

Algorithmique et Programmation Impérative

TD3 (séances 4 et 5) – Recherche, Tri et Complexité

Nous rappelons que l'objectif de ce module est d'apprendre les concepts de l'algorithmique et non pas un langage de programmation précis. Cependant, nous allons utiliser Python pour les TDs puisque c'est un langage que vous avez appris à l'ENSIBS.

Nous tenons à noter que nous faisons de notre mieux pour adapter ce cours à l'enseignement à distance tout en tenant compte des contraintes actuelles et en adaptant les modalités du cours, la charge de travail ainsi que les modalités d'évaluation pour s'assurer que chacun de vous a acquis les compétences minimales nécessaires pour valider cette matière. Nous restons à votre écoute si vous avez des suggestions d'amélioration et/ou des critiques constructives.

I. Produit des matrices

Soient deux matrices $A = (a_{ij})$ et $B = (b_{ij})$ de taille $m \times n$ et $n \times p$. Le produit de ces deux matrices, $AB = (c_{ij})$ de taille $m \times p$, est donné par :

$$\forall i, j : c_{ij} = \sum_{k=1}^n a_{ik} b_{kj} = a_{i1}b_{1j} + a_{i2}b_{2j} + \dots + a_{in}b_{nj}$$

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

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

2. Quel est le nombre d'opérations "élémentaires" (addition, multiplication) réalisées par votre programme pour cet exemple ?
3. Quelle est la complexité algorithmique dans le cas général du produit de deux matrices $(m \times p) * (p \times n)$?

II. 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" et le "dérouler" sur un exemple.

1. 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
2. Combien de comparaisons sont effectuées par votre programme dans chacun de ces cas ?
3. Écrivez un nouveau programme qui recherche un mot dans un "dictionnaire" (tableau de mots **trié**).
4. Écrivez un programme qui recherche si une chaîne de caractères est présente à l'intérieur d'une autre.
5. Écrivez un programme qui remplace toutes les occurrences d'une chaîne de caractères par une autre dans un texte.

III. 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,...

1. Écrivez un programme qui permet de jouer au jeu de la pyramide. Le programme doit vérifier qu'un mot proposé par un joueur est valable...

IV. 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 chacun des tri suivants, commencez par :

- A) Réfléchir à vos algorithmes : penser à la façon dont vous procéderiez si vous deviez réaliser vous même la tri "à la main" et écrire l'algorithme en langage "naturel". 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.
- B) Puis, implémenter votre algorithme en Python 3

1. Tri par sélection : le plus simple, mais pas le plus efficace...

Le principe est le suivant :

- a) Rechercher le plus petit élément, puis l'échanger avec l'élément d'indice 0
- b) Rechercher le second plus petit élément, puis l'échanger avec l'élément d'indice 1
- c) 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.

- a) Avec un tableau, on parcourt le tableau à trier du début à la fin
- b) Au moment où on considère le i ème élément, les éléments qui le précèdent sont déjà triés
- c) L'objectif d'une étape est d'insérer le i ème 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 par fusion : algorithme le plus efficace des trois, qui utilise le principe de la récursivité (l'algorithme s'appelle lui-même).

Le principe est le suivant :

- a) Découper le tableau en deux parties
- b) Trier chaque partie (en utilisant le tri fusion lui-même)
- c) Fusionner les deux parties
- d) La récursivité s'arrête quand on arrive à un tableau composé d'un seul élément