223092870

27.73306917

6469693230

32.59105016

200560490130

37.54524647

7420738134810

42.75469984

304250263527210

48.11225184

14299762385778870

53.66684069

757887406446280110

59.39476115

44715356980330526490

65.27740420

2727636775800162115890

71.20814153

182751663978610861764630

77.27423072 (5.14.20)

```
> chrem([1,3],[2,13]);
```

3 (5.14.21)

```
> classes(13,61*128,[2,3,5,7,11,17],[[0,-1],[1,-1],[2,-1],[3,
  -1]]);
```

240, [[13, [11]], [2, [0, 1]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 3, 5, 7, 8, 9, 10]], [17, [0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 16]]] (5.14.22)

```
> chrem([1,0,0,11],[2,3,5,13]);
```

375 (5.14.23)

```
> classes(13,30*128,[2,3,5,7,11,17],[[0,-1],[1,-1],[2,-1],[3,
  -1],[4,-1]]);
```

240, [[13, []], [2, [0, 1]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 10]], [17, [0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14]]] (5.14.24)

```
> classes(13,30*128,[2,3,5,7,11,17],[[0,-1],[1,-1],[2,-1],[3,
  -1]]);
```

240, [[13, [8]], [2, [0, 1]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 9, 10]], [17, [0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 16]]] (5.14.25)

```
> classes(17,30*128,[2,3,5,7,11,13],[[4,-1]]);
```

240, [[17, [2, 4, 5, 6, 9, 10, 11, 13]], [2, [0, 1]], [3, [0, 2]], [5, [0, 2, 3, 4]], [7, [0, 1, 2, 3, 5, 6]], [11, [0, 1, 2, 3, 4, 5, 6, 7, 8, 10]], [13, [0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12]]] (5.14.26)

```
> chrem([1,0,0,8],[2,3,5,13]);
```

$$255 \tag{5.14.27}$$

```
> classes(13,29*64,[2,3,5,7,11,17,19,23,29,31],[[0,-1],[1,-1],
  [2,-1],[3,-1]]); cl13:=%[2];
```

55440, [[13, [11]], [2, [0, 1]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 1, 3, 6, 7, 9, 10]], [17, [0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 16]], [19, [0, 1, 2, 4, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18]], [23, [0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]], [29, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 25, 27, 28]], [31, [0, 1, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]]]

$$cl13 := [[13, [11]], [2, [0, 1]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 1, 3, 6, 7, 9, 10]], [17, [0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 14, 16]], [19, [0, 1, 2, 4, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18]], [23, [0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]], [29, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 25, 27, 28]], [31, [0, 1, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]]] \tag{5.14.28}$$

```
> classes(17,29*64,[2,3,5,7,11,13,19,23,29,31],[[4,-1],[5,-1]])
  ; cl17:=%[2];
```

55440, [[17, [2, 5, 11, 13]], [2, [0, 1]], [3, [0]], [5, [0, 2, 4]], [7, [0, 2, 3, 5, 6]], [11, [0, 2, 3, 4, 5, 7, 8, 9, 10]], [13, [0, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12]], [19, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18]], [23, [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]], [29, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 26, 28]], [31, [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]]]

$$cl17 := [[17, [2, 5, 11, 13]], [2, [0, 1]], [3, [0]], [5, [0, 2, 4]], [7, [0, 2, 3, 5, 6]], [11, [0, 2, 3, 4, 5, 7, 8, 9, 10]], [13, [0, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12]], [19, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18]], [23, [0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]], [29, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 26, 28]], [31, [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]]] \tag{5.14.29}$$

29, 30]]]

```
> {op(cl13[6][2])}; {op(cl17[6][2])}; % intersect %%; nops(%);
```

$$\{0, 1, 3, 6, 7, 9, 10\}$$

$$\{0, 2, 3, 4, 5, 7, 8, 9, 10\}$$

$$\{0, 3, 7, 9, 10\}$$

$$5 \tag{5.14.30}$$

```
> {op(cl13[8][2])}; {op(cl17[8][2])}; % intersect %%; nops(%);
```

$$\{0, 1, 2, 4, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18\}$$

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 17, 18\}$$

$$\{0, 1, 2, 4, 7, 8, 9, 10, 13, 14, 16, 17, 18\}$$

$$13 \tag{5.14.31}$$

```
> {op(cl13[9][2])}; {op(cl17[9][2])}; % intersect %%; nops(%);
```

$$\{0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22\}$$

$$\{0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22\}$$

$$\{0, 3, 5, 7, 9, 10, 11, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22\}$$

$$17 \tag{5.14.32}$$

```
> {op(cl13[10][2])}; {op(cl17[10][2])}; % intersect %%; nops(%)
  ;
```

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 25, 27, 28\}$$

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 28\}$$

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 28\}$$

$$23 \tag{5.14.33}$$

```
> {op(cl13[11][2])}; {op(cl17[11][2])}; % intersect %%; nops(%)
  ;
```

$$\{0, 1, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30\}$$

$$\{0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30\}$$

$$\{0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30\}$$

$$26 \tag{5.14.34}$$

```
> classes(17,1983*128,[2,3,5,7,11,13],[[4,-1],[5,-1],[6,-1]]);
```

240, [[17, [11]], [2, [0, 1]], [3, [0]], [5, [0, 2]], [7, [0, 3, 5, 6]], [11, [0, 3, 5, 6, 7, 8, 9, 10]], [13, [0, 1, 2, 3, 4, 5, 6, 7, 8, 10]]] (5.14.35)

```
> classes(5,1983*128,[2,3,7,11,13],[[0,-1],[0,+1]]);
```

60, [[5, [3]], [2, [0, 1]], [3, [0]], [7, [0, 2, 3, 4, 5]], [11, [0, 1, 3, 4, 5, 6, 7, 8, 10]], [13, [0, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]]] (5.14.36)

```
> classes(5,1983*128,[2,3,7,11,13],[[0,-1],[0,+1],[1,+1]]);
```

60, [[5, [3]], [2, [0, 1]], [3, [0]], [7, [0, 2, 4, 5]], [11, [0, 3, 4, 5, 6, 7, 8, 10]], [13, [0, 2, 3, 4, 5, 7, 8, 9, 10, 11]]] (5.14.37)

```
> chrem([1,0,3,0,0,0],[2,3,5,7,11,13]);
```

$$3003 \tag{5.14.38}$$

```
> classes(5,247*128,[2,3,7,11,13],[[0,+1],[1,+1],[2,+1]]);
```

60, [[5, [3, 5]], [2, [0, 1]], [3, [0]], [7, [0, 1, 2, 4]], [11, [0, 1, 2, 4, 5, 8, 9, 10]], [13, [0, 1, 2, 3, 4, 6, 7, 8, 11, 12]]] (5.14.39)

```
> classes(5,61*128,[2,3,7,11,13,17],[[0,+1],[1,+1],[2,+1],[3,+1]]);
```

240, [[5, [5]], [2, [0, 1]], [3, [0]], [7, [0, 1, 2, 4]], [11, [0, 1, 2, 3, 4, 6, 8]], [13, [0, 1, 2, 3, 4, 6, 7, 8, 12]], [17, [0, 1, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15]]] (5.14.40)

```
> chrem([1,0,0],[2,3,5]);
```

$$15 \tag{5.14.41}$$

```
> classes(5,30*128,[2,3,7,11,13,17,19],[[0,+1],[1,+1],[2,+1],[3,+1],[4,+1]]);
```

720, [[5, [5]], [2, [0, 1]], [3, [0]], [7, [0, 1, 2, 4]], [11, [0, 1, 3, 6, 7, 9]], [13, [0, 1, 2, 5, 7, 9, 10, 11]], [17, [0, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15]], [19, [0, 3, 5, 6, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18]]] (5.14.42)

```
> classes(5,29*64,[2,3,7,11,13,17,19,23,29,31],[[0,+1],[1,+1],[2,+1],[3,+1],[4,+1],[5,+1]]);
```

55440, [[5, [5]], [2, [0, 1]], [3, [0]], [7, [0, 1, 2, 4]], [11, [0, 1, 2, 4, 8]], [13, [0, 1, 2, 3, 4, 7, 8]], [17, [0, 3, 5, 6, 7, 10, 11, 12, 13, 14, 15]], [19, [0, 1, 2, 3, 5, 6, 9, 10, 11, 12, 15, 17, 18]], [23, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 16, 18, 20]], [29, [0, 1, 5, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]], [31, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 28]]] (5.14.43)

```
> classes(17,0,[2,3,5,7,11],[[0,-1]]);
```

240, [[17, [4, 6, 7, 8, 11, 12, 13, 15]], [2, [0]], [3, [0, 2]], [5, [0, 2, 3, 4]], [7, [0, 2, 3, 4, 5, 6]], [11, [0, 2, 3, 4, 5, 6, 7, 8, 9, 10]]] (5.14.44)

```
> classes(17,0,[2,3,5,7,11],[[0,-1],[1,-1]]);
```
240, [[17, [4, 6, 12, 15]], [2, [0]], [3, [0]], [5, [0, 2, 4]], [7, [0, 2, 3, 5, 6]], [11, [0, 2, 3, 4, 5, 7, 8, 9, 10]]] (5.14.45)

```
> classes(17,0,[2,3,5,7,11],[[0,-1],[1,-1],[2,-1]]);
```
240, [[17, [6]], [2, [0]], [3, [0]], [5, [0, 2]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 7, 8, 9, 10]]] (5.14.46)

```
> classes(17,0,[2,3,5,7,11],[[0,-1],[1,-1],[2,-1],[3,-1]]);
```
240, [[17, []], [2, [0]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 9, 10]]] (5.14.47)

```
> classes(29,0,[2,3,5,7,11],[[0,-1],[1,-1],[2,-1],[3,-1]]);
```
420, [[29, [20]], [2, [0]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 9, 10]]] (5.14.48)

```
> classes(29,0,[2,3,5,7,11],[[0,-1],[1,-1],[2,-1],[3,-1],[4,-1]
  ]);
```
420, [[29, []], [2, [0]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 10]]] (5.14.49)

```
> classes(53,0,[2,3,5,7,11],[[0,-1],[1,-1],[2,-1],[3,-1]]);
```
780, [[53, [21]], [2, [0]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 9, 10]]] (5.14.50)

```
> classes(53,0,[2,3,5,7,11],[[0,-1],[1,-1],[2,-1],[3,-1],[4,-1]
  ]);
```
780, [[53, []], [2, [0]], [3, [0]], [5, [0]], [7, [0, 3, 5, 6]], [11, [0, 2, 4, 5, 8, 10]]] (5.14.51)

5.15. n^2+1 es n^4+1 alakú prímek.

```
> #
  # A pretest will be done for prime candidates having the
  # form n^2+1, to be more exact, for (h0+c*h)^2*b^(2*n)+1,
  # where h runs from 0 to H-1. All prime divisors from P0
  # to P1 will be tried. P0 must be odd, and b,c must
  # have prime factors smaller then P0. The values h, for
  # which (h0+c*h)^2*b^(2*n)+1 is not divisible with primes
  # from P0 to P1 are given back.
  #

  presquareplus:=proc(h0,H,c,b,n,P0,P1)
```

```
  local T,h,p,hh,L,nn,bb,bbb,cc,bT,cT,i,j,R,r;
  T:=array(0..H-1);
  for h from 0 to H-1 do T[h]:=1 od;
  bT:=array(0..b-1);                    # table of inverses for
  b
  for i from 0 to b-1 do bT[i]:=0 od;
  for i from 0 to b-1 do
    for j from 0 to b-1 do
      if irem(i*j+1,b)=0 then bT[i]:=j fi
    od
  od;
  cT:=array(0..c-1);                    # table of inverses for
  c
  for i from 0 to c-1 do cT[i]:=0 od;
  for i from 0 to c-1 do
    for j from 0 to c-1 do
      if irem(i*j+1,c)=0 then cT[i]:=j fi
    od
  od;
  for p from P0 to P1 by 2 do
    if isprime(p) then
      r:=msqrt(p-1,p);
      if r<>FAIL then
        R:=[r,p-r];
        cc:=(cT[p mod c]*p+1)/c;
        nn:=n mod (p-1);
        bb:=(bT[p mod b]*p+1)/b;
        bbb:=bb&^nn mod p;
        for r in R do
          hh:=(r*bbb-h0)*cc mod p;
          for h from hh to H-1 by p do
            T[h]:=0;
          od;
        od;
      fi;
    fi;
  od;
  L:=[];
  for h from 0 to H-1 do
    if T[h]=1 then L:=[op(L),h0+c*h] fi
  od; L end;
```

$$presquareplus := \mathbf{proc}(h0, H, c, b, n, P0, P1) \tag{5.15.1}$$

$\mathbf{local}\ T, h, p, hh, L, nn, bb, bbb, cc, bT, cT, i, j, R, r;$

$T := array(0..H-1);$

$\mathbf{for}\ h\ \mathbf{from}\ 0\ \mathbf{to}\ H-1\ \mathbf{do}$

$T[h] := 1$

```
    end do;
    bT := array(0..b - 1);
    for i from 0 to b - 1 do
        bT[i] := 0
    end do;
    for i from 0 to b - 1 do
        for j from 0 to b - 1 do
            if irem(i*j + 1, b) = 0 then
                bT[i] := j
            end if
        end do
    end do;
    cT := array(0..c - 1);
    for i from 0 to c - 1 do
        cT[i] := 0
    end do;
    for i from 0 to c - 1 do
        for j from 0 to c - 1 do
            if irem(i*j + 1, c) = 0 then
                cT[i] := j
            end if
        end do
    end do;
    for p from P0 by 2 to P1 do
        if isprime(p) then
            r := numtheory:-msqrt(p - 1, p);
            if r <> FAIL then
                R := [r, p - r];
                cc := (cT[mod(p, c)]*p + 1) / c;
                nn := mod(n, p - 1);
                bb := (bT[mod(p, b)]*p + 1) / b;
                bbb := mod(bb &^ nn, p);
```

```
                for r in R do
                    hh := mod((r*bbb - h0)*cc, p);
                    for h from hh by p to H - 1 do
                        T[h] := 0
                    end do
                end do
            end if
        end if
    end do;
    L := [];
    for h from 0 to H - 1 do
        if T[h] = 1 then
            L := [op(L), h0 + c*h]
        end if
    end do;
    L
end proc
```

```
> #
  # This filter left only those h's from the list L in the
  # resulting list for which (h0+c*h)^2*2^(2*n)+1 is
  # probably prime.
  #

  squareplusprob:=proc(h0,c,n,L) local LL,h,N;
  LL:=[];
  for h in L do
    N:=(h0+c*h)^2*2^(2*n)+1;
    if modp(2&^(N-1),N)=1 then LL:=[op(LL),h] fi
  od; LL end;
```

$$squareplusprob := \mathbf{proc}(h0, c, n, L) \qquad (5.15.2)$$

```
    local LL, h, N;
    LL := [];
    for h in L do
        N := (h0 + c*h)^2*2^(2*n) + 1;
        if modp(2 &^ (N - 1), N) = 1 then
            LL := [op(LL), h]
```

```
        end if
    end do;
    LL
end proc
```

```
> #
  # This filter left only those h's from the list L in the
  # resulting list for which (h0+c*h)^2*2^(2*n)+1 is
  # prime.
  #

  squareplusproth:=proc(h0,c,n,L) local LL,h,N;
  LL:=[];
  for h in L do
    N:=(h0+c*h)^2*2^(2*n)+1;
    if proth((h0+c*h)^2,2*n) then LL:=[op(LL),h] fi
  od; LL end;
```

$$squareplusproth := \mathbf{proc}(h0, c, n, L) \tag{5.15.3}$$

```
    local LL, h, N;
    LL:= [ ];
    for h in L do
        N:= (h0+ c* h)^2 *2^(2 *n) + 1;
        if proth( (h0+ c* h)^2, 2 *n) then
            LL:= [op(LL), h]
        end if
    end do;
    LL
end proc
```

```
> #
  # This is a test to measure densities and times.
  # First pretests are done for N where
  # N=(h0+2*h)^2*2^(2*n)+1 and h runs from 0 to H-1.
  # The pretest is repeated for prime limit P=2^e where
  # e is integer from the interval eint. The time
  # and the number of remaining candidates after the pretest
  # is reported, and p*ln(P) also calculated, where p is the
  # relative frequency of numbers passed the pretest. The
  # result of the last pretest became the input of the
  # probabilistic test for N. The time and the result
  # of this test is also reported.
  #
```

```
squareplustest:=proc(h0,H,n,eint) local e,p,P,L,t,N;
print(`Squareplustest; h0=`,h0,`H=`,H,`n=`,n);
N:=(h0+H)^2*2^(2*n);
print(`Pretest for N`);
for e from op(1,eint) to op(2,eint) do
  P:=2^e;
  t:=time();
  L:=presquareplus(h0,H,2,2,n,3,P);
  t:=time()-t;
  p:=evalf(nops(L)/H);
  print(`P=`,2^e,`p=`,p,`p*ln(P)=`,evalf((ln(P))*p),`time=`,
t)
od;
print(`Probabilistic test for N`);
t:=time();
L:=squareplusprob(h0,2,n,L);
t:=time()-t;
p:=evalf(nops(L)/H);
print(`p=`,p,`p*ln(N)=`,evalf((ln(N))*p),`time=`,t);
print(`Proth test for N`);
t:=time();
L:=squareplusproth(h0,2,n,L);
t:=time()-t;
p:=evalf(nops(L)/H);
print(`p=`,p,`p*ln(N)=`,evalf((ln(N))*p),`time=`,t);
L end;
```

$$squareplustest := \mathbf{proc}(h0, H, n, eint) \tag{5.15.4}$$

$$\mathbf{local}\ e, p, P, L, t, N;$$

$$print(Squareplustest;\ h0=, h0, H=, H, n=, n);$$

$$N := (h0 + H)\hat{}2 * 2\hat{}(2 * n);$$

$$print(Pretest\ for\ N);$$

$$\mathbf{for}\ e\ \mathbf{from}\ op(1, eint)\ \mathbf{to}\ op(2, eint)\ \mathbf{do}$$

$$P := 2\hat{}e;$$

$$t := time();$$

$$L := presquareplus(h0, H, 2, 2, n, 3, P);$$

$$t := time() - t;$$

$$p := evalf(nops(L) / H);$$

$$print(P=, 2\hat{}e, p=, p, p*ln(P)=, evalf(\ln(P) * p), time=, t)$$

$$\mathbf{end\ do};$$

$$print(Probabilistic\ test\ for\ N);$$

$$t := time();$$

```
    L := squareplusprob(h0, 2, n, L);
    t := time() - t;
    p := evalf(nops(L) / H);
    print(p=, p, p*ln(N)=, evalf(ln(N) * p), time=, t);
    print(Proth test for N);
    t := time();
    L := squareplusproth(h0, 2, n, L);
    t := time() - t;
    p := evalf(nops(L) / H);
    print(p=, p, p*ln(N)=, evalf(ln(N) * p), time=, t);
    L
end proc
```

```
> squareplustest(1,20000,1000,2..15);
```

```
                Squareplustest; h0=, 1, H=, 20000, n=, 1000

                              Pretest for N

              P=, 4, p=, 1., p*ln(P)=, 1.386294361, time=, 9.873

           P=, 8, p=, 0.6000000000, p*ln(P)=, 1.247664925, time=, 3.789

          P=, 16, p=, 0.5076500000, p*ln(P)=, 1.407504665, time=, 2.739

          P=, 32, p=, 0.4171000000, p*ln(P)=, 1.445558445, time=, 1.935

          P=, 64, p=, 0.3490500000, p*ln(P)=, 1.451658140, time=, 1.442

         P=, 128, p=, 0.3069500000, p*ln(P)=, 1.489330690, time=, 1.180

         P=, 256, p=, 0.2757500000, p*ln(P)=, 1.529082680, time=, 1.018

         P=, 512, p=, 0.2450000000, p*ln(P)=, 1.528389533, time=, 0.871

        P=, 1024, p=, 0.2213500000, p*ln(P)=, 1.534281284, time=, 0.814

        P=, 2048, p=, 0.2011500000, p*ln(P)=, 1.533692109, time=, 0.765

        P=, 4096, p=, 0.1856000000, p*ln(P)=, 1.543777401, time=, 0.765

        P=, 8192, p=, 0.1710000000, p*ln(P)=, 1.540866182, time=, 0.822

       P=, 16384, p=, 0.1595500000, p*ln(P)=, 1.548282857, time=, 0.984

       P=, 32768, p=, 0.1484500000, p*ln(P)=, 1.543465484, time=, 1.385

                          Probabilistic test for N

          p=, 0.0004000000000, p*ln(N)=, 0.5624405745, time=, 108.150

                              Proth test for N

          p=, 0.0004000000000, p*ln(N)=, 0.5624405745, time=, 0.297
```

$$[15509, 15699, 23649, 31899, 33795, 36009, 38189, 38199] \tag{5.15.5}$$

```
> #
  # Estimate of the powplusone density Cpowplusone for
  # polynomial (X*2^n)^(2^e)+1 (n fix) by calculating the
  product for
  # primes below x.
  #

  Cpowplusone:=proc(e::posint,x::posint) local P,p;
  P:=2.; p:=nextprime(2);
  while p<x do
    if modp(p,2^(e+1))=1 then
      P:=P*(1-2^e/p)/(1-1/p)
    else
      P:=P/(1-1/p)
    fi;
    p:=nextprime(p)
  od; P end;
```

$$Cpowplusone := \mathbf{proc}(e::posint, x::posint) \tag{5.15.6}$$

$$\begin{aligned}
&\mathbf{local}\ P, p;\\
&P := 2.;\\
&p := nextprime(2);\\
&\mathbf{while}\ p < x\ \mathbf{do}\\
&\quad \mathbf{if}\ modp(p, 2\wedge(e+1)) = 1\ \mathbf{then}\\
&\qquad P := P * (1 - 2\wedge e / p) / (1 - 1 / p)\\
&\quad \mathbf{else}\\
&\qquad P := P / (1 - 1 / p)\\
&\quad \mathbf{end\ if};\\
&\quad p := nextprime(p)\\
&\mathbf{end\ do};\\
&P\\
&\mathbf{end\ proc}
\end{aligned}$$

```
> Cpowplusone(1,10); Cpowplusone(1,100); Cpowplusone(1,1000);
  Cpowplusone(1,10000); Cpowplusone(1,100000); Cpowplusone(1,
  1000000);
```

$$2.625000000$$

$$2.703091852$$

$$2.740907689$$

$$2.742045031$$

2.744701015

2.745621017 (5.15.7)

```
> Cpowplusone(2,10); Cpowplusone(2,100); Cpowplusone(2,1000);
  Cpowplusone(2,10000); Cpowplusone(2,100000); Cpowplusone(2,
  1000000);
```

4.375000000

4.964215420

5.307605624

5.345408691

5.351678486

5.357012604 (5.15.8)

```
> #
  # This procedure calculate the factor qeAB.
  #

  qeAB:=proc(e::posint,A::posint,B::posint) local P,p;
  P:=1.; p:=nextprime(A-1);
  while p<B do
    if modp(p,2^(e+1))=1 then P:=P*(1-2^e/p) fi;
    p:=nextprime(p)
  od; P end;
```

$$qeAB := \mathbf{proc}(e::posint, A::posint, B::posint) \quad (5.15.9)$$

```
local P, p;
P:= 1.;
p:= nextprime(A - 1);
while p < B do
    if modp(p, 2^(e + 1)) = 1 then
        P:= P* (1 - 2^e / p)
    end if;
    p:= nextprime(p)
end do;
P
end proc
```

```
> # square plus one prime
> f:=h->((2*h+1)*2.^(340000.*4))^2+1;
  g:=h->1/ln(f(h));
```

$$f := h \rightarrow (2\,h + 1)^2 \left(2.^{3.40000\,10^5\,4}\right)^2 + 1$$

$$g := h \to \frac{1}{\ln(f(h))} \tag{5.15.10}$$

```
> H:=2.^21; evalf(H/6*(g(0)+4*g(H/2)+g(H)));
```

$$H := 2.097152\,10^{6}$$

$$1.112320400 \tag{5.15.11}$$

```
> %*2.745621017; # expected primes
```

$$3.054010268 \tag{5.15.12}$$

```
> qeAB(1,3,1000000);
```

$$0.1115771215 \tag{5.15.13}$$

```
> %*(ln(1000000.)/(40*ln(2.)));
```

$$0.05559767622 \tag{5.15.14}$$

```
> %*H*160*20/3.054010268; # SPU time in sec
```

$$1.221704109\,10^{8} \tag{5.15.15}$$

```
> %/24/3600/16; # time for one Cell blade in days
```

$$88.37558659 \tag{5.15.16}$$

```
> # quad plus one prime
> f:=h->((2*h+1)*2.^(340000.*2))^4+1;
  g:=h->1/ln(f(h));
```

$$f := h \to (2\,h+1)^4 \left(2.^{3.40000\,10^5\,2}\right)^4 + 1$$

$$g := h \to \frac{1}{\ln(f(h))} \tag{5.15.17}$$

```
> H:=2.^15; evalf(H/6*(g(0)+4*g(H/2)+g(H)));
```

$$H := 32768.$$

$$0.01737990828 \tag{5.15.18}$$

```
> %*5.357012604; # expected primes
```

$$0.09310438771 \tag{5.15.19}$$

```
> qeAB(2,3,1000000);
```

$$0.2176994122 \tag{5.15.20}$$

```
> %*(ln(1000000.)/(40*ln(2.)));
```

$$0.1084772690 \tag{5.15.21}$$

```
> %*H*160*20/0.09310438771; # SPU time in sec
```

$$1.221711067\,10^{8} \tag{5.15.22}$$

```
> %/24/3600/16; # time for one Cell blade in days
```

$$88.37608992 \tag{5.15.23}$$

5.16. Egyéb speciális alakú prímek.

```
> #
  # Cullen and Woodall numbers: n*2^n+-1; all known below n<=
  8000000.
  #

  x:=20000000; evalf(int(y->2/y/ln(2.)/ln(ln(y)),8000000..x));
```

$$x := 20000000$$

$$0.94619675 + 0.\mathrm{I} \tag{5.16.1}$$

```
> #
  # This procedure calculate the factor qsAB.
  #

  qsAB:=proc(s::posint,A::posint,B::posint) local P,p;
  P:=1.; p:=nextprime(A-1);
  while p<B do P:=P*(1-s/p); p:=nextprime(p) od;
  P end;
```

$$qsAB := \mathbf{proc}(s::posint, A::posint, B::posint) \tag{5.16.2}$$

$$\mathbf{local}\ P, p;$$

$$P := 1.;$$

$$p := nextprime(A - 1);$$

$$\mathbf{while}\ p < B\ \mathbf{do}$$

$$P := P * (1 - s / p);$$

$$p := nextprime(p)$$

$$\mathbf{end\ do};$$

$$P$$

$$\mathbf{end\ proc}$$

```
> qsAB(1,3,1000000);
```

$$0.08127641616 \tag{5.16.3}$$

```
> %*(ln(1000000.)/(50*ln(2.))); # decreasing factor of sieve
```

$$0.03239932923 \tag{5.16.4}$$

```
> %*12000000*160*16*20; # SPU time in sec
```

$$1.990614788\ 10^{10} \tag{5.16.5}$$

```
> %/24/3600/16; # time for one Cell blade in days
```

$$14399.70188 \tag{5.16.6}$$

```
> #
  # Primorial numbers: p#+-1; p#><exp(p); record (02.2011):
  843301#-1.
  # Expected number of such primes for +1 and -1, respectively,
```

```
# up to x is exp(gamma)*ln(x), hence the next expected by
#

x:=evalf(exp(exp(-gamma)+ln(843301.))); # around this p

log[10.](exp(x)); # so many decimal digits

x/ln(x)-843301/ln(843301.); # so many to test, without sieve
```

$$x := 1.478500136\ 10^{6}$$

$$6.421044506\ 10^{5}$$

$$42269.23308 \tag{5.16.7}$$

```
> %*300*128*3; # SPU time in sec
```

$$4.869415651\ 10^{9} \tag{5.16.8}$$

```
> %/24/3600/420; # time for all PS3 in days
```

$$134.1880415 \tag{5.16.9}$$

```
> #
  # Factorial numbers: n!+-1; n!><n^n/exp(n); record (02.2011):
  103040!-1.
  # Expected number of such primes for +1 and -1, respectively,
  # up to x is exp(gamma)*ln(x), hence the next expected by
  #

  x:=evalf(exp(exp(-gamma)+ln(102030.))); # around this n

  log[10.](x^x/exp(x)); # so many decimal digits

  x-103040; # so many to test, without sieve
```

$$x := 1.788820001\ 10^{5}$$

$$8.619021610\ 10^{5}$$

$$75842.0001 \tag{5.16.10}$$

```
> %*300*200; # SPU time in sec
```

$$4.550520006\ 10^{9} \tag{5.16.11}$$

```
> %/24/3600/420; # time for all PS3 in days
```

$$125.4001324 \tag{5.16.12}$$

```
> #
  # Primes in arithmetic sequences, 3 primes; record (02.2011):
  # 1185*2^529445-1539*2^263359+1, d=1185*2^529444-1539*
  2^263359
  # (159382 decimal digits).
  #
```

```
evalf(2^64/3/5/7/11/13/17/19/23/29/31); # so many to sieve
out

(2/1)*(3/2)*(5/4)*(7/6)*(11/10)*(13/12)*(17/16)*(19/18)*
(23/22)*(29/28)*(31/30); evalf(%); # so many times more dense
```

$$1.839519245\ 10^{8}$$

$$\frac{86822723}{13271040}$$

$$6.542269709 \tag{5.16.13}$$

```
> qsAB(1,32,1000000);
```

$$0.2658661300 \tag{5.16.14}$$

```
> %*(ln(1000000.)/(50*ln(2.))); # decreasing factor of sieve
```

$$0.1059825800 \tag{5.16.15}$$

```
> 1/200000/ln(10.); # natural density of primes having 200000
digits
```

$$0.000002171472410 \tag{5.16.16}$$

```
> %*6.542269709/0.1059825800; 1/%; # increased density and
expected tests
```

$$0.0001340442757$$

$$7460.221593 \tag{5.16.17}$$

```
> # after 30000000 test we have 400 primes, 80000 pairs, seems
to be enough
> 3000000*160*5; # SPU time in sec
```

$$2400000000 \tag{5.16.18}$$

```
> %/24/3600/16; evalf(%); # time for one Cell blade in days
```

$$\frac{15625}{9}$$

$$1736.111111 \tag{5.16.19}$$

```
> 1/160000/ln(10.); # natural density of primes having 160000
digits
```

$$0.000002714340512 \tag{5.16.20}$$

```
> %*6.542269709/0.1059825800; 1/%; # increased density and
expected tests
```

$$0.0001675553446$$

$$5968.177275 \tag{5.16.21}$$

```
> # after 1800000 test we have 300 primes, 45000 pairs, seems
to be enough
> 1800000*300*12; # SPU time in sec
```

(5.16.22)

$$6480000000 \tag{5.16.22}$$

```
> %/24/3600/420; evalf(%); # time for all PS3 in days
```

$$\frac{1250}{7}$$

$$178.5714286 \tag{5.16.23}$$

```
> # A ten-year jump:

  1/310000/ln(10.); # natural density of primes having 310000
  digits
```

$$0.000001400949942 \tag{5.16.24}$$

```
> %*6.542269709/0.1059825800; 1/%; # increased density and
  expected tests
```

$$0.00008648017787$$

$$11563.34347 \tag{5.16.25}$$

```
> # after 5800000 test we have 500 primes, 125000 pairs, seems
  to be enough

  5800000*300*24; # SPU time in sec
```

$$41760000000 \tag{5.16.26}$$

```
> %/24/3600/420; evalf(%); # time for all PS3 in days
```

$$\frac{72500}{63}$$

$$1150.793651 \tag{5.16.27}$$

```
> repunit:=proc(n) local i;
  for i to n do if isprime((10^i-1)/9) then print(i) fi od;
  end;
```

$$repunit := \mathbf{proc}(n) \tag{5.16.28}$$

```
    local i;
    for i to n do
        if isprime(1/9*10^i-1/9) then
            print(i)
        end if
    end do
end proc
```

```
> repunit(1000);
```

$$2$$

$$19$$

$$23$$

5.17. Kátai egy problémája.

```
> #
  # This is a simple procedure to get all the prime pairs (p,q)

  # for which the quotient (p+1)/(q+1)=n is an integer,
  # 1<n<=N and 2<q<=Q. The limit Q is supposed to be small
  # and N is supposed to be large, hence primality testing is
  used.
  # The result is a list of pairs; each pair consist of n
  # and a list of all pairs [p,q] corresponding to this n.
  # The pairs are in increasing order.
  #

  prime_plus_one_quotient:=proc(Q,N)
  local q,n,p,QQ,LL,LLL; QQ:=[];
  for q from 3 by 2 while q<=Q do
    if isprime(q) then QQ:=[op(QQ),q] fi;
  od;
  LL:=[];
  for n from 2 while n<=N do
    LLL:=[];
    for q in QQ do
      p:=n*q+n-1;
      if isprime(p) then LLL:=[op(LLL),[p,q]] fi;
    od;
    LL:=[op(LL),[n,LLL]];
  od; LL end;
```

$$prime\_plus\_one\_quotient := \mathbf{proc}(Q, N) \tag{5.17.1}$$

local q, n, p, QQ, LL, LLL;

$QQ := [\,]$;

for q **from** 3 **by** 2 **while** $q <= Q$ **do**

if $isprime(q)$ **then**

$QQ := [op(QQ), q]$

end if

end do;

$LL := [\,]$;

for n **from** 2 **while** $n <= N$ **do**

$LLL := [\,]$;

for q **in** QQ **do**

```
            p := n*q + n - 1;
            if isprime(p) then
                LLL := [op(LLL), [p, q]]
            end if
        end do;
        LL := [op(LL), [n, LLL]]
    end do;
    LL
end proc
```

```
> L:=prime_plus_one_quotient(10,100);
```

$$L := [[2, [[7, 3], [11, 5]]], [3, [[11, 3], [17, 5], [23, 7]]], [4, [[23, 5], [31, 7]]], [5, [[19, 3], [29, 5]]], [6, [[23, 3], [47, 7]]], [7, [[41, 5]]], [8, [[31, 3], [47, 5]]], [9, [[53, 5], [71, 7]]], [10, [[59, 5], [79, 7]]], [11, [[43, 3]]], [12, [[47, 3], [71, 5]]], [13, [[103, 7]]], [14, [[83, 5]]], [15, [[59, 3], [89, 5]]], [16, [[127, 7]]], [17, [[67, 3], [101, 5]]], [18, [[71, 3], [107, 5]]], [19, [[113, 5], [151, 7]]], [20, [[79, 3]]], [21, [[83, 3], [167, 7]]], [22, [[131, 5]]], [23, [[137, 5]]], [24, [[191, 7]]], [25, [[149, 5], [199, 7]]], [26, [[103, 3]]], [27, [[107, 3]]], [28, [[167, 5], [223, 7]]], [29, [[173, 5]]], [30, [[179, 5], [239, 7]]], [31, []], [32, [[127, 3], [191, 5]]], [33, [[131, 3], [197, 5], [263, 7]]], [34, [[271, 7]]], [35, [[139, 3]]], [36, []], [37, []], [38, [[151, 3], [227, 5]]], [39, [[233, 5], [311, 7]]], [40, [[239, 5]]], [41, [[163, 3]]], [42, [[167, 3], [251, 5]]], [43, [[257, 5]]], [44, [[263, 5]]], [45, [[179, 3], [269, 5], [359, 7]]], [46, [[367, 7]]], [47, [[281, 5]]], [48, [[191, 3], [383, 7]]], [49, [[293, 5]]], [50, [[199, 3]]], [51, []], [52, [[311, 5]]], [53, [[211, 3], [317, 5]]], [54, [[431, 7]]], [55, [[439, 7]]], [56, [[223, 3]]], [57, [[227, 3]]], [58, [[347, 5], [463, 7]]], [59, [[353, 5]]], [60, [[239, 3], [359, 5], [479, 7]]], [61, [[487, 7]]], [62, []], [63, [[251, 3], [503, 7]]], [64, [[383, 5]]], [65, [[389, 5]]], [66, [[263, 3]]], [67, [[401, 5]]], [68, [[271, 3]]], [69, []], [70, [[419, 5]]], [71, [[283, 3]]], [72, [[431, 5]]], [73, []], [74, [[443, 5]]], [75, [[449, 5], [599, 7]]], [76, [[607,$$ (5.17.2)

```
7]]], [77, [[307, 3], [461, 5]]], [78, [[311, 3], [467, 5]]], [79, [[631,
7]]], [80, [[479, 5]]], [81, [[647, 7]]], [82, [[491, 5]]], [83, [[331,
3]]], [84, [[503, 5]]], [85, [[509, 5]]], [86, []], [87, [[347, 3], [521,
5]]], [88, []], [89, []], [90, [[359, 3], [719, 7]]], [91, [[727,
7]]], [92, [[367, 3]]], [93, [[557, 5], [743, 7]]], [94, [[563, 5], [751,
7]]], [95, [[379, 3], [569, 5]]], [96, [[383, 3]]], [97, []], [98, [[587,
5]]], [99, [[593, 5]]], [100, [[599, 5]]]]
```

```
> #
  # This routine collect statistics from the results of the
  # above routine. For all number 0<=m<=M, where M is
  # the number of odd prime less then Q, the number of those
  # numbers 1<n<=N given back, for which exactly m solutions
  # of n=(p+1)/(q+1) is found. L is the list calculated by the
  # above program.
  #

  stat_prime_plus:=proc(Q,L) local q,M,MM,LL; M:=0;
  for q from 3 by 2 while q<Q do
    if isprime(q) then M:=M+1 fi;
  od;
  MM:=array(0..M);
  for q from 0 to M do MM[q]:=0; od;
  for LL in L do MM[nops(LL[2])]:=MM[nops(LL[2])]+1; od;
  convert(MM,listlist) end;
```

$$stat\_prime\_plus := \mathbf{proc}(Q, L) \tag{5.17.3}$$

$\mathbf{local}\ q, M, MM, LL;$

$M := 0;$

$\mathbf{for}\ q\ \mathbf{from}\ 3\ \mathbf{by}\ 2\ \mathbf{while}\ q < Q\ \mathbf{do}$

$\quad \mathbf{if}\ isprime(q)\ \mathbf{then}$

$\quad\quad M := M + 1$

$\quad \mathbf{end\ if}$

$\mathbf{end\ do};$

$MM := array(0..M);$

$\mathbf{for}\ q\ \mathbf{from}\ 0\ \mathbf{to}\ M\ \mathbf{do}$

$\quad MM[q] := 0$

$\mathbf{end\ do};$

$\mathbf{for}\ LL\ \mathbf{in}\ L\ \mathbf{do}$

$\quad MM[nops(LL[2])] := MM[nops(LL[2])] + 1$

```
    end do;
    convert(MM, listlist)
end proc
```

```
> stat_prime_plus(10,L);
```

$$[11, 52, 32, 4] \tag{5.17.4}$$

▶ 5.18. Példa.

▼ 5.19. Feladat.

```
> #
  # This is a simple factorization
  # procedure using trial division.
  # The result is a sequence of pairs
  # [p,e] where the p's are the prime
  # factors and the e's are the exponents.
  # The factors are anyway in increasing order.
  # Only primes <= P are tried, hence the
  # last "factor" may composite, if
  # it is greater then P^2;
  #

  trialdiv:=proc(n::posint,P::posint) local L,p,i,d,nn;
  L:=[]; nn:=n;
  if type(nn,even) and 2<=P then
    for i from 0 while type(nn,even) do nn:=nn/2; od;
    L:=[[2,i]];
  fi;
  if nn mod 3=0 and 3<=P then
    for i from 0 while nn mod 3=0 do nn:=nn/3; od;
    L:=[op(L),[3,i]];
  fi;
  d:=2; p:=5;
  while p<=P and nn>=p^2 do
    if nn mod p=0 then
      for i from 0 while nn mod p=0 do nn:=nn/p; od;
      L:=[op(L),[p,i]];
    fi;
    p:=p+d; d:=6-d;
  od;
  if nn>1 then L:=[op(L),[nn,1]] fi;
  L;
  end;
```

$$trialdiv := \mathbf{proc}(n::posint, P::posint) \tag{5.19.1}$$

```
    local L, p, i, d, nn;
```

```
    L := [ ];
    nn := n;
    if type(nn, even) and 2 <= P then
        for i from 0 while type(nn, even) do
            nn := 1 / 2 * nn
        end do;
        L := [[2, i]]
    end if;
    if mod(nn, 3) = 0 and 3 <= P then
        for i from 0 while mod(nn, 3) = 0 do
            nn := 1 / 3 * nn
        end do;
        L := [op(L), [3, i]]
    end if;
    d := 2;
    p := 5;
    while p <= P and p^2 <= nn do
        if mod(nn, p) = 0 then
            for i from 0 while mod(nn, p) = 0 do
                nn := nn / p
            end do;
            L := [op(L), [p, i]]
        end if;
        p := p + d;
        d := 6 - d
    end do;
    if 1 < nn then
        L := [op(L), [nn, 1]]
    end if;
    L
end proc
```

```
> trialdiv(2^107-2^54+1,1000);
```

$$[[5, 1], [857, 1], [37866809061660057264219253397, 1]] \qquad (5.19.2)$$

```
> n0:=%[3][1];
```

$$n0 := 37866809061660057264219253397 \tag{5.19.3}$$

```
> modp(3&^(n0-1),n0);
```

$$1 \tag{5.19.4}$$

```
> trialdiv(n0-1,1000);
```

$$[[2, 2], [19, 1], [107, 1], [353, 1], [13191270754108226049301, 1]] \tag{5.19.5}$$

```
> n1:=%[5][1];
```

$$n1 := 1319127075410822604930 1 \tag{5.19.6}$$

```
> modp(3&^(n1-1),n1);
```

$$12346330842015364008789 \tag{5.19.7}$$

```
> trialdiv(n1,100000);
```

$$[[91813, 1], [143675413657196977, 1]] \tag{5.19.8}$$

```
> n2:=%[2][1];
```

$$n2 := 143675413657196977 \tag{5.19.9}$$

```
> modp(3&^(n2-1),n2);
```

$$1 \tag{5.19.10}$$

```
> trialdiv(n2-1,1000);
```

$$[[2, 4], [3, 2], [547, 1], [1824032775457, 1]] \tag{5.19.11}$$

```
> n3:=%[4][1];
```

$$n3 := 1824032775457 \tag{5.19.12}$$

```
> modp(3&^(n3-1),n3);
```

$$1527852257227 \tag{5.19.13}$$

```
> trialdiv(n3,10000);
```

$$[[1103, 1], [1653701519, 1]] \tag{5.19.14}$$

```
> n4:=%[2][1];
```

$$n4 := 1653701519 \tag{5.19.15}$$

```
> modp(3&^(n4-1),n4);
```

$$1 \tag{5.19.16}$$

```
> trialdiv(n4-1,1000);
```

$$[[2, 1], [7, 1], [19, 1], [23, 1], [137, 1], [1973, 1]] \tag{5.19.17}$$

```
> trialdiv(2^107+2^54+1,1000);
```

$$[[162259276829213381405976519770113, 1]] \tag{5.19.18}$$

```
> n5:=%[1][1];
```

$$n5 := 162259276829213381405976519770113 \tag{5.19.19}$$

```
> modp(3&^(n5-1),n5);
```

$$43364179560026952317517299954583 \tag{5.19.20}$$

```
> trialdiv(2^107+2^54+1,1000000);
```

$$[[843589, 1], [192343993140277293096491917, 1]] \quad (5.19.21)$$

```
> n6:=%[2][1];
```

$$n6 := 192343993140277293096491917 \quad (5.19.22)$$

```
> modp(3&^(n6-1),n6);
```

$$181705897546165034210519386 \quad (5.19.23)$$

▶ 5.20. Prímszámkódolás.

▼ 5.21. Feladat.

```
> p:=safeprime
  (156345678881425617888676566155526134298764532134566543212345
  678877209127512109876123212233332123343412343212334445432123432
  094872546784546778885981234236 5); log[2.](p);
  q:=safeprime
  (298415244751590016765614534678909876512342543214564909987676267
  678818251432567890998723651423415423239658787477899337772200498
  8837666788276715636388837762667772888 8); log[2.](q);
  n:=p*q;

  e:=28763541324536789099876534321234098876354231 25;
  igcdex(e,(p-1)*(q-1),'d'); d; d*e mod (p-1)*(q-1);
```

$p :=$

1563456788814256178886765661555261342987645321345665432123456\
5432123456788772091275121098761232122333321233434123432123344454321234320948725467845467788859812363179\

508.8997379

$q :=$

2984152447515900167656145346789098765123425432145649099876762678818251432567890998723651423415423239658787477899337772200498837666788276715636388837766267728888\

533.0858164

$n :=$

4665593402925412424182224876400472112892773327813449453351019945622674603742441009009573960222116531745940349543348422686605488315168714111637189152515085

80667842804378950087286351064335682692081770931613 3\
37266867667869486510555871282216710657562621632875 5\
38434596976778130582126186403663969270237481961374 9\
31132333

$e := 28763541324536789099876534321234098876354231 25$

$$\frac{1}{195712962247828637661157257250153348977923993149711\backslash}$$
$$216419886784552067125136182062835458415232964006497\backslash$$
$$161778669423464044068221493208412327356863857893715\backslash$$
$$807237232345759942482997119965921948965140136400026\backslash$$
$$278153423494508152965571832240980895842023669648156\backslash$$
$$871374368010120465251673272075317985600807326891442\backslash$$
$$00483831$$

$$1 \qquad (5.21.1)$$

5.22. Feladat.

```
> M:="Mint víz alatti, elmerült harangok
  hintáznak-e hajnalonként ágyadnál
  a tizennyolc éves iskolások
  kiket felakasztattál";

  convert(M,'bytes');
  m:=sum(%[i]*256^(i-1),i=1..nops(%));

  c:=m&^e mod n;
```

$M :=$ "Mint víz alatti, elmerült harangok

hintáznak-e hajnalonként ágyadnál

a tizennyolc éves iskolások

kiket felakasztattál"

[77, 105, 110, 116, 32, 118, 195, 173, 122, 32, 97, 108, 97, 116, 116, 105, 44, 32, 101, 108, 109, 101, 114, 195, 188, 108, 116, 32, 104, 97, 114,

97, 110, 103, 111, 107, 10, 10, 104, 105, 110, 116, 195, 161, 122, 110, 97, 107, 45, 101, 32, 104, 97, 106, 110, 97, 108, 111, 110, 107, 195, 169, 110, 116, 32, 195, 161, 103, 121, 97, 100, 110, 195, 161, 108, 10, 10, 97, 32, 116, 105, 122, 101, 110, 110, 121, 111, 108, 99, 32, 195, 169, 118, 101, 115, 32, 105, 115, 107, 111, 108, 195, 161, 115, 111, 107, 10, 10, 107, 105, 107, 101, 116, 32, 102, 101, 108, 97, 107, 97, 115, 122, 116, 97, 116, 116, 195, 161, 108]

$m :=$

19528680046240611054074111890960392345823897844641211232\
61273221169061744378240835537054015631394774815478\
29828081050733626289546862197460959443059138312535\
25019845996711281074221991389403117085548216837925\
02717827769015312568809104897659048954978097532759\
69487743773948534981673170799653012616885791655618\
893

$c := modp($ (5.22.1)

19528680046240611054074111890960392345823897844641211232\
61273221169061744378240835537054015631394774815478\
29828081050733626289546862197460959443059138312535\
25019845996711281074221991389403117085548216837925\
02717827769015312568809104897659048954978097532759\
69487743773948534981673170799653012616885791655618\
893 &^ e, n)

```
> c&^d mod n; convert(%,base,256); convert(%,'bytes');
```

19528680046240611054074111890960392345823897844641211232\
61273221169061744378240835537054015631394774815478\
29828081050733626289546862197460959443059138312535\
25019845996711281074221991389403117085548216837925\
02717827769015312568809104897659048954978097532759\
69487743773948534981673170799653012616885791655618893

[77, 105, 110, 116, 32, 118, 195, 173, 122, 32, 97, 108, 97, 116, 116, 105, 44, 32, 101, 108, 109, 101, 114, 195, 188, 108, 116, 32, 104, 97, 114, 97, 110, 103, 111, 107, 10, 10, 104, 105, 110, 116, 195, 161, 122, 110,

97, 107, 45, 101, 32, 104, 97, 106, 110, 97, 108, 111, 110, 107, 195, 169, 110, 116, 32, 195, 161, 103, 121, 97, 100, 110, 195, 161, 108, 10, 10, 97, 32, 116, 105, 122, 101, 110, 110, 121, 111, 108, 99, 32, 195, 169, 118, 101, 115, 32, 105, 115, 107, 111, 108, 195, 161, 115, 111, 107, 10, 10, 107, 105, 107, 101, 116, 32, 102, 101, 108, 97, 107, 97, 115, 122, 116, 97, 116, 116, 195, 161, 108]

"Mint víz alatti, elmerült harangok (5.22.2)

hintáznak-e hajnalonként ágyadnál

a tizennyolc éves iskolások

kiket felakasztattál"

```
> #
  # This simple procedure generates a pair of safe primes p and
  q
  # congruent 3 mod 4, and calculates the product n=p*q,
  # a Blum integer. The output is a sequence of these numbers
  # in this order. The input is two large numbers, these are
  the
  # starting points for the search for p and q, respectively.
  # For example, 100 and 104 digit starting points may be used.
  # The best to choose them randomly and independently.
  #

  setBlum:=proc(p0,q0) local p,q,n,e,d,k;
  p:=safeprime(p0);
  while p mod 4 <> 3 do p:=safeprime(p) od;
  q:=safeprime(q0);
  while q mod 4 <> 3 do q:=safeprime(q) od;
  n:=p*q; p,q,n; end;
```

$$setBlum := \mathbf{proc}(p0, q0) \quad (5.22.3)$$

$$\mathbf{local}\ p, q, n, e, d, k;$$

$$p := numtheory:-safeprime(p0);$$

$$\mathbf{while}\ mod(p, 4) <> 3\ \mathbf{do}$$

$$p := numtheory:-safeprime(p)$$

$$\mathbf{end\ do};$$

$$q := numtheory:-safeprime(q0);$$

```
    while mod(q, 4) <> 3 do
        q := numtheory:-safeprime(q)
    end do;
    n := p*q;
    p, q, n
end proc
```

```
> #
  # This inner routine make the encryption and decryption
  # for an arbitrary texts. Here n is the Blum integer,
  # L is the text, x0 is the first quadratic residue,
  # k is the number of bits used in one step.
  #

  Blumi:=proc(n,L,x0,k) local xi,b,i,j,nL;
  nL:=[]; xi:=x0;
  for i from 0 to nops(L)/k-1 do
    b:=convert(xi mod 2^k,base,2);
    for j to k do
      if nops(b)>j then nL:=[op(nL),L[i*k+j]+b[j] mod 2]
      else nL:=[op(nL),L[i*k+j]] fi
    od;
    xi:=xi &^ 2 mod n;
  od; xi,nL end;
```

```
Blumi := proc(n, L, x0, k)                                    (5.22.4)
    local ξ, b, i, j, nL;
    nL := [ ];
    ξ := x0;
    for i from 0 to nops(L) / k − 1 do
        b := convert(mod(ξ, 2^k), base, 2);
        for j to k do
            if j < nops(b) then
                nL := [op(nL), mod(L[i* k + j] + b[j], 2)]
            else
                nL := [op(nL), L[i* k + j]]
            end if
        end do;
        ξ := mod(ξ &^ 2, n)
    end do;
```

```
    ξ, nL
end proc
```

```
> #
  # This make the encryption for arbitrary texts.
  # Here n is the Blum integer, L is the text, x is the
  # random number to generate the first quadratic residue,
  # k is the number of bits used in one step.
  #

  Blume:=proc(n,x,L,k) local pt,i,j,nL;
  if igcd(x,n)>1 then ERROR(`x is not relative prime to n`) fi;
  convert(L,'bytes');
  pt:=sum(%[i]*256^(i-1),i=1..nops(%));
  pt:=convert(pt,base,2);
  if k>1 then pmod(nops(pt),k);
    if %<>0 then pt:=[op(pt),0$i=1..k-%] fi;
  fi;
  Blumi(n,pt,x &^ 2 mod n,k) end;
```

$$Blume := \mathbf{proc}(n, x, L, k) \tag{5.22.5}$$

```
    local pt, i, j, nL;
    if 1 < igcd(x, n) then
        ERROR(x is not relative prime to n)
    end if;
    convert(L, 'bytes');
    pt := sum(%[i] * 256^(i - 1), i = 1 ..nops(%));
    pt := convert(pt, base, 2);
    if 1 < k then
        pmod(nops(pt), k);
        if %<>0 then
            pt := [op(pt), $(0, i = 1 ..k - %)]
        end if
    end if;
    Blumi(n, pt, mod(x &^ 2, n), k)
end proc
```

```
> #
  # This make the decryption for arbitrary texts.
  # Here p and q are the primes, x is the last quadratic
  residue,
  # L is the string, and k is the number of bits used in one
```

```
   step.
   #

   Blumd:=proc(p,q,x,L,k) local a,b,u,v,i,n,t,nL;
   igcdex(p,q,'a','b'); t:=nops(L);
   u:=x &^ (((p+1)/4) &^ t mod (p-1)) mod p;
   v:=x &^ (((q+1)/4) &^ t mod (q-1)) mod q;
   nL:=Blumi(p*q,L,a*u+b*v mod p*q,k)[2]; n:=0;
   for i from t to 1 by -1 do n:=n+n+nL[i]; od;
   convert(convert(n,base,256),'bytes'); end;
```

$$Blumd := \mathbf{proc}(p, q, x, L, k) \tag{5.22.6}$$

$\mathbf{local}\ a, b, u, v, i, n, t, nL;$

$igcdex(p, q, 'a', 'b');$

$t := nops(L);$

$u := mod(x\ \&\hat{}\ (mod((1/4*p+1/4)\ \&\hat{}\ t, p-1)), p);$

$v := mod(x\ \&\hat{}\ (mod((1/4*q+1/4)\ \&\hat{}\ t, q-1)), q);$

$nL := Blumi(p*q, L, mod(a*u+b*v, p*q), k)[2];$

$n := 0;$

$\mathbf{for}\ i\ \mathbf{from}\ t\ \mathbf{by}\ -1\ \mathbf{to}\ 1\ \mathbf{do}$

$n := 2*n+nL[i]$

$\mathbf{end\ do};$

$convert(convert(n, base, 256), 'bytes')$

$\mathbf{end\ proc}$

```
> p,q,n:=setBlum(10^100,10^104);
```

$$p, q, n := \tag{5.22.7}$$

100000000000000000000000000000000000000000000000000\
00000000000000000000000000000000000000000043723,
100000000000000000000000000000000000000000000000000\
00000000000000000000000000000000000000000000040\
783,
100000000000000000000000000000000000000000000000000\
00000000000000000000000000000000000000000437270\
78300000000000000000000000000000000000000000000000\
00000000000000000000000000000000000000178315510\
9

```
> x,L:=Blume(n,55555,M,1);
```

$$x, L := \tag{5.22.8}$$

$$18436030386116447173266333402980395666731282469764\backslash$$
$$08201745875707371972897260497475102933664349313129\backslash$$
$$41703478879852116319162302813158000605564521749618\backslash$$
$$617678314506549917655600378158393966228778721497838524,$$

$$[1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,$$
$$1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1,$$
$$1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,$$
$$0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,$$
$$0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,$$
$$0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0,$$
$$1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,$$
$$1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,$$
$$0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0,$$
$$0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,$$
$$1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,$$
$$0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,$$
$$1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,$$
$$0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0,$$
$$1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,$$
$$0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0,$$
$$0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0,$$
$$0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,$$
$$0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0,$$
$$0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,$$
$$0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,$$
$$1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1,$$
$$0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0,$$
$$0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,$$
$$0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,$$
$$1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0,$$

1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0,
1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0,
0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0,
1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0,
0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,
1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1]

```
> Blumd(p,q,x,L,1);
```

"Mint víz alatti, elmerült harangok (5.22.9)

hintáznak–e hajnalonként ágyadnál

a tizennyolc éves iskolások

kiket felakasztattál"

```
> #
  # This simple procedure generates a safe prime q, and a
  generator
  # g of the reduced residue classes. For the given exponent d,
  # calculates e=g^d mod q. The output is a sequence
  # of the numbers [q,g,e,d] in this order. The input is
  # three large numbers, these are the starting points for the
  search
  # for q and e, respectively, and the exponent d.
  # For example, 200 digit starting points may be used.
  # The best to choose them randomly and independently.
  #

  setElGamal:=proc(q0,g0,d) local f,p,q,g,e,P;
  q:=safeprime(q0); P:=factorset(q-1); f:=false; g:=g0;
  while not f do f:=true;
    for p in P while f=true do
      if modp(g &^ ((q-1)/p),q) = 1 then f:=false; g:=g+1 fi
    od;
  od; e:=modp(g &^ d,q); [q,g,e,d]; end;
```

setElGamal := **proc**(*q0*, *g0*, *d*) (5.22.10)

```
    local f, p, q, g, e, P;
    q := numtheory:-safeprime(q0);
    P := numtheory:-factorset(q - 1);
    f := false;
    g := g0;
    while not f do
        f := true;
        for p in P while f = true do
            if modp(g &^ ((q - 1) / p), q) = 1 then
                f := false;
                g := g + 1
            end if
        end do
    end do;
    e := modp(g &^ d, q);
    [q, g, e, d]
end proc
```

```
> #
  # This make the encryption for short texts. Here q is the
  modulus,
  # g is the generator, e the encryption exponent, k the random
  # exponent, L the text.
  #

  ElGamale:=proc(q,g,e,k,L) convert(L,'bytes');
  modp(g &^ k,q),modp(sum(%[i]*256^(i-1),i=1..nops(%))*(e &^ k)
  ,q) end;
```

```
ElGamale := proc(q, g, e, k, L)                                (5.22.11)
    convert(L, 'bytes');
    modp(g &^ k, q), modp((sum(%[i]*256^(i - 1),
    i = 1 ..nops(%)))*e &^ k, q)
end proc
```

```
> #
  # This make the decryption for short texts. Here q is the
  modulus,
  # d the decryption exponent, x the number.
  #
```

```
ElGamald:=proc(q,g,d,x,y)
convert(convert(modp((x &^ (q-d-1))*y,q),base,256),'bytes');
end;
```

$$ElGamald := \mathbf{proc}(q, g, d, x, y) \quad (5.22.12)$$

$$convert(convert(modp(x \,\&\hat{}\, (q-d-1)) * y, q), base, 256), 'bytes')$$

end proc

```
> L:=setElGamal(10^200,2*10^199,3*10^199); q:=L[1]; g:=L[2];
  e:=L[3]; d:=L[4];
```

$L :=$ [

100000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000063234\
7,

200000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000,

544074621785157933767590206711282019518310813694548\
462194152770277382146869223284290838297276185369017\
566781100076197917836670850901090352692670188317\
639746638536641247150793145153552830707943395913 29,

300000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000]

$q :=$

100000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000000000000\
000000000000000000000000000000000000000000063234\
7

$g :=$

```
2000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000
```

$e :=$

```
54407462178515793376759020671128201951831081369454\
84621941527702773821468692232842908382972761853690\
17566781100076197917836670850901090352692670188317\
63974663853664124715079314515355283070794339591329
```

$d :=$ (5.22.13)

```
3000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000\
0000000000000000000000000000000000000000000000000
```

```
> LL:=ElGamale(q,g,e,5555,M[1..36]);
```

$LL :=$ (5.22.14)

```
65165693481487536607506215144369365717835711271805\
27508442784812334238582911629923019617458603467219\
41097906208897710643080925220562384173497335106877\
78140867703515669954152782946446076233390661152795,
56770284399222909337540688867509040123068328377425\
00229613942444564440855008406108780204511531072476\
48132417413840488044857357147082319364345997525328\
42778554649620887923098112324658034971325986318774
```

```
> ElGamald(q,g,d,LL[1],LL[2]);
```

"Mint víz alatti, elmerült harangok" (5.22.15)

```
> debug(ElGamal);
>
```

▶ 6. Számok és polinomok

▶ 7. Gyors Fourier-transzformáció

▶ 8. Elliptikus függvények

▶ 9. Számolás elliptikus görbéken

▶ 10. Faktorizálás elliptikus görbékkel

▶ 11. Prímteszt elliptikus görbékkel

▶ 12. Polinomfaktorizálás

▶ 13. Az AKS–teszt

▶ 14. A szita módszerek alapjai

▶ 15. Számtest szita

▶ 16. Vegyes problémák