Algorithmique - TP2

Produit de matrices

Soient deux matrices A = (a_{ij}) et B = (b_{ij}) de taille m×n et n×p. Le produit de ces deux matrices, AB = (c_{ij}) de taille m×p, est donné par $\forall i, j : c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj}$.

Écrivez un programme qui réalise un produit de matrices, et testez le avec l'exemple suivant :

$$\begin{pmatrix} 5 & 1 \\ 2 & 3 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 \\ 4 & 3 & -1 \end{pmatrix} = \begin{pmatrix} 9 & 13 & -1 \\ 14 & 13 & -3 \\ 19 & 18 & -4 \end{pmatrix}.$$

Quel est le nombre d'opérations "élémentaires" (addition, multiplication) réalisées par votre programme pour cet exemple ?

Algorithmes de recherche

Pour chacun des programmes suivants, commencez par **réfléchir** à vos algorithmes: penser à la façon dont vous procéderiez si vous deviez réaliser vous même la tâche "à la main", écrire l'algorithme en langage "naturel", le "dérouler" sur un exemple.

Soit un tableau d'entiers quelconque. Écrivez un programme qui recherche une valeur dans ce tableau et affiche son indice (-1 si la valeur ne figure pas dans le tableau). Testez votre programme dans différentes conditions :

- la valeur recherchée figure au début, au milieu ou à la fin du tableau
- la valeur recherchée ne figure pas dans le tableau

Combien de comparaisons sont effectuées par votre programme dans chacun de ces cas ?

Écrivez un nouveau programme qui recherche un mot dans un "dictionnaire" (tableau de mots **trié**).

Écrivez un programme qui recherche si une chaine de caractères est présente à l'intérieur d'une autre.

Écrivez un programme qui remplace toutes les occurences d'une chaine de caractères par une autre dans un texte.

Jeu de la pyramide

Écrivez un programme qui permet de jouer au jeu de la pyramide. Dans ce jeu, chaque joueur propose un mot formé des lettres du mot du joueur précédent, en lui ajoutant une lettre. Par exemple : pal, lape, palie, paille,etc.

Le programme doit vérifier qu'un mot proposé par un joueur est valable en cherchant dans une liste de mots triés par ordre alphabétique (Lexique Gutenberg).

Implémenter une recherche séquentielle et une recherche dichotomique.

Algorithmes de tri

Le tri est un problème récurrent en algorithmique. Tout programmeur devra un jour ranger des valeurs dans un ordre donné. Pour trier des données, il existe de nombreuses stratégies éprouvées. Vous implémenterez les algorithmes suivants pour des tableaux de caractères. Pour réfléchir à vos algorithmes, fabriquez un "jeu de cartes", chacune comportant une lettre ; mélangez ces cartes, puis tentez de les ranger en appliquant l'algorithme présenté.

1. Tri par sélection : le plus simple, mais pas le plus efficace... Le principe est le suivant :

1. rechercher le plus petit élément, puis l'échanger avec l'élément d'indice 0
2. rechercher le second plus petit élément, puis l'échanger avec l'élément d'indice 1
3. continuer jusqu'à ce que le tableau soit entièrement trié

2. Tri par insertion : c'est la technique "naturellement" utilisée pour trier un jeu de carte. On prend les cartes mélangées une à une sur la table, et on forme une main en insérant chaque carte à sa place. Avec un tableau, on parcourt le tableau à trier du début à la fin. Au moment où on considère le ième élément, les éléments qui le précèdent sont déjà triés. L'objectif d'une étape est d'insérer le ième élément à sa place parmi ceux qui précèdent, en décalant au fur et à mesure les éléments qui sont plus grands que lui.

3. Tri fusion : algorithme le plus efficace des trois, qui utilise le principe de la récursivité (l'algorithme s'appelle lui-même). L'algorithme consiste à :

1. découper le tableau en deux parties,
2. trier chaque partie (en utilisant le tri fusion lui-même),
3. fusionner les deux parties

La récursivité s'arrête quand on arrive à un tableau composé d'un seul élément.

4. **Pour finir**, évaluer la complexité empirique de ces trois algorithmes de tri.