Travaux Dirigés NSI : Cours Algo1

Exercice 1

La leçon a montré que le tri par sélection a une complexité en temps en $O(n^2)$. On suppose que sur une machine, notre fonction de tri par sélection met 6,8 secondes à trier une liste de 16000 nombres sur notre ordinateur. Combien de temps faudra-t-il approximativement pour trier un tableau de un million de nombres avec la même fonction?

Exercice 2

Écrire une fonction *estDecroissant(tableau)* qui renvoie True si le tableau est trié en ordre décroissant et False sinon.

Exercice 3

Que se passe-t-il quand le tri par insertion est appliqué à un tableau en rangé en ordre décroissant ?

Exercice 4

Plutôt que de trier en place le tableau donné en entrée comme cela a été fait dans la leçon, on peut aussi renvoyer un nouveau tableau sans modifier le tableau donné en entrée. Écrire une fonction *triInsertionExterne(tableau)* qui réalise cette idée avec le tri par insertion.

Exercice 5

Écrire une fonction prenant un tableau de nombres en argument et qui affiche son contenu sous la forme d'un histogramme. Les nombres du tableau doivent être compris entre 0 et 10000.

Le résultat attendu est de la forme :
1 fois 0
2 fois 3
3 fois 4
1 dois 7
etc

Exercice 6

En se servant du tri par insertion, écrire une fonction prenant en entrée un tableau de nombre positifs triés et qui renvoie la valeur la plus fréquente de ce tableau et le nombre d'occurrences de cette valeur. **Indication :** une fois le tableau trié, les valeurs égales se trouvent côte à côte et il devient facile de compter les occurrences. Un simple boucle et quelques variables suffisent !

Exercice 7

Écrire une fonction *trier2valeurs(tableau)* qui trie un tableau de nombres aléatoires composé exclusivement de 0 et de 1. On cherche une fonction la plus performante possible. Tenter de calculer la complexité de la fonction produite….on peut le faire en $O(n)$!

Exercice 8

On se place dans le cadre des tableaux aléatoires de nombres dont les valeurs sont comprises entre deux entiers de tailles raisonnables...par exemple entre -1000000 et 1000000. On peut trier ces tableaux en comptant les occurrences de chacun des éléments de ce tableau dans un autre tableau, puis en les réécrivant ces valeurs en un seul passage dans le tableau donné en entrée. Écrire une fonction *triDenombrement(tableau)* qui réalise cet algorithme. Quelle est sa complexité ? Tenter une justification..