Feuille de TP N°02 – Algorithmes de tri

Exercice 01 : La fonction *« qsort »*

L'objectif de cet exercice est d'apprendre à utiliser la fonction prédifine ***« qsort »***. Cette dernière utilise l'algorithme « tri rapide » (à voir dans la section 2.7 du cours) pour trier les éléments d'un tableau.

La fonction ***« qsort »*** est définie dans la bibliothèque ***« stdlib.h »***, la fonction prend quatre arguments :

- Le premier paramètre est le tableau à trier.
- Le deuxième paramètre est le nombre d'éléments du tableau.
- Le troisième paramètre est la taille en octets de chaque élément du tableau.
- Le quatrième paramètre est un pointeur sur une fonction de comparaison.

La fonction de comparaison prend deux paramètres et retourne un entier. Elle doit retourner un entier négatif si le premier paramètre doit être placé avant le deuxième dans le tableau trié, un entier positif si le premier paramètre doit être placé après le deuxième et zéro si les deux paramètres sont égaux.

1. Ecrire un programme qui utilise la fonction « qsort » pour trier un tableau d'entiers saisis par l'utilisateur dans un ordre croissant puis dans un ordre décroissant. Le programme doit demander à l'utilisateur le nombre d'entiers à saisir, puis allouer dynamiquement le tableau et enfin le remplir avec les entiers saisis par l'utilisateur. Le tableau doit être affiché sur écran avant le tri, puis dans un ordre croissant et décroissant.
2. Modifier le programme précédent pour lire et trier un tableau de chaines de caractères dans un ordre alphabétique croissant.

Indication : utiliser la fonction ***« strcmp »*** de la bibliothèque ***« string.h »*** pour comparer deux chaines de caractères.

Exercice 02 : Chercher trois nombres avec une somme donnée

L'objectif de cet exercice est d'écrire une fonction « somme3 » qui prend comme argument un tableau T et un entier S et qui retourne vrai si le tableau T contient trois nombres dont la somme est égale à S. Par exemple, si T = [1, 2, 3, 4, 5] et S = 9, la fonction doit retourner vrai car 2 + 3 + 4 = 9.

- Commencer par implémenter une version naïve de cette fonction qui consiste à tester toutes les triplets possibles de nombres dans le tableau. Quelle est la complexité de cette fonction ?
- Comment peut-on utiliser les algorithmes de tri pour améliorer la complexité de cette fonction ? Implémenter cette version.
- Comparer le temps d'exécution des deux versions de la fonction ***« somme3 »*** en fonction de la taille du tableau ***T***. Pour cela, générer des tableaux aléatoires de taille croissante et calculer le temps d'exécution de chaque version de la fonction ***« somme3 »*** pour chaque tableau.

Indication : utiliser la fonction ***« qsort »*** pour trier le tableau.

Exercice 03 : L'élément majoritaire

L'objectif de cet exercice est d'écrire une fonction ***« majoritaire »*** qui prend comme argument un tableau ***T*** et qui retourne l'élément majoritaire du tableau s'il existe, et -1 sinon. Un élément majoritaire est un élément qui apparaît plus de ***n/2*** fois dans le tableau où ***n*** est la taille du tableau (au moins ***n / 2 + 1*** fois si ***n*** est pair). Par exemple, si ***« T = [1, 2, 3, 4, 7, 7, 7, 7, 7] »***, la fonction doit retourner 7 car il apparaît 5 fois dans le tableau et 5 > 9 / 2.

- Commencer par implémenter une version naïve de cette fonction qui teste chaque élément du tableau. Quelle est la complexité de cette fonction ?
- Comment peut-on utiliser les algorithmes de tri pour améliorer la complexité de cette fonction ? Implémenter cette version.
- Comparer le temps d'exécution des deux versions de la fonction ***« majoritaire »*** en fonction de la taille du tableau ***T***. Pour cela, générer des tableaux aléatoires de taille croissante et calculer le temps d'exécution de chaque version de la fonction pour chaque tableau.

Indication : utiliser la fonction ***« qsort »*** pour trier le tableau.

Exercice 04 : Tri de trois couleurs

Soit un tableau ***T*** contenant ***n*** objets de trois couleurs différentes : rouge, blanc et bleu. L'objectif de cet exercice est d'écrire une fonction ***« tri3couleurs »*** qui prend comme argument un tableau ***T*** et qui trie les objets de ce tableau par couleur. Nous allons utiliser la convention suivante pour représenter les couleurs :

- 0 pour le rouge.
- 1 pour le blanc.
- 2 pour le bleu.

Par exemple, si ***T = [1, 0, 2, 1, 0, 2, 1, 0, 2]***, la fonction doit retourner ***[0, 0, 0, 1, 1, 1, 2, 2, 2]***.

- Comment peut-on utiliser la fonction ***« qsort »*** pour trier le tableau ***T*** ?
- Serait-il possible de trier le tableau T en un seul passage ? Si oui, écrire une fonction ***« tri3couleurs »*** qui trie le tableau T en un seul passage (avec une complexité ***O(n)***).
- Comparer le temps de la fonction ***« tri3couleurs »*** avec la fonction ***« qsort »***. Pour cela, générer des tableaux aléatoires de taille croissante et calculer le temps d'exécution de chaque fonction pour chaque tableau. Qu'observez-vous ? Donnez une explication.

Indication : faites des recherches sur le problème du drapeau hollandais et l'algorithme de Partitionnement à trois voies.