DS d'informatique
Nombres premiers, nombres composés

Correction

28 mai 2011

1 Tests de composition

1. a)
```
Puissance := proc(a,q,N)
  local m,i:
  m := 1:
  for i from 1 to q do
    m := m*a mod N
  end do:
return m end;
```
Une solution récursive est également possible ; on peut aussi utiliser l'exponentiation rapide (et c'est mieux).

   b)
```
TestFermat := proc(a,N)
  return Puissance(a,N-1,N)<>1
end;
```

2. a) On a $13^{\frac{561-1}{2}} = 13^{280} = 1 \pmod{561}$. Calculons le symbole $J(13, 561)$.

   D'après la propriété vii), comme 13 est impair, on a $J(13, 561) = (-1)^{\frac{(13-1)(561-1)}{4}} J(561, 13)$. Comme 12 est divisible par 4 et 560 par 2, la puissance totale de -1 est paire et on a donc $J(13, 561) = J(561, 13)$.

   On utilise maintenant la propriété i) pour réduire 561 : $561 = 43 \times 13 + 2$ par division euclidienne, et donc $J(561, 13) = J(561 - 43 \times 13, 13) = J(2, 13)$.

   La propriété vi) donne $J(2, 13) = (-1)^{\frac{13^2-1}{8}}$. On calcule la puissance : on a $13 = -3 \pmod{16}$ d'où l'égalité $13^2 - 1 = 8 \pmod{16}$, et on a donc $\frac{13^2-1}{8} = 1 \pmod 2$ ce qui donne la valeur $J(13, 561) = J(2, 13) = -1$.

   Le nombre 561 est donc détecté comme composé par le test (bien entendu il est évident que 561 est divisible par 3, mais le test devient réellement intéressant pour des grands entiers).

   b)
```
Jacobi := proc(M,N)
  if M<0 or M>=N then
    return Jacobi(M mod N,N)                  #propriété i)
  elif M=0 then return 0                      #propriété iii)
  elif M=1 then return 1                      #propriété iv)
  elif M mod 2 = 0 then
    return (-1)^((N^2-1)/8)*Jacobi(M/2,N)     #propriétés ii) et vi)
  else                          #ici on sait que 1<M<N et M est impair
    return (-1)^((M-1)*(N-1)/4)*Jacobi(N,M)   #propriété vii)
end;
```
   On s'assure que les hypothèses des propriétés sont vérifiées quand on les utilise. Comme dans l'algorithme d'Euclide pour le PGCD, on sait que la procédure va s'arrêter car dans tous les cas de récursivité, ou bien N est constant et M diminue strictement (premier et quatrième cas), ou bien N diminue strictement (dernier cas).

   c)
```
TestSolovayStrassen := proc(a,N)
  return Puissance(a,(N-1)/2,N)<>(Jacobi(a,N) mod N)
end;
```

2 Tables de nombres premiers

1. a)
```
EstPremier := proc(N)
local d:
for d from 2 to N-1 do
  if N mod d = 0 then
    return false
  end if
end do:
return true
end;
```
   b)
```
EstPremier2 := proc(N)
local d:
for d from 2 to floor(sqrt(N)) do
  if N mod d = 0 then
    return false
  end if
end do:
return true
end;
```
   c)
```
TablePremiers := proc(B)
local N,Sp:
Sp := NULL:
for N from 2 to B do
  if EstPremier2(N) then
    Sp := Sp,N
  end if
end do:
return [Sp]
end;
```
2.
```
TablePremiers2 := proc(B)
local raye,N,Sp,m:
raye := [seq(false, N=1..B)]:
Sp := NULL:
for N from 2 to B do
  if not raye[N] then
    Sp := Sp,N:
    for m from 2*N to B by N do
      raye[m] := true
    end do
  end if
end do:
return [Sp]
end;
```
3. La procédure `EstPremier2` effectue $O(\sqrt{N})$ opérations (taille de la boucle).On en déduit que la procédure `TablePremiers` utilise $\sum_{N=2}^{B} O(\sqrt{N}) = O(B^{\frac{3}{2}})$ opérations.

   Dans la procédure `TablePremiers2`, la boucle intérieure est effectuée si et seulement si N est premier, et elle prend $O(\frac{B}{N})$ opérations (de $2N$ à B par pas de N). La procédure utilise donc un nombre total d'opérations de

$$\sum_{N \leq B \text{ premier}} O\left(\frac{B}{N}\right) = O\left(B \sum_{p \leq B} \frac{1}{p}\right) = O(B \log\log B).$$