Informatique tronc commun
Algorithmique

Corrigé

30 septembre 2006

Question 1 *Écrire une fonction* `ordre(k,n)` *qui renvoie l'ordre d'un élément k dans le groupe multiplicatif $(\mathbb{Z}/n\mathbb{Z})^*$. (On supposera que k est bien inversible modulo n.)*

```
    ordre := proc(k : :integer, n : :integer) : :inte-
ger ;
        local r : :integer, j : :integer ;
        r := k mod n ; j := 1 ;
        while r <> 1 do
            r := (r * k) mod n ;
            j := j + 1 ;
        od ;
        RETURN(j) ;
    end proc ;
```

Question 2 *On représente un polynôme $P \in \mathbb{Z}[X]$ de degré n par le tableau* `a[i]`, $i \in [\![1, n]\!]$ *de ses coefficients (*`a[i]` *contenant le coefficient de X^i). Écrire une fonction* `evalue(a,x)`, *ou* `evalue(a,n,x)`, *qui calcule la valeur de P en un point $x \in \mathbb{Z}$, en temps $O(n)$.*

```
    evalue := proc(a : :array, x : :integer) : :inte-
ger ;
        local n : :integer, i : :integer, v : :integer ;
        n := op(op(2,eval(a))[1])[2] ;
        v := 0 ;
        for i from n to 0 by -1 do
            v := v * x + a[i] ;
        od ;
        RETURN(v) ;
    end proc ;
```

On rappelle que la suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ est définie par $F_0 = F_1 = 1$ et $\forall n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$.

Question 3 *Écrire une fonction* `Fa(n)` *qui calcule récursivement F_n à l'aide de la définition ci-dessus. Calculer F_{30}. Regarder la barre d'état de Maple. Évaluer (ou minorer) la complexité de votre fonction pour rendre compte de ce que vous observez.*

```
Fa := proc(n : :integer) : :integer ;
  if n <= 1 then
    RETURN(1)
  else
    RETURN( Fa(n-2) + Fa(n-1) ) ;
  end if ;
end proc ;
```

L'exécution prend beaucoup de temps et utilise beaucoup de mémoire. Notons T_n le nombre d'opérations élémentaires qu'exécute `Fa` pour calculer F_n. On a $T_{n+2} \geq T_{n+1} + T_n$, et donc

$$T_n \geq F_n = \Theta_{n\to+\infty}\left(\frac{1+\sqrt{5}}{2}\right)^n,$$

soit une complexité (au moins) exponentielle en n.

Question 4 *Écrire une fonction* `Fb(n)` *de complexité linéaire en n qui calcule itérativement F_n.*

```
Fb := proc(n : :integer) : :integer ;
  local i : :integer,  # indice de boucle
        a : :integer, b : :integer,  # F(n-1) et F(n)
        c : :integer ;  # temporaire
  a := 0 ; b := 1 ;
  for i from 1 to n do
    c := a + b ;
    a := b ;
    b := c ;
  end do ;
  RETURN(c) ;
end proc ;
```