TD4 : Tris

Exercice 1 (Casiers)**.**
Soit t un tableau de taille n contenant des valeurs comprises entre 0 et un certain entier k. Proposez un algorithme de tri pour t dont la complexité soit $O(n)$.

Exercice 2 (Stabilité)**.**
On dit qu'un tri est **stable** si l'ordre des valeurs égales n'est pas modifié. Par exemple, si l'on possède la liste des étudiants ordonnée alphabétiquement et que l'on trie par note, les étudiants ayant la même note seront toujours ordonnés alphabétiquement. **Parmi les algorithmes vus en cours, lesquels sont stables ?**

Exercice 3 (Le tri pas rapide)**.**
Voici un nouvel algorithme de tri :

```
NouveauTri
Input :
    - t, un tableau de taille n
    - i un entier entre 0 et n-1
    - j un entier entre 0 et n-1
Procédé :
    Si j-i = 1 et t[j] < t[i]:
        t[i], t[j] <- t[j], t[i]
    Si j-i <= 1:
        Retourner
    k <- (j+1-i)/3
    NouveauTri(t,i,j-k)
    NouveauTri(t,i+k,j)
    NouveauTri(t,i,j-k)
```

(1) Soit `t = [9,3,6,1]`, afficher les étapes d'exécutions de `NouveauTri(t,0,3)`.
(2) On veut Prouver par récurrence que si t est un tableau de taille n, l'appel de `NouveauTri(t,0,n-1)` trie le tableau `t`,
   (a) Prouver le cas initial et exprimer l'hypothèse de récurrence.
   (b) Soit u et v deux valeurs du tableau telles que $u < v$, si u est avant v au début de l'algorithme, peuvent-ils être inversés ?
   (c) Étudier le cas où v est avant u et que les deux valeurs apparaissent dans les 2 premiers tiers du tableau.
   (d) Même question pour les deux derniers tiers.
   (e) Enfin, étudier le cas où v est dans le premier tiers et u dans le dernier tiers.
(3) Exprimer le nombre d'appels récursifs sous la forme d'une fonction récursive.
(4) Un théorème d'analyse d'algorithme, le *master theorem* nous dit que si la complexité d'un algorithme récursif s'exprime sous la forme

$$f(n) = af\left(\frac{n}{b}\right) \tag{0.1}$$

avec $a \geq 1$ et $b > 1$, alors la complexité de l'algorithme est de $n^{\frac{\log(a)}{\log(b)}}$. En comparant avec les algorithmes connus, l'algorithme `NouveauTri` est-il efficace ?

Exercice 4 (Parité)**.**
Voici un algorithme

```
Parite
Input :
    - t, un tableau de taille n
Procédé :
    j <- 0
    Pour i allant de 0 à n-1:
        Si t[i] est pair:
            t[i],t[j] <- t[j],t[i]
            j <- j+1
    Retourner j
```

(1) Appliquer l'algorithme au tableau `t = [9, 3, 6, 1, 2, 4, 3, 5, 6, 4]`

(2) Quel est l'effet de l'algorithme sur le tableau ? Proposer un **invariant de boucle** pour prouver votre hypothèse.

(3) Que retourne l'algorithme ?

(4) L'ordre des valeurs paires est-il conservé ? Même question pour les valeurs impaires.

Exercice 5 (Quand l'écriture est coûteuse)**.**
Un professeur de la Grèce Antique a noté ses élève et a inscrit leurs noms et leurs notes sur une plaque de cire. Il voudrait à présent les ranger par ordre croissant. Cependant, il n'a plus de plaques de cire disponible... Il peut modifier la plaque existante (et recouverte par les noms et les notes) avec certaines contraintes :

— En raclant patiemment la cire, il peut effacer le nom et la note d'un élève.
— Il peut ensuite ré-écrire sur la ligne effacée exactement une fois (la cire est trop fine pour être à nouveau effacée et gravée).
— Une fois que la cire a été effacée (et éventuellement re-gravée), elle a une couleur différente (car elle est plus fine) et on peut donc savoir qu'elle a été modifiée.
— Le professeur possède un petit morceau de papyrus de quelques lignes (bien moins que le nombre d'élève) sur lequel il peut écrire avec du charbon et effacer à volonté.

Décrivez l'algorithme à suivre pour ordonner la liste des élèves en fonction de leurs notes.

Exercice 6 (Drapeau)**.**
On considère un tableau d'éléments munis chacun d'une couleur, bleu, blanc ou rouge. On souhaite réordonner les éléments en fonction de leur couleur. **Proposer un algorithme qui réponde au problème en un unique parcours.**