J1MI2013: Algorithmes et Programmes: feuille 6

Tableaux(3)

Travaux dirigés

Dans les exercices qui suivent on suppose que les tableaux ont une taille maximale fixée par une constante "assez grande" `TAILLE_MAX`.

Pour chacun des exercices, indiquer, en fonction de la taille `n` du tableau, le nombre de décalages nécessaires.

Exercice 1. Écrire une fonction `int inserer(int t[], int n, int e, int k)` qui, étant donné un tableau `t` contenant `n` éléments (avec $n <$ `TAILLE_MAX` -1) insère un élément `e` à la position `k` avec $k \geq 0$.
Si $k \geq n$, l'élément sera ajouté à l'indice n. L'ordre initial des éléments du tableau sera conservé. La fonction doit renvoyer le nouveau nombre d'éléments dans le tableau.
Exemple :
si `TAILLE_MAX`= 10 et `t` contient la suite `[5,4,3,12,0,4,7,2]`, l'appel `inserer(t,8,6,4)` doit
- insérer l'élément 6 à l'indice 4, après l'insertion le tableau contiendra `[5,4,3,12,6,0,4,7,2]`,
- renvoyer le nouveau nombre d'éléments dans `t`, c'est à dire 9.

Exercice 2. Écrire une fonction `int supprimer(int t[], int n, int k)` qui supprime l'élément situé à la position `k` du tableau `t` contenant `n` éléments en supposant $0 \leq k < n$. L'ordre initial des éléments du tableau sera conservé. La fonction doit renvoyer le nouveau nombre d'éléments dans le tableau.

Exercice 3. On considère un tableau `t` de n entiers.
Écrire une fonction `int supprimerPremiereOccurrence(int t[],int n, int x)` permettant d'enlever du tableau `t` la première occurrence d'un élément `x` passé en paramètre (le tableau `t` ne sera pas modifié si `x` n'appartient pas à `t`). L'ordre initial des éléments du tableau sera conservé. La fonction doit renvoyer le nouveau nombre d'éléments dans le tableau.

Exercice 4. Soit `t` un tableau contenant `n` nombres entiers rangés dans l'ordre croissant.
Écrire une fonction `int insererOrdonne(int t[], int n, int e)` qui insère un nouvel élément `e` dans `t` en respectant l'ordre croissant et qui renvoie le nouveau nombre d'éléments dans le tableau.
Exemple : si `TAILLE_MAX`= 10 et `t` = `[2,4,7,10]`, l'appel `insererOrdonne(t,4,6)` va renvoyer 5 (nombre d'éléments dans `t` après l'insertion) et va aussi modifier le contenu de `t` en `[2,4,6,7,10]`.

Exercice 5.

1. Écrire une fonction `void rotationAdroite(int t[], int n)` qui applique aux `n` éléments d'un tableau `t` un décalage d'une position à droite. L'élément en position `n-1` est placé en position 0 après l'appel de la fonction.
   Exemple : si t = [0,1,2,3,4,5,6] après l'appel de la fonction `rotationAdroite(t,7)`
   `t` = `[6,0,1,2,3,4,5]`. Si $n <= 1$ le tableau ne sera pas modifié.

2. On veut maintenant écrire un algorithme qui applique aux éléments d'un tableau une rotation à droite de `k` positions.
   Exemple : si t = [0,1,2,3,4,5,6] et qu'on demande une rotation à droite de 3 positions, après l'appel de la fonction on aura t = [4,5,6,0,1,2,3].

   Deux algorithmes sont possibles : le premier utilise un tableau auxiliaire de taille égale au nombre d'éléments dans `t`, le deuxième utilise une unique variable auxiliaire de type `int`. Expliquer les deux méthodes. Comparer le nombre de recopies d'éléments du tableau effectuées par chaque méthode.

   Qu'est-ce qu'on peut dire de la complexité en temps de chaque algorithme ? Et de la complexité en espace ?
   (Remarque : si `t` contient `n` éléments avec `n<TAILLE_MAX`, les cases de `t` d'indice supérieur à `n-1` ne seront jamais utilisées par les algorithmes demandés).