DS 3
Informatique
Suite de Robinson associée à une liste
Extrait E3A PSI 2018
Correction

Simon Dauguet
simon.dauguet@gmail.com

Mercredi 11 Février 2026

1. On propose :

```
1  def maxi(L) :
2      Vmax = -1        # L contient des entiers naturels
3      for i in range( len(L) ) :
4          if Vmax < L[i] :
5              Vmax = L[i]
6      return Vmax
```

On pose n le nombre d'éléments de la liste `L`. Le booléen de la ligne 4 effectue 2 opérations. Et la ligne 5 également. Donc la structure `if` effectue $2 + \max(2, 0) = 4$ opérations dans le pire des cas. La fonction `maxi` effectue donc $1 + 4n + 1 = 4n + 2$ opérations.

La complexité de `maxi` est donc linéaire en $O(n)$.

2. On propose :

```
1  def ind(L) :
2      M = []
3      for i in range( len(L) ) :
4          if L[i] != 0 :
5              M += [i]     # <=> M.append( i )
6      return M
```

3. On propose :

```
1  def nb_oc(L) :
2      M = maxi(L) + 1
3      T = M * [0]
4      for i in range( M ) :
5          for j in range( len(L) ) :
6              if L[j] == i :
7                  T[i] += 1
8      return T
```

4. (a) À chaque tour de boucle principale, on effectue `len(L)` tours de boucle secondaire. La boucle principale entraîne M tours.

Ainsi la liste `L` est parcourue $1 + M$ fois (1 fois dans `maxi` et M fois dans la boucle).

(b) On propose :

```
1       def nb_oc(L) :
2           M = maxi(L) + 1
3           T = M * [0]
4           for j in range( len(L) ) :
5               T[ L[j] ] += 1
6           return T
```

Dans ce cas, la liste **L** est parcourue deux fois (1 fois dans **maxi** et 1 fois dans la boucle).

(c) La ligne 5 correspond à **T[L[j]] = T[L[j]]+1**, et donc effectue 6 opérations. En posant n le nombre d'éléments de la liste **L**, la boucle **for** effectue donc $6n$ opérations. La ligne 3 effectue $1 + (M - 1) = M$ opérations. Et la ligne 2 effectue $2 + O(n)$ opérations d'après le calcul de la complexité de la fonction **maxi**.

Par conséquent, la fonction **nb_oc** a une complexité linéaire en $O(n)$.

5. (a) $L_1 = [1, 4, 3, 3, 2, 2, 3, 1, 1, 0]$; $L_2 = [1, 4, 3, 3, 2, 2, 3, 1, 1, 0]$. Donc le processus de construction de la suite de Robinson ne change pas la liste L. Autrement dit, la suite de Robinson est constante, et donc $L_1 = L_3 = L_{2026}$

(b) La configuration $L_1 = [2, 4, 1, 1, 1, 2]$ est impossible puisque les nombres 4, 1, 2 ne sont pas donnés dans l'ordre décroissant.

(c) Si $L_1 = [2, 4, 1, 0]$, L_0 est de longueur 3 et contient deux 4 et un 0.

Il peut y avoir 3 possibilités : $L_0 = [0, 4, 4]$, $L_0 = [4, 0, 4]$ et $L_0 = [4, 4, 0]$.

On considère le code suivant :

```
1   def rob(A,n) :
2       i = 0
3       L = A[:]     # <=> A.copy()
4       while i < n :
5           T = nb_oc(L)
6           I = ind(T)
7           L = []
8           for e in I :
9               L = [T[e], e] + L
10          i += 1
11      return L
```

(d) On effectue le jeu de test suivant, pour $A = [3, 1, 4, 1]$ et $n = 2$:

ligne	i	L	i<n	T	I	e	T[e]	Sortie
2	0							
3		[3,1,4,1]						
4			True					
5				[0,2,0,1,1]				
6					[1,3,4]			
7		[]						
8						1		
9		[2,1]					2	
8						3		
9		[1,3,2,1]					1	
8						4		
9		[1,4,3,2,1]					1	
10	1							
4			True					
5				[0,3,1,1,1]				
6					[1,2,3,4]			
7		[]						
8						1		
9		[3,1]					3	
8						2		
9		[1,2,3,1]					1	
8						3		
9		[1,3,1,2,3,1]					1	
8						4		
9		[1,4,1,3,1,2,3,1]					1	
10	2							
4			False					
11								[1,4,1,3,1,2,3,1]

(e) L'entier `n-i` est un variant de la boucle `while` de la fonction `rob`. En effet, à chaque passage dans la boucle `while`, l'algorithme passe par la ligne 10 et donc la variable `i` est incrémenté. Donc la valeur de `n-i` est une suite d'entier strictement décroissante. Et donc la boucle s'arrête bien.

(f) Soit $i \in \mathbb{N}$ le nombre de passage dans la boucle `while`. On pose, pour tout $i \in \mathbb{N}$, L_i l'état de la variable `L` après le i-ème passage dans la boucle `while`. On va montrer que pour tout $i \in \mathbb{N}$, tant qu'on passe par la boucle `while`, L_i correspond au terme d'indice i de la suite Robinson associée à la liste `A`.

L_0 correspond à la liste `L` avant l'entrée dans la boucle `while`. Donc $L_0 = A$. Donc L_0 est bien le premier terme de la suite de Robinson associée à A.

Supposons $\exists i \in \mathbb{N}$ tel que L_i est le terme d'indice i de la suite de Robinson associée à A. Donc, par définition de L_i, à la fin du passage numéro i dans la boucle `while`, la liste `L` contient L_i le terme d'indice i de la suite de Robinson associée à A.

Alors, au début du passage $i+1$ dans la boucle `while`, on a toujours `L` qui contient le terme d'indice i de la suite de Robinson associée à A. Alors, à la ligne 5, la variable `T` contient le nombre d'occurrence des éléments de la liste `L`, donc `T` contient le nombre d'occurrences des éléments du terme d'indice i de la suite de Robinson associée à A. Et donc, à la ligne 6, la variable `I` contient les indices des termes non nuls de la liste des occurrences des termes non nuls de la liste `T`. On rappelle que dans `T`, chaque valeur correspond aux nombre d'occurrence de l'indice de la position de cette valeur. Donc `I` est la liste des nombres qui apparaissent dans la liste `L`, et `T` est le nombre d'occurrence de ces valeurs. Alors, pour tout $e \in I$, `[T[e],e]` est le couple du nombre d'occurrence de la valeur `e` dans la liste `L`. Donc, à la fin de la boucle `for`, la variable `L` contient la liste de toutes les occurrences de chaque valeurs de l'ancienne valeur de `L`. Autrement dit, à la ligne 10, la variable `L` contient la liste de Robinson de l'ancienne valeur de `L`, c'est-à-dire du terme d'indice i de la suite de Robinson associée à A. Donc, à la fin de la boucle `while`, la variable `L` contient le terme d'indice $i+1$ de la suite de Robinson associée à A, par définition de la construction de la suite de Robinson.

D'où, par principe de récurrence, tant qu'on reste dans la boucle `while`, pour tout $i \in \mathbb{N}$, la variable `L` contient, à la fin de la boucle `while`, la liste d'indice de i de la suite de Robinson associée à A.

Donc, à la sortie de la boucle `while`, c'est-à-dire après n passage dans la boucle `while`, l'algorithme renvoie la valeur de `L`, et donc, d'après l'étude précédente, le terme d'indice n dans la suite de Robinson associée à A.