Lycée Vaugelas MP 2023 - 2024
TP d'informatique n° 1 du 5 septembre 2023 **Diviser pour régner + tris**

Consignes

- Toujours venir en informatique avec sa clé USB
- Créer un dossier Info_MP_2023-2024
- Dans le dossier précédent, créer un dossier TP1
- La recherche préalable avec papier/crayon est préférable à l'écriture de code aléatoire. On évite d'affirmer sans preuve, comme dans toute discipline scientifique.
- Les programmes doivent être testés : réaliser un jeu de tests unitaires est une bonne habitude. On les écrit entre la docstring et le corps de la fonction. Ils sont introduits par `>>>` et ne sont réalisés que si la fonction est appelée avec `doctest`. https://docs.python.org/fr/3/library/doctest.html

Exercice 1 (Diviser pour régner)

1. Écrire une fonction `recherche_dicho` qui prend en argument une liste triée `L` et un élément `x` et qui renvoie -1 si l'élément n'est pas dans la liste, sinon l'indice où l'élément est présent pour la première fois. La fonction doit avoir une complexité temporelle logarithmique en la taille de la liste.
2. Écrire une fonction `exp_rap` qui prend en argument un flottant `x` et un entier `n` et qui calcule x^n avec une complexité temporelle logarithmique en n. Présenter une version impérative et une version récursive de cette fonction.

Exercice 2 (Tri par insertion)

Le tri par insertion d'une liste consiste à parcourir la liste dans l'ordre et chaque élément est inséré (déplacé) à la bonne place parmi les éléments déjà triés. C'est le tri naturel d'une main de cartes à jouer.

1. Écrire une fonction `tri_insertion` qui prend en argument une liste `L` qui réalise un tri par insertion par des échanges adjacents pour repousser l'élément à gauche jusqu'à sa position triée (pas de `del`, pas de `insert`, pas de découpage de liste)
2. Déterminer la complexité du tri par insertion
3. Trouver et démontrer un invariant de boucle, prouver la correction du tri par insertion.

Exercice 3 (Tri par sélection)

Le tri par sélection d'un tableau consiste à déterminer le minimum du tableau et à l'échanger avec le premier élément. On poursuit avec le minimum des éléments restants, que l'on place en seconde position, puis avec tous les autres de la même manière.

1. Écrire une fonction `tri_selection` qui trie un tableau (une liste Python) de nombres à l'aide du tri par sélection.
2. Déterminer la complexité du tri par sélection
3. Montrer l'invariant de boucle : « après i itérations de la boucle `for`, la liste `L[0:i]` est triée ». En déduire la correction du tri.

Exercice 4 (Diviser pour régner : le tri fusion)

Le tri fusion consiste à partager le tableau à trier en deux parties égales, à trier récursivement les deux parties et à fusionner les deux moitiés triées.

1. Écrire une fonction `fusion` qui réalise la fusion de deux listes triées en temps linéaire.
2. Écrire la fonction récursive `tri_fusion` qui tri une liste Python selon cet algorithme.
3. Déterminer la complexité temporelle du tri fusion.
4. Montrer la correction du tri fusion.
5. Écrire une version impérative du tri fusion qui repose sur la méthode suivante :
   — Former la liste des singletons (liste de listes qui ne contiennent qu'un élément et qui sont donc triée)
   — Fusionner ces listes deux par deux de gauche à droite (les listes de deux éléments obtenues sont triées, il peut rester une liste d'un seul élément)
   — Recommencer la fusion tant qu'il reste plus d'une seule liste
   — Renvoyer la liste obtenue, qui est triée.
6. Vérifier la complexité temporelle de la fonction précédente.