Algorithmique

TD3 : Complexité et récursivité

# 1 Calculs de complexité

Évaluer la complexité des algorithmes suivants.

**Algorithme 1 :** Algo1

**Entrées :** $n$: entier
$i \leftarrow 1$
**tant que** $i \leq n$ **faire**
  **pour** $j$ *de* $1$ *à* $i$ **faire**
    `OpérationElementaire()`
  **fin**
  $i \leftarrow i * 2$
**fin**

**Algorithme 2 :** Algo2

**Entrées :** $n$: entier
$i \leftarrow 0$
$j \leftarrow 1$
**tant que** $i \leq n$ **faire**
  `OpérationElementaire()`
  $i \leftarrow i + j$
  $j \leftarrow j + 1$
**fin**

**Algorithme 3 :** Algo3

**Entrées :** $n$: entier
$i \leftarrow 2$
**tant que** $i \leq n$ **faire**
  `OpérationElementaire()`
  $i \leftarrow i^2$
**fin**

**Algorithme 4 :** Algo4

**Entrées :** $n$: entier
$i \leftarrow n$
$j \leftarrow 0$
**tant que** $i \geq 1$ **faire**
  **pour** $k$ *de* $1$ *à* $i$ **faire**
    **pour** $\ell$ *de* $1$ *à* $j$ **faire**
      `OpérationElementaire()`
    **fin**
  **fin**
  $i \leftarrow i/2$
  $j \leftarrow j + 1$
**fin**

# 2 Opérations sur les matrices

1. Écrivez un algorithme `Somme`, qui calcule la somme de deux matrices (tableaux bidimentionnels) $A$ et $B$ qu'on suppose de même taille. Quelle est la complexité de cette fonction, en fonction de la taille de $A$ et $B$?

2. Écrivez un algorithme `Produit`, qui calcule le produit de deux matrices carrées $A$ et $B$. Quelle est la complexité de votre fonction ?

3. Écrivez un algorithme `Puissance`, qui prend en entrée une matrice $A$ supposée carrée et un entier $n$, et retourne la matrice $A^n$. Pour cela, on fera plusieurs appels à la fonction `Produit`, et on utilisera une exponentiation rapide, qui repose sur le principe suivant :

$$A^n = \begin{cases} \left(A^{n/2}\right)^2 & \text{si } n \text{ est pair ;} \\ A * \left(A^{\lfloor n/2 \rfloor}\right)^2 & \text{si } n \text{ est impair.} \end{cases}$$

   Combien d'appels à `Produit` fait-on dans le pire cas ? Quelle est la complexité de `Puissance` dans le pire cas ?

## 3 Recherche dichotomique

1. Écrire un algorithme `EstPrésent` qui prend comme paramètre une valeur `x` et un tableau `tab` trié dans l'ordre croissant, et qui renvoie `Vrai` si `x` est présent dans `tab`, et `Faux` sinon, en utilisant une recherche dichotomique.

   Le principe de la recherche dichotomique consiste à comparer l'élément recherché `x` avec l'élément central `y`. S'ils sont égaux, on renvoie `Vrai`. Si $x < y$ on continue la recherche dichotomique sur la moitié gauche de `tab`, et si $x > y$ on continue la recherche dichotomique sur la moitié droite de `tab`.

2. Quelle est la complexité dans le pire cas de la recherche dichotomique ?

## 4 Fibonacci

Les nombres de Fibonacci sont définis par $F_0 = F_1 = 1$, et $F_n = F_{n-1} + F_{n-2}$ pour tout $n \geq 2$. On considère l'algorithme suivant.

**Algorithme 5 :** Fibo

```
Entrées : n ≥ 0: entier
si n ≤ 1 alors
    retourner 1
sinon
    retourner Fibo(n-1)+Fibo(n-2)
fin
```

1. Énumérer les appels récursifs de `Fibo(5)` (sous la forme d'un arbre). Que remarque-t-on ?

2. La complexité de l'algorithme `Fibo` est exponentielle, à cause de redondances dans les appels récursifs. Pour pallier cela, on peut implémenter un algorithme récursif `Fibo2` qui prend comme paramètre un entier $n$, et qui retourne la paire $(F_n, F_{n+1})$. Écrire un tel algorithme.

3. Quelle est la complexité de `Fibo2` ?