

**M1.L4** : Série d'exercices sur les algorithmes / récursivité**1** Que font ces algorithmes?

Voici quatre algorithmes :

algo1	algo2
entrée : entier naturel $n$ sortie :??	entrée : entier naturel $n$ sortie :??
Si $n = 0$ sortir : 0 $k \leftarrow 0$ Pour $j$ allant de 1 à $2n - 1$ Si $j$ est impair $k \leftarrow k + j$ sortir : $k$	sortir : $n^2$
algo3	algo4
entrée : entier naturel $n$ sortie :??	entrée : entier naturel $n$ sortie :??
Si $n = 0$ sortir : 0 sortir : $2n - 1 + \text{algo3}(n - 1)$	Si $n = 0$ sortir : 0 Si $n = 1$ sortir : 1 sortir : $n + \text{algo4}(n - 2)$

Pour chacun des quatre algorithmes ci-dessus, répondre aux questions suivantes :

- En général, que fait l'algorithme?  
(si vous répondez juste à cette question, toutes les suivantes sont faciles)
- Que se passe-t-il si on exécute l'algorithme avec la valeur d'entrée 6?
- Quelle est l'ordre de complexité de l'algorithme? (utiliser la notation de Landau  $O(\cdot)$ )
- L'algorithme est-il récursif ou non?

Voici quatre autres algorithmes :

algo5
entrée : entier naturel $n$ sortie :??
<p>Si <math>n = 0</math>  sortir : 0  sortir : <math>n + \text{algo5}(n)</math></p>
algo7
entrée : entier naturel $n$ sortie :??
<p>Si <math>n = 0</math>  sortir : 0  Si <math>n = 1</math>  sortir : 1  sortir : <math>2 \cdot \text{algo7}(n-1) - \text{algo7}(n-2)</math></p>

algo6
entrée : entier naturel $n$ sortie :??
<p>Si <math>n = 0</math>  sortir : 0  sortir : <math>n + \text{algo6}(n+1)</math></p>
algo8
entrée : entier naturel $n$ sortie :??
<p>Si <math>n = 0</math>  sortir : 0  Si <math>n = 1</math>  sortir : 1  sortir : <math>\text{algo8}(2n) + \text{algo8}(2n - 1)</math></p>

f) Un seul de ces algorithmes fonctionne correctement : lequel?

g) Et celui qui fonctionne a un gros défaut : lequel?

h) Question subsidiaire : que calcule-t-il?

## 2 Au temps des Egyptiens, troisième partie

Proposez une version récursive de l'algorithme vu à l'exercice 5 de la série M1.L1 et à l'exercice 2 de la série M1.L2 sous la forme d'une boucle conditionnelle :

devinette
entrée : $a, b$ deux entiers naturels non nuls sortie :??
<p><math>x \leftarrow a</math>  <math>y \leftarrow b</math>  <math>z \leftarrow 0</math>  Tant que <math>y \geq 1</math>    Si <math>y</math> est pair      <math>x \leftarrow 2x</math>      <math>y \leftarrow y/2</math>    Sinon      <math>z \leftarrow z + x</math>      <math>y \leftarrow y - 1</math>  sortir : <math>z</math></p>

### 3 Au temps des Grecs

Proposez une version récursive de l'algorithme d'Euclide, qui calcule le plus grand diviseur commun de deux entiers naturels  $a$  et  $b$ . Voici une variante différente de celle vue à la série M1.L2 :

Algorithme d'Euclide
entrée : $a, b$ deux entiers naturels non nuls sortie : pgcd ( $a, b$ )
$r \leftarrow 0$ Tant que $b > 0$ $r \leftarrow a \bmod b$ $a \leftarrow b$ $b \leftarrow r$ sortir : $a$

### 4 Tri-Fusion

- Dessiner les étapes de l'exécution du tri-fusion vu en cours sur la liste {20, 3, 4, 12, 2, 16, 3}.
- Cet algorithme s'exécute-t-il avec le même nombre d'étapes pour la liste {1,2,3,4,5,6,7} ?

### 5 Taille de liste : le retour

Cet exercice reprend la donnée de l'exercice 3 de M1.L3. Proposez une version récursive de l'algorithme **TailleDichotomique(L, a, b)** qui cherche la taille inconnue  $t$  d'une liste  $L$  sachant qu'elle appartient à l'intervalle  $[a, b[$ .

TailleDichotomique
entrée : $L, a, b$ sortie : le nombre d'éléments de $L$
<b>Tant que</b> $a < b - 1$ $c \leftarrow a + \lfloor \frac{b-a}{2} \rfloor$ <b>Si</b> $a\_element(L, c)$ $a \leftarrow c$ <b>Sinon</b> $b \leftarrow c$ <b>sortir</b> $a$

Rappel : on dispose de l'algorithme  $a\_element(L, i)$  qui nous dit (vrai ou faux) s'il existe un  $i^e$  élément dans la liste : il répond donc **vrai** si  $i$  est inférieur ou égal la taille de la liste et **faux** sinon.<sup>1</sup>

---

<sup>1</sup> . Notez qu'il n'est pas évident de toujours avoir un tel algorithme (sans avoir taille, bien sûr!). En réalité cela dépend de la représentation effective de  $L$ . Mais nous faisons ici l'hypothèse qu'un tel algorithme existe pour  $L$ .