Diviser pour régner

Exercice 1 *Nombre d'inversions d'un tableau*

Pour un tableau T indexé de 1 à n, le nombre d'inversions de T est défini par

$$\text{inv}(T) = |\{1 \leqslant i < j \leqslant n \ : \ T[i] > T[j]\}|.$$

1. Adapter l'algorithme de tri fusion pour concevoir un algorithme qui trie un tableau T donné en entrée et retourne $\text{inv}(T)$.

Exercice 2 *Multiplication rapide de matrices*

Le but de cet exercice est de concevoir un algorithme de produit de deux matrices carrées, plus efficace que l'algorithme naïf. L'idée est la même que pour le produit rapide de polynômes vu en cours.

1. Rappeler quelle est la complexité du produit naïf de deux matrices de tailles $n \times n$.
2. Comment calculer les quatre coefficients de la matrice

$$\begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{pmatrix} \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

   avec seulement 7 multiplications (et sans faire commuter les éléments) ?
   Indication : Montrer que les éléments cherchés peuvent être obtenus comme combinaison linéaire des produits suivants :
   - $(a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2})$,
   - $(a_{2,1} + a_{2,2})b_{1,1}$,
   - $a_{1,1}(b_{1,2} - b_{2,2})$,
   - $a_{2,2}(b_{2,1} - b_{1,1})$,
   - $(a_{1,1} + a_{1,2})b_{2,2}$,
   - $(a_{2,1} - a_{1,1})(b_{1,1} + b_{1,2})$,
   - $(a_{1,2} - a_{2,2})(b_{2,1} + b_{2,2})$.

3. En déduire un algorithme récursif pour faire le produit de deux matrices de taille $n \times n$, où n est une puissance de 2.
4. Déterminer la complexité de l'algorithme décrit ci-dessus. Comparer à la méthode naïve.

Exercice 3 *Algorithme de sélection*

Étant donné un tableau $T[1 \dots n]$, considérons π une permutation de $\{1, \dots, n\}$ qui trie le tableau, c'est-à-dire que

$$T[\pi(1)] \leqslant T[\pi(2)] \leqslant \dots \leqslant T[\pi(n)].$$

Le problème de la sélection du i-ème plus petit élément d'un tableau consiste, étant donné T, à calculer $T[\pi(i)]$. Noter qu'il peut exister plusieurs permutations π triant le tableau (c'est le cas si et seulement si le tableau contient plusieurs fois le même élément), mais $T[\pi(i)]$ est indépendant du choix de π puisqu'il n'existe qu'une version triée du tableau T. Nous appellerons cet élément l'élément de rang i de T.

1. Proposer un algorithme simple qui étant donné T et i retourne l'élément de rang i de T. Quelle est sa complexité ?

Le but de la suite est de construire un algorithme plus efficace. Cet algorithme est récursif. Dans la suite, nous noterons $|A|$ la taille d'un tableau A, c'est-à-dire le nombre d'éléments qu'il contient.

Étant donné un élément v du tableau $T[1, \dots, n]$, on peut collecter dans un tableau A les éléments de T qui sont strictement inférieurs à v, dans X les éléments égaux à v et dans un tableau B les éléments qui sont strictement supérieurs à v. Trois cas se présentent alors :

— Si $|A| < i \leqslant |A| + |X|$, l'élément de rang i de T est v. Nous pouvons donc retourner v et terminer l'algorithme.

— Si $i \leqslant |A|$, l'élément de rang i de T est l'élément de rang i de A. Dans ce cas, on calcule récursivement l'élément de rang i de A.

— Si $i > |A| + |X|$, l'élément de rang i de T est l'élément d'un certain rang r de B, que nous pouvons calculer récursivement.

2. Déterminer la valeur de r dans le troisième cas ci-dessus.

La partie délicate consiste à calculer un élément v qui partitionne le tableau "à peu près en deux parties égales". Pour cela, on effectue les étapes suivantes :

— On partitionne le tableau T en paquets de 5 éléments (le dernier paquet peut comporter éventuellement moins de 5 éléments, si n n'est pas un multiple de 5) ;

— On calcule par un algorithme naïf la médiane de chaque paquet ;

— On calcule récursivement la médiane des médianes : c'est cet élément que nous choisissons pour v.

3. Écrire l'algorithme de sélection (à un haut niveau) basé sur le principe ci-dessus.
4. Comment démontrer sa correction ?

Dans la suite, on s'intéresse à la complexité de cet algorithme.

5. Montrer que l'élément v vérifie la propriété suivante :

$$|A| \leqslant \frac{7n}{10} + 3$$

et de même pour B.

6. En déduire que la complexité de l'algorithme de sélection est $O(n)$.