TD 11 : Sujets de concours

ECE3 Lycée Carnot

6 avril 2010

HEC 2009

Fonction complète :

```
Function f(n : integer) : integer ;
var temp,u,v,k : integer ;
Begin
u := 0 ; v := 1 ;
for k := 1 to n-1 do
  Begin ;
  temp := v ; v := u+v ; u := temp ;
  end ;
f := v ;
end ;
```

Pour la fin de l'exercice :

1. Écrivons donc les premiers termes de la suite (en excluant u_0 et u_1) : 1 2 3 5 8 13 21 34 55 89 144 233. On observe que $37 = 34 + 3$ et $272 = 233 + 34 + 5$ sont deux décompositions convenables.
2. Vérifions la propriété si $r = 1$. Cela signifie que n peut se décomposer à l'aide d'un seul terme de la suite, autrement dit que $n = u_{k_1}$. La suite (u_n) étant strictement croissante, on a bien dans ce cas $n < u_{k_1+1}$. Supposons désormais $n = u_{k_1} + \dots + u_{k_r} + u_{k_{r+1}}$, avec par hypothèse de récurrence $u_{k_1} + \dots + u_{k_r} < u_{k_{r+1}}$ (attention à ne pas confondre ce dernier nombre avec $u_{k_{r+1}}$). Or, une Z -décomposition ne pouvant faire apparaître deux termes consécutifs de la suite (u_n) , on a $k_r + 1 \leq k_{r+1} - 1$, donc $u_{k_r+1} \leq u_{k_{r+1}-1}$, et on en déduit que $n < u_{k_{r+1}-1} + u_{k_r} = u_{k_r} + 1$ (ceci par définition de la suite (u_n)), ce qui achève la récurrence. On a donc en fait pas le choix sur u_{k_r} : il s'agit nécessairement du plus gros terme de la suite inférieur ou égal à n . Une fois choisi u_{k_r} chercher une Z -décomposition de n revient à en chercher une de $n - u_{k_r}$, ce pour quoi on a pas le choix non plus. On peut faire une jolie récurrence si on le souhaite, mais la conclusion est celle-ci : la Z -décomposition, si elle existe, est nécessairement unique.
3. Encore une récurrence ? Mais oui. Pour 2, la décomposition est évidente. Supposons que tout entier strictement inférieur à n possède une Z -décomposition (récurrence forte), et notons p le plus grand entier pour lequel $u_p \leq n$ (cet entier existe forcément), alors $n - u_p < n$, donc il admet une Z -décomposition $u_{k_1} + \dots + u_{k_r}$, et $n = u_{k_1} + \dots + u_{k_r} + u_p$ aussi. L'unicité a déjà été prouvée à la question précédente.
4. Evidemment, le programme va être très moche si on s'interdit d'utiliser la récursivité, mais c'est tout de même faisable :

```
PROGRAM hec ;
VAR r,s,i : integer ;
BEGIN
s := p ;
```

```

REPEAT
r := 2;
WHILE t[r] <= n DO r := r+1;
FOR i := r TO s DO t[i] := 0;
s := r-1;
n := n-t[r];
UNTIL n=0;
FOR i := 2 TO r-1 DO t[i] := 0;

```

EMLyon 2009

Cette question est en fait difficilement faisable si on ne connaît pas la valeur de la limite, en l'occurrence $\ln 2$. Si on la connaît, pas contre, c'est un programme idiot :

```

PROGRAM emlyon;
USES wincrt;
VAR u : real; n : integer;
BEGIN
u := 1; n := 0;
REPEAT n := n+1; u := u/(exp(u)-1);
UNTIL abs(u-ln(2)) < 0.000 000 001;
WriteLn(n);
END.

```

EDHEC 2009

Voilà le programme complété :

```

Program edhec2009;
Var x,t,lancer : integer;
Begin
  Randomize; x := 0;
  Repeat lancer := random; x := x+1; until (lancer <= p);
  If (x mod 2=0) then t := x else t := (1+x)/2;
  WriteLn(t);
End.

```

Ecricome 2009

On peut bien sûr faire une belle dichotomie mais aussi, la méthode n'étant pas imposée, des choses beaucoup plus rudimentaires et moches, comme le programme qui suit, qui se contente de tester des valeurs de x écartées de 0.01 jusqu'à ce que la fonction change de signe :

```

PROGRAM ecricome;
USES wincrt;
VAR x : real;
FUNCTION phi (t : real) : real;
BEGIN
phi := 2*ln(x/2)+1/x;
END;
BEGIN
x := 0;

```

```
REPEAT x := x+0.01 UNTIL phi(x) < 0 ;  
WriteLn('alpha est compris entre ',x-0.01,' et ',x) ;  
END.
```

ESC 2009

```
PROGRAM esc ;  
USES wincrt ;  
VAR u : real ; i,n : integer ;  
BEGIN  
WriteLn('Choisissez la valeur de n') ;  
ReadLn(n) ;  
u := 0 ;  
FOR i := 1 TO n DO u := (u*u*u*u+1)/4 ;  
WriteLn('u_n=',u) ;  
END.
```