Série d'exercices - 3
Paradigme « diviser pour régner »

Exercice-1 : Maximum d'un tableau

Etant donné un tableau X composé de N éléments entiers. On voudrait déterminer son maximum par un algorithme récursif basé sur le paradigme « diviser pour régner » :

a- En considérant que le maximum est le plus grand entre le dernier terme et le maximum des (n-1) premiers termes. Estimer sa complexité.
b- En considérant que le maximum est le plus grand entre les maximums des deux moitiés du tableau. Estimer sa complexité.

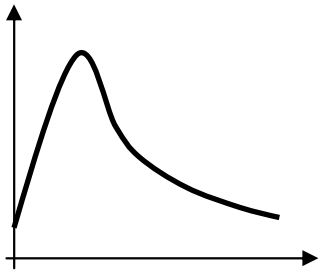
Exercice-2 : Recherche binaire récursive

Soit X un tableau composé de N entiers ordonnés par ordre croissant et Y un entier.

a- Elaborer un algorithme récursif permettant de trouver la position de Y dans X en opérant selon le paradigme « diviser pour régner ».
b- Estimer sa complexité.

Exercice-3 : Recherche récursive du maximum d'une fonction

Soit une fonction à une dimension définie par N échantillons équidistants se trouvant dans un tableau X. On suppose que la fonction est unimodale (croissante puis décroissante) (Voir figure). On voudrait déterminer l'échantillon qui donne sa valeur maximale.



1- Ecrire un algorithme récursif permettant de déterminer l'élément du tableau X contenant l'échantillon maximal selonle paradigme « diviser pour régner ».
2- Estimer sa complexité en terme de comparaisons.

Exercice-4 : Puissance X^n

Considérer la méthode suivante pour calculer X^n :

$$X^n = \begin{cases} X^{n/2} * X^{n/2} & si\_n\_est\_pair \\ X^{n/2} * X^{n/2} * X & si\_n\_est\_impair \end{cases}$$

n/2 représente la division entière de n par 2.

a- Ecrire un algorithme récursif pour calculer X^n
b- Donner une formule récurrente estimant sa complexité
c- Calculer sa complexité en utilisant le théorème de résolution des récurrences.

Exercice-5 : Maximum d'une matrice carrée

Soit une matrice entière carrée T d'ordre NxN avec $N=2^k$

a- Ecrire un algorithme récursif basé sur le paradigme « diviser pour régner » permettant de déterminer le maximum de la matrice T. L'algorithme doit procéder par la division de T en quatre sous-matrices d'ordre (N/2 x N/2) chacune. Pour cela adopter le prototype suivant pour l'algorithme : **Maximum (T, a, b, N)** avec : a : début de l'indice des lignes, b : début de l'indice des colonnes, N : le nombre de lignes et le

nombre de colonnes. Et supposer que vous avez à votre disposition une fonction **Sup(x1,x2,x3,x4)** qui vous retourne le maximum des 4 paramètres.

b- Quelle est la formule de récurrence donnant sa complexité. Expliquer.
c- Estimer cette complexité ?

Exercice-6 : Multiplication de deux matrices

Soient trois matrices réelles A, B et C carrées d'ordre NxN avec $N=2^k$

d- Ecrire un algorithme récursif basé sur le paradigme « diviser pour régner » pour multiplier A par B et mettre le résultat dans une matrice C. L'algorithme doit procéder par la division de T en quatre sous matrices d'ordre (N/2 x N/2) chacune.
e- Quelle est la formule de récurrence donnant sa complexité. Expliquer.
f- Estimer cette complexité ?

Exercice-7 : Tri par insertion binaire

Etant donné un tableau X composé de N entiers et numérotés de 0 à (N-1), on se propose d'écrire une fonction « Trier » permettant de trier le tableau X en opérant par « insertion binaire ». Pour cela, on vous propose de suivre les étapes suivantes :

a- Etant donné un tableau Y composé de M entiers **triés par ordre croissant**, écrire une fonction **« Position »** permettant de chercher, selon le paradigme **« diviser pour régner »**, la position que devrait occuper un entier Z dans Y de telle sorte que le tableau Y reste trié. La fonction « position » doit retourner la position recherchée.
b- Etant donné un tableau Y composé de M entiers, écrire une fonction **« Décaler »** permettant de décaler d'une position vers la droite tous les éléments du tableau Y[k] pour k allant de P à M-1 (P étant un indice compris entre 0 et M-1). On suppose que la position Y[M] est libre.
c- Ecrire une fonction **« trier »** permettant de trier par ordre croissant un tableau X composé fe N entiers. La fonction trier doit bien sûr appeler les fonctions « Position » et « Décaler »
d- Etablir une formule récurrente puis estimer la complexité T(n) en nombre de comparaisons pour trier un tableau X composé de N entiers.

Exercice-8 : Tri récursif par partition

Etant donné un tableau composé de N entiers :

a- Ecrire un algorithme récursif permettant de le trier par ordre croissant selon le méthode par partition (tri rapide)
b- Estimer sa complexité.

Exercice-9 : Couverture par des tuiles

Soit une matrice de taille $2^N x 2^N$ représentant un échiquier (figure-2a). Toutes les cases sont de couleur blanche sauf une de couleur noire.
Soit un motif pouvant avoir 4 orientations possibles (Figure-1) notées A, B, C et D. Il est prouvé qu'on peut couvrir la totalité de l'échiquier avec ces formes. Le problème est comment le faire ? Pour cela, on vous propose l'idée suivante :

- on divise l'échiquier en 4 parties égales P1, P2, P3 et P4
- On place une des formes du motif (A ou B ou C ou D) au centre de telle sorte que chaque partie contienne une case noire (Figure-2b).

2.1/ Comment appliquer le paradigme « diviser pour régner » pour couvrir la totalité de l'échiquier ?

2.2/ Elaborer un algorithme basé sur le paradigme « diviser pour régner » pour couvrir l'échiquier. La valeur N est un paramètre de l'algorithme.

2.3/ Quelle est sa complexité (avec démonstration) ?

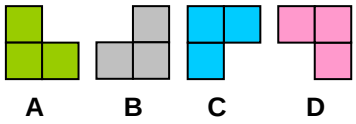


Figure-1

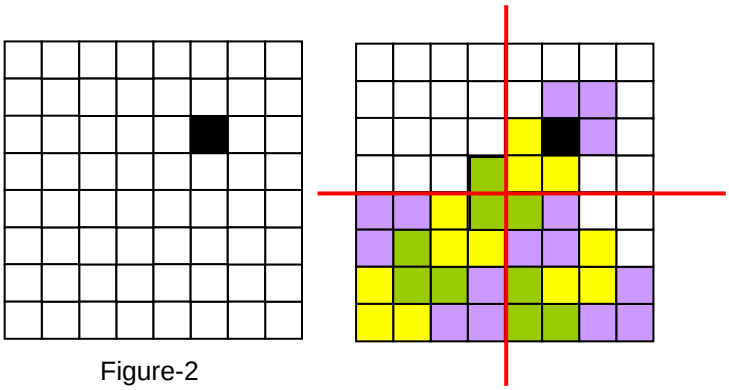


Figure-2

Exercice-10 : Elément majoritaire

Soit X un tableau d'entiers de taille N. Un élément de X est majoritaire si l'ensemble {i tel que X[i] = e} est de cardinalité strictement supérieure à n/2.

1- Proposer un algorithme simple qui recherche un élément majoritaire dans un tableau X. Analyser sa complexité.
2- Proposer un 2ème algorithme opérant selon le paradigme "diviser pour régner" qui recherche un élément majoritaire dans un tableau X. Analyser sa complexité.

Exercice-11 : Détermination de zones homogènes

Soit X une matrice carrée entière (entier court : unsigned char) de dimension NxN représentant une image. Chaque élément X(i,j) représente un pixel dont la valeur appartient à l'intervalle [0,255] et représente un niveau de gris d'un point de l'image allant du noir (valeur 0). Une zone est dite homogène s'il n'existe pas une grande variation entre ses pixels. Cette variation peut être exprimée par une différence réduite entre la valeur minimale et la valeur maximale. L'objectif est de segmenter l'image (c'est-à-dire déterminer les zones homogènes et d'affecter à tous les pixels d'une même zone la même valeur). On suppose que $N = 2^k$. Pour déterminer ces zones homogènes, on se propose d'appliquer la méthode « Split » ou « Quadtree » dont le principe est le suivant :

- Initialiser la couleur C à 0
- Vérifier le critère d'homogénéité de toute l'image m.
- Si le critère est vérifié, incrémenter C et affecter à tous les pixels la même valeur C.
- Sinon : diviser la matrice m en 4 sous-matrices égales et appliquer le même processus à chaque sous-matrice.

Comme critère d'homogénéité, on vous propose de vérifier que la différence entre le maximum et le minimum de la matrice est inférieure à une valeur V donnée.

a- Ecrire une fonction « Split » basée sur le paradigme « diviser pour régner » pour segmenter une image m en régions homogènes.
b- Estimer sa complexité entre nombre d'appels récursifs.