4TPM205U: Algorithmique des tableaux: feuille 4

Tableaux : 2

Travaux dirigés sur machine

Rappel : le fichier `bibTableau.py` *doit se trouver dans le même répertoire que vos programmes* Python.

Depuis la page `http://dept-info.labri.fr/ENSEIGNEMENT/algotab/src/` téléchargez le fichier `tp04.py`. Il contient des fonctions utilisées pour les tests. Vous devez y ajouter les fonctions demandées.

Sauf indication contraire les tableaux manipulés contiennent des nombres et au moins un élément.

Exercice 1 – Écrire et tester les fonctions des exercices 1 à 4 de la feuille de TD#4.

Exercice 2 – Écrire une fonction booléenne `estCroissant(t, n)` qui retourne `True` si le tableau `t` contenant `n` nombres est trié dans l'ordre croissant et `False` sinon.

Exercice 3

1. Écrire une fonction `rotationAdroite(t, 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` a 10 cases et 7 éléments : `t = [0,1,2,3,4,5,6,None,None,None]`, après l'appel de la fonction `rotationAdroite(t,7)` on aura `t = [6,0,1,2,3,4,5,None,None,None]`.
   Le tableau ne sera modifié que s'il contient au moins deux éléments.

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,None,None,None]` 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,None,None,None]`.

   Deux algorithmes sont possibles : le premier consiste à réutiliser la fonction `rotationAdroite`. Le deuxième utilise un tableau auxiliaire de taille égale au nombre d'éléments dans `t`.

   i) Analysez les deux méthodes. Comparer le nombre de recopies d'éléments du tableau effectuées par chaque méthode.
   ii) Comment éviter d'effectuer des recopies inutiles si $k > n$?
   iii) Écrire et tester les deux versions de la fonction `rotationADroiteDeK_V1(t, n, k)` et `rotationADroiteDeK_V2(t, n, k)` (respectivement avec la première méthode et avec la seconde méthode).
   (Remarque : si `t` contient `n` éléments avec $n <$ `len(t)`, les cases de `t` d'indice supérieur à `n-1` ne seront jamais utilisées par les algorithmes demandés).

Exercice complémentaire 1 – Écrire une troisième version `rotationADroiteDeK_V3(t, n, k)` permettant toujours d'effectuer une rotation de `t` de `k` positions vers la droite, en temps linéaire et sans avoir recours à un tableau auxiliaire. (Aide : vous pourrez utiliser plusieurs appels à la fonction `renverser(t,n)`, déjà programmée au TP#3, permettant de renverser en temps linéaire les `n` premiers éléments de `t`.)

Exercice complémentaire 2 – En modifiant la fonction `estSection` de la feuille de TD#4, écrire une nouvelle fonction `nombreSections(t, nt, s, ns)` qui calcule et retourne le nombre de fois où le tableau non vide `s` apparaît comme section du tableau `t`.

Par exemple, le tableau `[1, 2]` correspond à deux sections du tableau `[5, 1, 2, 3, 1, 2, 1]`. Autre exemple, le tableau `[1, 2, 1]` correspond à deux sections du tableau `[5, 1, 2, 1, 2, 1]`. Dans ce dernier cas, les deux sections se chevauchent :
`[5, 1, 2, 1, 2, 1]` et `[5, 1, 2, 1, 2, 1]`.