Énoncé

La *suite de Fibonacci* est définie par :

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ \forall n \in \mathbb{N} : F_{n+2} = F_n + F_{n+1} \end{cases}$$

Dans ce problème[1] on appellera *nombre de Fibonacci* une des valeurs de la suite de Fibonacci.

Partie I. Relations

1. On considère une suite de nombres entiers $(x_n)_{n\in\mathbb{N}}$ vérifiant

$$(1) \quad \forall n \in \mathbb{N} : x_{n+2} = x_{n+1} + x_n + 1$$

   a. Quelle est la relation vérifiée par la suite $(y_n)_{n\in\mathbb{N}} = (x_n + 1)_{n\in\mathbb{N}}$?

   b. Pour tout entier n, exprimer x_n en fonction d'un nombre de Fibonacci lorsque $x_0 = 0$ et $x_1 = 0$.

   c. Pour tout entier n, exprimer x_n en fonction d'un nombre de Fibonacci lorsque $x_0 = 0$ et $x_1 = 1$.

2. On définit :

$$\begin{aligned} \forall n \in \mathbb{N} : s_n &= F_0 + F_1 + F_2 + \cdots + F_n \\ \forall p \in \mathbb{N} \setminus \{0\} : u_p &= F_2 + F_4 + \cdots + F_{2p} \\ \forall p \in \mathbb{N} \setminus \{0\} : v_p &= F_3 + F_5 + \cdots + F_{2p+1} \end{aligned}$$

   a. Pour tout entier n, exprimer s_n en fonction d'un nombre de Fibonacci.

   b. Pour tout entier p non nul, exprimer u_p et v_p en fonction d'un nombre de Fibonacci.

Partie II. Procédures.

1. On propose la procédure suivante en syntaxe Maple.

```
fib :=proc(n)
  local temp,u,v,k;
  u:=0;
  v:=1;
  for k from 1 to n-1 do
    temp := *--- ;
    v:= *--- ;
    u:= *--- ;
  od;
  return *---;
end proc;
```

   Compléter cette procédure aux quatre places signalées par "*—" pour que la valeur renvoyée par l'appel `fib(m)` soit F_m lorsque `m` est assigné à un entier supérieur ou égal à 1.

2. a. Écrire en syntaxe Maple une procédure *récursive* de nom `fib_r` telle que la valeur renvoyée par l'appel `fib_r(m)` soit F_m lorsque `m` est assigné à un entier.

   b. On note a_m le nombre d'additions exécutées par l'appel de `fib_r(m)`. Former une relation vérifiée par les a_m. En déduire

$$\forall n \in \mathbb{N} : a_n = F_{n+1} - 1$$

3. Écrire en syntaxe Maple une procédure de nom `fib_g` telle que l'appel `fib_g(m)` :
   – assigne à une variable globale de nom n la valeur de `m` supposée entière (≥ 2)
   – assigne à une variable globale de nom `F` un tableau indexé de 1 à n dont les valeurs sont les nombres de Fibonacci de $F_1, F_2, \cdots, F_m$.

Partie III. Représentation de Zeckendorf et algorithme glouton

On définit une relation notée $<<$ entre deux entiers i et j par :

$$i << j \Leftrightarrow i + 2 \leq j$$

On dira qu'un entier x admet une *représentation de Zeckendorf* (Z-représentation) si et seulement si il existe des entiers $k_1, k_2, \cdots, k_r$ tels que

$$\begin{cases} 0 << k_1 << k_2 << \cdots << k_r \\ x = F_{k_1} + F_{k_2} + \cdots + F_{k_r} \end{cases}$$

[1] d'après BCE ECE (HEC) 2009 maths 3 et *Mathématiques Concrètes Fondations pour l'informatique* (Vuibert)

Rémy Nicolai Azeck

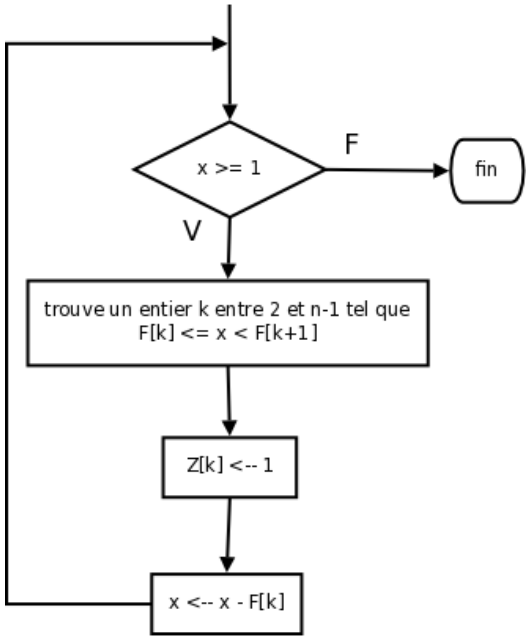

FIG. 1 – algorithme pour obtenir une Z-représentation

1. Exemples.
   a. Donner une Z-représentation pour les entiers entre 1 et 5.
   b. Donner une Z-représentation pour $F_{2p+2} - 1$ et pour $F_{2p+3} - 1$.
2. On considère un algorithme présenté en figure 1 dans lequel
   – `n` désigne un entier,
   – `F` désigne un tableau indexé de 1 à la `n` dont les valeurs sont $F_1, F_2, \cdots, F_n$,
   – `Z` désigne un tableau indexé de 1 à la `n` dont les valeurs sont toutes nulles,
   – `x` désigne un entier strictement inférieur à F_n.
   a. Lors de l'exécution de cet algorithme, expliquez pourquoi on peut toujours trouver un `k` satisfaisant aux conditions.
   b. Après la fin de l'exécution de cet algorithme montrer que `x` est nul et que le produit de deux valeurs consécutives de `Z` est nul.
   c. La question précédente assure-t-elle l'existence d'une Z-représentation de x ? l'unicité ? les deux propriétés ? aucune des deux ?
3. Soient n et $k_1, k_2, \cdots, k_r$ des entiers tels que $n > 2$ et

$$0 << k_1 << k_2 << \cdots << k_r < n$$

   Montrer que

$$F_{k_1} + F_{k_2} + \cdots + F_{k_r} < F_n$$

   En déduire que tout entier admet une unique Z-représentation.
   Expliquez en langage usuel ou avec un diagramme comment obtenir cette représentation.

Corrigé

Partie I. Relations.

1. a. En remplaçant, on trouve que la suite $(y_n)_{n\in\mathbb{N}}$ vérifie la même relation de récurrence que la suite de Fibonacci.
Dans les questions b. et c., les suites proposées et celle de Fibonacci vérifient la même relation de récurrence mais les termes sont décalés.

   b. Si $x_0 = 0$ et $x_1 = 0$:

$$\left.\begin{array}{l} y_0 = 1 = F_1 \\ y_1 = 1 = F_2 \end{array}\right\} \Rightarrow \forall n \in \mathbb{N} : y_n = F_{n+1} \Rightarrow \forall n \in \mathbb{N} : x_n = F_{n+1} - 1$$

   c. Si $x_0 = 0$ et $x_1 = 1$:

$$\left.\begin{array}{l} y_0 = 1 = F_2 \\ y_1 = 2 = F_3 \end{array}\right\} \Rightarrow \forall n \in \mathbb{N} : y_n = F_{n+2} \Rightarrow \forall n \in \mathbb{N} : x_n = F_{n+2} - 1$$

2. a. Formons une relation de récurrence vérifiée par $(s_n)_{n\in\mathbb{N}}$

$$\begin{array}{rccccccccc} s_n = F_0 & + & F_1 & + & \cdots & + & F_n & & & \\ & \searrow & & & & & & \searrow & & \\ s_{n+1} = F_0 & + & F_1 & + & & \cdots & & + & F_{n+1} & \\ & & & \searrow & & & & & & \searrow \\ s_n + s_{n+1} = F_0 & & & + & F_2 & & \cdots & & + & F_{n+2} \end{array}$$

   On en déduit $s_n + s_{n+1} = s_{n+2} - 1$. La suite $(s_n)_{n\in\mathbb{N}}$ vérifie donc la relation de récurrence de la question 1. Comme de plus, $s_0 = F_0 = 0$ et $s_1 = F_0 + F_1 = 1$, on obtient d'après 1.c :

$$\forall n \in \mathbb{N} : s_n = F_{n+2} - 1$$

   Autre solution, on peut simplifier s_n "en dominos" :

$$s_n = (F_2 - F_1) + (F_3 - F_2) + \cdots + (F_{n+2} - F_{n+1}) = F_{n+2} - 1$$

   b. On commence par former $v_p - u_p$ en exploitant la relation de récurrence de Fibonacci avec des différences cette fois :

$$\begin{aligned} v_p - u_p &= (F_3 - F_2) + (F_4 - F_3) + \cdots + (F_{2p+1} - F_{2p}) \\ &= F_1 + F_2 + \cdots + F_{2p-1} = v_{p-1} + 1 = v_p - F_{2p+1} + 1 \end{aligned}$$

   On en déduit :

$$\forall p \in \mathbb{N} : v_p = F_{2p+1} - 1$$

   Pour obtenir u_p, on considère la somme

$$\begin{aligned} u_p + v_p = s_{2p+1} - 1 = F_{2p+3} - 2 \Rightarrow u_p &= F_{2p+3} - 2 - F_{2p+1} + 1 \\ &= F_{2p+2} - 1 \end{aligned}$$

   On en déduit :

$$\forall p \in \mathbb{N} : u_p = F_{2p+2} - 1$$

Partie II. Procédures.

1. Les compléments de code à fournir sont :

```
for k from 1 to n-1 do
  temp := v;
  v := u+v ;
  u := temp;
od;
return v;
```

   Il est à noter que cette procédure ne renvoie F_m que pour $m \geq 1$. Pour $m = 0$, on n'entre pas dans la boucle "for" et la procédure renvoie la valeur intiale de v c'est à dire 1 au lieu de 0. Pour $m = 1$, on n'entre pas non pus dans la boucle mais la valeur renvoyée est bien $F_1 = 1$.

2. a. Dans la procédure récursive, il ne faut surtout pas oublier de faire renvoyer directement le résultat pour $m = 0$ ou 1

```
fib_r := proc(m)
  if m=0 then
    return 0 ;
  elif m=1 then
    return 1 ;
  else
    return fib_r(m-2) + fib_r(m-1);
  fi;
end proc;
```

   b. Pour m égal à 0 ou 1, la procédure renvoie directement le résultat sans calcul. Pour une autre valeur de m, il y a une seule addition à part celles

des appels récursifs. On en déduit :

$$\left.\begin{array}{l} a_0 = a_1 = 0 \\ \forall n \geq 2 : a_n = a_{n-1} + a_{n-2} + 1 \end{array}\right\} \Rightarrow \forall n \in \mathbb{N} : a_n = F_{n+1} - 1$$

d'après la question I.1.b.

3. La procédure `fib_g` utilise une variable globale et un tableau

```
fib_g := proc(m)
  global n, F;
  local k;
  n:=m;
  F:= array(1..n,[]);
  F[1]:=1; F[2]:=1;
  for all k from 3 to n do
    F[k] := F[k-2] + F[k-1]
  od;
  return ();
end proc;
```

Noter le `return` vide à la fin pour que la procédure remplisse le tableau sans rien renvoyer.

Partie III. Représentation de Zeckendorf.

1. Exemples. Les premiers entiers se représentent facilement

$$1 = F_2 \qquad 2 = F_3 \qquad 3 = F_4 \qquad 4 = F_4 + F_2 \qquad 5 = F_5$$

2. a. La suite des nombres de Fibonacci est strictement croissante et la recherche de k se fait dans un boucle pour laquelle x est entre 1 et F_n. Il existe donc toujours un tel k.

   b. Pourquoi x est-il nul à la fin ?
   Remarquons d'abord que x est négatif ou nul car $x \geq 1$ est faux pour sortir de la boucle. D'autre part, juste avant la sortie de boucle, on a trouvé un k tel que $F[k] \leq x$ donc $y = x - F[k] \geq 0$. Ensuite cette valeur y a été assignée à x et c'est ce nombre qui a échoué au test. La valeur de x après sortie de boucle est donc 0.
   Pourquoi la boucle se termine-t-elle ?
   Parceque la suite des valeurs de x est strictement décroissante.
   Pourquoi $Z[k]Z[k-1] = 0$?
   Supposons $Z[k] = 1$. Alors $F[k] \leq x < F[k+1]$. Posons $y = x - F[k]$ qui est la valeur que prendra x ensuite. On a

$$y < F[k+1] - F[k] = F[k-1]$$

   Par conséquent, le "bon " k de l'étape suivante ne peut être $k-1$. Le $Z[k-1]$ reste donc à 0.

   c. La question précédente assure l'existence d'une Z-représentation. Elle fournit un algorithme permettant d'en trouver une. Elle ne dit rien sur l'unicité.

3. La suite des nombres de Fibonacci est strictement croissante donc :

$$F_{k_1} + \cdots + F_{k_r} \leq F_{n-1} + F_{n-3} + \cdots + \left\{\begin{array}{l} F_2 \\ \text{ou} \quad \text{selon parité de } n \\ F_3 \end{array}\right.$$

Avec les résultats de I.2.b., on trouve dans les deux cas que

$$F_{k_1} + \cdots + F_{k_r} \leq F_n - 1$$

L'inégalité précédente permet de montrer l'unicité d'une Z-représentation. Si il en existait deux égales, il existerait des suites se terminant avec $k_r < k'_{r'}$ telles que

$$k_r + 1 \leq k'_{r'}$$

$$F_{k'_{r'}} \leq F_{k'_1} + \cdots + F_{k'_{r'}} = F_{k_1} + \cdots + F_{k_r} < F_{k_r+1}$$

Ce qui est évidemment absurde car $k_r + 1 \leq k'_{r'}$.
On aurait aussi pu raisonner en évaluant le nombre de suites d'indices.
Notons r_n le nombre de ces suites pour un $n \geq 2$. Il est bien clair que $r_2 = r_3 = 1$.
Pour un n fixé, on peut classer les suites suivant la manière dont elles se terminent. Le nombre de suites dont le dernier k est n est a_{n-2} car alors le k précédent ne peut être $n-1$. Quant aux suites qui se terminent strictement avant n, elles sont au nombre de a_{n-1}.
On retombe encore sur $a_n = a_{n-2} + a_{n-1}$ ce qui donne avec les conditions initiales $a_n = F_{n+1}$. Un nombre ne saurait donc admettre plusieurs Z-représentation. Pour obtenir pratiquement une Z-représentation d'un nombre x, il faut d'abord calculer et stocker dans un tableau des nombres de Fibonacci $F_0, \cdots, F_n$ avec $x < F_n$. Il faut ensuite utiliser l'algorithme étudié en question 2.