

Collatz conjecture : Geography of the Pascal trihedron.

Programming with PARI

Hubert Schaetzel

Abstract We give a few programs for the computer algebra system PARI/GP that enables localization of integers in the “Pascal trihedron”.

Conjecture de Collatz : Géographie du Trièdre de Pascal. Programmation avec PARI/GP.

Résumé Nous proposons plusieurs programmes pour le système de calcul formel PARI/GP qui permettent de repérer les nombres entiers dans le « trièdre de Pascal ».

Date Version 01 : October 21 of 2017

1	Framework.	1
2	Construction of the v-plane.	1
3	Search for the position of an integer in the Pascal trihedron from its value.	4
4	Search for the value of an integer from its position in the Pascal trihedron.	5
5	Getting a sequence of numbers answering successive additions of 1 at the beginning of the parity vector.	6
6	Search for the value of an integer from its parity vector.	10
7	Getting the parity vector from the integer value.	11

1 Framework.

The Collatz conjecture is also called Syracuse conjecture, Ulam conjecture, Czech conjecture or 3x+1 problem. We studied this conjecture in other articles (see references [1], [2], [3]) which enabled us to build a classification of integers within an architecture that we called “Pascal Trihedron”.

The computer programs for Pari/GP below are parametrable and enable to build or locate the integers within this said architecture.

2 Construction of the v-plane.

The plane $v = 1$ includes a single item : the integer 1.

The plane $v = 2$ includes a single item : the integer 3.

The choice of v , greater than or equal to 3, below enables to calculate the v -plane v line by line, each element being in its exact position in the columns, the empty positions being viewed by the number 0. The number of elements increasing exponentially with v , there is a rapid saturation of the system of calculation for comprehensive planes. Hence, the need for alternative code programming given afterwards.

Program 1

```
/*Choose v >= 3*/
/*Recommending to limit to v = 14 max with already 8045 elements*/
v = 6;
lm = log(3)/log(2); w = 1+floor(lm*v); infini = 10^1000;
qt1 = qt2 = vector(v); cdplt=2; for(i=3, v, qt1[1] = qt = 1; for(j=2, cdplt, qt1[j] = qt1[j-1]+qt2[j]; qt = qt+qt1[j]);
for(j=1, cdplt, qt2[j] = qt1[j]); dw = floor(lm*i)-floor(lm*(i-1)); if(dw ==2, cdplt++));
print("The \"v\"-plane of the trihedron contains \"qt\" integers (0 is void):");
sgorig = 0;
for(i = 1, v, sgorig = concat(sgorig,1));
for(i = 1, w-v-1, sgorig = concat(sgorig,0));
sgorig = sgorig[^1];
```

```

sgref = 0;
for(t = 1,v, wp = 1+floor(lm*(t-1)); wc = 1+floor(lm*t); dww = wc-wp;
if(dww == 1,sgref = concat(sgref,1), sgref = concat(sgref,1); sgref = concat(sgref,0)););sgref = sgref[^1];
invtaborig = 0;
for(i = 1, v, invtaborig = concat(invtaborig,i));
invtaborig = invtaborig[^1];
invtabref = 0;tt= 0;
for(t = 1,v, wp = 1+floor(lm*(t-1)); wc = 1+floor(lm*t); dww = wc-wp; tt++; invtabref = concat(invtabref,tt);
if(dww == 2,tt++));
invtabref = invtabref[^1];
taborig = vector(v); tabref = vector(v);
for(i = 1, v, j= v-i+1; taborig[j] = invtaborig[i]; tabref[j] = invtabref[i]);
tabact = tabdpl = vector(qt); tabact[1] = taborig; tabdpl = taborig;
i = 0; r = 1;
for(t = 1,infini, i++; if(i > v, break);
if(tabdpl[i] < tabref[i],
tabdpl[i]++; r++);
if(i > 1,for(j = 1, i-1, x = i-j; tabdpl[x] = tabdpl[x+1]+1)); i = 0;
tabact[r] = tabdpl;
if(tabdpl[1] == v+1, lig = 1, lig = lig+1);
if(tabdpl == tabref, break));
ligm = lig;
col = 1; lig = 0; tabplan = matrix(ligm,v-1);
for(t = 1,qt, tabdpl = tabact[t];
res = 0; long = 2;
for(i = 1, v, ii = v-i+1;
if(tabdpl[ii]<>v-ii+1, res = concat(res, tabdpl[ii+1]); long = v-i+3; break));
for(j = v+1-ii, v, iii = v-j+1; res = concat(res, tabdpl[iii]-if(long == 2,0,1)));
res = res[^1];
res = concat(res, w);
resp = res;
resp[2] = res[2]-res[1];
for(i = 3, long, resp[i] = res[i]-res[i-1]-1);
sec = resp[long]; add = sec;
for(i = 1, long-2, ii = long-i; add = add + resp[ii]+1; sec = concat(add,sec));
sec = concat(res[1],sec);
nb1 = 0; nb2 = 0;
if(long > 2, for(i = 1, long-2, ii = long-i+1;
for(rech = 1, infini, nb1 = (rech*(2^sec[ii])+nb2*(2^resp[ii])+1)/3;
if(nb1 == floor(nb1), nb2 = 2*nb1-1;break)));
for(rech = 1, infini, nb1 = (rech*(2^sec[2])+nb2*(2^resp[2])+1)/(3^resp[1]);
if(nb1 == floor(nb1), nb2 = nb1*(2^resp[1])-1;break));
if(tabdpl[1] == v+1, col = col+1;lig = 1, lig = lig+1);
tabplan[lig,col] = nb2);
for(i = 1,ligm, print(tabplan[i,]));}

```

Example 1

The plane v = 6 :

The 6-plane of the trihedron contains 12 integers (0 is void):
[575, 287, 367, 999, 923]
[0, 735, 815, 423, 347]
[0, 0, 975, 583, 507]

The plane v = 10 :

The 10-plane of the trihedron contains 476 integers (0 is void):
[25599, 12799, 26367, 13951, 28095, 49311, 15599, 63335, 36635]
[0, 52735, 767, 53887, 2495, 23711, 55535, 37735, 11035]
[0, 1535, 15103, 2687, 16831, 38047, 4335, 52071, 25371]
[0, 30207, 43775, 31359, 45503, 1183, 33007, 15207, 54043]
[0, 22015, 35583, 23167, 37311, 58527, 24815, 7015, 45851]
[0, 0, 27903, 15487, 29631, 50847, 17135, 64871, 38171]
[0, 0, 42239, 29823, 43967, 65183, 31471, 13671, 52507]
[0, 0, 5375, 58495, 7103, 28319, 60143, 42343, 15643]
[0, 0, 62719, 50303, 64447, 20127, 51951, 34151, 72987]
[0, 0, 30975, 18559, 32703, 53919, 20207, 67943, 41243]
[0, 0, 59647, 47231, 61375, 17055, 48879, 31079, 69915]
[0, 0, 51455, 39039, 53183, 8863, 40687, 22887, 61723]
[0, 0, 37119, 24703, 38847, 60063, 26351, 8551, 47387]
[0, 0, 28927, 16511, 30655, 51871, 18159, 65895, 39195]
[0, 0, 0, 56191, 4799, 26015, 57839, 40039, 13339]
[0, 0, 0, 4991, 19135, 40351, 6639, 54375, 27675]
[0, 0, 0, 33663, 47807, 3487, 35311, 17511, 56347]
[0, 0, 0, 25471, 39615, 60831, 27119, 9319, 48155]
[0, 0, 0, 59263, 7871, 29087, 60911, 43111, 16411]
[0, 0, 0, 22399, 36543, 57759, 24047, 6247, 45083]
[0, 0, 0, 14207, 28351, 49567, 15855, 63591, 36891]
[0, 0, 0, 65407, 14015, 35231, 1519, 49255, 22555]
[0, 0, 0, 57215, 5823, 27039, 58863, 41063, 14363]
[0, 0, 0, 9599, 23743, 44959, 11247, 58983, 32283]
[0, 0, 0, 38271, 52415, 8095, 39919, 22119, 60955]
[0, 0, 0, 30079, 44223, 65439, 31727, 13927, 52763]
[0, 0, 0, 15743, 29887, 51103, 17391, 65127, 38427]
[0, 0, 0, 7551, 21695, 42911, 9199, 56935, 30235]
[0, 0, 0, 47487, 61631, 17311, 49135, 31335, 70171]
[0, 0, 0, 39295, 53439, 9119, 40943, 23143, 61979]
[0, 0, 0, 0, 33087, 54303, 20591, 68327, 41627]
[0, 0, 0, 0, 47423, 3103, 34927, 17127, 55963]
[0, 0, 0, 0, 10559, 31775, 63599, 45799, 19099]
[0, 0, 0, 0, 2367, 23583, 55407, 37607, 76443]
[0, 0, 0, 0, 36159, 57375, 23663, 5863, 44699]
[0, 0, 0, 0, 64831, 20511, 52335, 34535, 73371]
[0, 0, 0, 0, 56639, 12319, 44143, 26343, 65179]
[0, 0, 0, 0, 42303, 63519, 29807, 12007, 50843]
[0, 0, 0, 0, 34111, 55327, 21615, 3815, 42651]
[0, 0, 0, 0, 52031, 7711, 39535, 21735, 60571]
[0, 0, 0, 0, 15167, 36383, 2671, 50407, 23707]
[0, 0, 0, 0, 6975, 28191, 60015, 42215, 15515]
[0, 0, 0, 0, 58175, 13855, 45679, 27879, 66715]
[0, 0, 0, 0, 49983, 5663, 37487, 19687, 58523]
[0, 0, 0, 0, 24383, 45599, 11887, 59623, 32923]
[0, 0, 0, 0, 16191, 37407, 3695, 51431, 24731]
[0, 0, 0, 0, 43071, 64287, 30575, 12775, 51611]
[0, 0, 0, 0, 6207, 27423, 59247, 41447, 14747]
[0, 0, 0, 0, 63551, 19231, 51055, 33255, 72091]
[0, 0, 0, 0, 49215, 4895, 36719, 18919, 57755]
[0, 0, 0, 0, 41023, 62239, 28527, 10727, 49563]
[0, 0, 0, 0, 15423, 36639, 2927, 50663, 23963]
[0, 0, 0, 0, 7231, 28447, 60271, 42471, 15771]
[0, 0, 0, 0, 31199, 63023, 45223, 18523]
[0, 0, 0, 0, 45535, 11823, 59559, 32859]
[0, 0, 0, 0, 8671, 40495, 22695, 61531]
[0, 0, 0, 0, 479, 32303, 14503, 53339]
[0, 0, 0, 0, 34271, 66095, 48295, 21595]
[0, 0, 0, 0, 62943, 29231, 11431, 50267]
[0, 0, 0, 0, 54751, 21039, 68775, 42075]

```

[0, 0, 0, 0, 0, 40415, 6703, 54439, 27739]
[0, 0, 0, 0, 0, 32223, 64047, 46247, 19547]
[0, 0, 0, 0, 0, 50143, 16431, 64167, 37467]
[0, 0, 0, 0, 0, 13279, 45103, 27303, 66139]
[0, 0, 0, 0, 0, 5087, 36911, 19111, 57947]
[0, 0, 0, 0, 0, 56287, 22575, 4775, 43611]
[0, 0, 0, 0, 0, 48095, 14383, 62119, 35419]
[0, 0, 0, 0, 0, 22495, 54319, 36519, 75355]
[0, 0, 0, 0, 0, 14303, 46127, 28327, 67163]
[0, 0, 0, 0, 0, 41183, 7471, 55207, 28507]
[0, 0, 0, 0, 0, 4319, 36143, 18343, 57179]
[0, 0, 0, 0, 0, 61663, 27951, 10151, 48987]
[0, 0, 0, 0, 0, 47327, 13615, 61351, 34651]
[0, 0, 0, 0, 0, 39135, 5423, 53159, 26459]
[0, 0, 0, 0, 0, 13535, 45359, 27559, 66395]
[0, 0, 0, 0, 0, 5343, 37167, 19367, 58203]
[0, 0, 0, 0, 0, 61135, 43335, 16635]
[0, 0, 0, 0, 0, 9935, 57671, 30971]
[0, 0, 0, 0, 0, 38607, 20807, 59643]
[0, 0, 0, 0, 0, 30415, 12615, 51451]
[0, 0, 0, 0, 0, 64207, 46407, 19707]
[0, 0, 0, 0, 0, 27343, 9543, 48379]
[0, 0, 0, 0, 0, 19151, 66887, 40187]
[0, 0, 0, 0, 0, 4815, 52551, 25851]
[0, 0, 0, 0, 0, 62159, 44359, 17659]
[0, 0, 0, 0, 0, 14543, 62279, 35579]
[0, 0, 0, 0, 0, 43215, 25415, 64251]
[0, 0, 0, 0, 0, 35023, 17223, 56059]
[0, 0, 0, 0, 0, 20687, 68423, 41723]
[0, 0, 0, 0, 0, 12495, 60231, 33531]
[0, 0, 0, 0, 0, 52431, 34631, 73467]
[0, 0, 0, 0, 0, 44239, 26439, 65275]
[0, 0, 0, 0, 0, 5583, 53319, 26619]
[0, 0, 0, 0, 0, 34255, 16455, 55291]
[0, 0, 0, 0, 0, 26063, 8263, 47099]
[0, 0, 0, 0, 0, 11727, 59463, 32763]
[0, 0, 0, 0, 0, 3535, 51271, 24571]
[0, 0, 0, 0, 0, 43471, 25671, 64507]
[0, 0, 0, 0, 0, 35279, 17479, 56315]

```

3 Search for the position of an integer in the Pascal trihedron from its value.

The choice of the number nb gives its modulo 2^w representative within the Pascal trihedron, possibly different from nb, and coordinates (plane, column, row).

Program 2

```

/* Give integer : nb */
nb = 40160091701359;

infini = 10^10000; nbt = nb; sg = 0; v = 0; lm = log(3)/log(2);
for(i = 1, infini,
if(nbt/2 == floor(nbt/2), nbt = nbt/2; sg = concat(sg,0); if(nbt <= nb, break), nbt = (3*nbt+1)/2; v = v+1; sg =
concat(sg,1));
w = 1+floor(lm*v);
sg = sg[^1]; sg = sg[^w];
for(i = 1, w-1, if(sg[i] == 0, rep = i-1 ;break));col = v+1-rep;
vv = 0; tabrech = 0;
for(i = 1, w-1, vv = vv+1;if(sg[i] == 1, tabrech = concat(vv,tabrech)));
tabrech = tabrech[^v+1];
sgorig = 0;
for(i = 1, rep, sgorig = concat(sgorig,1)); sgorig = concat(sgorig,0);
for(i = 1, v-rep, sgorig = concat(sgorig,1));

```

```

for(i = 1, w-v-1, sgorig = concat(sgorig,0));
sgorig = sgorig[^1];
sgref = 0;
for(t = 1,v, wp = 1+floor(lm*(t-1)); wc = 1+floor(lm*t); dww = wc-wp;
if(dww == 1,sgref = concat(sgref,1), sgref = concat(sgref,1); sgref = concat(sgref,0)););sgref = sgref[^1];
invtaborig = 0;j = 1;
for(i = 1, w-1,
if(sgorig[i]== 1,invtaborig = concat(invtaborig,i);j = j+1));
invtaborig = invtaborig[^1];
invtabref = 0;tt= 0;
for(t = 1,v, wp = 1+floor(lm*(t-1)); wc = 1+floor(lm*t); dww = wc-wp; tt++; invtabref = concat(invtabref,tt);
if(dww == 2,tt++));
invtabref = invtabref[^1];
taborig = vector(v); tabref = vector(v);
for(i = 1, v, j= v-i+1; taborig[j] = invtaborig[i]; tabref[j] = invtabref[i]);
tabdpl = taborig; tabdpl[1] = tabdpl[1]-1;
i = 0; r = 1;
for(t = 1,infini, i++; if(i > v, break);
if(tabdpl[i] < tabref[i],
tabdpl[i]++; r++;
if(i > 1,for(j = 1, i-1, x = i-j; tabdpl[x] = tabdpl[x+1]+1)); i = 0;
if(tabdpl == tabrech, break));
nbpl = nb-(2^w)*floor(nb/(2^w));
if(nbpl <> nb, print("Integer "nb" is represented by "nbpl" in the trihedron."));
print("The location of integer "nbpl" in the trihedron is plane "v", column "col", line "r-1".")

```

Examples 2

Integer nb = 40160091701359 :

Integer 40160091701359 is represented by 111 in the trihedron.
The location of integer 111 in the trihedron is plane 19, column 16, line 150386.

Integer nb = 110420130004991 :

The location of integer 110420130004991 in the trihedron is plane 30, column 10, line 5.

4 Search for the value of an integer from its position in the Pascal trihedron.

The choice of the position (plane v, column, row) gives the modulo 2^w representative within the trihedron.

Program 3

```

/* Give plan rank : v */
v = 7;
/* Give column : column (<= v-1) */
column = 2;
/* Give line : line (must be not void)*/
line = 2;
infini = 10^10000;lm = log(3)/log(2); w = 1+floor(lm*v);alert = 0;
sgorig = 0; rep = v+1-column;
for(i = 1, rep, sgorig = concat(sgorig,1)); sgorig = concat(sgorig,0);
for(i = 1, v-rep, sgorig = concat(sgorig,1));
for(i = 1, w-v-1, sgorig = concat(sgorig,0));
sgorig = sgorig[^1];
sgref = 0;
for(t = 1,v, wp = 1+floor(lm*(t-1)); wc = 1+floor(lm*t); dww = wc-wp;
if(dww == 1,sgref = concat(sgref,1), sgref = concat(sgref,1); sgref = concat(sgref,0)););sgref = sgref[^1];
invtaborig = 0;j = 1;
for(i = 1, w-1,
if(sgorig[i]== 1,invtaborig = concat(invtaborig,i);j = j+1));

```

```

invtaborig = invtaborig[^1];
invtabref = 0;tt= 0;
for(t = 1,v, wp = 1+floor(lm*(t-1)); wc = 1+floor(lm*t); dww = wc-wp; tt++; invtabref = concat(invtabref,tt);
if(dww == 2,tt++));
invtabref = invtabref[^1];
taborig = tabref = tabdpl = vector(v);
for(i = 1, v, j= v-i+1; taborig[j] = invtaborig[i]; tabref[j] = invtabref[i]);
tabdpl = taborig; tabdpl[1] = tabdpl[1]-1;
i = 0; r = 1;
for(t = 1,infini, i++; if(i > v, break);
if(tabdpl[i] < tabref[i],
tabdpl[i]++; r++);
if(tabdpl[column] == archive+1, if(column >=1, alert = 1;print("Wrong choice : Line position outside the trihedron.")));
if(i > 1,for(j = 1, i-1, x = i-j; tabdpl[x] = tabdpl[x+1]+1)); i = 0;
archive = tabdpl[column];
if(r-1 == line, break));
if(alert == 1, break,
parv = 0; absc = 0;
for(i = 1, w-1, if(tabdpl[v-absc] == i, parv = concat(parv,1); absc = absc+1; if(absc > v-1, absc = v-1), parv = concat(parv,0)));
parv = parv[^1];
res = 0; long = 2;
for(i = 1, v, ii = v-i+1;
if(tabdpl[ii]<>v-ii+1, res = concat(res, tabdpl[ii+1]); long = v-i+3; break));
for(j = v+1-ii, v, iii = v-j+1; res = concat(res, tabdpl[iii]-if(long == 2,0,1)));
res = res[^1];
res = concat(res, w);
resp = res;
resp[2] = res[2]-res[1];
for(i = 3, long, resp[i] = res[i]-res[i-1]-1);
sec = resp[long]; add = sec;
for(i = 1, long-2, ii = long-i; add = add + resp[ii]+1; sec = concat(add,sec));
sec = concat(res[1],sec);
nb1 = 0; nb2 = 0;
if(long > 2, for(i = 1, long-2, ii = long-i+1;
for(rech = 1, infini, nb1 = (rech*(2^sec[ii])+nb2*(2^resp[ii])+1)/3;
if(nb1 == floor(nb1), nb2 = 2*nb1-1;break)));
for(rech = 1, infini, nb1 = (rech*(2^sec[2])+nb2*(2^resp[2])+1)/(3^resp[1]);
if(nb1 == floor(nb1), nb2 = nb1*(2^resp[1])-1;break));
print("The element of the "v"-plane in line "line" and column "column" is equal to "nb2"."));
}

```

Exemple 3

The integer of plan v = 19 with coordinates (341651,16) is :

The element of the 19-plane in line 341651 and column 16 is equal to 1507847631

Integer of plane v = 14 with coordinates (3388,12) doesn't exist :

Wrong choice : Line position outside the trihedron.

5 Getting a sequence of numbers answering successive additions of 1 at the beginning of the parity vector.

The choice of an integer aa, whose parity vector is vp, gives by the algorithm below a sequence of numbers having parity vectors [1, vp], then [1,1,vp,0], then [1,1,1,vp,0],... the number of final 0 being adjusted to the relationship linking v the number of multiplication steps to w the number of division steps, that is w = int(ln(3)/(2).v)+1 (in altitude flight).

The calculations are done without any recourse to the parity vectors (aiming a very high speed of execution). This program allows for example to recursively calculate the rest of the first elements of planes $a_v(1,1)$.

It also allows to recursively calculate the series of the second elements of planes $a_v(1,2)$.

Similarly, each element of a first line $a_v(1,i)$ gives an analog series $a_{v+n}(1,i)$.

This continues throughout the second block of a Pascal trihedron plane : $a_v(i$ in second block, $j)$ gives $a_{v+n}(i,j)$.
In addition, calculation gives jointly for $a_v(i,j)$ its flight altitude output value $m_v(i,j)$.

Program 4

```
/*Choose starting integer */
aa = 5;
/*Choose number of steps */
steps = 100;
infini = 10000000; lm = log(3)/log(2);
mm = aa; m = a = vector(steps); id = 0;
for(t = 1,infini,
if(mm/2 == floor(mm/2), mm = mm/2, id = id+1; mm = 3*mm+1);
if(aa >= mm, break););
wid = 1+floor(lm*id); ret1 = 2^wid; acal= aa-ret1*floor(aa/ret1); ret2 = 3^id; mcal= mm-ret2*floor(mm/ret2);
if(acal <> aa, print("Integer "aa" is represented by "acal" in the trihedron."));
a[1] = acal; m[1] = mcal;
print("v  m(v)  a(v)");
print(id" "m[1]" "a[1]);
limit = steps+id-1;
for(v = id, limit-1, w = 1+floor(lm*v) ; wp = 1+floor(lm*(v+1)); dw = wp-w;
for(r = 0, 5, vv = v-id+1; mm = (m[vv]+r*(3^v))/(2^(dw-1)) ; aa = (2*(a[vv]+r*(2^w))-1)/3;
if(aa == floor(aa), if(mm == floor(mm), t = v+1; tt = t-id+1; m[tt] = mm; a[tt] = aa; ret = 2^wp;
m[tt]= m[tt]-ret*floor(m[tt]/ret);
a[tt]= a[tt]-ret*floor(a[tt]/ret);
print(t" "m[tt]" "a[tt]);break))))}
```

Example 4

The series of the 50 first elements of each plane (starting with 5) :

Integer 5 is represented by 1 in the trihedron.

v	m(v)	a(v)
1	1	1
2	2	3
3	20	23
4	10	15
5	91	95
6	410	575
7	205	383
8	205	255
9	3383	5631
10	23066	25599
11	70582	104447
12	35291	69631
13	566732	745471
14	1877689	3293183
15	1877689	2195455
16	8113298	12648447
17	94206740	97910783
18	47103370	65273855
19	23551685	43515903
20	1185813152	1460666367
21	4079690977	6700400639
22	4079690977	4466933759
23	49111434902	71697432575
24	24555717451	47798288383
25	589414790413	764873277439
26	718351699928	1242923270143
27	3260217528257	3760646520831
28	5442907506622	8371159695359
29	5442907506622	5580773130239
30	2721453753311	3720515420159
31	310197425018629	565430297034751
32	927870821302576	1127553469251583
33	2316955599503129	3754102064414719

34	13435076732614175	14512333715931135
35	48410492615473510	69722884175560703
36	24205246307736755	46481922783707135
37	24205246307736755	30987948522471423
38	237244576099367059	404965800550596607
39	2938948011445351237	3344434545985323007
40	3495751582232163752	5304080376275140607
41	3495751582232163752	3536053584183427071
42	74693868545457654682	100740004115906559999
43	256184912535753545759	460690542983074807807
44	912698847324827701013	1094188109133657407487
45	2918276679121441932709	4664764141813809283071
46	8826902092223109329995	9406331405035399806975
47	8844920105937805212962	12567376247183126822911
48	31011274411926405894268	58750159982064966828031
49	190544160565671425620990	240654409923814840336383
50	334571409513453242400578	563411546487419618459647

Number 1000th of the series

1000	11027773766223939255599638243685383983123058040229118529895194635849895633361467411138240247775072661
9658150157035536038452913603129669634101848918205329307515073658919130294253278150609776924383406962978845	
0644321665401676961180452393456153975938038355963675587842309211959117160252848025523124344888826836411307	
8890677027695508662577537129844756989406360976065975196388266745940517417488868572248510967648397119266815	
58901600440324252134496987219467406007352136396783848666645	
1131817111979028005238968810662181601027096670	
4539395695038961943294809801395772938409808943142435536912466639492491375516278778672506461456066473815462	
9935851620439894616383876878359541056977271721011852700968538880599743458426880194928762980647025433476829	
5217540415359597797337275682099623440815404654638275837876230714470651737163334210306119645953170309462667	
2958818724087812388466870693936800254400938270212681887607569528364847105839757038671880257837927976119270	
80534015	

Number 10000th of the series

10000	1355804477459747676955643086236374860704684364747777944908196135411095481544526486642744900255633638
8372676434100570775113547079492777406822381936174631349226325946736161321618852289063601097532088765276022	
322009130640959525465979083084753684655958449014252861975561389610300194971825240439397174824453359249470	
8982676745502490643105792221326101763608769188242975163684927267234266616931753355743304879494909369465467	
1636722715291477956419362070101846723708237180594789967683973660476147889526215841535390818828871670969602	
7862697260665365987786376199816082415678409091216040475535421744944158855174666052331443956021266752594155	
897878103412503180391298256795505972969283589024018822930858565093422551351887465488700263912644032932478	
219508842520094253023230245504587579598558747329981969891452577582247818916311087415561999480896266328839	
695076555641601871746901258828453963024355749962408479521500306851408993494724023031818939390906027675439	
621906981948139143879954025460746314263678984307838431923084629535402521012590745419055568082650259394739	
7207543728059268252784668523876824678067036267571330978031772214027132327679714789485762989780543180222794	
468942389592746752853482400504185081487538785910234488586335567707045533052059857753649148177948722483776	
5551817079931160918665201976487010432353942523497611200733496818751352175220392765114073276179197073994193	
0334272323353869979822617840376755082567058636354287963628850584511769640841676440499920287717563062363233	
7077773171582252515862902650600461014784676754718769832329929029950233271987503018924721071493838417870146	
126532628582000290280671190712729598783382805142209410661592381248310023279066919748668608508643048888961	
1917155999071847939267668574801309992978294745512453029002072604419721939984496637929883912999204250365074	
1471209577171970495066014083964941153905803437799017843327850513691090223383443584510801231752223269527861	
8347317834300111889280000355919562627260594246183852498668974818806042545206361243413293064633573807877476	
3905997061386143573187714919025090322500269984659629049035042696369624626151629100717657804894292379176948	
2777924042445695113854021372896959541630724083506193279558248551023693949538549696233759051253788391884187	
8397312615323498695442835240463465961109532267039621345888896657692005407298237875241045196289661186392989	
616893563469310501422395092362938366561184754393454715581884226466714176646047791159898783756436588725805	
7419234662337657055709262695062125229393005182709405442736347632039124294974241610475541949342520476576404	
9427415330783406907936319942878245786379149868765097755301741923980962683982042090752035297908643526754078	
3930388237385570780693930734805029804079253946325787529248547240103264346716531803818946250424917858610315	
1365419556882824048084083179849706089663002612766874033874705121072073399300400255519521282664070074401129	
7040111495928013795302167531230771504470465556593295779091751264484339647438551365159556029277908979539675	
18586318407288822911043498220633304655471076814825871806296552686105896995314241610901554496637291480109	
6655484324463506517513294310510250856807959814630402088981289985308511030146070625137093170307624525022549	
9088725611012551314963219695112367095324314833590378814052361077158554746499963240964033027428357836872477	
6019090133702485637035598931831510599973227218275203418329130764498608973244251164005061630706769322453412	
5987961683613165107787810325755067158490714310565675492076598731177536762507503982496993551379751663124148	
5953263608647512813095991708203388372501009800681593741212409209819860857090644637867891951463817681347895	

6912520368749692416213209252393881909178394485081948739100188316039816254659988678454604369182424605740711
7513048392967537603554025652034180627949182985385131937434670727418382138387819133225688240999348339695807
8742097154469236840395372505165590611063549135021518792604235743619408874907641575115329950699609749464287
9248986867468414162422872634972635308694878378377036842333586447847566839962255247249445650064961802548815
1538359477293403985012598660253512052930107743421748123667188690913625015988687514599781062647185731996047
478309974434126635681706611000359943473265483248039851829696915407452488553937094221274587371055890543874
2429045837209851442581510338496771611041903349709795343607374014062889443067845595926749404036785245420524
335890739433903060278578259279527504327116578367862049004362088347854754169420690083926722405773055032430
9680029828275499080491235930162432024037903092477774890098441693201481429627993857184714267381278375794786
2095588099755125468116996558584123889187601262294114985102485953603032602017133827979796260333313310752457
3100474731319701278283698303404075256990785935238605580263630404761719268568223089097182474832029414672820
86072349 1758252085792249411761416308381793239188059313865951574886800061126501048635126307069232775447522
5012225957038147782118263795077512987772825065924348903635209341611653932304996754387661362184278182789969
7439838394151569806005197756985415467382648816432808797272113913757193491318743157452355529745137903126098
403849486601915039265732915076335217274306795421386492251206491047338998938364322453273218166915461734372
6195733849177763798243792796727640920817765261449320458018144664071203353955944147241958319447866118694684
8821114743162110358828656606385550472092755881514049387660963165662385189153027370285480564931011668627850
9724898584341004007589958529077454373784574262035530938878958410970268641412091500668561370597141935595108
8539573705765550713779382940562304468879007017369618779746346545957061995934340047318655497687582267770231
8270877306100710827481474272742975853498624505690456566146233457000094798913286073215189116997986424764098
232864575945875830964190756442660719087093701975414725901269028483772944927993499313643680036653001542938
2337938627395313465709811541220995354948824833895523007106176846353596304760585963860228625276978372971738
9210518185472538139862377859183054885030398430214315382658990895762484712525957261936283688936429220504207
7997571327590712402968272433247397607036717877851742949920149848576469186487495417716108356220075549706724
9130119318142132413028002004356426481298988724810346677070344606332455719052051118402024164650787517566051
6737212317137548620091477430742224011088014707940362951705560707843655260506060519319697473560669289796956
116340127744494021400745217435771272730514898941918790865284642332556328210770002377482893680402078268822
1219376324683339710638568324429465443409598203955931943094204149590046978747627789064460969626486972948020
8979890151767862846483916185830857878393564885006649922370457991376722209109556254341989090017472329866040
5217585181435038391034639547118843944055969852614748363343564344273780073813566406288334379393693389109063
8037350614640087694959109717006461814884470046405213213240943436368690952881821982559313054119504902678262
4484148440573901802268879579112792244507029184034692366832600242704847420581586635657648034266873748404502
312422017382088072927273721370976076241105487967723147235597695222949874365844916585227191374765106474903
3516564699334470090690597332048756294028304037832345718997195558235891057720046792004289672098539481886775
447688579718517953676532065327450112959653334025698838972454963988569621407037981098677592223643798389
735790227689095059947988396553924133928558173973723945445974511523001602478682786174531081990173077364952
9483272027896916032411229227969976081260883413070357952313263954300530262644120449303729466320212448971645
4944083207576075868909582036153856688276819600799641290222489794327346167591280525432056692625481791738296
458786372494272382250042056279377644525881723386130894941671324665000179573946681319211674825930892448533
0619608057805022179103626689758604834123553788068651941831264549079151111960832996494187901583980858227019
4928562410100973927766058942480093484976905500829253944968114918267143981121725051072586744934991245020062
513115536938816947170137238623561904986878834740521104569613629848322564811943289532033410576353474468365
306296985006283533819721978387239479568468430979182938186892916731326625051145635053829758465499616521589
6564082631746680654633130265148412667703353184819163807054090830915453647715377945876348368609438112523790
750135938981537059110858059784247076609273386320855564160438392277472039896061626881642981138913238055865
699285530447465350181494029393795723459317330401232272363891366785243675894759687706888497015228645391497
8016001214283328764302428613593730643678158206624628459429175228516389415445646792823483451542534817804767
338094233436892587534516418900242894037850088041425603844329267506519283241123835088118414441021959746538
545956321885327998734852780389733276118622678336558040021882539682138252858046142165016972201277223255
83713299967865720215061666941319864702730140406795914485422989257083239022219478336435548278569648998943
8683395629595179794641751496116714464828021681954849910117449396399148400654205753652265331641563267700075
821534780135122585634124752554563158217488421151118627242577642053173974609421509793823406567722634628763
6625800882173410339589364529979492368234855778118163941298931848978463727236078231234276169920992705641660
4274163722172621204681258831264315926116502384215442794741384933407076427059059470553145766389738654928457
9910644589115598118604009414241832571516616338588679348504812793136744205959139178197817434978912056471227
1538335198407412477320296603936935940033731211486142647844101503089065009280511171677896217241585114995732
01733681151

6 Search for the value of an integer from its parity vector.

The choice of licit parity vector “vectpar” gives the integer nb2 that generates it.

The choice of a parity vector without precaution is often wrong. The number w of divisions by 2 and v of multiplications by 3 plus 1 are linked by $w = \text{int}(\ln(3)/\ln(2).v) + 1$. Moreover, a bad choice can also come from the fact that too many divisions by 2 are selected at the beginning of the vector. A bad choice is detected (but not corrected).

Program 5

```

 $\text{/* Choose a valid parity vector */}$ 
vectpar = [1,0,1,1,1,0,1,1,1,0,0,0];
lg = #vectpar; lm = log(3)/log(2); nbi = 0; nbp = 0;
for(i = 1, lg, if(vectpar[i] == 1, nbi = nbi+1, nbp = nbp+1));
wred = 1+floor(lm*nbi)-nbi;
alert = 0;
if(wred > nbp, print("Missing "wred-nbp" zero(s) in the parity vector "vectpar". Retype valid vector."); alert = 1);
if(wred < nbp, print(nbp-wred" zero(s) in excess in parity vector "vectpar". Retype valid vector."); alert = 1);
nbi = 0; nbp = 0;
if(alert == 0, for(i = 1, lg-1, if(vectpar[i] == 1, nbi = nbi+1, nbp = nbp+1); wred = 1+floor(lm*nbi)-nbi;
if(wred <= nbp, print("Premature zero(s) in the parity vector "vectpar". Retype valid vector."); alert = 1;break));
if(alert == 1, break, vectpar = vectpar[^lg];
v = nbi; w = 1+floor(lm*v); tabdpl = 0; infini = 10^1000;
for(i = 1, w-1, if(vectpar[i] == 1, tabdpl = concat(i, tabdpl)));
tabdpl = tabdpl[^v+1];
res = 0; long = 2;
for(i = 1, v, ii = v-i+1;
if(tabdpl[ii]<>v-ii+1, res = concat(res, tabdpl[ii+1]); long = v-i+3; break));
for(j = v+1-ii, v, iii = v-j+1; res = concat(res, tabdpl[iii]-if(long == 2,0,1)));
res = res[^1]; res = concat(res, w); resp = res; resp[2] = res[2]-res[1];
for(i = 3, long, resp[i] = res[i]-res[i-1]-1;
sec = resp[long]; add = sec;
for(i = 1, long-2, ii = long-i; add = add + resp[ii]+1; sec = concat(add,sec));
sec = concat(res[1],sec);
nb1 = 0; nb2 = 0;
if(long > 2, for(i = 1, long-2, ii = long-i+1;
for(rech = 1, infini, nb1 = (rech*(2^sec[ii])+nb2*(2^resp[ii])+1)/3;
if(nb1 == floor(nb1), nb2 = 2*nb1-1;break)));
for(rech = 1, infini, nb1 = (rech*(2^sec[2])+nb2*(2^resp[2])+1)/(3^resp[1]);
if(nb1 == floor(nb1), nb2 = nb1*(2^resp[1])-1;break));
print("The integer corresponding to the parity vector "vectpar" is "nb2"."))}

```

Examples 5

is

37673421724097038780882884211240197520960230005261015864955325865342591310042774114581230646205
75874987337922452989160224604545786177507888654818856289857023223150693458377197935460529496000
34319471751557130633267779954550335374416876680786772400054796794305013349731326384733862501764
190704390025360137350466904678483092399321921516271.

Premature zero(s) in the parity vector [1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0] . Retype valid vector.

2 zero(s) in excess in parity vector [1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0] . Retype valid vector.

Missing 1 zero(s) in the parity vector [1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0]. Retype valid vector.

7 Getting the parity vector from the integer value.

The use of the Collatz algorithm gives the parity vector without difficulty.

Program 6

```

/* Choose a number */
nb = 27+2^59;
nbr = nb; infini = 10^1000; vectpar = 0; v = 1; nbr = (3*nbr+1)/2; vectpar = concat(vectpar, 1);
for(i = 1, infini,
if(nbr > nb,
if(nbr/2 == floor(nbr/2), nbr = nbr/2; vectpar = concat(vectpar, 0), v = v+1; nbr = (3*nbr+1)/2; vectpar =
concat(vectpar, 1)),
break));
vectpar = vectpar[^1];
lm = log(3)/log(2); w = 1+floor(lm*v); ret = 2^w;
nbpl = nb-ret*floor(nb/ret);
if(nbpl <> nb, print("Integer "nb" is represented by "nbpl" within the trihedron."));
print("Integer "nbpl" is located in plane "v" and as parity vector "vectpar".")}
```

Examples 6

Integer 576460752303423515 is represented by 27 within the trihedron.

Integer 27 is located in plane 37 and has parity vector [1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0].

8 Interest note

The hereby architecture is not limited to a simple localization of integers. The elements are linked together by common properties :

- Plane : Same altitude flight time (stopping time)
 - Lines: Difference between lines of same value modulo 2^w ,
 - Columns: Difference between columns of same value modulo 2^w .

For example:

Let us consider the plane $v \equiv 5$, $w \equiv \text{int}(\ln(3)/\ln(2), v) + 1 \equiv 8$, $2^w \equiv 256$, whose elements are:

95	175	39	219
	79	199	123

Lines (1)-(2) : $175-79 = 39-199 = 219-123 = 96 \bmod 256$.

Columns (3)-(2) : $39-175 = 199-79 = 120 \bmod 256$.

Columns (4)-(3) : $219-39 = 123-199 = 180 \bmod 256$.

References

- [1] <https://sites.google.com/site/schaetzelhubertdiophantien/>
Altitude flight with Collatz numbers : the genesis of a Pascal trihedron.
- [2] <https://sites.google.com/site/schaetzelhubertdiophantien/>
Pascal trihedron and Collatz algorithm.
- [3] <https://sites.google.com/site/schaetzelhubertdiophantien/>
Collatz conjecture : Geography of the Pascal trihedron.