Série d'exercices N^{o}5

ING1 – Algorithmique procédurale

1 Complexité : exemples simples

1 Calculez la complexité des algorithmes suivants :

1. Calcul du maximum et du minimum dans un tableau d'entiers.
2. Multiplication de deux matrices. □

2 Comparaison des tris simples

Le **tri par insertion** considère chaque élément du tableau et l'insère à la bonne place parmi les éléments déjà triés. Ainsi, au moment où on considère un élément, les éléments qui le précèdent sont déjà triés, tandis que les éléments qui le suivent ne sont pas encore triés. Pour trouver la place où insérer un élément parmi les précédents, il faut le comparer à ces derniers, et les décaler afin de libérer une place où effectuer l'insertion.

Principe : A tout moment, le tableau est séparé en deux parties :

— de $A[1]$ à $A[deb-1]$: partie déjà triée du tableau

— de $A[deb]$ à $A[taille(A)]$ partie non triée du tableau

Au cours d'une étape :

— prendre un élément non encore trié ;

— l'insérer à sa place dans l'ensemble des éléments triés.

Le **tri par sélection** sélectionne la valeur minimum (ou maximum) du tableau afin de la placer en début (ou en fin) de celui-ci, la plaçant ainsi à sa position finale. Le reste du tableau est cependant encore en désordre. Il faut donc répéter les sélections de minimums (ou maximums) dans tableau, sans prendre en compte les éléments déjà bien positionnés, jusqu'à ce que tous les éléments soient placés à leur position définitive.

Principe :

— On commence par trouver le plus petit élément du tableau et on le permute avec $A[1]$.

— Puis on trouve ensuite le deuxième plus petit élément du tableau et on le permute avec $A[2]$.

— On continue de procédé pour les $n-1$ premiers éléments de A.

Le **tri à bulles** consiste à comparer répétitivement les éléments consécutifs d'un tableau, et à les permuter lorsqu'ils ne sont pas dans l'ordre. Après un premier parcours complet du tableau, le plus grand élément est forcément en fin de tableau, à sa position définitive. Le reste du tableau est cependant encore en désordre. Il faut donc répéter les parcours du tableau, jusqu'à ce que tous les éléments soient placés à leur position définitive.

Principe :

— On effectue une succession de parcours du tableau.

— A chaque parcours, on compare tout couple de cases consécutives $A[j]$ et $A[j+1]$. Si ces deux cases ne sont pas ordonnées, alors on les échange.

— Lorsqu'aucun échange n'est effectué lors d'un parcours, alors le tableau est trié et l'algorithme s'arrête.

(2) Pour chacun de ces trois tris simples :

1. Écrivez l'algorithme.
2. Calculez la complexité dans le meilleur et dans le pire des cas.

Comparez ces trois tris. □

3 Complexité moyenne

(3) Rappelez la complexité (au pire des cas) de la recherche d'un élément dans un tableau d'entiers (quelconque) de taille n. □

(4) Calculez la complexité moyenne de cet algorithme en supposant que :

— la probabilité que l'entier recherché est présent dans le tableau est égale à p ($0 \leq p \leq 1$).

— si l'entier est présent dans le tableau, alors toutes les positions sont équi-probables. □

4 Complexité optimale

Notre objectif dans cet exercice est de calculer la complexité **optimale** du tri en admettant que les opérations de bases sont la **comparaison** et la **permutation**.

(5) Expliquez (brièvement) pourquoi trier un tableau revient à appliquer une permutation σ à l'ensemble des indices des éléments du tableau $\{1, 2, ..., n\}$. Rappelez le nombre de permutations sur cet ensemble. □

(6) Expliquez comment on peut modéliser la suite des opérations du tri (conduisant du tableau initial au tableau trié) par un **arbre binaire**. □

(7) Construisez un tel arbre dans le cas d'un tableau à 3 éléments.
Quelle(s) relations y-a-t-il entre cet arbre et les permutations de la première question ? □

(8) Déduisez-en le nombre de feuilles puis la hauteur de cet arbre.
Que peut-on en déduire concernant la complexité du tri ? □