

Exercice I :

On voudrait représenter un arbre n-aire sur un arbre binaire (méthode vue en cours).

Nous utilisons, ci-après, les deux règles :

Le fils aîné d'un noeud n-aire est représenté par le fils gauche du noeud binaire.

Le frère cadet d'un noeud n-aire est représenté par le fils droit du noeud binaire.

Ecrire les algorithmes suivants :

- Afficher les descendants d'un noeud.
- Afficher le nombre de noeuds d'un arbre n-aire.
- Afficher les feuilles d'un arbre n-aire.

Exercice II :

Soit un tableau de N nombres. On voudrait avoir les I plus grands d'entre eux, mais affichés triés.

Pour chacune des deux méthodes suivantes, donner la complexité en fonction de N et de I . Quelle est la meilleure méthode ?

1. Trier les N nombres et renvoyer les I plus grands.
2. Trouver le I ème plus grand élément (à l'aide d'un algorithme linéaire), partitionner et trier les I plus grands.

Exercice III :

Le responsable de la bibliothèque veut réorganiser les étagères.

Il possède N livres b_1, b_2, \dots, b_n .

Chaque livre b_i a une largeur w_i et une hauteur h_i . Les livres doivent être rangés dans l'ordre croissant de i sur des étagères de largeur L .

1. On ne prend pas en considération la hauteur des livres. Ecrire un algorithme pour ranger tous les livres sur les étagères en minimisant le nombre d'étagères. Toutes les étagères ont la même largeur L .
2. Votre algorithme est-il optimal ? Si c'est le cas, démontrez.

3. Prenons en compte maintenant la hauteur des étagères. Nous cherchons maintenant à minimiser l'encombrement, défini comme la somme des hauteurs du plus grand livre de chaque étagère utilisée. Donnez un exemple où l'algorithme précédent ne trouve pas la solution optimale.

Exercice IV:

Soient $A[1..n]$ et $B[1..n]$ deux tableaux triés par ordre croissant. On cherche à trouver l'élément médian de ces deux tableaux (élément qui a autant d'éléments supérieurs stricts que d'éléments inférieurs ou égaux).

Proposez un algorithme diviser pour régner qui trouve le médian en $O(\log n)$.

Justifiez la complexité.

Exercice V :

Soient deux méthode de représentation d'un ensemble de valeurs entières.

1. Une matrice d'entiers.
2. Un tableau de $N \times M$ enregistrements. Chaque enregistrement contient trois champs (Numéro de ligne, Numéro de colonne et la valeur entière).

Ecrire, pour chacun des deux types de représentation, un algorithme qui calcule la somme des valeurs. Ensuite, comparer leur complexité spatiale (taille mémoire occupée) et leur complexité temporelle (nombre d'opérations à effectuer).

Exercice VI:

Soit un tableau de 1000 entiers.

Ecrire les algorithmes de recherche suivants :

1. Recherche séquentielle d'une valeur sur le tableau non trié.
2. Recherche séquentielle d'une valeur sur le tableau trié.
3. Recherche dichotomique d'une valeur sur le tableau non trié.
4. Recherche dichotomique d'une valeur sur le tableau trié.

Pour chacun de ces algorithmes, combien de comparaisons sont exécutées pour :

- trouver un élément qui existe ?
- trouver un élément qui n'existe pas ?

Quels sont les cas où le tableau est parcouru entièrement et les cas où un parcours partiel est suffisant ? Donner, alors, la complexité temporelle pour chaque algorithme.

Exercice VII:

Soit un tableau de caractères en majuscule.

On voudrait calculer le nombre d'occurrences de chacun des 26 caractères de l'alphabet en majuscule.

- Ecrire l'agorithme qui fait le calcul en parcourant 26 fois le tableau.
- Ecrire l'agorithme qui fait le calcul en parcourant une seule fois le tableau.

Comparez les deux complexités.