Les tris (séance 3). Exercices

Tri arborescent

1. Proposer un algorithme qui stocke dans un tableau l'arbre de tri d'un tableau d'entiers.
   - entrée = un tableau d'entiers de longueur N à trier. Exemple :
     `int[] aTrier = {44, 55, 12, 42, 94, 18, 06, 67}`
   - sortie = un tableau d'entiers de longueur 2N-1 qui contient l'arbre de tri du tableau donné en entrée. L'arbre de tri des éléments à trier donnés en exemple sera le suivant :
     `int[] arbreTri = {44, 55, 12, 42, 94, 18, 06, 67, 44, 12, 18, 06, 12, 06, 06}`

   Le tableau correspondant à l'arbre de tri contient donc d'abord les feuilles de l'arbre (le tableau à trier), puis les noeuds de hauteur 1, ... jusqu'à la racine de l'arbre.
2. Ecrire une fonction qui prend en argument un tableau d'entiers (contenant un arbre de tri) et qui renvoie un tableau contenant les entiers (correspondant aux feuilles de l'arbre) triés. On appliquera l'algorithme de parcours de l'arbre vu en cours.
   `int[] trié = {06, 12, 18, 42, 44, 55, 67, 94}`

Tri rapide

1. Écrire une fonction `partition` prenant en argument un tableau, un indice de début, un indice de fin et un pivot. Cette fonction met à la droite du pivot tous les entiers supérieurs à la valeur du pivot et à sa gauche tous les entiers qui lui sont inférieurs. La fonction renvoie la (nouvelle) position du pivot à la fin de la partition.
2. Écrire une fonction `trouverPivot` prenant en argument un indice de début et un indice de fin. La fonction renvoie 0 si le tableau est trié (dans la fonction, cela correspond au cas où début et fin renvoient à la même position ou au cas où début et fin sont contigus et ont la même valeur), l'indice de la plus grande des deux premières clés différentes sinon.
3. Écrire une fonction récursive `triRapide` qui prend en argument un tableau d'entiers et le tri en utilisant les deux fonctions définies précédemment.