1 art;maple_mode(0);cas_setup(0,0,0,1,0,1e-10,10,[1,50,0,25],0,0,0); //radians,pas de cmplx, pas de Sqrt

2 l:=%{1,4,5,6,7%};a:=rand(10):;

([1, 4, 5, 6, 7], Done)

3 l:=set[1,4,5,6,7];a:=rand(10):;

([1, 4, 5, 6, 7], Done)

4 member(a,l);

1

5 a; //verification.

1

6 maple_mode(1); // en mode maple

Warning: some commands like subs might change arguments order

7 ln(exp(1));ln(e); //ca n'est que la lettre e, pas le reel.

(1, ln(e))

8 maple_mode(0);

Warning: some commands like subs might change arguments order

9 evalf(e); //la doc fait reference au mode xcas.

2.718281828

10 evalf(exp(1));

2.718281828

11 ln(exp(1));

1

12 log(exp(1)); // les 2 marchent

1

13 \qu Comment obtenir un entier entre 0 et 20? Attention, que donne

Syntax compatibility mode xcas
Parse error line 1 at Comment

undef

14 \verb/rand(2^30);/ et \verb/rand(2^31);/

Syntax compatibility mode xcas
Parse error line 1 at /

(verb, undef && (verb), undef)
741678871 281437032.0

15 rand(2^30);rand(2^31); //il passe parfois en scientifique quand c'est trop grand?

(518467323 , 1852766699)

16 bete comme le crible d'eratostene sont en O(p), et bete n'use pas de memoire.
donc asymptotiquement, le crible n'est pas avantageux.

17 Prog Edit Add | 1 | nxt | OK (F9) | Save

```
bete:=proc(n)
j:=3;
a:=0;
SQ:=evalf(sqrt(n)); //pour ne le faire qu'une fois. ne pas le mettre dans le while!
while (j<SQ)
   { if (irem(n,j)) <>0 then j:=j+2;
      else a:=j;j:=n;
      fi ;
   };
if a<>0 then a; else n fi;
end;
```

// Warning: j,a,SQ, declared as global variable(s)
// End defining bete

```
(n)->
{ local NULL;
  j:=3;
  a:=0;
  SQ:=evalf(sqrt(n));
  while(j<SQ){
     if ((irem(n,j))!=0) {
        j:=j+2;
      }
      else {
        a:=j;
        j:=n;
      }; ;
   };
```

M

18 j:=4;N:=nextprime(10^j+5432)*nextprime(2*10^j+1234);

(4 , 328032433) M

19 bete(N);

15439 M

20 j:=5;N:=nextprime(10^j+5432)*nextprime(2*10^j+1234); //j=6 passe encore mais...

(5 , 21218879939) M

21 bete(N);

Evaluation time: 0.9

105437 M

22 Attention, pour que la suite r\'ecurrente soit bien programmee, il ne faut pas recalculer tous les termes jusque u_i ni u_2i a chaque etape!

23 Prog Edit Add | 1 | nxt | OK (F9) | Save

```
pollard := proc(n)
local x,y ;
x:=1; y:=x^2+1;
while (member ( igcd(y-x,n) , set[1,n] ))
   {
     x:=irem((x^2+1), n) ;
     y:=irem((y^2+1)^2+1, n) ;
    };
igcd(y-x,n);
end_proc;
```

// End defining pollard

```
(n)->
{ local x,y;
  x:=1;
  y:=x^2+1;
  while(member(igcd(y-x,n),set[1,n])){
     x:=irem(x^2+1,n);
```

M

24 N:=nextprime(10^6)*nextprime(2*10^6):;

Done

25 pollard(N);

1000003

26 N:=nextprime(10^8+5*rand(1000))*nextprime(2*10^8+11*rand(1000)):;

Done

27 pollard(N);

100001569

28 pollard est en O(sqrt(p)) ssi la fonction suivante est bornee.

29 Prog Edit Add | 1 | nxt | | OK (F9) | Save

```
comptep:=proc(n)
local x,y,j ;
x:=1; y:=x^2+1 ;j:=0;
while (member ( igcd(y-x,n) , %{1,n%} ))
  {
     x:=irem(x^2+1,n) ;
     y:=irem((y^2+1)^2+1, n) ;
     j:=j+1;
  }
evalf(j/sqrt(igcd(y-x,n)));
end_proc;
```

// Success
// End defining comptep

```
(n)->
{ local x,y,j;
  x:=1;
  y:=x^2+1;
  j:=0;
  while(member(igcd(y-x,n),set[1,n])){
```

30 On cr\'ee une liste de valeurs. Il ne faut pas des nombres trop petits pour que pollard ait de bonne chance d'aboutir. On fait imprimer a chaque etape pour voir s'il blocque a une etape.
On met des floor pour les grands rand car il bascule en flottant a partir de 2^31

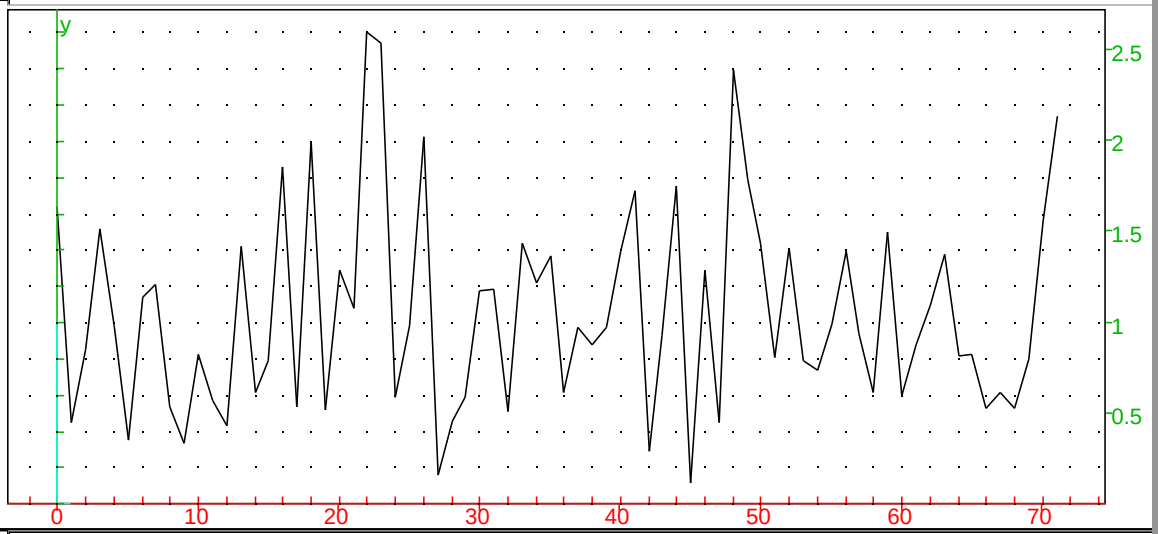
31 for j from 7 to 30 do

```
N:=nextprime(floor(alea(3^(j))))*nextprime(floor(alea(2^(j+1))));
t:=comptep(N);
afficher(j,t);
l:=append(l,t);
N:=nextprime(floor(alea(3^(j))))*nextprime(floor(alea(2^(j+1))));
t:=comptep(N);
afficher(j,t);
l:=append(l,t);
N:=nextprime(floor(alea(3^(j))))*nextprime(floor(alea(2^(j+1))));
t:=comptep(N);
afficher(j,t);
l:=append(l,t);
od;
```

26,0.6152805577
26,1.497623139
27,0.595442883
27,0.8776031913
27,1.09839073
28,1.379445264
28,0.8187177288
28,0.8248802713
29,0.527697553
29,0.6172450279
29,0.5302144649
30,0.8036918912
30,1.560217665
30,2.136468104
Evaluation time: 15.31

[[], [1.641105448 , 0.4527923591 , 0.8516583167 , 1.515985071 , 0.9837827088 , 0.3545937657 ,

32 plotlist(l); //ca a bien l'air borne



33 P:=i0->product(1-j/p,j=1..i0-1);

// Warning: j,p, declared as global variable(s)
// End defining P

$$i0 \;\; -> \;\; \Pi(1 - (\frac{j}{p}), j = (1 \;..\; (i0 - 1)))$$

M

34 i0:=2;limit(log(P(i0))/(i0*(i0-1)/2/p),p,+infinity);

(2 , -1)

M

35 On devine que P(i0) equivaut a: -i0*(i0-1)/(2p), on l'illustre ainsi:

36 l:=limit(log(P(2))/(i0*(i0-1)/2/p),p,+infinity);

-1

M

37 for i0 from 3 to 50 do l:=l,limit(log(P(i0))/(i0*(i0-1)/2/p),p,+infinity) od;

-1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1 , -1

M

38 Puisque ln est croissante, on se demande quand ln(P(i0))>ln(0.5). On compare donc -ln(0.5) avec i0^2/2p

39 sqrt(-2*ln(0.5)); //ca fait a peu pres le 1.18 de l'enonce

1.177410023

M

40 u_2i=u_i est mauvais pour cette suite recurent, c'est en O(p) comme la methode bete. En effet u_2i=a^i u_i +c(a^i-1)/(a-1) [p], donc u_2i=u_i ssi (a^i-1)(-ui+c/(a-1))=0[p]. Mais a^i=1[p] par ex si a generateur, alors i>=p-1

41

undef

M

42 illustration: on fait afficher le rapport entre le nombre de tours et sqrt(p).

43 a:=54321123;c:=nextprime(10^4);

(54321123 , 10007)

M

44

45 Prog Edit Add | 1 | nxt | OK (F9) | Save

```
comptelin:=proc(n,a,c)
local x,y ;
x:=1; y:=irem(a*x+c, n) ;j:=0;
while (member ( igcd(y-x,n) , %{1,n%} ))
   {
     x:=irem(a*x+c, n) ;
     y:=irem((a*(a*y+c)+c) , n) ;
     j:=j+1;
   };
evalf(j/sqrt(igcd(y-x,n)));
end proc;
```

```
// Warning: j, declared as global variable(s)
// End defining comptelin
```

```
(n,a,c)->
{ local x,y;
  x:=1;
  y:=irem(a*x+c,n);
  j:=0;
  while(member(igcd(y-x,n),set[1,n])){
```

M

46

```
l:=[];rand(3^4);
```

```
( [] , 69 )
```

M

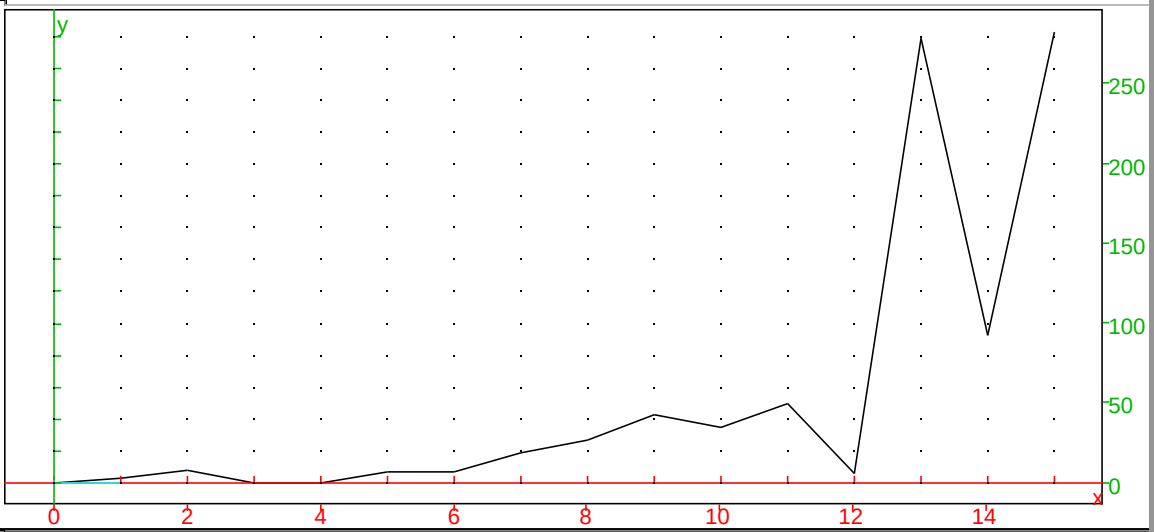
47

```
for i0 from 4 to 19 do
N:=nextprime(alea(3^i0))*nextprime(alea(2^(i0+1)));
t:=comptelin(N,a,c);
afficher(i0,t);
l:=append(l,t);
od:
```

```
6,7.941013883
7,0.2457695762
8,0.2560737599
9,7.258741748
10,7.345819086
11,19.24192072
12,26.70622584
13,43.20882014
14,34.73111668
15,49.93885229
16,6.478084885
17,278.2319177
18,93.00836295
19,282.5182393
Evaluation time: 14.97
```

[0.3015113446 , 3.049971407 , 7.941013883 , 0.2457695762 , 0.2560737599 , 7.258741748 , 7.3458

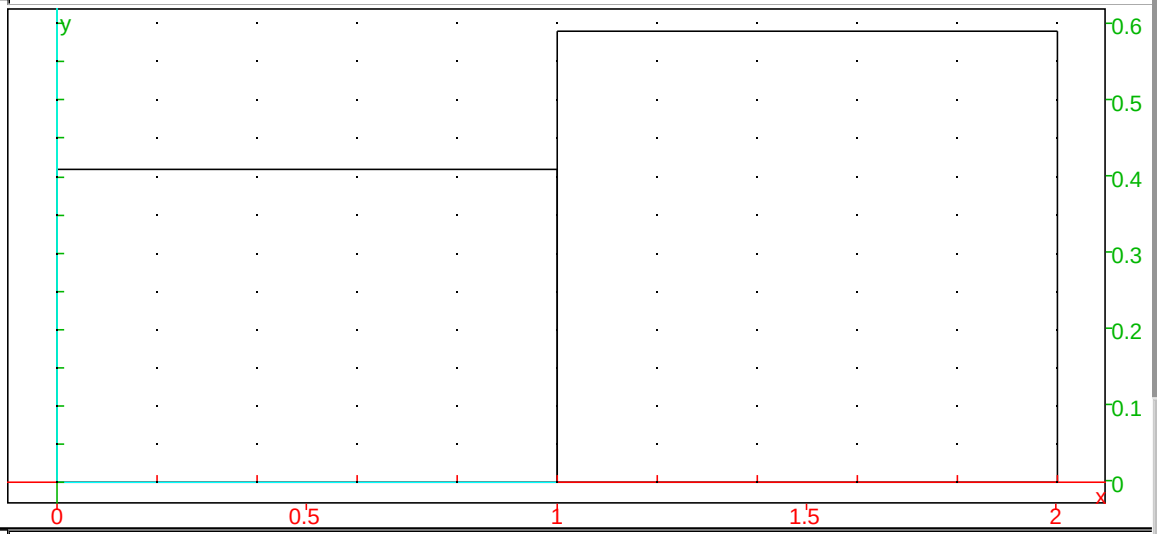
48 plotlist(l); // ca n'a plus l'air borne



49 l:=[seq(rand(2),j=1..100)]:;

Done

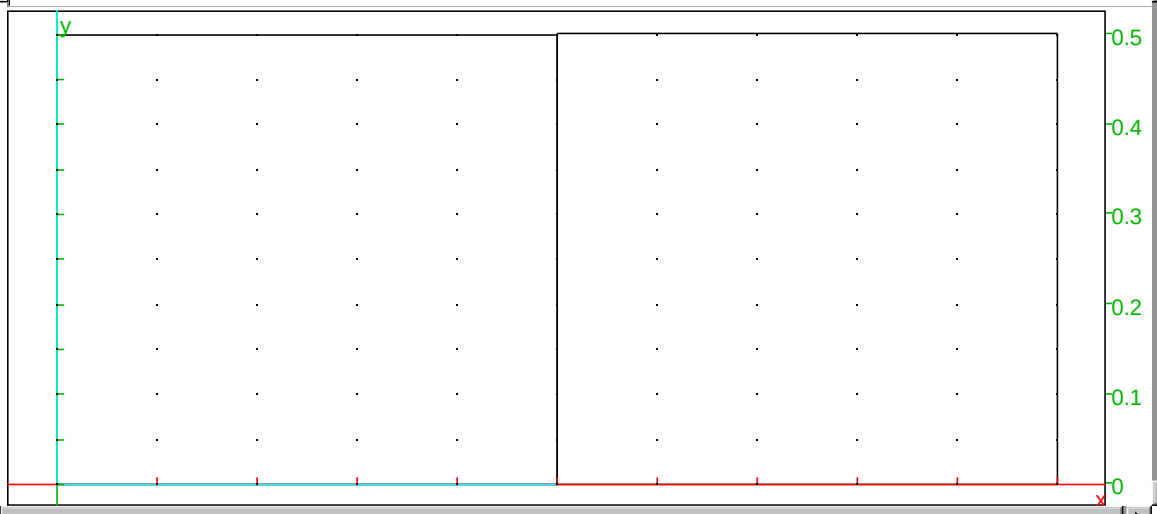
50 histogram(classes(l,0,1)); //On commence a 0, largeur constante 1.



51 l:=[seq(rand(2),j=1..1000)]:;

Done

52 histogram(classes(l,0,1));



53
```
N:=nextprime(10^6)*nextprime(2*10^6);
```

2000009000009

54
```
u:=n->if n=0 then 2 else irem(((u(n-1))^2+1) ,N) fi;
```

```
// Warning: u,N, declared as global variable(s)
// End defining u
u: recursive definition
```

```
                    if ((n==0)) {
                        2;
                    }
                    else {
                        irem((u(n-1))^2+1,N);
n    ->             };
```

55
```
u(10); // u(10000);Error, (in u) too many levels of recursion
```

1997309223146

56

57
```
local x,l;
x:=12345;l:=[];
for i0 from 1 to M do
    x:=irem((x^2+1), n) ;
    l:=[op(l),x];
    od ;
l;
end;
```

```
// Warning: i0, declared as global variable(s)
// End defining etudesuite
```

```
(n,M)->
{ local x,l;
  x:=12345;
  l:=[];
  for (i0:=1;i0<=M;i0:=i0+abs(1)) {
      x:=irem(x^2+1,n);
      l:=[op(l),x];
    };
  l;
}
```

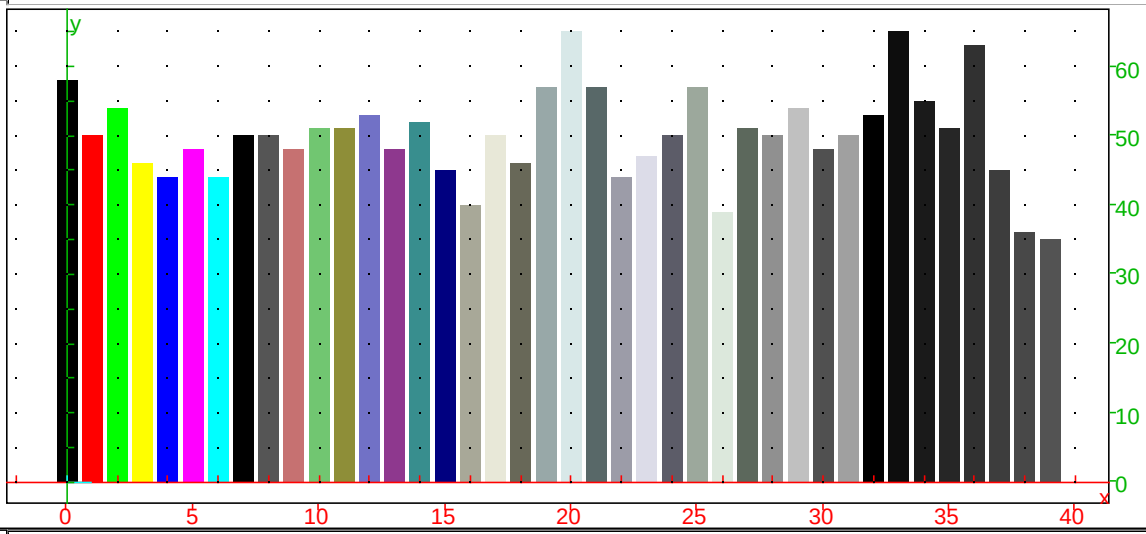
58
```
donnees:=(etudesuite(N,2000));
```

[152399026 , 1358617644169 , 1831711515025 , 190274858284 , 1218880810174 , 893513062655

59
```
cldonnees:=classes(donnees,0,N/40); //40 classes
```

1.50000675e+11	.. 2.000009e+11	,	46
2.000009e+11	.. 2.50001125e+11	,	44
2.50001125e+11	.. 3.0000135e+11	,	48
3.0000135e+11	.. 3.50001575e+11	,	44
3.50001575e+11	.. 4.000018e+11	,	50
4.000018e+11	.. 4.50002025e+11	,	50
4.50002025e+11	.. 5.0000225e+11	,	48
5.0000225e+11	.. 5.50002475e+11	,	51
5.50002475e+11	.. 6.000027e+11	,	51
6.000027e+11	.. 6.50002925e+11	,	53
6.50002925e+11	.. 7.0000315e+11	,	48
7.0000315e+11	.. 7.50003375e+11	,	52
7.50003375e+11	.. 8.000036e+11	,	45
8.000036e+11	.. 8.50003825e+11	,	40
8.50003825e+11	.. 9.0000405e+11	,	50
9.0000405e+11	.. 9.50004275e+11	,	46

60 diagramme_batons(cldonnees); //on peut cacher les noms dans le menu Cfg



61 ----------Test d'une variante de pollard----------
cette variante de pollard pour minimiser le nombre de igcd n'apporte pas d'amalioration
sous xcas, elle me semble meme un peu plus longue!!!!!!!

62 Prog Edit Add | 1 | nxt | OK (F9) | Save

```
pollard2 := proc(n)
local x,y,c,yy,xx,pp ;
xx:=1; yy:=xx^2+1 ;pp:=1;
while ((igcd(pp,n)==1))
  {
   x:=xx;y:=yy;
   for c from 1 to 20 do
     xx:=irem((xx^2+1) , n) ;
     yy:=irem((yy^2+1)^2+1, n) ;
     pp:=irem(pp*(xx-yy) , n);
   od ;
  }
  while ( igcd(y-x,n)==1 )
   {
     x:=irem(x^2+1 , n) ;
     y:=irem((y^2+1)^2+1 , n) ;
   }
     pp:=igcd(y-x,n);
if (pp<>n) then pp else afficher("pas trouve") fi;
end proc:;
```

63 N:=nextprime(10^6)*nextprime(2*10^6):;

Done

64 pollard2(N);

1000003

65 N:=nextprime(10^8+5*rand(1000))*nextprime(2*10^8+11*rand(1000)):;

Done

66 pollard2(N);

100004393

67 pollard(N);

100004393