les entiers parfaits seraient donc plutôt rares

Sacré Euclide ! Tu tiens le voyage. Dix numéros déjà qu'à travers cette série au pays des nombres, tu nous amènes de nouvelles idées de programmes. Mais là, c'est décidé, tu débarques. Sûr, tu nous manqueras. Mais on n'est pas triste. Toi dont les lumières éclipsaient voici deux mille ans celles-là même du phare d'Alexandrie, on sait que tu seras toujours là pour éclairer quelques sous-bois obscurs des forêts de nombres. Bref ! Salut à toi, Euclide... Et, merci encore !

Dans ce dernier article de la série sur la DIVISION EUCLIDIENNE et ses prolongements, vous trouverez des solutions aux exercices n° 26 à 30.

Commençons par l'exercice n° 26. Il s'agissait de calculer les PLOUTONS, c'est-à-dire les entiers qui ont plus de diviseurs que ceux qui les précédent.

01 N ← 1
02 NBMAX ← 1 * NBMAX = nombre de diviseurs du dernier plouton
03 *
04 N ← N+1
05 Calcul de NDIVN ; * NDIVN = nombre de diviseurs de N
06 Si (NBMAX ⩾ NDIVN) alors aller à 04
07 *
08 NBMAX ← NDIVN
09 Ecrire N
10 Aller à 04
11 *
12 Fin
100 ***
101 *Calcul de NDIVN
102 DIV ← 1 ; NDIVN ← 1
103 *
104 DIV ← DIV+1
105 Si (DIV > (√N)) Alors aller à 117 ; * fini
106 *
107 Si (DIV = >N) Alors aller à 115
108 Q ← N/DIV
109 *
110 Si (Q = Int (Q)) Alors aller à 104
111 NDIVN ← NDIVN+2
112 Aller à 104
113 *
114 *
115 NDIVN ← NDIVN+1
116 *
117 Retour

Afin de limiter le nombre de pas de programme et de réduire le temps de calcul, le sous-programme qui calcule le nombre de diviseurs de N a été inclus dans le programme principal (pas 06 à 34)

Mode d'emploi

N°	Instructions ou données	Touches	Affichage
01 à 04 : voir programmes précédents			
05	Introduire 40 en R7 41 en R1 0 en R0	40 STO 7 41 STO 1 0 STO 0	40. 41. 0.
06	Afficher les P(n)	R/S	P(n).

Registres

R0	n
R1	P (n+1)
R7	40 (test)

Affichage		
Pas	**Codes**	**Touches**
00	01	1
01	31 0	STO 0
02	32 4	STO 4
03	86 11	Lbl 1
04	01	1
05	34 0	SUM 0
Calcul de NDIVN		
06	32 1	STO 1
07	32 5	STO 5
08	33 0	RCL 0
09	24	$\sqrt{x}$
10	32 2	STO 3
11	86 4	Lbl 4
12	01	1
13	34 1	SUM 1
14	33 1	RCL 1
15	32 7	STO 7
16	33 3	RCL 3
17	–76	INV x ⩾ t
18	51 6	GTO 6
19	66	x=t
20	51 5	GTO 5
21	33 0	RCL 0
22	45	÷

Affichage		
Pas	**Codes**	**Touches**
23	33 1	RCL 1
24	85	=
25	32 7	STO 7
26	49	Int
27	–66	INV x=t
28	51 4	GTO 4
29	02	2
30	34 5	SUM 5
31	51 4	GTO 4
32	86 5	Lbl 5
33	01	1
34	34 5	SUM 5
....	....	....
35	86 6	Lbl 6
36	33 5	RCL 5
37	32 7	STO 7
38	33 4	RCL 4
39	76	x ⩾ t
40	51 1	GTO 1
41	33 0	RCL 0
42	81	R/S
43	33 5	RCL 5
44	32 4	STO 4
45	51 1	GTO 1

Voici quelques résultats obtenus avec ce programme :
2? 4, 6
12, 24, 36, 48, 60
120,180,240,360,720,840, etc.

Je n'ai pas réussi à faire tenir dans la mémoire programme de la TI.57, un programme, solution de l'exercice n° 27, qui engendre la suite des ploutons en utilisant la version optimisée du programme de l'exercice n° 25. Je suis intéressé par une solution de ce type.

Voici cependant un programme utilisant la solution donnée à l'exercice n° 24.

01 N ← 1
02 NBMAX ← 1
03 *
04 N ← N+1
05 NDIVN ← 1
06 DIV ← 1
07 N' ← N
08 *
09 DIV ← DIV+1
10 CPTR ← 1
11 Q ← N'/DIV
12 Si (Q = Int (Q)) Alors aller à 27
13 *
14 CPTR ← CPTR+1
15 N' ← Q
16 Si (N' = 1) Alors aller à 11
17 *
18 NDIVN ← NDIVN × CPTR
19 *
20 *
21 *
22 Si (NBMAX ⩾ NDIVN) Alors aller à 04
23 Ecrire N
24 NBMAX ← NDIVN
25 Aller à 04
26 *
27 Si (CPTR = 1) alors aller à 09
28 NDIVN ← × CPTR
29 Aller à 09
30 *
31 Fin

Registres	
R0	N'
R2	NBMAX
R3	DIV
R4	CPTR
R5	NDIVN
R6	N
R7	Tests

Ce programme est moins intéressant car bien plus lent que le précédent.

Pour l'exercice n° 28, nous vous présentons une démarche peu « orthodoxe ». En effet, nous allons dégrader la qualité des programmes écrits précédemment afin d'en écrire un qui tienne dans les 50 pas de la mémoire du calculateur. Il est ainsi plus concis mais aussi plus lent. Voyons cela.

01 N ← 1
02 *
03 N ← N+1
04 *
05 K ← N
06 Calcul de SdK
07 Si (SdK < N) Alors aller à 03
08 *
09 K ← SdK
10 Calcul de SdK
11 Si (SdK = N) Alors aller à 03
12 *
13 Ecrire N, K
14 Aller à 03
15 *
16 Fin
100 ***
101 * Calcul de SdK
102 DIV ← 1
103 SdK ← 1
104 *
105 DIV ← DIV+1
106 Si (DIV^2 > K) Alors aller à 119
107 *
108 Si (DIV^2 = K) Alors aller à 116
109 Q ← K/DIV
110 *
111 Si (Q = Int(Q)) Alors aller à 105
112 SdK ← SdK+DIV+Q
113 Aller à 105
114 *
115 *
116 SdK ← SdK+DIV

Mode d'emploi			
N°	**Instructions ou données**	**Touches**	**Affichage**
01	Passer en mode « Programme »	LRN	
02	Introduire le programme		
03	Passer en mode « Calcul »	LRN	
04	Initialiser le pointeur	RST	O.
05	Afficher les ploutons	R/S	N.

Affichage		
Pas	Codes	Touches
00	01	1
01	32 6	STO 6
02	32 2	STO 2
03	86 1	Lbl 1
04	01	1
05	34 6	SUM 6
06	32 5	STO 5
07	32 3	STO 3
08	86 3	Lbl 3
09	01	1
10	34 3	SUM 3
11	32 4	STO 4
12	33 6	RCL 6
13	32 0	STO 0
14	86 4	Lbl 4
15	33 0	RCL 0
16	45	÷
17	33 3	RCL 3
18	85	=
19	32 7	STO 7
20	49	Int
21	–66	INV x=t
22	51 5	GTO 5
23	32 0	STO 0
24	01	1

Affichage		
Pas	Codes	Touches
25	34 4	SUM 4
26	32 7	STO 7
27	33 0	RCL 0
28	–66	INV x=t
29	51 4	GTO 4
30	33 4	RCL 4
31	39 5	Prd 5
32	33 5	RCL 5
33	32 7	STO 7
34	33 2	RCL 2
35	76	x ⩾ t
36	51 1	GTO 1
37	33 5	RCL 5
38	32 2	STO 2
39	33 6	RCL 6
40	81	R/S
41	51 1	GTO 1
42	86 5	Lbl 5
43	01	1
44	32 7	STO 7
45	33 4	RCL 4
46	66	x=t
47	51 3	GTO 3
48	39 5	Prd 5
49	51 3	GTO 3

Registres	
R0	K
R1	DIV
R5	SdK
R6	N
R7	Tests

117 Aller à 119
118 *
119 Retour

Vous remarquerez que le sous-programme de calcul de SdK, aux pas 20 à 49 comporte deux instructions de RETOUR en 32 et 49. On gagne ainsi 1 pas en évitant un Label.

Mode d'emploi			
Nº	Instructions ou données	Touhes	Affichage
01	Passer en mode « programme »	LRN	
02	Introduire le programme		
03	Passer en mode « calcul »	LRN	
04	Initialiser N à 1	1 STO 6	1.
05	Initialiser le pointeur	RST	...
06	Afficher un entier	R/S	N.
07	Afficher un « ami »	R/S	K.
08	Pour le couple suivant, aller en 05		

En 08, on profite de l'appel de R5 pour l'enregistrer aussitôt en R0, ce qui permet d'appeler le sous-programme en 11 sans avoir à initialiser R0. Il aurait été plus lisible d'écrire :

07 RCL 5
08 INV x ⩾ t
09 RST
10 RCL5
11 STO 0
12 SBR 3
13 RCL 6
etc.

On perd alors un pas qu'il est bien difficile de récupérer ensuite. Mais la partie la plus « inélégante » de ce programme se trouve dans le sous-programme, où les tests d'arrêt se font sur DIV^2, ce qui oblige à reclaculer ce carré à chaque boucle alors qu'en

Affichage		
Pas	Codes	Touches
00	01	1
01	34 6	SUM 6
02	33 6	RCL 6
03	32 0	STO 0
04	61 3	SBR 3
05	33 6	RCL 6
06	32 7	STO 7
07	33 5	RCL 5
08	32 0	STO 0
09	–76	INV x ⩾ t
10	71	RST
11	61 3	SBR 3
12	33 6	RCL 6
13	32 7	STO 7
14	33 5	RCL 5
15	–66	INV x=t
16	71	RST
17	81	R/S
18	33 0	RCL 0
19	81	R/S
20	86 3	Lbl 3
21	01	1
22	32 1	STO 1
23	32 5	STO 5
24	86 4	Lbl 4

testant DIV par rapport à $\sqrt{N}$, le calcul de ce dernier nombre ne se fait qu'une seule fois, et en dehors de la boucle. Malgré mes efforts, je n'ai pas trouvé d'autres moyens pour faire tenir ce programme dans les 50 pas. Je suis intéressé par une solution « élégante » qui tiendrait en 50 pas ou moins.

On trouve les nombres suivants :
6 et 6 ; 28 et 28.

Ce qui n'est pas étonnant, puisque 6 et 28 sont parfaits.
220 et 284
496 et 496 ; 496 est parfait.

Et avec une dose de patience que je suis loin de posséder, on peut encore obtenir :
1184 et 1210.

Affichage		
Pas	Codes	Touches
25	01	1
26	34 1	SUM 1
27	33 1	RCL 1
28	23	x^2
29	32 7	STO 7
30	33 0	RCL 0
31	−76	INV x ⩾ t
32	−61	INV SBR
33	66	x=t
34	51 5	GTO 5
35	45	÷
36	33 1	RCL 1
37	85	=
38	32 7	STO 7
39	49	Int
40	−66	INV x=t
41	51 4	GTO 4
42	34 5	SUM 5
43	33 1	RCL 1
44	34 5	SUM 5
45	51 4	GTO 4
46	86 5	Lbl 5
47	33 1	RCL 1
48	34 5	SUM 5
49	−61	INV SBR

Affichage		
Pas	Codes	Touches
00	01	1
01	32 0	STO 0
02	86 1	Lbl 1
03	01	1
04	34 0	SUM 0
05	61 3	SBR 3
06	33 0	RCL 0
07	32 7	STO 7
08	33 5	RCL 5
09	−66	INV x=t
10	51 1	GTO 1
11	81	R/S
12	51 1	GTO 1
13	86 3	Lbl 3
14	01	1
15	32 1	STO 1
16	32 5	STO 5
17	33 0	RCL 0
18	24	$\sqrt{x}$
19	32 3	STO 3
20	86 4	Lbl 4
21	01	1
22	34 1	SUM 1
23	33 1	RCL 1
24	32 7	STO 7

Affichage		
Pas	Codes	Touches
25	33 3	RCL 3
26	−76	INV x⩾t
27	51 6	GTO 6
28	66	x=t
29	51 5	GTO 5
30	33 0	RCL 0
31	45	÷
32	33 1	RCL 1
33	85	=
34	32 7	STO 7
35	49	Int
36	−66	INV x=t
37	51 4	GTO 4
38	34 5	SUM 5
39	33 1	RCL 1
40	34 5	SUM 5
41	51 4	GTO 4
42	86 5	Lbl 5
43	33 1	RCL 1
44	34 5	SUM 5
45	86 6	Lbl 6
46	−61	INV SBR

Registres	
R0	N
R1	DIV
R3	$\sqrt{N}$
R5	SdN

Passons à la solution de l'exercice n° 29. Il s'agissait d'écrire un programme qui affiche les entiers parfaits.

01 N ← 1
02 *
03 N ← N+1
04 Calcul de SdN
05 Si (SdN = N) Alors aller à 03 ; * N non parfait
06 *
07 Ecrire N ; * N est parfait
08 Aller à 03
09 Fin

Le calcul de SdN se fait comme dans le sous-programme de l'exercice n° 28, à ceci près qu'ici, on teste DIV par rapport à $\sqrt{N}$, ce qui est plus « sérieux ».

Le mode d'emploi est réduit à sa plus simple expression : on introduit le programme, puis on appuie sur RST et enfin sur R/S pour obtenir l'affichage des entiers parfaits.

Pour gagner en temps d'exécution, on peut inclure le sous-programme de calcul de SdN (pas 13 à 46) dans le programme principal. Même ainsi, il faudra attendre plusieurs heures avant d'obtenir le troisième entier parfait, qui est 496. Les trois premiers sont donc 6, 28 et 496.

L'exercice n° 30 permet d'en obtenir d'autres, et surtout plus rapidement.

Le programme principal est organisé de la façon suivante :
+ une première boucle de calcul engendre les entiers p impairs

(00-01) et 2^{p-1} (02-03) que nous noterons DP.
01 P ← 1 ; DP ← 1
02 *
03 P ← P+2 ; DP ← DP×4

Le programme appelle alors le sous-programme à l'étiquette 1 qui teste si p est premier (21 à 44).

Si p n'est pas premier, on sort du sous-programme par l'instruction RST (pas 38) qui vide la pile d'adresses de retour de sous-programme et rebranche le pointeur au pas 00 pour engendrer un nouveau p ;
04 Si (p non premier) Aller à 03
05 *

Affichage		
Pas	Codes	Touches
00	02	2
01	34 0	SUM 0
02	04	4
03	39 4	Prd 4
04	33 0	RCL 0
05	32 2	STO 2
06	61 1	SBR 1
07	33 4	RCL 4
08	55	x
09	02	2
10	65	–
11	01	1
12	85	=
13	32 2	STO 2
14	61 1	SBR 1
15	33 4	RCL 4
16	55	x
17	33 2	RCL 2
18	85	=
19	81	R/S
20	71	RST
21	86 1	Lbl 1
22	01	1
23	32 1	STO 1
24	33 2	RCL 2
25	24	$\sqrt{x}$
26	49	Int
27	32 2	STO 3

Mode d'emploi			
N°	Instructions ou données	Touches	Affichage
01	Passer en mode « programme »	LRN	
02	Introduire le programme		
03	Passer en mode « calcul »	LRN	
04	Initialiser p à 1	1 STO 0	1.
05	Initialiser DP à 1	1 STO 4	1.
06	Initialiser le pointeur	RST	1.
07	Afficher un entier parfait	R/S	N.
08	Pour afficher l'entier parfait suivant, aller en 07		

Si p est premier, on calcule 2^p-1 au pas 07 à 12 en utilisant le fait que $2^p-1 = 2\times DP-1$, et on teste s'il est premier.

06 M ← 2xDP – 1
07 Si (M non premier) Aller à 03
08 *
09 Ecrire MxDP
10 Aller à 03

+ Si $M = 2^p-1$ est premier, on fait afficher DP×M aux pas 15 à 19 et on recommence un nouveau calcul.

Le sous-programme qui teste un nombre pour « savoir » s'il est où non premier n'est autre que le programme de l'exercice n° 14 (OI N°), légèrement modifié pour l'adapter à notre problème.

Affichage		
Pas	Codes	Touches
28	86 2	Lbl 2
29	02	2
30	34 1	SUM 1
31	33 2	RCL 2
32	45	÷
33	33 1	RCL 1
34	85	=
35	32 7	STO 7
36	49	Int
37	66	x=t
38	71	RST
39	33 3	RCL 3
40	32 7	STO 7
41	33 1	RCL 1
42	–76	INV x ≥ t
43	51 2	GTO 2
44	–61	INV SBR

Les entiers 6 et 28 ne sont pas obtenus avec ce programme. On obtient :
496
8 128
33 550336

Registres	
R0	p
R1	DIV
R2	nombre testé N
R3	N
R4	$DP = 2^{p-1}$
R7	Tests

Les entiers parfaits sont donc plutôt rares. En allant chercher ses deux derniers chiffres dans la réserve du registre d'affichage, on obtient encore :
8 589 869 056

On pense que les entiers parfaits se terminent tous par 6 ou 8, mais ce n'est pas encore démontré. Une étape de plus et on obtient :
137 438 691 32.,

Le point étant mis à la place du dernier chiffre, 6 ou 8 que ne permet pas d'obtenir TI.57.

Le même programme, tournant sur une TI.58 a donné :
137 438 691 328

Ce qui confirme que le dernier chiffre est un 8.

Cet exercice termine donc cette série sur la DIVISION. Nous espérons que vous y avez trouvé de l'intérêt et des idées de nouveaux programmes.

Christophe Haro