

TP3 : Tests de primalité

N'oubliez pas d'exécuter (valider avec la touche Entrée) les commandes Maple (texte en rouge) avant de les utiliser.

On sait que si m est un nombre premier alors $a^{(m-1)} \bmod m = 1$ pour a entre 1 et $m-1$ (Fermat), donc si on trouve un a entre 1 et $m-1$ tel que $a^{(m-1)} \bmod m \neq 1$ on peut en déduire que m n'est pas un nombre premier. Quelle chance ce test a-t-il de réussir pour des a pris au hasard ? C'est ce que nous allons expérimenter.

Les fonctions ou opérateurs de Maple dont vous aurez besoin

Le très bon test de primalité de Maple. On lui fera confiance.

```
[ > isprime(561576002580391);  
                                     false  
[ > isprime(561576002580401);  
                                     true
```

Le calcul de puissance modulaire se fait avec l'opérateur $\&\wedge$ qui réduit par le modulo en cours de calcul de la puissance ce que ne fait pas l'opérateur usuel \wedge .

```
[ > 416545^456454 mod 4545445;  
Error, object too large  
[ > 416545&^456454 mod 4545445;  
                                     3296255
```

Test utilisant la relation de Fermat

Exercice 1 : On dit qu'un nombre m est a -pseudopremier, si m n'est pas un nombre premier mais $a^{(m-1)} \bmod m = 1$ et donc le test rate. Ecrire une fonction qui rende **true** si m est a -pseudopremier **false** sinon (vous pouvez utiliser **isprime**).

Solution

Une première solution :

```
[ > pseudoprem:=proc(m,a)  
    if isprime(m) then false  
    else if a&^(m-1) mod m = 1 then true else false fi fi  
end:
```

Quelques remarques :

- écrire `if isprime(m)=true` est correct mais lourd, puisque **isprime** rend un booléen,

- attention, certaines constantes de Maple s'écrivent en minuscules (**true**, **false**), d'autres en majuscules (**NULL**), d'autres un peu des deux (**Pi**),
- ne pas oublier que la puissance modulaire s'écrit $a \&^m$ et pas a^m ,
- il est tentant mais incorrect d'écrire $a \&^{(m-1)} = 1 \text{ mod } m$: le modulo est fait après le test.

Une deuxième solution plus concise utilisant le calcul booléen :

```
[ > pseudoprem:=proc(m,a) (not isprime(m)) and (a&^(m-1) mod m
= 1) end:
```

Exercice 2 : En utilisant cette fonction construire les listes des nombres 2-pseudopremiers, 3-pseudopremiers et 5-pseudopremiers inférieurs à 1200 (vous pouvez utiliser **select**). Que déduisez-vous de la comparaison de ces trois listes ?

Solution

On peut construire ces listes en faisant une boucle par exemple pour la liste des nombres 2-pseudopremiers inférieurs à 1200 :

```
[ > s:=NULL:
for i from 2 to 1200 do if pseudoprem(i,2) then s:=s,i fi
od:
L2pp:=[s];
```

$L2pp := [341, 561, 645, 1105]$

mais on peut aussi utiliser la fonction **select** : on commence par construire la liste des entiers de 2 à 1200 (qu'on ne veut pas voir)

```
[ > entiers:= [seq(i, i=2..1200) ]:
```

puis on lui applique la fonction **select**

```
[ > L2pp:=select(pseudoprem,entiers,2);
```

$L2pp := [341, 561, 645, 1105]$

select applique la fonction **pseudoprem** à la liste **entiers** avec comme second argument (constant) 2, soit **pseudoprem(i,2)** pour *i* allant de 2 à 1200, et retourne la liste des *i* pour lesquels **pseudoprem(i,2)** est vrai.

Remarquons qu'il n'y a pas beaucoup de nombres 2-pseudopremiers.

Donc de même la liste des nombres 3-pseudopremiers inférieurs à 1200 :

```
[ > L3pp:=select(pseudoprem,entiers,3);
```

$L3pp := [91, 121, 286, 671, 703, 949, 1105]$

et des nombres 5-pseudopremiers inférieurs à 1200 :

```
[ > L5pp:=select(pseudoprem,entiers,5);
```

$L5pp := [4, 124, 217, 561, 781]$

On remarque qu'aucun entier n'appartient aux trois listes.

On voit donc que ce test simple appliqué avec *a* valant 2, 3 et 5 donne les nombres premiers inférieurs à 1200 et uniquement eux.

Exercice 3 : Les nombres 561 et 1105 résistent mieux que les autres aux tests : ils apparaissent dans deux des trois listes. Montrer qu'ils vérifient la propriété suivante : $a^{(m-1)} \text{ mod } m = 1$ pour tout *a* premier avec *m* . Ce sont des nombres de Carmichael.

Solution

pour $m = 561$

On met une variable *test* à true. Si on trouve un *a* premier avec *m* tel que $a^{(m-1)} \bmod m \neq 1$ on met *test* à faux.

```
> test:=true:  
for a from 1 to 560 do if igcd(561,a)=1 and  
not(pseudoprem(561,a)) then test:= false fi od:  
test;
```

true

pour $m = 1105$

De même pour 1105

```
> test:=true:  
for a from 1 to 1104 do if igcd(1105,a)=1 and  
not(pseudoprem(1105,a)) then test:= false fi od:  
test;
```

true

Exercice 4 : En prenant *a* au hasard entre 1 et $m - 1$ quelle chance a-t-on de détecter que 561 n'est pas premier ? Même question pour 1105.

Solution

pour $m = 561$

On a 560 choix pour *a* et $\phi(561)$ nombres *a* premiers avec *m* pour lesquels le test rate.

```
> with(numtheory):  
Warning, new definition for order  
> phi(561);
```

320

```
> evalf((560-320)/560);
```

0.4285714286

Donc moins d'une chance sur deux.

pour $m = 1105$

De même on a 1104 choix pour *a* et $\phi(1105)$ nombres *a* premiers avec *m* pour lesquels le test rate.

```
> phi(1105);
```

768

```
> evalf((1104-768)/1104);
```

0.3043478261

Donc moins d'une chance sur trois.

Nous allons améliorer ce test avec la méthode de Miller vue en cours : on veut tester si *m* (impair) est un nombre premier. On pose $m - 1 = 2^s t$ avec *t* impair. On calcule la suite

$a_0 = a^t \bmod m, a_1 = a^{(2t)} \bmod m, \dots, a_s = a^{(2^s t)} \bmod m$. Si $a_s \neq 1$ ou s'il existe un *i* tel que $a_i = 1$ et

$a_{i-1} \neq 1$ et $a_{i-1} \neq -1$ alors m n'est pas un nombre premier.

– Test amélioré (Miller)

Exercice 5 : Ecrire une fonction qui rende **false** si m satisfait ce test pour le nombre a , **true** sinon.

– Solution

On commence par calculer la décomposition $m-1 = 2^s t$ puis la suite des a_i qu'on range dans un tableau (`array`) car on connaît le nombre d'éléments, l'accès est plus rapide et les indices collent à la notation mathématique. On parcourt la suite des a_i en ordre décroissant puis on regarde si l'on s'est arrêté sur un élément qui vaut -1 modulo m , c'est à dire $m-1$, ou sur a_0 et qu'il vaut 1.

```
> testfort:=proc(m,a)
  local s,t,suite,i;
  t:=m-1;s:=0;while irem(t,2)=0 do t:=iquo(t,2);s:=s+1 od;
  suite:=array(0..s);suite[0]:=a &^t mod m;
  for i from 1 to s do suite[i]:=suite[i-1]&^2 mod m od;
  if suite[s]<>1 then false else
    i:=s-1;
    while suite[i]=1 and i<>0 do i:=i-1 od;
    suite[i]=m-1 or (i=0 and suite[0]=1)
  fi
end:
```

Une autre version du test qui ne construit pas forcément toute la suite :

- soit $a_0 = 1$ ou $a_0 = -1$ et ce n'est pas la peine de calculer la suite qui vaudra tout le temps 1, on pense que m est premier mais ça n'est pas sur,
- soit il existe un i entre 1 et $s-1$ tel que $a_i = -1$, on ne calcule pas les éléments suivants, on pense aussi que m est premier mais ça n'est pas sur,
- soit il existe un i entre 1 et $s-1$ tel que $a_i = 1$ alors qu'on n'a pas eu -1 auparavant, on ne calcule pas les éléments suivants, on est sur que m n'est pas premier
- sinon on est sur que m n'est pas premier.

```
> testfort2:=proc(m,a)
  local s,t,ai,i;
  t:=m-1;s:=0;while irem(t,2)=0 do t:=iquo(t,2);s:=s+1 od;
  ai:=a &^t mod m;
  if ai=1 or ai=m-1 then true
  else
    for i from 1 to s-1 do
      ai:=ai&^2 mod m;
      if ai=m-1 then RETURN(true) elif ai=1 then RETURN(false)
    fi od;
  false fi
end:
```

Exercice 6 : En utilisant cette fonction construire comme dans l'exercice 2 des listes de nombres qui résistent à ce test (vous pouvez agrandir l'intervalle testé).

Solution

Comme le test amélioré ne s'applique qu'aux nombres impairs (connaissez-vous beaucoup de nombres premiers pairs ?) on commence par construire une liste des entiers impairs jusqu'à 1200 :

```
[ > entiersimpairs:=select(type,entiers,odd):
```

De cette liste on extrait la liste des nombres premiers pour comparaison :

```
> lprem:=select(isprime,entiersimpairs);
```

```
lprem := [3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79,
83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173,
179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269,
271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373,
379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467,
479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593,
599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691,
701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821,
823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937,
941, 947, 953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033,
1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123,
1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193]
```

Puis on calcule la liste des nombres qui satisfont le test amélioré pour $a = 2$. On en profite pour regarder les performances des deux versions programmées :

```
> t:=time():l2:=select(testfort,entiersimpairs,2):time()-t;
t:=time():l22:=select(testfort2,entiersimpairs,2):time()-t;
;
```

0.030

0.016

On constate que la fonction `testfort2` a de meilleures performances, c'est elle qu'on utilisera désormais. On vérifie que les deux fonctions donnent bien le même résultat :

```
> l2;l22;
```

```
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,
101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181,
191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277,
281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383,
389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487,
491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601,
607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709,
719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827,
829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049,
```

1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193]

[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193]

On fait les mêmes calculs pour $a=3$ et pour $a=5$:

```
> l3:=select(testfort2,entiersimpairs,3);l5:=select(testfort2,entiersimpairs,5);
```

$l3 := [5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 121, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 703, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193]$

$l5 := [3, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 781, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997, 1009, 1013, 1019, 1021, 1031, 1033,$

1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109, 1117, 1123,
1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193]

On compare ces listes à la liste des nombres premiers. Pour cela on utilise la fonction `minus` qui fait la différence de deux ensembles :

```
> convert(lprem,set) minus convert(l2,set);  
convert(lprem,set) minus convert(l3,set);  
convert(lprem,set) minus convert(l5,set);  
  
{}  
{3}  
{5}
```

Evidemment on détecte tous les nombres premiers sauf le nombre premier $m = a$.
A-t-on d'autres nombres que les nombres premiers ?

```
> convert(l2,set) minus convert(lprem,set);  
convert(l3,set) minus convert(lprem,set);  
convert(l5,set) minus convert(lprem,set);  
  
{}  
{121, 703}  
{781}
```

Pour $a = 2$ on n'a que les nombres premiers, pour $a = 3$ et pour $a = 5$ on récupère quelques nombres non premiers mais beaucoup moins que dans l'exercice 2.

Exercice 7 : Les nombres 561 et 1105 résistent-ils à ce test ?

Solution

Non, ni pour $a = 2$, ni pour $a = 3$, ni pour $a = 5$.

On va estimer l'amélioration pour les nombres 561 et 1105 qu'on avait étudiés dans les exercices 3 et 4 : on compte le nombre de a compris entre 2 et $m - 1$ qui permettent de s'apercevoir que m n'est pas un nombre premier.

pour $m = 561$

```
> c561:=0:for a from 2 to 560 do if not(testfort2(561,a))  
then c561:=c561+1 fi od:c561;  
  
550
```

Ou encore si on mesure le nombre de réussites sur le nombre de cas possibles :

```
> evalf(c561/559);  
  
0.9838998211
```

On est passé de moins d'une chance sur deux à plus de 98 chances sur 100.
L'amélioration est notable !

pour $m = 1105$

```
> c1105:=0:for a from 2 to 1104 do if  
not(testfort2(1105,a)) then c1105:=c1105+1 fi od:c1105;  
  
1074
```

Ou encore si on mesure le nombre de réussites sur le nombre de cas possibles :

```
[ > evalf(c1105/1103);
```

```
0.9737080689
```

On est passé de moins d'une chance sur trois à plus de 97 chances sur 100.
L'amélioration est notable !