2 Recherche des entiers premiers inférieurs ou égaux à un entier A fixé.

2.1 Une première idée :

L'entier 2 étant le seul entier pair premier, on peut limiter l'étude aux entiers impairs inférieurs ou égaux à A. Pour cela, il suffira de tester leur primalité avec le(s) sous-programme(s) NPREM1 vus dans la partie précédente.

Cela donne pour l'algorithme et les programmes :

Effacer l'écran
Demander A
Mettre {2} dans liste L
Pour I allant de 1 à $\frac{A-1}{2}$
Mettre 2*I + 1 dans N
Tester la primalité de N
Si N est premier
alors
Aficher N puis le mettre dans L
Fin du Si
Fin du Pour
afficher liste L

Programme APREM

TI-83

```
ClrHome
Prompt A
Disp 2 :Pause
{2}→L1
For(I,1,(A-1)/2)
2I+1→N
prgmNPREM1
If W=0
Then
augment(L1,{N})→L1
Disp N :Pause
End
End
Pause L1
```

Casio

```
ClrText
''A='' ?→A
2▲
{2}→List 1
For 1→I To (A-1)/2 Step 1
2I+1→N
Prog ''NPREM1''
If W=0
Then Augment(List 1,{N})→List 1
N▲

IfEnd
Next
List 1
```

TI-92

```
aprem(a)
Prgm
ClrIo
Disp 2 :Pause :{2}→l1
For i,1,(a-1)/2
2i+1→n
nprem1(n)
If w=0
Then
augment(l1,{n})→l1
Disp n :Pause
EndIf
EndFor
Pause l1
EndPrgm
```

Remarque 9 *Certaines Casio ne possédant pas la fonction augment*
Il faut alors créer un sous-programme faisant ce travail :

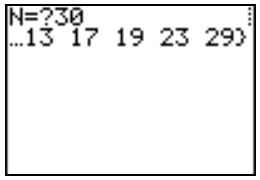
Sous-programme Augmen (Pour les Casio ne disposant pas de l'instruction Augment)

```
Dim List 1→U :List 1→List 2
U+1→Dim List 1
For 1→Z To U
List 2[Z]→List 1[Z]
Next
N→List 1[U+1]
```

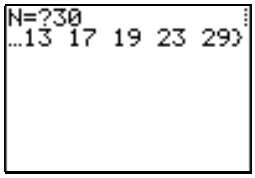
La ligne `Then Augment(List 1,{N})→List 1` *se transforme alors en* `Then Prog ''AUGMEN''`

Exemples d'écrans

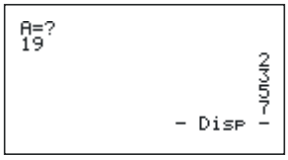
TI-83



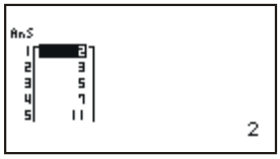
TI-83



Casio



Casio



2.2 Le crible d'ERATOSTHENE

On considère les entiers impairs inférieurs ou égaux à A. Soit pour notre exemple $A = 30$.

3	5	7	9	11	13	15	17	19	21	23	25	27	29

Au premier tour, on supprime les multiples de 3 autres que 3.
Il reste donc :

3	5	7	11	13	17	19	23	25	29

Au second tour, on élimine les multiples du deuxième terme de la liste restante différents de celui-ci.
Il reste donc :

3	**5**	7	11	13	17	19	23	29

On recommence, et ceci tant que la dimension de la liste des termes restants est strictement supérieure au rang du terme choisi.
On en déduit donc l'algorithme suivant :

```
Effacer l'écran. Demander A
Créer la liste L des entiers impairs inférieurs ou égaux à A
Mettre 1 dans I ( I représentera le rang du terme dont on veut éliminer les multiples)
Tant que I ≤ dimension de liste L
Mettre le terme de rang I dans X
Mettre I dans K
Pour J allant de I+1 à dim(L)
Si X ne divise pas L(J)
Alors mettre L(J) dans L(K+1) ( Ce terme étant à conserver)
Mettre K+1 dans K
Fin du Si
Fin du Pour
I augmente de 1
K va dans Dim(L) ( On a éliminé les multiples de L(I) = X)
Fin du tant Que
Ajouter 2 à la liste L
Afficher la liste L
```

Programmes : (ERATOS)

TI-83
```
ClrHome
Prompt A
seq(I,I,1,A,2)→L1
1→I
While I≤dim(L1)
L1(I)→X
I→K
For(J,I+1,dim(L1),1)
If fPart(L1(J)/X)≠0
Then
L1(J)→L1(K+1) :K+1→K
End
End
1+I→I :K→dim(L1)
End
augment({2},L1)→L1
Pause L1
```

Casio
```
ClrText
"A=" ?→A
Seq(I,I,1,A,2)→List 1
1→I
While I≤dim List 1
List 1[I]→X
I→K
For I+1→J To Dim List 1
If Frac (List 1[J]/X)≠0
Then List 1[J]→List 1[K+1]
K+1→K
IfEnd
Next
1+I→I :List 1→List 2 :K→Dim List 1
For 1→H To K
List 2[H]→List 1[H]
Next
WhileEnd
Augment({2},List 1)→List 1
List 1
```

TI-92
```
eratos(a)
Func
Local m,i,l1,x,k,j
seq(i,i,1,A,2)→l1
1→i
While i≤dim(l1)
l1[i]→x :i→k
For j,i+1,dim(l1)
If mod(l1[j],x)≠0 Then
l1[j]→l1[k+1] :k+1→k
EndIf
EndFor
i+1→i :mid(l1,1,k)→l1
EndWhile
augment({2},l1)→l1
EndFunc
```

On peut améliorer la performance de ce programme en remarquant que les nombres premiers impairs distincts de 3 sont de la forme $6k+1$ ou $6k-1$, k étant un entier naturel non nul. Cela donne initialement une liste plus courte. Seule la création de la liste L doit donc être modifiée ainsi que la ligne finale puisqu'il faudra réunir les listes L et $\{2,3\}$ dans L. La première partie de ce programme pouvant nous être utile plus tard, on peut en faire un sous-programme.

Sous-programme Saisie

TI-83
```
ClrHome
Prompt A
seq(I,I,7,A,6)→L1
seq(I,I,5,A,6)→L2
augment(L1,L2)→L1 :SortA(L1)
```

Casio
```
ClrText
"A=" ?→A
Seq(I,I,7,A,6)→List 1
Seq(I,I,5,A,6)→List 2
augment(List 1,List 2)→List 1
SortA(List 1)
```

TI-92
```
saisie(a)
Func
Local m,i,l1,l2,o
seq(i,i,7,a,6)→l1
seq(i,i,5,a,6)→l2
augment(l1,l2)→l1 :SortA(l1)
EndFunc
```

Le programme Eratos s'écrit alors :

Programme ERATOS1

TI-83
```
PrgmSAISIE
1→I
etc

End
augment({2,3},L1)→L1
Pause L1
```

Casio
```
Prog ''SAISIE''
1→I
etc

WhileEnd
Augment({2,3},List 1)→List 1
List 1
```

TI-92
```
eratos1(a)
prgm
saisie(a)→l1


augment({2,3},l1)→l1
pause l1
```

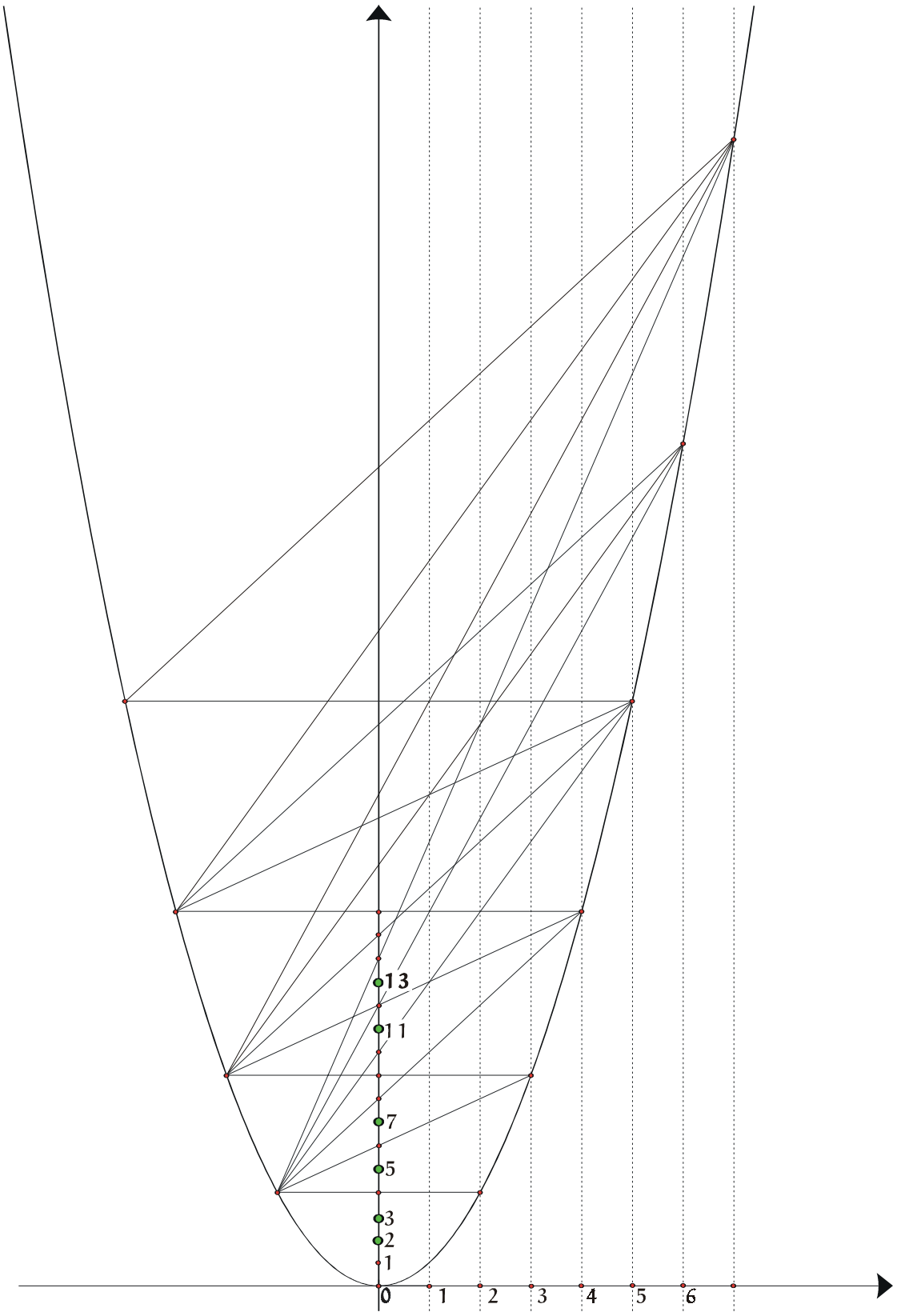
2.2.1 Interprétation géométrique du crible d'Eratosthène (cf Fig ??)

Le plan étant muni du repère orthonormé $\left(O, \vec{i}, \vec{j}\right)$. Considérons la parabole (P) d'équation $x = y^2$.
Quels que soient les entiers naturels, supérieurs ou égaux à deux, n et m, considérons la droite passant par les points A et

B de (P) d'abscisses respectives n^2 et m^2.

Exercice 10

1. *Donner une équation de la droite* (AB).
2. *Déterminer l'abscisse du point d'intersection de* (AB) *et de* (Ox).
3. *En déduire une trace sur le dessin précédent de l'ensemble des multiples de* 2 *et* 3 *inférieurs ou égaux à* 20
4. *En déduire alors une trace sur le dessin précédent des entiers premiers inférieurs ou égaux à* 20.



Interprétation géométrique du crible d'Eratosthène.

2.3 Une troisième idée

On peut reprendre la première idée mais en remarquant que pour tester la primalité de N, il suffit de tester les diviseurs premiers inférieurs ou égaux à $\sqrt{N}$ déjà trouvés (cela donne le programme ERATOS4).

TI-83
```
PrgmSAISIE
2→I :{2,3}→L2
While I≤Dim(L2)
L1(I)→X :0→W :2→J
While (L2(J))^2≤X
If Fpart (X/L2(J))=0
Then
1→W :1→X
End
1+J→J
End
1+I→I
If W=0
Then
Augment(L2,{X})→L2
End
End
Pause L2
```

Casio
```
Prog ''SAISIE''
2→I :{2,3}→List 2
While I≤Dim List 1
List 1[I]→X :0→W :2→J
While List 2[J]≤X
If Frac (X/List 2[J])=0
Then 1→W :1→X
IfEnd
1+J→J
WhileEnd
1+I→I
If W=0
Then Augment(List 2,{X})→List 2
IfEnd
WhileEnd
List 2
```

TI-92
```
eratos4(a)
Func
local
saisie(a)→l1
1→i :{2,3}→l2
While i≤dim(l1)
l1[i]→x :0→w :2→j
While (l2[j])^2≤x
If Fpart (x/l2[j])=0
Then
1→w :1→x
EndIf
1+j→j
EndWhile
1+i→i
If w=0 then
Augment(l2,{X})→l2
IfEnd :WhileEnd
l2
```

On pourrait de même rectifier le programme du crible d'Eratosthène en remarquant que le travail d'élimination est terminé dès que l'on recherche les multiples du premier entier strictement supérieur à $\sqrt{A}$. Il suffit de rectifier la ligne `While I≤dim(L1) en While I≤dim(L1) and L1(I)^2≤ A` (cela donne le programme ERATOS11)

Remarque 11 *Tester pour $A = 100$ et $A = 1000$. Comparer les temps de calcul avec les programmes précédents.*

2.4 Quatrième idée : Le crible de Sundaram (1934)

On considère le tableau suivant :

4	7	10	13	16	19	22	···
7	12	17	22	27	32	37	···
10	17	24	31	38	45	52	···
13	22	31	40	49			
16	27	38	49	60			
19	32	45					
22	37	54					
25	42						
⋮							

On reconnaîtra la suite arithmétique de raison 3 et de premier terme 4
On reconnaîtra la suite arithmétique de raison 5 et de premier terme 7
On reconnaîtra la suite arithmétique de raison 7 et de premier terme 10

On remarquera immédiatement que ce tableau est symétrique par rapport à la diagonale issue du sommet gauche du tableau.

Calculons le terme $U_{n,k}$ situé ligne n et colonne k ($n \geq 1$ et $k \geq 1$)
La colonne k est une suite arithmétique de premier terme $4 + 3(k-1)$ et de raison $1 + 2k$. On en déduit que :

$$U_{n,k} = 4 + 3(k-1) + (n-1) \times (1+2k) = n + k + 2n \times k$$

Montrons que si un entier N figure dans la table alors $2N + 1$ n'est pas premier.
Soit un entier N de la table, il existe donc deux entiers n et k supérieurs ou égaux à 1 tels que $N = U_{n,k}$.

On a donc :

$$\begin{aligned} 2N+1 &= 2U_{n,k}+1 \\ &= 2n+2k+4n\times k+1 \\ &= (2n+1)(2k+1) \end{aligned}$$

or $2n+1$ et $2k+1$ sont deux entiers strictement supérieurs à 1 d'où N est composé.

Remarque 12 *Pour tout entier k non nul, on a :*

$$2U_{k,k}+1 = (2k+1)(2k+1) = (2k+1)^2$$

Réciproquement, montrons que si N impair (>1) est absent de cette table alors $2N+1$ est premier.
Soit N un nombre impair non premier, on peut donc l'écrire sous la forme $N = A\times B$ avec A et B impairs strictement supérieurs à 1. D'où :

$$N = (2a+1)\times(2b+1) \text{ avec } a>1 \text{ et } b>1$$

De ce fait $N = 2U_{a,b}+1$ donc N provient de la table.

Conséquence : On obtient donc ainsi un moyen de savoir si un nombre impair est premier ou composé.
On peut déduire du critère précédent un algorithme permettant de trouver les nombres composés impairs inférieurs ou égaux à un entier donné M. Reste à l'écrire. On se propose de rechercher la liste des entiers composés inférieurs ou égaux à M.
Remarques préliminaires : Vu la symétrie du tableau, on travaillera avec les termes $U_{n,k}$ avec $k\leq n$. D'autre part, le travail sera terminé dès que $2U_{k,k+1}$ sera strictement supérieur à M.
Cela nous donne l'algorithme suivant :

Demander M : Mettre 1 dans K
Tant que $2U_{k,k+1}$ sera strictement inférieur à M
Mettre K dans N
Jusqu'à ce que $B>M$
Mettre $U_{n,k}$ dans A
Mettre $2A+1$ dans B (Traitement de la colonne K)
Si $B\leq M$ alors afficher B
Mettre $N+1$ dans N
Fin du jusqu'à ce que
Mettre $K+1$ dans K
Fin du tant que

Programmes (SUNDARA1) :

TI-83
```
ClrHome
Prompt M :1→K
While (2K+1)²≤M
K→N
Repeat B>M
N+K+2*K*N→A
2A+1→B
If B≤M
Then
Disp B :Pause
End
N+1→N
End
K+1→K
End
```

Casio
```
ClrText
''M='' ?→M :1→K
While (2K+1)²≤M
K→N
Do
N+K+2*K*N→A
2A+1→B
If B≤M
Then B ▲
IfEnd
N+1→N
LpWhile B≤M
K+1→K
WhileEnd
```

TI-92
```
sundara1(m)
Prgm
local k,n,a,b
ClrIo :1→k
While (2k+1)²≤m
k→n
Loop
n+k+2k*n→a
2a+1→b
If b≤m
Then
disp b :pause
EndIf :n+1→n
If b>m
Exit
EndLoop
k+1→k
EndWhile :EndPrgm
```

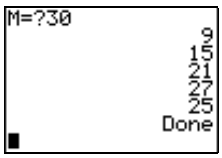
On peut améliorer sérieusement le programme précédent en utilisant les listes. Dans les programmes qui suivent, on fait apparaître la liste des nombres premiers et la liste des nombres composés impairs. Cela donne le programme SUNDARA2.

TI-83	Casio	TI-92

ClrHome
Prompt M
1→K :{1}→L_1 :{2}→L_2
While 4*K*(K+1)≤M
K→N
While (2N+1)*(2*K+1)≤M
(2N+1)*(2K+1)→B :0→R
For(I,1,dim(L_1)
If B=L_1(I)
Then
1→R :Dim(L_1)→I
End
End
If R=0
Then
augment(L_1,{B})→L_1
End
N+1→N
End
K+1→K
End
SortA(L_1)
Pause L_1: {2} →L_2
For(I,1,dim(L_1)-1,1)
(L_1(I)+1)/2→X :(L_1(I+1)-1)/2→Y
If X≤Y-1
augment(L_2,seq(2K+1,K,X,Y-1)→L_2
End
(L_1(I)+1)/2→X :M/2→Y
If X≤int(Y-1)
augment(L_2,seq(2K+1,K,X,int(Y-1))→L_2
Pause L_2
	ClrText
	''N=''?→M
	1→K
	{1}→List 1 :{2}→List2
	While 4*K*(K+1)≤M
	K→N
	While (2N+1)*(2K+1)≤M
	(2N+1)*(2K+1)→B :0→R
	For 1→I To Dim List 1
	If B=List 1[I]
	Then 1→R :Dim List 1→I
	IfEnd
	Next
	If R=0
	Then
	augment(List 1,{B})→List 1
	IfEnd
	N+1→N
	WhileEnd
	K+1→K
	WhileEnd
	SortA(List 1) :List 1▲
	For 1→I To Dim List 1-1
	(List 1[I]+1)/2→X
	(List 1[I+1]-1)/2→Y
	If X≤Y-1
	Then seq(2K+1,K,X,Y-1,1)→list 3
	augment(List 2,list 3)→List 2
	IfEnd
	Next
	List 1[dim(list 1)]+1)/2→X
	M/2→Y
	If X≤int(Y-1)
	Then
	seq(2K+1,K,X,int(Y-1),1)→list 3
	augment(List 2,list 3)→List 2
	IfEnd
	List 2
		prem(m)
		prgm
		ClrIo :local k,n,b,r,i
		1→k :{1}→l1 :{2}→l2
		While 4*k*(k+1)≤m
		k→n
		While (2n+1)*(2*k+1)≤m
		(2n+1)*(2k+1)→b :0→r
		For i,1,dim(l1)
		If B=l1[i] then
		1→r :Dim(l1)→i
		EndIf
		EndFor
		If r=0 then
		augment(l1,{B})→l1
		EndIf
		n+1→n
		EndWhile
		k+1→k
		EndWhile
		SortA(l1)
		Pause l1 :{2}→l2
		For i,1,dim(l1)-1,1
		(l1[i]+1)/2→x
		(l1[i+1]-1)/2→y
		If x≤y-1
		augment(l2,seq(2k+1,k,x,y-1)→l2
		EndFor
		(l1[i]+1)/2→x :m/2→y
		If x≤int(y-1)
		augment(l2,seq(2k+1,k,x,int(y-1))→l2
		Pause l2
		EndPrgm

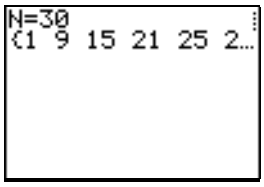
Exemples d'écrans :

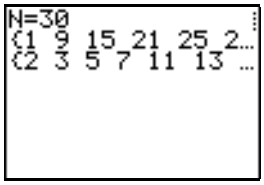
TI-83

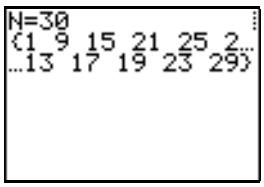


Casio

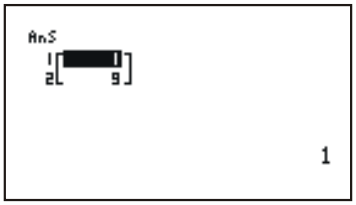
Seconde version (pour $M = 30$)

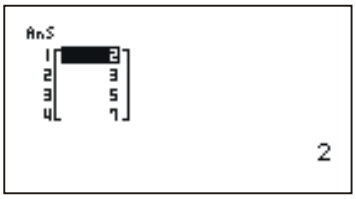






Idem pour $M = 9$





3 Spirale D'ULAM

On considère le tableau suivant :

						50
31	30	29	28	27	26	49
32	13	12	11	10	25	48
33	14	3	2	9	24	47
34	15	4	1	8	23	46
35	16	5	6	7	22	45
36	17	18	19	20	21	44
37	38	39	40	41	42	43

On encadre les nombres premiers, cela donne :

91	90	89	88	87	86	85	84	**83**	82
92	57	56	55	54	**53**	52	51	50	81
93	58	**31**	30	**29**	28	27	26	49	80
94	**59**	32	**13**	12	**11**	10	25	48	**79**
95	60	**33**	14	**3**	**2**	9	24	**47**	78
96	**61**	34	15	4	1	8	**23**	46	77
97	62	35	16	**5**	6	**7**	22	45	76
98	63	36	**17**	18	**19**	20	21	44	75
99	64	**37**	38	39	40	**41**	42	**43**	74
100	65	66	**67**	68	69	70	**71**	72	**73**
101	102	103	104	105	106	**107**	108	**109**	110

A partir de là, on peut faire certaines constatations. Il restera bien sûr à les démontrer :
On voit apparaître des lignes constituées uniquement de nombres composés :

1. Celle démarrant par $1, 9, 25, 49$ qui sont les nombres de la forme $(2n+1)^2$.
2. Celle démarrant par $15, 35, 63, 99$ qui sont les nombres de la forme $(n+1)(4n+5)$
3. Et celles que vous trouverez !

Programmation sur une TI-89 (l'adaptation aux autres machines est laissée aux soins du lecteur !)

Principe : Le pixel correspondant au point de coordonnées $(79, 38)$ représente le nombre 1. On noircit les pixels correspondant aux nombres premiers. Pour une meilleure lisibilité, on va de 2 pixels en deux pixels. Le dessin initial sera donc :

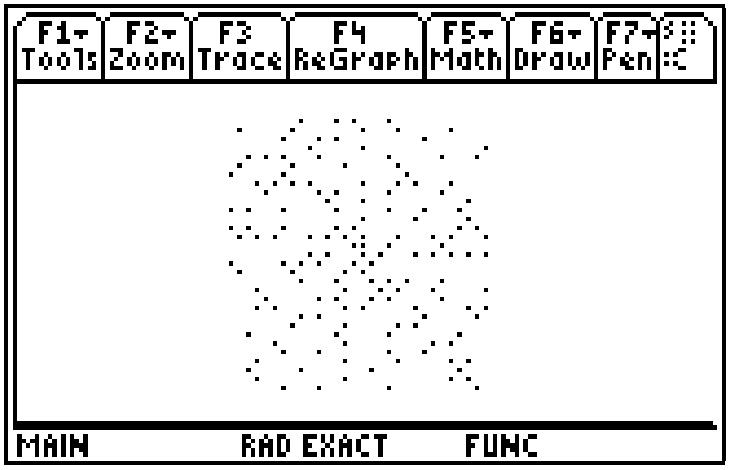
	noir(=11)	blanc
noir	noir=2	blanc
blanc	blanc=1	blanc
noir	blanc	noir

Algorithme	TI-89
c désigne le nombre d'enroulements désirés.	`ulam(c)`
	`prgm`
Effacer l'écran	`ClrDraw :1→d :{}→l1 :1→n`
Initialisations	`PtOn 79,38 :79→a :38→o`
Pour t allant de 1 à c (t=numéro de l'enroulement)	`For t,1,c,1`
Pour u allant de 1 à d	`For u,1,d`
augmenter n de 1 : Augmenter l'ordonnée de 2 (on va vers le haut)	`n+1→n :o+2→o`
Si n est premier alors	`If IsPrime(n)=true Then`
l'ajouter à la liste , allumer le point correspondant	`PtOn a,o :augment({n},l1)→l1`
Fin du Si :Fin du Pour	`EndIf :EndFor`
Pour v allant de 1 à d	`For v,1,d`
augmenter n de 1 : Diminuer l'abscisse de 2 (on va vers la gauche)	`n+1→n :a-2→a`
Si n est premier alors	`If IsPrime(n)=true Then`
..	`PtOn a,o :augment({n},l1)→l1`
..	`EndIf :EndFor`
Pour w allant de 1 à d+1	`For w,1,d+1`
augmenter n de 1 : Diminuer l'ordonnée de 2 (on va vers le bas)	`n+1→n :o-2→o`
	`If IsPrime(n)=true Then`
	`PtOn a,o :augment({n},l1)→l1`
	`EndIf :EndFor`
augmenter n de 1 : Augmenter l'abscisse de 2 (on va vers la droite)	`For x,1,d+1`
	`n+1→n :a+2→a`
	`If IsPrime(n)=true Then`
	`PtOn a,o :augment({n},l1)→l1`
	`EndIf :EndFor`
augmenter d de 2 (à chaque tour le nombre d'éléments sur une rangée augmente de 2)	`d+2→d`
Fin du pour	`EndFor`
Fin du programme	`EndPrgm`

En supposant qu'aucune fonction ou graphe statistique ne soit actif, on obtient le dessin suivant en ayant choisi pour fenêtre :

$$Xmin = 0 : Xmax = 158 : Xscl = 0 : Ymin = 0 : Ymax = 76 : Yscl = 0$$

Image obtenue avec c=11



Remarque 13 *a) On peut améliorer le programme précédent en utilisant un sous-programme qui teste la primalité et trace si nécessaire le point correspondant.*
b) On peut également décider de ne démarrer cette spirale qu'à partir d'un entier donné. Essayez, par exemple avec 41 *pour point de départ.*

4 Répartition des nombres premiers.

N désignant un entier naturel donné, on désigne par $p(N)$ le nombre d'entiers premiers inférieurs ou égaux à N. On se propose d'examiner le rapport $\frac{p(N)}{N}$.

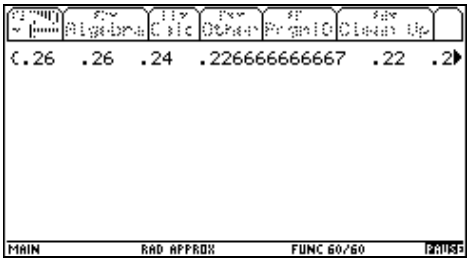
Pour $N = 50$, on constate que $p(50) = 0.26$. Pour la suite, on va créer un programme calculant $p(50k)$ pour tout entier k compris entre 1 et n.

Algorithme
n désigne le nombre de cinquantaines désirées
nb désigne le nombre de nombre premiers
pour n = 50, la proportion est de 0.26
Pour i allant de 1 à n
Si n est premier alors
nb augmente de 1
Fin du Si
Si on arrive à un multiple de 50 Alors
on stocke la proportion dans l1
Fin du Si
Fin du Pour
Afficher l1
Fin du programme

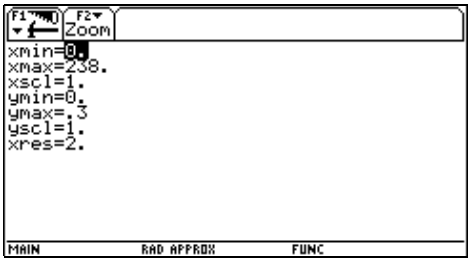
TI-92
`reparpre(n)`
`Prgm`
`ClrIO :13→nb :DelVar l1`
`{0.26}→l1`
`For i,51,50*n,2`
`If IsPrime(i)=true Then`
`nb+1→nb`
`EndIf`
`If mod(i-1,50)=0 Then`
`augment(l1,{nb/i})→l1`
`EndIf`
`EndFor`
`Disp l1`
`EndPrgm`

Exemples d'écrans obtenus :

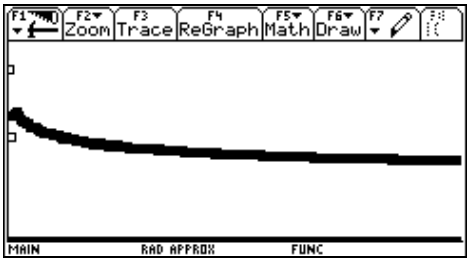
Premiers éléments de la liste l1



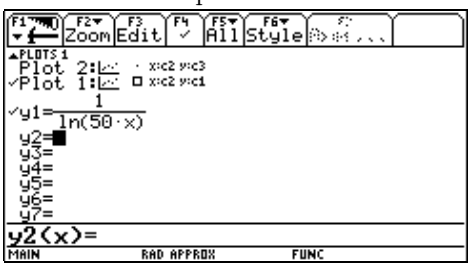
Ecran à choisir pour afficher le diagramme statistique



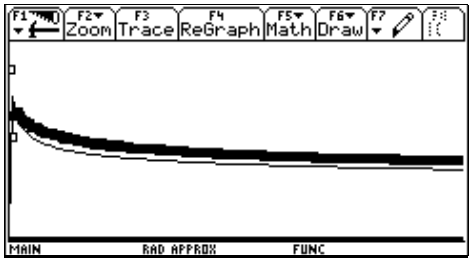
En abscisses 50k
(k compris entre 1 et 238), en ordonnées p(50k)



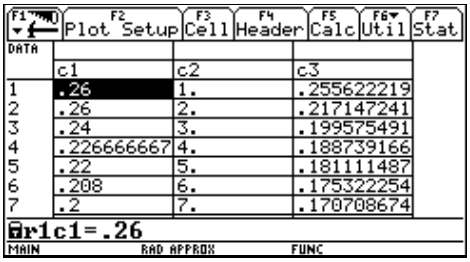
En y1, estimation de p(50x)
donnée par Hadamard

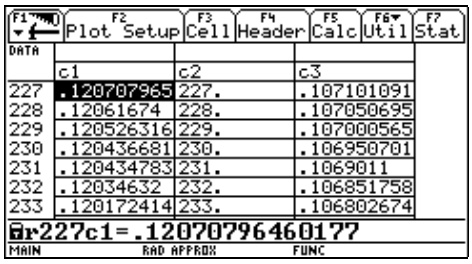


Comparaison graphique

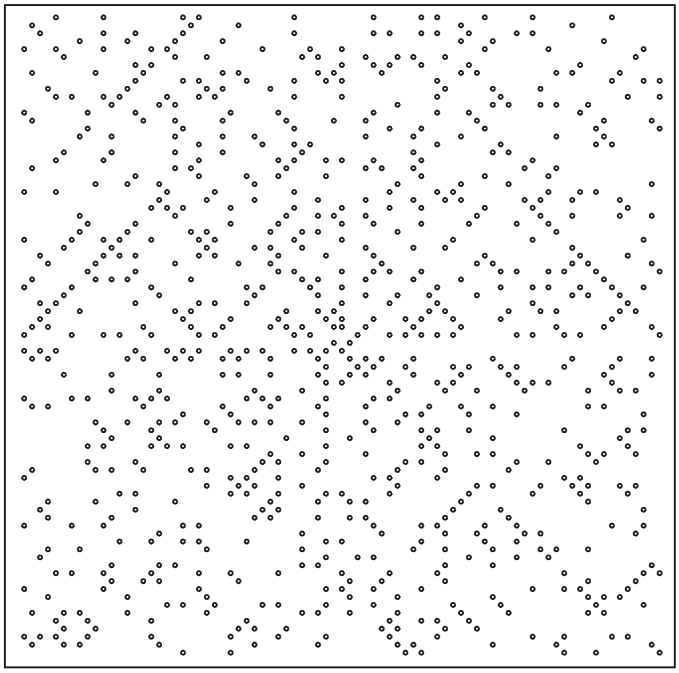


Comparaison numérique

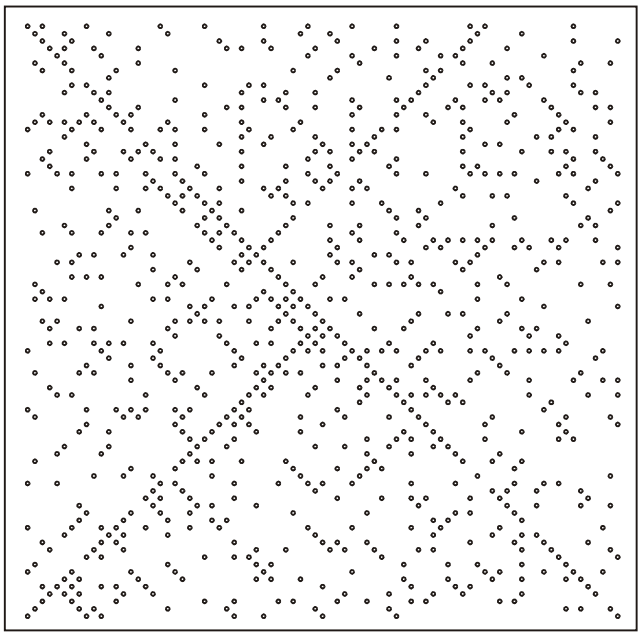




5 Figures pour la spirale d'Ulam



Spirale d'Ulam des 8000 premiers entiers



La même en commençant à l'entier 41

6 A propos des différences finies

Le dernier graphique présente la spirale d' Ulam lorsqu'elle commence à l'entier $n = 41$. Pourquoi avoir choisi cet entier. En fait, ce choix est inspiré d'une formule découverte par Euler qui fournit un grand nombre d'entiers premiers. Cette formule est la suivante

$$n^2 + n + 41$$

qui donne des entiers premiers pour n variant de 0 à 40.
Cette formule donne un prétexte à une digression sur les différences finies.

6.1 Position du problème

Considérons n nombres $a_0, a_1, ...a_i, ...a_{n-1}$ que l'on suppose classés par ordre croissant par exemple. A partir de ces n nombres, construisons la suite $(b_j)_{0\leq j\leq n-2}$ définie par :

$$\forall j \in [0, n-2],\ b_j = a_{j+1} - a_j$$

Examinons cette transformation lorsque la suite $(b_j)_{0\leq j\leq n-2}$ est une suite constante. Cela signifie, en fait, que la suite $(a_i)_{0\leq i\leq n-1}$ est une suite arithmétique de raison $b_0 = r$. On en déduit donc que

$$\forall i \in [0, n-1]$$

$$\begin{aligned} a_i &= a_0 + i \times b_0 \\ &= a_0 + i \times r \\ &= P(i) \text{ en posant } P(x) = a_0 + (a_1 - a_0)x \end{aligned}$$

On constate ainsi, que si la suite $(b_j)_{0\leq j\leq n-2}$ est une suite constante alors il existe un polynôme P de degré inférieur ou égal à 1 vérifiant

$$\forall i \in [0, n-1]\,,\ P(i) = a_i.$$

Dans le cas où la suite $(b_j)_{0\leq j\leq n-2}$ n'est pas une suite constante,on réitère le procédé. On construit alors la suite $(c_j)_{0\leq j\leq n-3}$ définie par

$$\forall j \in [0, n-2], c_j = b_{j+1} - b_j$$

La question que l'on se pose alors est la suivante

Lorsque la suite $(c_j)_{j\leq n-3}$ est constante, peut-on trouver un polynôme de degré inférieur à 2 tel que $P(i) = a_i$?

6.2 Etude du cas où $(c_j)_{j\leq n-3}$ est constante

On peut déjà appliquer la première partie et dire qu'il existe un polynôme P de la forme $P(x) = \alpha x + \beta$ tel que pour tout j de $[0, n-2]$

$$P(j) = b_j$$

Soit i un entier quelconque de l'intervalle $[0, n-1]$, on a

$$\begin{aligned} P(0) &= \beta = a_1 - a_0 \\ P(1) &= \alpha + \beta = a_2 - a_1 \\ P(3) &= \cdots \\ &\vdots \\ P(i-1) &= (i-1)\alpha + \beta = a_i - a_{i-1} \end{aligned}$$

ce qui donne en sommant

$$\begin{aligned} a_i &= \alpha\left(\sum_{k=1}^{i-1} k\right) + i \times \beta + a_0 \\ &= \alpha\left(\frac{i(i-1)}{2}\right) + i \times \beta + a_0 \\ &= Q(i) \end{aligned}$$

Où Q est le polynôme défini par

$$Q(x) = \frac{\alpha}{2}x^2 + (-\frac{\alpha}{2} + \beta)x + a_0$$

soit un polynôme de degré inférieur ou égal à 2 et ne dépendant pas de i.
En conclusion : si la suite $(c_j)_{j\leq n-3}$ est une suite constante alors, il existe un polynôme Q de degré au plus 2

$$\forall i \in [1, n],\ Q(i) = a_i$$

Remarque 14 *On peut généraliser ce résultat et entreprendre une étude complète des polynômes trouvés. Cela est une autre histoire qui conduit aux polynômes de Bernoulli.*

6.3 La formule d'Euler

Lors du tracé de la spirale d'Ulam à partir de 41, on voit apparaître la suite

$$a_0 = 41,\ a_1 = 43,\ a_2 = 47,\ a_3 = 53,\ a_4 = 61 \text{ et } a_5 = 71$$

Si l'on applique le procédé précèdent, on obtient

$$b_0 = 2,\ b_1 = 4,\ b_2 = 6,\ b_3 = 8 \text{ et } b_4 = 10$$

puis

$$c_0 = c_1 = c_2 = c_3 = 2$$

Il existe donc un polynôme Q de degré au plus 2 tel que

$$Q(0) = 41,\ Q(1) = 43,\ Q(2) = 47 \cdots$$

En appliquant ce qui précède, on trouve

$$P(x) = 2 + 2x$$

puis

$$Q(x) = x^2 + x + 41$$

Reste alors à trouver, quel est le premier entier n, pour lequel $Q(n)$ n'est pas premier.