

TP5 – Algorithmes de tri

Exercice 1 : Tableau trié ou non

- Ecrire une fonction « `est_trié(t)` » qui renvoie `True` si le tableau est trié par ordre croissant et `False` sinon.
- On testera la fonction à l'aide d'un tableau généré aléatoirement, et d'un tableau trié à l'aide de la fonction tri native de python.

Exercice 2 : Tri par insertion

1. Ecrire une fonction « `insère(t,i,v)` » qui insère la valeur `v` dans un tableau `t[0, ..., i, ...]` déjà trié jusqu'à la position `i`. Par exemple `insère([1,2,3],2,0)` donnera comme résultat : `[0,1,2]`. La valeur 3 est exclue du tableau et 0 est inclus afin de garder une liste triée. Par contre `insère([1,2,3],1,0)` va travailler sur les deux premiers éléments et insérer 0 dans la liste triée `[1,2]`, ainsi on aura `[0,1,3]`. (3 n'est pas affectée car considérée dans une liste non triée)
2. A l'aide de la fonction `insère`, écrire une fonction « `tri_insertion(t)` » qui trie un tableau quelconque `t` dans l'ordre croissant.
3. Ecrire une fonction « `plus_petits` » qui prend en paramètres un tableau `t` et un entier `k` supposé inférieur à la longueur de `t` et qui renvoie dans l'ordre les `k` plus petits éléments de `t`. On se servira de la fonction « `insère` ».

Exercice 3 : Tri à bulles

Le tri à bulles consiste à comparer les deux premiers éléments d'une liste `L` et à les échanger s'ils ne sont pas triés par ordre croissant. On recommence ensuite avec le deuxième et le troisième élément de la liste, et ainsi de suite...Après chaque parcours complet de la liste, on recommence l'opération, ainsi les plus grands éléments remontent de proche en proche vers la droite comme des bulles vers la surface. Lorsqu'aucun échange a lieu pendant un parcours, cela signifie que la liste est triée et on arrête l'algorithme.

1. Ecrire une fonction « `tri_bulles(L)` » réalisant cette opération
2. Evaluer la complexité du tri.

Exercice 4 : Comptage

Ecrire une fonction « `comptage(t)` » qui prend en argument un tableau d'entiers trié et affiche son contenu sous la forme d'un histogramme du type : 1 fois 0, 2 fois 2, 1 fois 3...

Exercice 5 : Tri au choix

On dispose de points dans le plan muni d'un repère orthonormé d'origine `O`. Ces points possèdent un couple de coordonnées $(x ; y)$ représenté par la liste `[x,y]`. Nous allons trier ces points en fonction de leur distance à `O`, de la plus petite à la plus grande distance

1. Ecrire une fonction `distance` qui prend en paramètre une liste de deux nombres nommée « `point` » et renvoie le carré de la distance de ce point `P` à `O`. « `point` » est la liste des coordonnées d'un point `P`.
2. Ecrire une fonction « `compare` » qui prend en paramètres deux listes `p1` et `p2` représentant deux points `P1` et `P2` et qui renvoie `-1` si `P1` est plus proche de `O` que `P2`, `1` si `P2` est plus proche de `O` que `P1`, et `0` si les deux points sont équidistants de `O`.
3. Ecrire une fonction « `tri` » qui prend en paramètre une liste composée de listes de deux nombres représentant des points du plan et qui trie cette liste suivant la distance entre les points et `O`. Utiliser un algorithme de tri du cours.

Exercice 6 : Temps d'exécution

Pour mesurer le temps d'exécution d'un programme, on importe la fonction `time` du module `time`. On souhaite comparer les temps d'exécution du tri insertion et du tri fusion sur deux types de listes.

1. Construire une liste de 3000 entiers pris au hasard entre 1 et 3000, bornes comprises. Mesurer les temps d'exécution des programmes du tri insertion et du tri fusion pour trier cette liste. Quel commentaire peut-on faire concernant les deux résultats ?
2. Construire la liste des 3000 entiers de 1 à 3000, bornes comprises. Mesurer le temps d'exécution du programme du tri insertion et du programme du tri fusion pour trier cette liste. Quel commentaire peut-on faire concernant les deux résultats ?