Algorithmique
TD n° 2 : Diviser pour règner

Exercice 1 : point fixe

Donner un algorithme *diviser pour régner* qui prend en entrée un tableau trié `t` de n entiers (relatifs) *tous distincts*, et qui décide s'il existe un indice i tel que `t[`i`]`$= i$. Analyser sa complexité.

Exercice 2 : minimum et maximum

Soit `t` un tableau de n éléments (comparables entre eux).

1. Montrer que le calcul du minimum de `t` demande au moins $n-1$ comparaisons.

2. Proposer un algorithme optimal. Mêmes questions pour le maximum.

On considère maintenant l'algorithme (écrit en Python) suivant, qui étant donné un tableau `t` et deux indices `g` et `d` retourne un couple d'entier.

```
def minMax2(t,g,d):
   if g == d: return t[g],t[g]
   else:
     if t[g] < t[g+1]: return t[g],t[g+1]
     else: return t[g+1],t[g]

def minMax(t,g,d):
   if d <= g + 2 : return minMax2(t,g,d)
   milieu = (g+d)//2
   ming, maxg = minMax(t,g, milieu)
   mind, maxd = minMax(t,milieu, d)
   return min(ming, mind), max(maxg, maxd)
```

3. Montrer que l'algorithme calcule le minimum et le maximum d'un tableau `t` si on l'appele avec paramètre `0` et `len(t)`

4. Combien de comparaisons fait-il (pour un tableau de taille $n = 2^k$) ? Est-ce contradictoire avec la question 2 ?

5. Proposer un algorithme non récursif aussi efficace.

6. Programmer l'algorithme récursif et l'algorithme non récursif. Comparer leur temps d'éxécution sur des tableaux de grandes tailles.

Exercice 3 : des pièces et une balance

Un tas de n pièces en comprend une fausse, et une seule, qui est plus légère que les autres. Pour l'identifier, on ne dispose que d'une balance à plateaux.

1. Combien de pesées sont nécessaires pour $n = 4$? $n = 8$? $n = 9$?

2. Proposer un algorithme *diviser pour régner* résolvant le problème.

3. Montrer sa correction.

4. Calculer sa complexité (nombre de pesées)

5. Prouver son optimalité.

Rappel du cours Le théorème suivant (*Master Theorème*) permet de résoudre les récurrences les plus fréquentes dans l'analyse des algorithmes de type *diviser pour régner*.

Théorème. *Soient deux rationnels $a \geqslant 1$ et $b > 1$. Soit $f(n)$ une fonction positive. On considère la fonction $t(n)$ définie par :*

$$t(n) \;=\; \begin{cases} a \cdot t(\frac{n}{b}) + f(n) & si\, n > 1 \\ \Theta(1) & si\, n = 1 \end{cases}.$$

où $\frac{n}{b}$ peut aussi désigner $\lceil \frac{n}{b} \rceil$ ou $\lfloor \frac{n}{b} \rfloor$.
Alors on a :

1. *Si $f(n) = O(n^{\log_b(a)-\varepsilon})$ pour $\varepsilon > 0$, on a : $t(n) = \Theta(n^{\log_b a})$.*
2. *Si $f(n) = \Theta(n^{\log_b a})$, on a : $t(n) = \Theta(n^{\log_b a} \cdot \log n)$.*
3. *Si $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ pour $\varepsilon > 0$ et si il existe $c < 1$ tel que $a \cdot f(\frac{n}{b}) \leqslant c \cdot f(n)$ pour n assez grand, alors $t(n) = \Theta(f(n))$.*

Exercice 4 :

Donner les bornes asymptotiques obtenues grâce au Master Theorem pour les fonctions suivantes, sachant qu'elles sont constantes non nulles pour $n \leqslant 2$:

- $A(n) = A(\frac{9n}{10}) + n$
- $B(n) = 2\, B(\frac{n}{4}) + \sqrt{n}$
- $C(n) = 7\, C(\frac{n}{12}) + \log_2 n$
- $D(n) = 2\, D(\frac{n}{2}) + \dfrac{n}{\log_2 n}$
- $E(n) = 5\, E(\frac{n}{3}) + n^2$
- $F(n) = 4\, F(\frac{n}{2}) + n^2 \log_2 n$

Exercice 5 :

On veut fusionner k tableaux d'entiers triés. Chaque tableau à taille n. Un algorithme naif est de fusionner le premier tableaux avec le deuxième, de fusionner le résultat avec le troisième, etc.

1. Analyser la complexité de cet algorithme (en terme de k et n).

2. Donner un algorithme *diviser pour régner* et analyser sa complexité.