INF1 - TP 05 Constructions de tableaux

UNIVERSITÉ DE RENNES 1

Le but du TP est de continuer à travailler sur les tableaux, et de réviser l'utilisation de boucles et l'écriture de fonctions.

Attention : Pour tester vos fonctions sur quelques exemples, vous aurez besoin d'afficher le contenu de tableaux. On recommande pour cela de programmer une fonction qui réalise l'affichage de tous les éléments d'un tableau passé en paramètre.

Fichiers de tests : On vous fournit également des fichiers de test JUnit pour ce TP. Ils sont dans le répertoire `test`. Avant d'exécuter les tests, il faudra décommenter les tests portant sur les fonctions que vous avez réussi à définir. Ces tests sont volontairement incomplets. À vous de les compléter !

Exercice 1 : Opérations sur les tableaux

1.a] Écrire une fonction `copieTableau` qui prend en paramètre un tableau `tab` de longueur quelconque. Cette fonction doit renvoyer une copie de `tab`, c'est-à-dire un nouveau tableau de la même taille et qui contient les mêmes valeurs.

Par exemple, pour `tab = {1,0,6,8,6,9,2,2,6}`, l'appel à la fonction doit renvoyer un *nouveau* tableau `{1,0,6,8,6,9,2,2,6}`.

1.b] Écrire une fonction `sousTableau` qui prend en paramètres un tableau `tab` de longueur quelconque et deux indices valides `i` et `j` (on suppose `j` plus grand que `i`). Cette fonction doit renvoyer un nouveau tableau qui contient les valeurs des cases de `tab` entre les indices `i` et `j` inclus.

Par exemple, pour `tab = {1,0,6,8,6,9,2,2,6}`, `i = 3`, et `j = 6`, la fonction doit renvoyer le tableau `{8,6,9,2}`.

1.c] Écrire une fonction `concatenation` qui prend en paramètres deux tableaux `tab1` et `tab2` de longueurs quelconques. Cette fonction doit renvoyer un nouveau tableau qui contient les éléments de `tab1` suivis des éléments de `tab2`.

Par exemple, pour `tab1 = {2, 5, 6}` et `tab2 = {1, 2, 3, 4, 5}`, la fonction doit renvoyer le tableau `{2, 5, 6, 1, 2, 3, 4, 5}`.

1.d] **(Challenge)** Écrire une fonction `fusion` qui prend en paramètres deux tableaux `tab1` et `tab2` de longueurs quelconques. Cette fonction doit renvoyer un nouveau tableau qui contient les éléments de `tab1` et `tab2` intercalés.

Par exemple, si `tab1 = {2, 5, 6}` et `tab2 = {1, 2, 3, 4, 5}`, la fonction doit renvoyer le tableau `{2, 1, 5, 2, 6, 3, 4, 5}`.

Exercice 2 : Tableaux triés

2.a] **(Challenge)** Écrire une fonction `fusionTrie` qui prend en paramètres deux tableaux triés par ordre croissant `tab1` et `tab2` de longueurs quelconques. Cette fonction doit renvoyer un nouveau tableau trié par ordre croissant qui contient les éléments de `tab1` et `tab2`.

Par exemple, si `tab1 = {2, 5, 6, 6}` et `tab2 = {1, 2, 3, 4, 5}`, la fonction doit renvoyer le tableau `{1, 2, 2, 3, 4, 5, 5, 6, 6}`.

Exercice 3 : Permutations

Dans la suite, on appelle permutation tout tableau `tab` tel que, si la longueur de `tab` est n, alors `tab` contient tous les entiers de 0 à $n-1$, mais pas nécéssairement dans l'ordre. Par convention, on considérera qu'un tableau vide est une permutation.

Pour cet exercice, vous pourrez réutiliser la fonction `premiereOccurrence` réalisée lors du TP4, qui prend en paramètre un tableau d'entiers `tab` et un entier `i`, et qui renvoie la position de la première occurrence de `i` dans `tab` si `tab` contient `i`, et -1 sinon.

3.a] Écrire une fonction `estPermutation` qui prend en paramètre un tableau `tab` de longueur quelconque. Cette fonction doit renvoyer `true` si le tableau `tab` est une permutation, et `false` sinon.

Par exemple, pour `tab1 = {1,3,4,2,0}`, l'appel à la fonction doit renvoyer `true`. Pour `tab2 = {5,2,3}`, l'appel à la fonction doit renvoyer `false`

3.b] Écrire une fonction `compose` qui prend en paramètre deux tableaux `tab1` et `tab2`. On suppose que `tab1` et `tab2` sont des permutations, et que toutes les valeurs présentes dans `tab1` sont des indices valides de `tab2`. Cette fonction doit renvoyer un nouveau tableau `tab` de même longueur que `tab1` et `tab2` tel que, pour tout `j`, `tab[j] = tab2[tab1[j]]`.

Par exemple, pour `tab1 = {1,3,4,2,0}` et `tab2 = {1,0,2,4,3}`, la fonction doit renvoyer le tableau `tab = {0,4,3,2,1}`.

3.c] **(Challenge)** Écrire une fonction `itere` qui prend en paramètre un tableau `tab` de longueur quelconque, et deux entiers `i` et `k`. On suppose que `tab` est une permutation et que `k` est positif. Cette fonction doit renvoyer `tab[tab[...[i]...]]` en effectutant `k` itérations.

Par exemple, pour `tab = {1,3,4,2,0}`, et `i = 0`, l'appel `itere(tab,0,0)` doit renvoyer `0` ; l'appel `itere(tab,0,1)` doit renvoyer `tab[0]`, donc `1` ; l'appel `itere(tab,0,2)` doit renvoyer `tab[tab[0]]`, donc `3` ; etc.

3.d] Écrire une fonction `inverse` qui prend en paramètre un tableau `tab` de longueur quelconque. On suppose que `tab` est une permutation. Cette fonction doit renvoyer une nouvelle permutation `tab2` telle que, pour tout `i`, si `tab[i] = j`, alors `tab2[j] = i`.

Par exemple, pour `tab = {1,3,4,2,0}`, la fonction doit renvoyer `tab2 = {4,0,3,1,2}`.

3.e] **(Bonus)** Si vous considérez une permutation `tab` de taille `n`, alors que doit renvoyer `compose(tab,inverse(tab))` ? Vérifiez en écrivant quelques exemples de permutation.