Fiche de TD : Recherche d'algorithmes 1/2

Ce TD porte sur la méthode de production **diviser pour régner** *et sa* **programmation récursive**.
*On insistera sur l'***analyse descendante** *et le* **calcul de la complexité.**

Exercice - 1 *Un exemple simple de tri*

Q-1 **:** Montrer (par récurrence sur l'entier $n \geq 1$ que

$$1 + 2 + 3 + \ldots + (n-2) + (n-1) = \frac{(n-1)n}{2}$$

Q-2 **:** On se propose d'ordonner les éléments d'un tableau T de n nombres réels. On propose la démarche suivante : Tan que le tableau n'est pas entièrement trié,
– on cherche le plus grand de tous les éléments de la partie non triée
– et on le place en dernière position de la partie non trie au moyen d'un échange.

Q-2-1 **:** Appliquer cette démarche sur le tableau suivant $T =$ | 6 | 3 | 7 | 2 | 3 | 5 |

Q-2-2 **:** Écrire l'algorithme itératif qui réalise ce tri (appelé *tri par sélection*).

Q-2-3 **:** Dénombrer le nombre total d'opérations élémentaires exécutées.

Q-2-4 **:** En déduire que la complexité en temps de cet algorithme est $\mathcal{O}(n^2)$.

Q-3 **:** Proposer un algorithme récursif qui traduit la démarche décrite à la question Q-2.

Q-3-1 **:** Écrire l'équation vérifiée par la fonction de complexité en temps $f(n)$ où n est la taille du tableau à trier.

Q-3-2 **:** Résoudre cette équation de récurrence et conclure.

Exercice - 2 *Calcul d'une factoriel*

La factoriel de x, notée $x!$, est le résultat de l'opération $1 \times 2 \times 3 \times \ldots \times x$. Ainsi, $4! = 24$, soit le résultat $1 \times 2 \times 3 \times 4$.

Q-1 **:** Écrire un algorithme itératif qui calcule la factorielle du nombre demandé à l'utilisateur.

Q-2 **:** Écrire un algorithme récursif qui effectue ce même calcul.

Exercice - 3 *Détecteur de Palindrome*

Un palindrome est un mot (ou une phrase) qui se lit aussi bien à l'envers qu'à l'endroit : par exemple " radar ", "kayak", ou bien la phrase "esope reste et se repose"(si on ne tient pas compte des espaces).

Q-1 **:** Écrire un algorithme récursif de détecteur de palindrome. Cet algorithme travaillera sur un tableau de caractères contenant la phrase. Il testera si deux caractères opposés sont identiques, puis appellera récursivement la fonction sur la partie de la chaîne qui ne contient pas ces deux caractères. Les espaces seront ignorés.

Exercice - 4 *La suite de Fibonacci*

La suite de Fibonacci pour un nombre entier n se définit comme la relation de récurrence :

$$F(n) = F(n-1) + F(n-2), \text{ pour } n \geq 2 \text{ avec } F(0) = F(1) = 1.$$

Q-1 **:** Écrire l'algorithme récursif qui implémente ce calcul.

Q-2 **:** Écrire une version itérative de cet algorithme en appliquant une règle de suppression de la récursivité (par exemple celle vue en cours).

Exercice - 5 *Quelques opérations sur les tableaux*

Pour chacune des questions ci-dessous, il faut fournir la complexité temporelle de l'algorithme.

Q-1 **: Recherche d'un élément dans un tableau**
Écrire une fonction récursive `estPresent`, qui retourne **VRAI** si un élément donné est un des éléments d'un tableau, T, d'entiers et **FAUX** sinon. Étudier le cas où T n'est pas un tableau trié et où T est un tableau trié.

Q-2 **: Plus petit élément et plus grand éléments d'un tableau**
Écrire deux fonctions récursives, `caculerMinMax`, qui calcule le plus grand et le plus petit élément d'un tableau T d'entiers basée sur les principes suivants :
– le plus grand élément est le maximum entre le premier élément et le reste du tableau.
– le plus grand éléments est le maximum entre le plus grand de chaque moitié de tableau (méthode dichotomique).

Q-3 **: Inversion d'un tableau**
Soit T un tableau d'entiers. Écrire une procédure, `inverserTableau`, qui change de place les éléments de ce tableau de telle sorte que le nouveau tableau T soit une sorte de "miroir" de l'ancien.
Exemple : $1\ 2\ 4\ 6 -> 6\ 4\ 2\ 1$.

Exercice - 6 *La tour de HANOI*

Les Tours de Hanoi sont un jeu constitué de trois tours symbolisées par des piques A,B,C. Sur le piquet A, sont enfilés N ($N \geq 3$) disques par ordre de diamètres décroissants, formant ainsi une pyramide. Les piquets B et C étant vides. Le but du jeu est de placer progressivement tous ces disques sur le piquet B en utilisant le piquet C comme intermédiaire. La règle du jeu stipule qu'on ne peut déplacer qu'un seul disque à la fois et qu'on ne doit jamais placer un disque sur un disque de diamètre inférieur.

Q-1 **:** Proposer un algorithme récursif pour résoudre le problème de la tour de Hanoi.

Q-2 **:** Étudier la complexité de cet algorithme. *On pourra montrer que la fonction complexité en temps de l'algorithme récursif est de la forme* $f(n) = 2f(n-1) + 1$

Exercice - 7 *Approximation par dichotomie de la racine d'une fonction dans l'intervalle* $]a, b[$

On considère une fonction réelle $f(x)$ strictement monotone sur l'intervalle $]a, b[$. avec $f(a)f(b) < 0.$;

Q-1 **:** Montrer que cette fonction admet une racine dans cet intervalle.

Q-2 **:** On découpe l'intervalle $]a, b[$ en deux intervalle $]a, c[$ et $]c, b[$, avec c le milieu de l'intervalle. Montrer que la racine de f n'appartient qu'à l'une de ces intervalle.

Q-3 **:** Le cas trivial étant obtenu lorsque $|f(c)| < \varepsilon$, où $\varepsilon > 0$ est une précision donnée. (C'est-à-dire que c dans ce cas est une approximation à ε près de la racine de f). Écrire un algorithme récursif qui détermine la racine de f à ε près.

Q-4 **:** Étudier la complexité de cet algorithme.

Exercice - 8 *Produits de Matrices*

On souhaite effectuer le produit de deux matrices X et Y de même taille et mettre le résultat dans une matrice Z. On décompose chacune des matrices de la façon suivante :

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad Y = \begin{bmatrix} E & F \\ G & H \end{bmatrix} \quad Z = \begin{bmatrix} I & J \\ K & L \end{bmatrix}$$

Q-1 **:** **Première méthode** On effectue un produit par block et on pose

$$\begin{array}{ll} I = A\,E + B\,G & J = A\,F + B\,H \\ J = C\,E + D\,G & L = C\,F + D\,H \end{array}$$

Q-1-1 **:** Vérifier la correction des équations proposées.

Q-1-2 **:** Écrire un algorithme `ProduitMat` utilisant la méthode décrite. Vérifier qu'il respecte le principe *diviser pour régner*. On justifiera la *condition de terminaison* choisie.

Q-1-3 : Vérifier que l'équation satisfaite par la fonction de complexité en temps est

$$f(n) = 8f(\frac{n}{4}) + n$$

où n est la taille de la matrice (9 pour une matrice carrée 3×3)

Q-1-4 : Vérifier que la solution de cette équation est $f(n) = \mathcal{O}(n^{\frac{3}{2}})$

On utilisera le résultat général suivant : soient $a, b > 1$ deux entiers on a

$$g(n) = ag(\frac{n}{b}) + n^k \Longrightarrow g(n) = \begin{cases} \mathcal{O}\left(n^{\log_b(a)} \log(n)\right) & \textit{si} \quad a = b^k \textit{ c'est-à-dire } k = \log_b(a) \\ \mathcal{O}\left(n^{\log_b(a)}\right) & \textit{si} \quad a > b^k \\ \mathcal{O}(n^k) & \textit{si} \quad a < b^k. \end{cases}$$

Q-1-5 : Calculer la complexité en espace de cet algorithme.

Q-1-6 : Conclure en comparant cet algorithme avec une autre solution à ce problème.

Q-2 : Seconde méthode : Méthode de Strassen Ici on pose :

$$\begin{array}{ll} P_1 = A\,(G - H) & P_5 = (A + D)(E - F) \\ P_2 = (A + B)\,H & P_6 = (B + D)(E + F) \\ P_3 = (C + D)\,E & P_7 = (A - C)(E + G) \\ P_4 = D\,(F - E) & \end{array}$$

Puis

$$\begin{array}{ll} I = P_5 + P_4 - P_2 + P_6 & J = P_1 + P_2 \\ K = P_3 + P_4 & L = P_5 + P_1 - P_3 - P_7 \end{array}$$

Q-2-1 : Vérifier la correction des équations proposées.

Q-2-2 : Écrire un algorithme `ProduitMat` utilisant la méthode décrite. Vérifier qu'il respecte le principe *diviser pour régner*. On justifiera la *condition de terminaison* choisie.

Q-2-3 : Vérifier que l'équation satisfaite par la fonction de complexité en temps est

$$f(n) = 7f(\frac{n}{4}) + n$$

où n est la taille de la matrice (9 pour une matrice carrée 3×3)

Q-2-4 : Vérifier que la solution de cette équation est $f(n) = \mathcal{O}(n^{\log_4(7)}) \approx \mathcal{O}(n^{1,403})$

Q-2-5 : Calculer la complexité en espace de cet algorithme.

Q-2-6 : Conclure en comparant cet algorithme à celui proposé précédemment.

Exercice - 9 *Inconvénients de la récursivité et essais de suppression*

Q-1 : Présenter quelques inconvénients de la récursivité

Q-2 : Donner deux exemples d'algorithmes récursifs, et présenter un moyen d'y supprimer la récursivité.