Informatique PCSI
TP 8 : algorithmes de tri

1 Énoncé des exercices

1.1 Exercice 1

Écrire une fonction `permute` qui prend en argument une liste de mots et modifie la liste en permutant le mot le plus court en nombre de lettres avec le premier mot de la liste. La fonction ne renvoie rien.

Tester la fonction avec la liste `['toto', 'bonjour', 'a', 'oui', 'non']`.

C'est une recherche de minimum, sur le nombre de lettres d'un mot. Une variable est nécessaire pour stocker l'indice du minimum.

1.2 Exercice 2

Ordre lexicographique

L'objectif est d'écrire un programme qui trie une liste de mots (type str) et les range suivant l'ordre lexicographique (l'ordre des dictionnaires).

1. Écrire la définition de la variable `alphabet` :
   alphabet ="AaàBbCcDdEeéèFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuùVvWwXxYyZz"
2. Écrire une fonction `ordre_alphabetique` qui prend en arguments deux caractères alphabétiques `c1` et `c2` et renvoie -1 si `c1` est avant `c2`, 1 si `c2` est avant `c1` et 0 si `c1` $=$ `c2`. On pourra utiliser la méthode `index` qui renvoie l'indice d'un élément dans une chaîne de caractères.
3. Écrire une fonction `ordre_lexicographique` qui prend en arguments deux mots `m1` et `m2` et renvoie -1 si "`m1` $<$ `m2`" pour l'ordre lexicographique, 0 si "`m1` $=$ `m2`" et 1 si "`m1` $>$ `m2`". On utilisera la fonction `ordre_alphabetique`.
4. Écrire une fonction `tri_lexicographique` qui prend en argument une liste de mots et trie cette liste. On utilisera la fonction `ordre_lexicographique` avec l'algorithme du tri par insertion.

Il faut tenir compte de la longueur des mots. Par exemple "bon" est avant "bonjour". La méthode `index` s'utilise simplement, par exemple : `"abcde".index("d")` vaut 3.

1.3 Exercice 3

On dispose de points dans le plan muni d'un repère orthonormé d'origine $\mathcal{O}$. 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 $\mathcal{O}$, de la plus petite à la plus grande.

1. Écrire une fonction `distance2` qui prend en paramètre une liste de deux nombres nommée `point` qui représente un point du plan, (`point` est la liste des coordonnées d'un point P), et renvoie le carré de la distance de ce point à $\mathcal{O}$.
2. Écrire une fonction `compare` qui prend en paramètres deux listes `p1` et `p2` représentant deux points P_1 et P_2 et qui renvoie -1 si P_1 est plus proche de $\mathcal{O}$ que P_2, 1 si P_2 est plus proche de $\mathcal{O}$ que P_1, et 0 si les deux points sont équidistants de $\mathcal{O}$.
3. Écrire une fonction `tri_points` 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 $\mathcal{O}$. Utiliser par exemple l'algorithme du tri par sélection.

1.4 Exercice 4

Programmation du tri par insertion de manière récursive.

1. Écrire une fonction récursive `insertion` qui prend trois paramètres, une liste, un élément de la liste `x` et son indice `n`. Si `n` est strictement positif, on suppose les éléments d'indice 0 à `n-1` triés et la fonction insère l'élément `x` à la bonne place.
2. Écrire une fonction récursive `tri_insertion` qui prend en paramètres une liste et la longueur de la liste et qui trie la liste.

Distinguer les cas extrêmes pour la condition d'arrêt.

1.5 Exercice 5

Programmer un algorithme du tri insertion avec une insertion qui utilise une recherche dichotomique.