Calcul efficace de x^n, application au codage RSA

. les programmes sont donnés pour les calculatrices TI, les algorithmes étant transposables sans difficulté sur les autres modèles

Introduction : *x* désigne un nombre réel et n un entier naturel non nul. Par exemple pour calculer x^{17} à l'aide de multiplications, on peut procéder de différentes manières :

La plus élémentaire utilise la définition $x^{17} = x \times x \times x ... \times x$, le résultat est progressivement calculé dans la mémoire r, ce qui donne l'algorithme :

début
  lire x,n
  $1 \to r$
  pour i variant de 1 à n
  faire
    $x \times r \to r$
  finpour
  afficher r
fin

Cet algorithme demande n-1 multiplications, On peut faire beaucoup mieux en remarquant que pour x^{17} : $x^{17} = \left(\left(\left(x^2 \right)^2 \right)^2 \right)^2 \times x$, ce qui ne coûte que 5 multiplications au lieu de 16.

(chaque carré donnant lieu à une multiplication du nombre par lui-même)(1)

L'algorithme suivant calcule progressivement le résultat dans *r*, à chaque étape :

- Si n est impair alors on remplace *r* par $r \times x$ et *n* par *n*–1 pour retrouver un exposant pair
- Si *n* est pair alors on remplace x par x^2 et *n* par $\frac{n}{2}$ -

Lire x,n
$1 \to r$
Répéter
  Si $\mathrm{E}(n/2) \neq n/2$
  Alors $x \times r \to r$
    $n-1 \to n$
  Sinon $x \times x \to x$
    $n/2 \to n$
  Finsi
Jusqu'à n=0
Afficher r

:prompt x, n
:$1 \to r$
:Repeat $n = 0$
:if $\mathrm{int}(n/2) \neq n/2$
:then : $x \times r \to r$
:$n-1 \to n$
:else : $x \times x \to x$
:$n/2 \to n$
:end
:end
:disp *r*

(2)

Le coût de cet algorithme est inférieur ou égal à E(2 ln(n)/ln(2)) multiplications (si n était du type 2^p on effectuerait p multiplications, et dans le cas le plus défavorable, s'il y alternance d'exposants pairs et impairs lors de l'exécution de l'algorithme, où l'on double le nombre de multiplications)

Par exemple pour n = 128, 7 multiplications suffisent pour calculer x^{128} !

Application au codage RSA

Lors des codages ou le décodage du type RSA, on est amené à calculer des congruences du type $c \equiv x^n (a)$, où a et n sont connus, x un nombre qui représente le caractère à coder avec $0 \leq x \leq a$, et c est le reste dans la division euclidienne de x^n par *a*.

Compte tenu de la compatibilité des congruences avec la multiplication, on peut reprendre l'algorithme précédent, en calculant à chaque étape les congruences modulo *a*, plus précisément le reste dans la division euclidienne par *a*. Ceci évite les débordements de capacité des machines lors des calculs de puissances puisqu'alors les nombres traités ne dépassent pas a^2 (lors du calcul de x^2 ou de $r \times x$).

Dans l'algorithme : *x* mod *a* désigne le reste de la division euclidienne par *a*(3)

Les nombres a et n sont donnés au début, x pourra être modifié pour coder plusieurs caractères sans rentrer de nouveau *a* et *n*, ce qui implique d'utiliser une mémoire *m* pour ne pas perdre la valeur initiale de n lors de l'exécution. On arrête le programme en rentrant la valeur zéro pour *t* :

Début
Lire a, n
Répéter
  Lire x
  $x \bmod a \to x$
  $1 \to r$
  $n \to m$
  Répéter
    Si $\mathrm{E}(m/2) \neq m/2$
    Alors $x \times r \to r$
      $r \bmod a \to r$
      $m-1 \to m$
    Sinon $x \times x \to x$
      $x \bmod a \to x$
      $m/2 \to m$
    Finsi
  Jusqu'à *m*=0
  Afficher *r*
Jusqu'à *x*=0
Fin

:prompt a, n
:repeat *x=0*
prompt *x*
$x - \mathrm{int}(x/a) \times a \to x$
:$1 \to r$
:$n \to m$
:Repeat $m = 0$
:if $\mathrm{int}(m/2) \neq m/2$
:then : $x \times r \to r$
:$r - \mathrm{int}(r/a) \times a \to r$
:$m-1 \to m$
:else : $x \times x \to x$
:$x - \mathrm{int}(x/a) \times a \to x$
:$m/2 \to m$
:End
:End
:Disp r
:Disp "STOP:X=0"
:End

Remarques :

(1) Une autre piste très intéressante consiste à écrire l'exposant n comme somme de puissances de 2. Ainsi $17= 2^4 + 2^0$

(2) Le test de parité choisi est int(n/2)= n/2; on peut aussi utiliser: FPART(n/2)=0

(3) Dans l'algorithme : *x* mod *a* désigne le reste de la division euclidienne de x par *a* , *sur TI 80, 89, 92 on peut l'obtenir avecREMAIN(x,a).*