```
1 ;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 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.
```

```
3 Prog  Edit  Ajouter | 1 | nxt | OK (F9) | Save
bete(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;
};
```

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

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

```
( 4 ,  328032433  )
```

```
5 bete(N);
```

```
15439
```

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

```
( 5 ,  21218879939  )
```

```
7 bete(N);
```

```
105437
```

```
8 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!
```

```
9 Prog  Edit  Ajouter | 1 | nxt | OK (F9) | Save
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) ;
      y:=irem((y^2+1)^2+1, n) ;
     };
 igcd(y-x,n);
};
```

```
(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);
```

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

```
Done
```

```
11 pollard(N);
```

```
1000003
```

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

13 pollard(N);

200008147

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

15 Prog Edit Ajouter | 1 | nxt | OK (F9) | Save

```
comptep(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)));
};
```

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

16 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

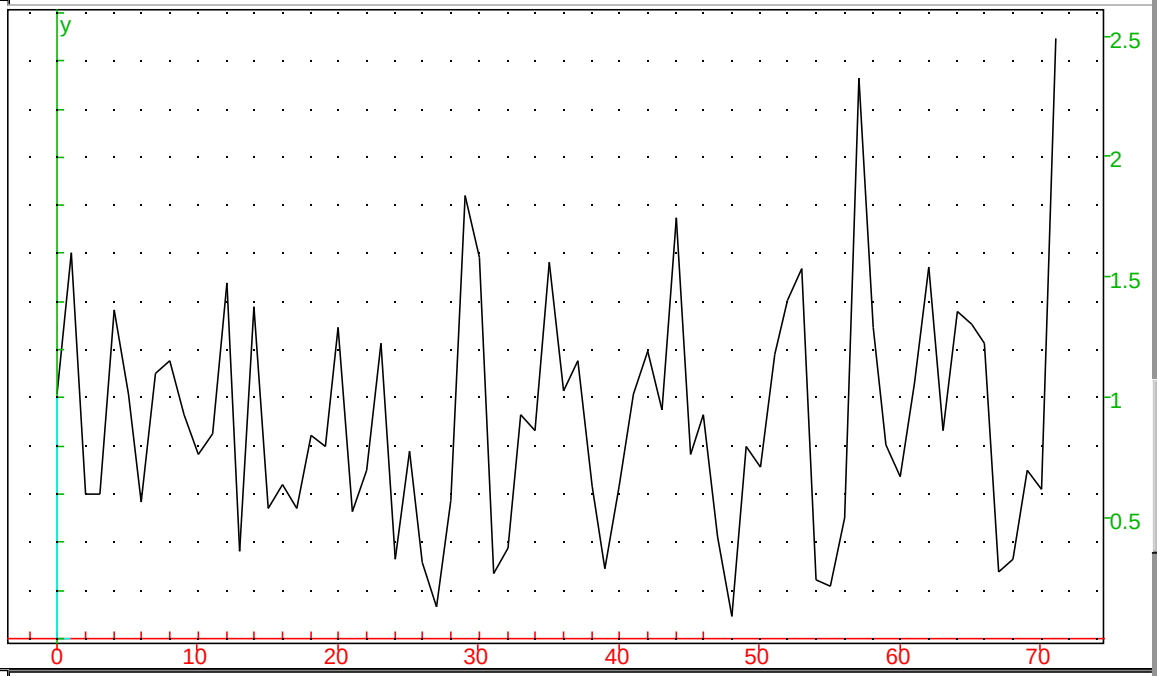
17

```
for(j:=1;j<31;j++){
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);
};
```

```
26,2.33009551141
26,1.29546823828
26,0.801739038209
27,0.674302659242
27,1.06272029207
27,1.54321996494
28,0.864073472319
28,1.35830043677
28,1.30734069559
29,1.22542837844
29,0.278104669224
29,0.328674270314
30,0.70139229939
30,0.623067826723
30,2.49257796647
Evaluation time: 4.34
```

[[] , [1.013004447 , 1.604514906 , 0.6026845772 , 0.6026845772 , 1.366972252 , 1.013004447 , 0.5

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



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

// Attention: j,p, declaree(s) comme variable(s) globale(s)
// End defining P

$$i0 \to \prod_{j=1}^{i0-1}\left(1-\frac{j}{p}\right)$$ M

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

(2 , -1) M

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

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

-1 M

23 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 M

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

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

1.177410023 M

26 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

27

undef M

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

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

(54321123 , 10007) M

31 Prog Edit Ajouter | 1 | nxt | | OK (F9) | Save

```
comptelin(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)));
};
```

```
(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

32

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

([] , 22)

M

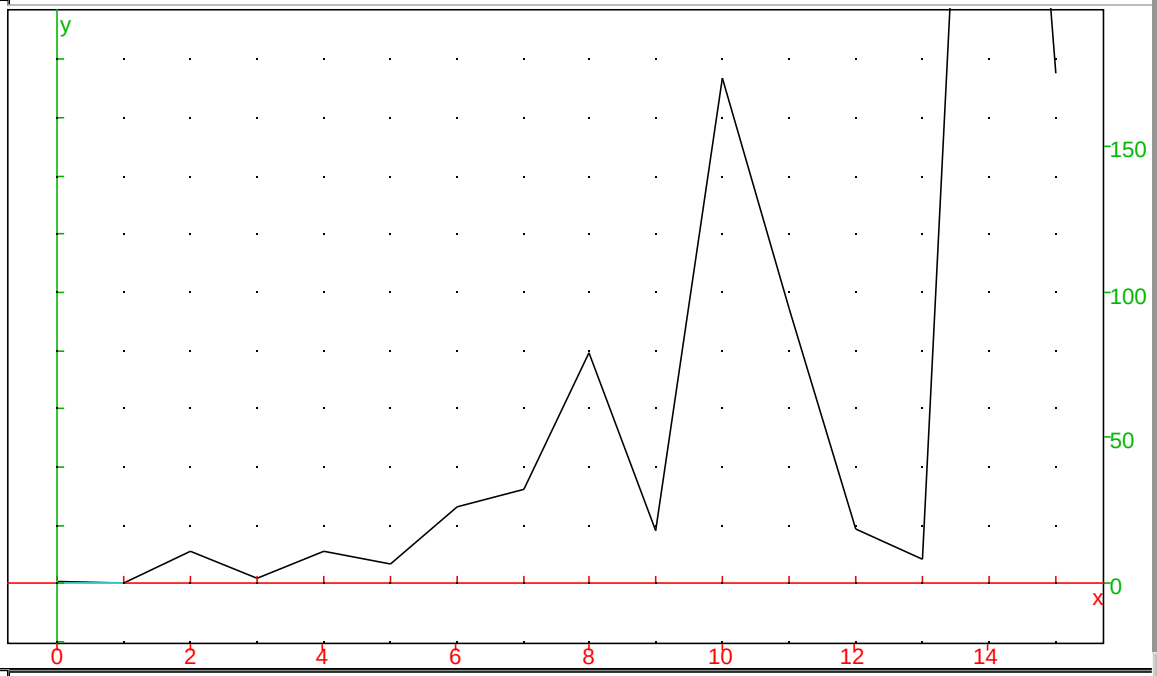
33

```
for(i0:=4;i0<20;i0++){
N:=nextprime(alea(3^i0))*nextprime(alea(2^(i0+1)));
t:=comptelin(N,a,c);
afficher(i0,t);
l:=append(l,t);
}:
```

```
5,0.256073759866
6,11.0919563677
7,1.95309230105
8,11.2707823309
9,6.57742881622
10,26.0577408176
11,32.0780902208
12,79.1517570118
13,18.0893013753
14,173.801611421
15,94.4258972056
16,18.8784419309
17,8.50335489107
18,467.776656128
19,175.3573138
Evaluation time: 4.68
```

[0.6246950476 , 0.2560737599 , 11.09195637 , 1.953092301 , 11.27078233 , 6.577428816 , 26.0577

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

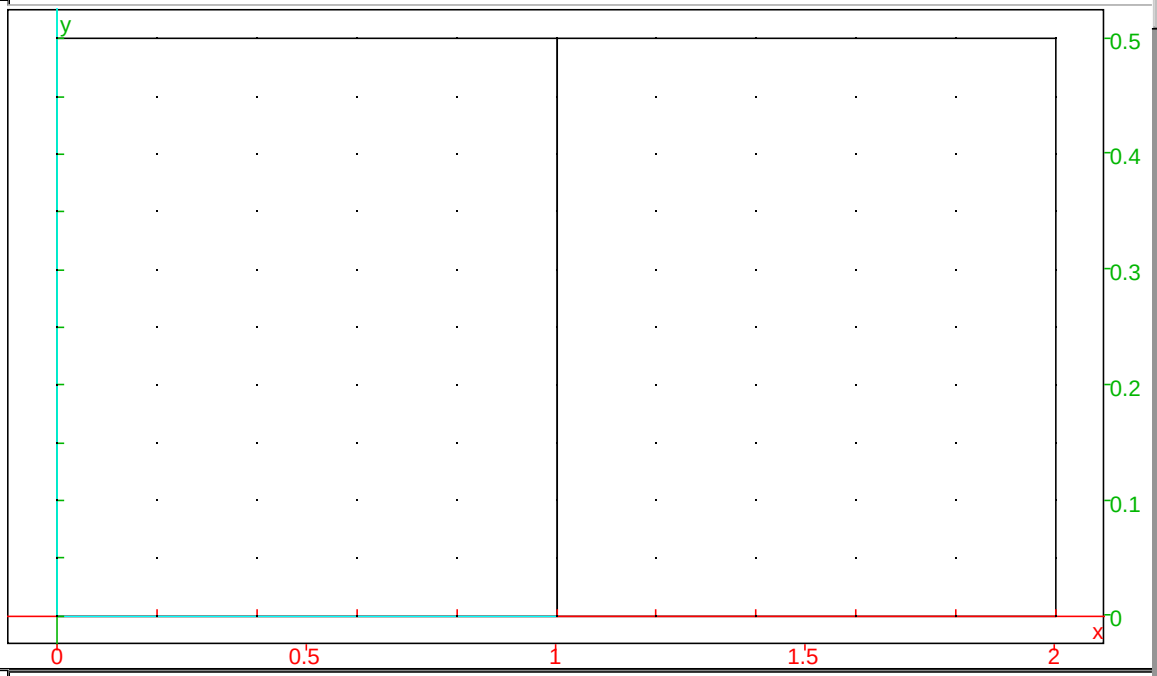


35 l:=[seq(rand(2),j=1..100)]:;

Done

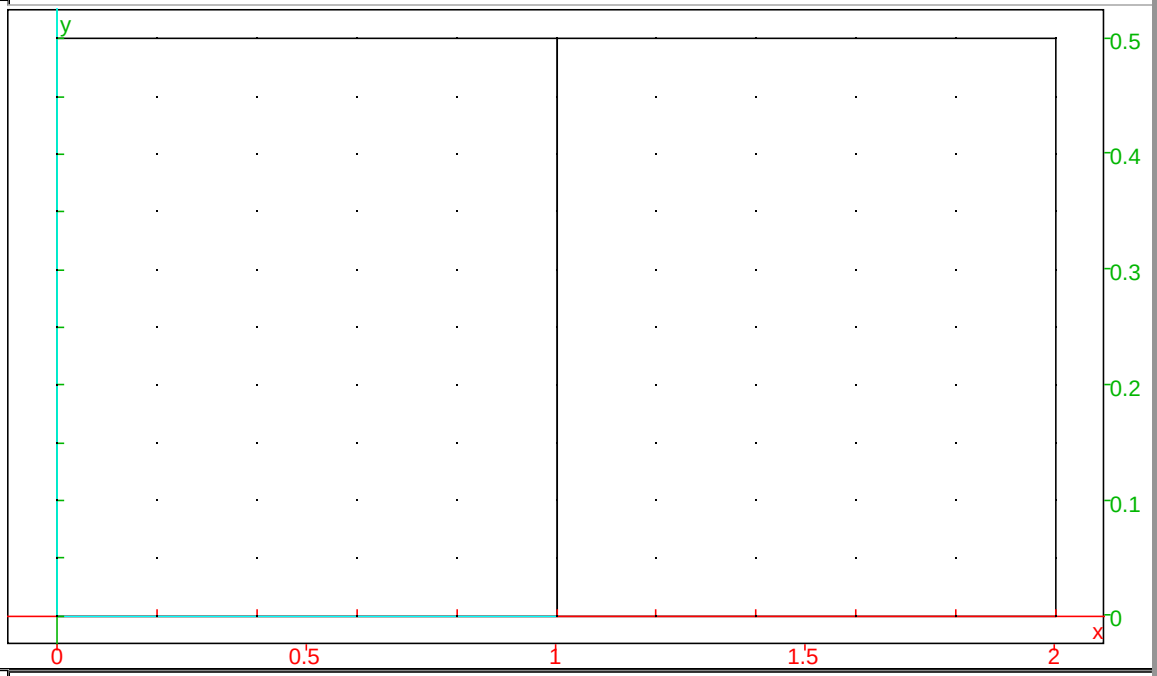
M

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



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

```
38 histogram(classes(l,0,1));
```




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

2000009000009 M

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

```
// Attention: u,N, declaree(s) comme variable(s) globale(s)
// End defining u
u: recursive definition
```

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

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

1997309223146 M

```
42
43 local x,l;
   x:=12345;l:=[];
   for(i0:=1;i0<=M;i0++){
       x:=irem((x^2+1), n) ;
       l:=[op(l),x];
       };
   l;
   };
```

```
// Interprete etudesuite
// Attention: i0, declaree(s) comme variable(s) globale(s) compiling etudesuite
```

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

M

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

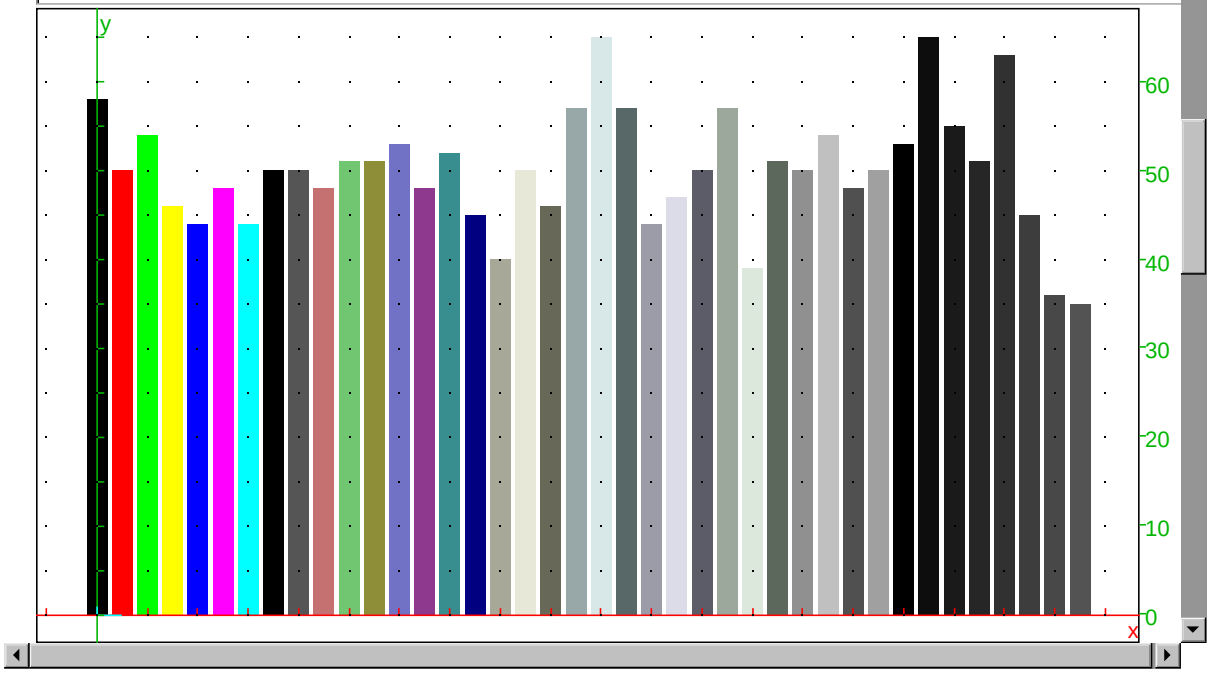
[152399026 , 1358617644169 , 1831711515025 , 190274858284 , 1218880810174 , 893513062655

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

5.0000225e+10	.. 1.0000045e+11	,	50
1.0000045e+11	.. 1.50000675e+11	,	54
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
9.50004275e+11	.. 1.0000045e+12	,	57

M

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



47
```
etudesuite2(n,M):={
local x,y,l;
x:=12345;l:=[];
for(i0:=1;i0<=M;i0++){
    x:=irem((x^2+1), n) ;
    y:=irem((y^2+1)^2+1, n);
    l:=[op(l),[x,y]];
    };
l;
```

```
// Interprete etudesuite2
// Attention: i0, declaree(s) comme variable(s) globale(s) compiling etudesuite2
```

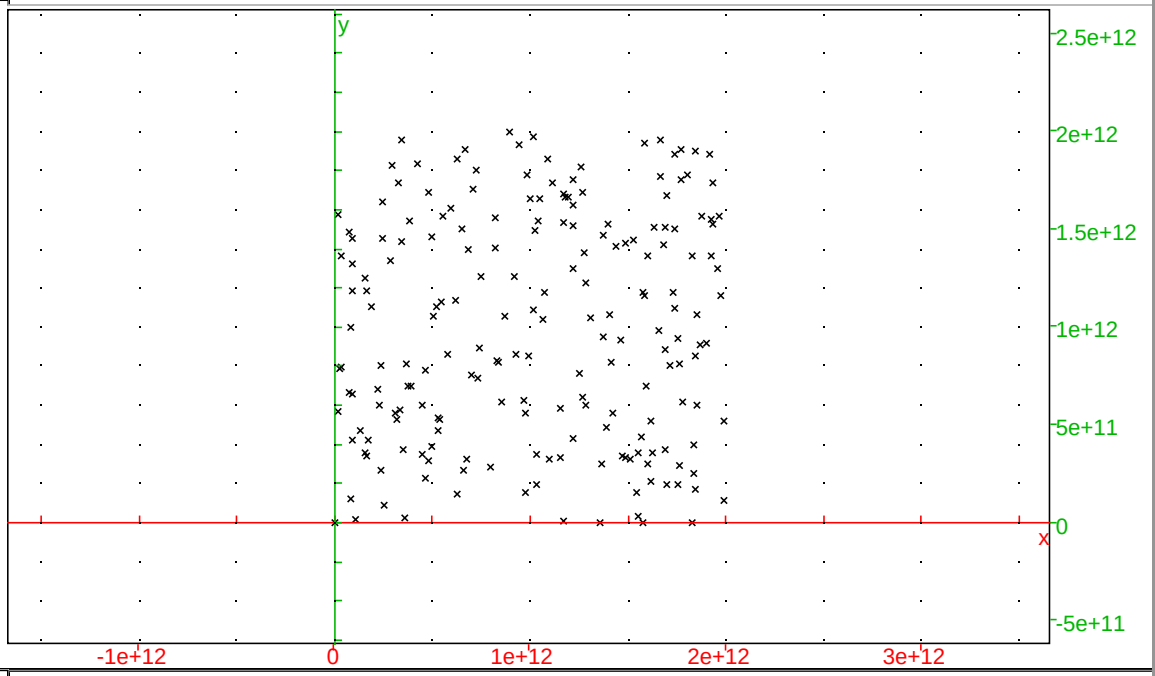
```
(n,M)->
{ local x,y,l;
  x:=12345;
  l:=[];
  for (i0:=1;i0<=M;i0++) {
     x:=irem(x^2+1,n);
     y:=irem((y^2+1)^2+1,n);
     l:=[op(l),[x,y]];
   };
  l;
}
```

M

48
```
couples:=(etudesuite2(N,200));
```

330544164034	,	173478610477
1157038865962	,	334379237733
498015076756	,	1458900729266
615817930864	,	1134762771112
833286708035	,	830583626248
423531302060	,	1833792431132
1026960438802	,	1494633196599
164146708858	,	1186977455442
698257570655	,	755965130392
344024644760	,	1434039102208
1805346551338	,	1778212124973
576380823433	,	862746417277
1694200791529	,	886015804412
1248381832471	,	760879913408
799139126097	,	288114878613
92286342622	,	1451750299031
1218662546699	,	1626307758286
1470946058285	,	337591628076

49 scatterplot(couples);



50 l'hypothese d'independance semble raisonable.

51 ----------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!!!!!!!

52 Prog Edit Ajouter | 1 | nxt | OK (F9) | Save

```
pollard2(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;
}:;
```

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

Done M

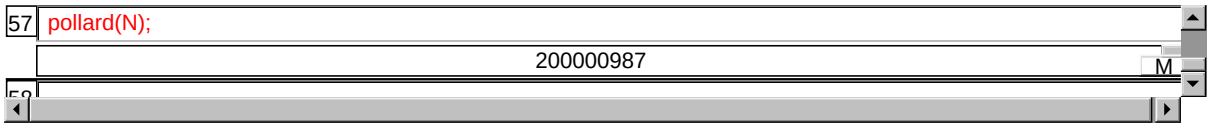
54 pollard2(N);

1000003 M

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

Done M

56 pollard2(N);


57
pollard(N);
200000987
M