R5.04 Qualité Algorithmique TDn°4

Diviser pour régner

Exercice 1 : Etude du tri rapide

Le tri rapide est un algorithme de tri récursif basé sur la technique de diviser pour régner. Il fonctionne en choisissant un élément appelé pivot dans la liste et en partitionnant la liste autour du pivot, plaçant tous les éléments plus petits à sa gauche et tous les éléments plus grands à sa droite. Ensuite, le tri rapide est appliqué récursivement aux sous-listes résultantes de part et d'autre du pivot jusqu'à ce que toute la liste soit triée.

Étapes du processus :

1. **Choix du Pivot :** On choisit un élément de la liste comme pivot. Le choix du pivot peut varier (premier élément, dernier élément, élément médian, etc.). Un bon choix de pivot peut améliorer les performances.
2. **Partitionnement :** On partitionne la liste autour du pivot de telle sorte que les éléments plus petits que le pivot soient à gauche et les éléments plus grands soient à droite. Cela peut être fait en déplaçant les éléments de manière appropriée.
3. **Appels Récursifs :** On applique récursivement le tri rapide aux sous-listes à gauche et à droite du pivot.
4. **Combinaison :** Les sous-listes triées résultantes sont combinées pour former la liste triée complète.

A. Illustrez par un schéma le tri rapide sur les deux listes ci-dessous de n=12 éléments.

	0	1	2	3	4	5	6	7	8	9	10	11
L1	23	24	23	60	63	17	48	12	15	59	40	20

	0	1	2	3	4	5	6	7	8	9	10	11
L2	6	7	10	13	16	18	21	24	25	27	28	29

A. Ecrire une fonction de partitionnement de la liste L en deux sous-listes L' et L'' suivant un pivot piv, tel que tous les éléments dans L' sont inférieurs à piv et tous les élément de L'' sont supérieurs à piv. Cette fonction retournera la position du pivot dans la liste ainsi partitionnée.

B. Quelle est le coût de cette fonction ?

C. Ecrire l'algorithme récursif du tri rapide faisant appel à cette fonction.

D. Définir le cas favorable et le cas défavorable du tri rapide. Illustrez par un schéma.

Exercice 2 : somme de trois...améliorer la complexité

Étant donné un tableau T de taille n, on veut écrire un algorithme qui trouve trois indices distincts i, j et k dans $\{0, \ldots, n-1\}$ tels que $T[i] + T[j] = T[k]$, ou qui signale si trois tels indices n'existent pas.

A. Écrire un premier algorithme naïf de complexité en temps $O(n^3)$ pour résoudre ce problème et testez le sur des tableau de taille croissante. Constatez les limites des algorithmes de telle complexité. Quelle est la taille de tableau pour laquelle le temps d'exécution est inférieure à 1 minute ?

B. On souhaite améliorer la qualité de notre solution et fournir un algorithme de compléxité quadratique. Pour cela on se propose de trouver un algorithme linéaire en temps, qui résolve le sous-problème suivant : étant donné un tableau ***T trié*** de taille n et un nombre x, existe t-il deux indices distincts i et j du tableau tels que $T[i] + T[j] = x$
(Indice : on peut commencer par comparer $T[0] + T[n-1]$ et x).

C. En déduire un algorithme de complexité en temps quadratique pour résoudre le problème initial. L'écrire en pseudocode et justifier sa complexité.

Exercice 3 : Elément majoritaire

On dit qu'un tableau a un élément majoritaire si une valeur du tableau est présent au moins $\lfloor n/2 \rfloor$+1 fois sur n éléments.

1 – Proposer un algorithme itératif qui retourne la valeur de l'élément majoritaire, dès que celui-ci est trouver et qui retourne -32000 si aucun élément majoritaire n'apparait dans le tableau. Vous ferez en sorte que le nombre de comparaisons d'éléments soit le plus petit possible, même dans cette version itérative. Enfin,vous en déterminerez la complexité.

2 - Proposer une méthode pour résoudre ce problème en **diviser pour régner** et faire un schéma afin de l'expliquer.

La division doit s'effectuer sur le tableau :

T	T_1	T_2
	n/2	n/2

Discutez pour cela des différents cas :

- si x est majoritaire dans T_1 et dans T_2
- si x est majoritaire dans T_1 mais pas dans T_2
- si x est majoritaire ni dans T_1 ni dans T_2

3 – Ecrire la méthode décrite par un algorithme.