TD 2 (algorithmique) - Listes.

Dans les exercices qui suivent, vous pouvez supposer que tous les éléments d'une liste sont comparables, et que les types des données de deux listes différentes sont compatibles.

Exercice 1

Écrivez une fonction `insertion_triee(T, x)` qui insère, en $O(|T|)$, un élément `x` dans une liste triée `T` de façon à ce que `T` reste triée après l'insertion. Attention, `T` peut être vide.

Exercice 2

Une permutation de n éléments est une liste contenant les entiers de 1 à n une et une seule fois. Écrivez une fonction qui vérifie, en $O(n)$, si une liste donnée est bien une permutation.

Exercice 3

Écrivez une fonction qui, étant donnée une liste de n entiers, déplace tous les 0 vers la fin de la liste. Il n'est pas nécessaire de préserver l'ordre des autres éléments. Par exemple, `[0, 1, 0, 2, 3]` sera transformée en `[3, 1, 2, 0, 0]`. Votre solution doit être en $O(n)$.

Exercice 4

Modifiez la solution de l'exercice précédent pour que l'algorithme préserve l'ordre des éléments non nuls (c'est-à-dire que `[0, 1, 0, 2, 3]` sera transformée en `[1, 2, 3, 0, 0]` et pas `[3, 1, 2, 0, 0]`). Votre solution doit être en $O(n)$.

Exercice 5

Soit `S` et `T` deux listes triées. Écrivez une fonction `fusion(S, T)` qui renvoie une liste triée contenant tous les éléments de `S` et de `T` (les doublons doivent être préservés). Votre fonction doit être en $O(|S| + |T|)$.

Exercice 6

Écrivez une fonction `reordonner(T)` qui réorganise la liste `T` de n éléments en plaçant tous les éléments inférieurs ou égaux au premier élément avant lui. Par exemple, si `T = [3, 1, 4, 2, 5, 2]`, alors `[1, 2, 2, 3, 4, 5]` est un résultat valide qu'on pourrait obtenir après avoir appelé la fonction. Il n'est pas nécessaire de préserver l'ordre des autres éléments, mais la fonction doit être en $O(n)$.