

# INFO2050 - Programmation avancée

## Répétition 2: Résolutions de récurrences

Jean-Michel BEGON

13 octobre 2017

### 1 Récurrences

#### Exercice 1

Trouver une solution analytique pour la récurrence suivante :

$$T(n) = \begin{cases} 2, & \text{if } n = 0 \\ \frac{1}{n} \sum_{p=0}^{n-1} T(n-1-p)T(p), & \forall n > 1 \end{cases} \quad (1)$$

#### Exercice 2

Trouver une solution analytique pour la récurrence suivante :

$$\begin{cases} T(1) = 1, \\ nT(n) = (n-2)T(n-1) + 2, \quad \forall n > 1 \end{cases} \quad (2)$$

#### Exercice 3

Trouver une solution analytique pour la récurrence suivante :

$$T(n) = \begin{cases} 13, & \text{if } n = 1. \\ 2T(n/8) + 21n, & \forall n > 1. \end{cases} \quad (3)$$

#### Exercice 4

Soit la récurrence (où  $n > 1$  est une puissance de 3)

$$T(n) = \begin{cases} 0, & \text{if } n = 1. \\ 6T(n/3) + 2n, & \forall n > 1. \end{cases} \quad (4)$$

- Trouver une solution analytique exacte à la récurrence.
- Vérifier que la solution satisfait à la borne obtenue par le *master theorem*.

## Exercice 5

Pour chacun des pseudo-codes suivants, déterminer ce que fait l'algorithme, puis la complexité asymptotique.

<pre>CODE1(<i>n</i>) 1  if <i>n</i> ≤ 1 2      return 1 3  else 4      return CODE1(<i>n</i> - 1) + CODE1(<i>n</i> - 1)</pre>	<pre>CODE1'(<i>n</i>) 1  if <i>n</i> ≤ 1 2      return 1 3  else 4      return 2 * CODE1'(<i>n</i> - 1)</pre>
<pre>CODE2(<i>n</i>) 1  if <i>n</i> == 0 2      return "" 3  else 4      tmp = CODE2(<i>n</i>/2) 5      if <i>n</i>%2 == 0 6          return tmp + tmp 7      else 8          return tmp + tmp + "x"</pre>	<pre>CODE3(<i>A</i>, <i>k</i>) 1  for <i>i</i> = 1 to <i>A.length</i> 2      if <i>A</i>[<i>i</i>] == <i>k</i> 3          return <i>i</i> 4  return -1</pre>

## Exercice 6

Soit une chaîne de caractères  $S$  de longueur  $N$ . Quelle serait la complexité minimale d'un algorithme qui imprime à l'écran :

- Toutes les sous-chaînes contiguës de longueur  $k$  ( $0 \leq k \leq N$ ) de  $S$ ?
- Toutes les sous-chaînes non nécessairement contiguës de  $S$ ?

## Exercice 8 — Exercice d'examen

Supposons que vous disposiez des trois algorithmes suivants pour résoudre un même problème de taille  $n$  :

- L'algorithme A qui divise le problème en 2 sous-problèmes de taille  $n/2$ , résout récursivement chacun des sous-problèmes, puis combine leur solution en utilisant  $n^3$  opérations exactement.
- L'algorithme B qui divise le problème en 8 sous-problèmes de tailles  $n/2$ , résout récursivement chacun des sous-problèmes, puis combine leur solution en utilisant  $n$  opérations exactement.
- L'algorithme C qui divise le problème en 4 sous-problèmes de tailles  $n/2$ , résout récursivement chacun des sous-problèmes, puis combine leur solution en utilisant  $n^2$  opérations exactement.

Lequel de ces trois algorithmes est le plus efficace pour des grandes valeurs de  $n$ ?

## Exercice 9

Soit un tableau de  $N$  entiers où chaque entier de l'intervalle  $1..N$  apparaît exactement une fois, à l'exception d'un entier apparaissant 2 fois et d'un entier manquant. Proposer un algorithme linéaire pour trouver l'entier manquant, en utilisant au plus  $O(1)$  d'espace mémoire supplémentaire.

## Bonus

### Bonus 1

Le premier problème du projet Euler ne nécessite pas l'utilisation de l'ordinateur. Il s'énonce comme suit :

*“Si on liste tous les naturels inférieurs à 10 qui sont multiples de 3 ou de 5, on obtient 3, 5, 6 et 9. La somme de ceux-ci est 23.*

*Trouver la somme de tous les multiples de 3 ou de 5 inférieure à 1000.”*