4TPM205U: Algorithmique des tableaux: feuille 3

Tableaux

Travaux dirigés sur machine

Le fichier `bibTableau.py` (à télécharger) est une bibliothèque rassemblant un ensemble de fonctions qui manipulent des tableaux. Une description de cette bibliothèque se trouve dans le fichier `bibTableau.pdf` consultable dans la section « Travaux Dirigés / Fichiers à télécharger » de la page WEB de l'UE.

Par la suite tout fichier utilisant les fonctions de la bibliothèque de manipulation des tableaux doit inclure la ligne : `from bibTableau import *`

Attention : 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 `tp03.py`. Il contient des fonctions utilisées pour les tests. Vous devez y ajouter les fonctions demandées.

Sauf indication contraire, dans tous les exercices on manipule des tableaux de nombres **qui contiennent au moins un élément**.

Exercice 1 – Écrire une fonction `appartient(x,t,n,i1,i2)` qui renvoie `True` si le tableau `t` contient la valeur `x` dans les cases d'indices compris entre `i1` et `i2` inclus (on supposera que $0 \leq$ `i1` $\leq$ `i2` $<$ `n` où `n` est le nombre d'élements de `t`), et renvoie `False` sinon.

Exercice 2 – Écrire une fonction `doubler(t,n)` qui modifie le tableau `t` en multipliant par deux la valeur de chacun de ses `n` éléments.

Exercice 3 – Écrire une fonction `cumuler(t,n)` qui modifie le tableau `t`, contenant `n` éléments, en plaçant dans chacune de ses premières `n` cases la somme des valeurs des termes précédents et du terme courant. Par exemple, si avant l'appel `t = [1, -3, 12, -183]`, après l'appel de `cumuler(t,4)` on aura `t = [1, -2, 10, -173]`.

Exercice 4 – Écrire une fonction `renverser(t,n)` qui « *renverse* » la suite de `n` éléments du tableau `t`. Par exemple, si avant l'appel `t = [1, -3, 12, -183]`, après l'appel de `renverser(t,4)` on aura `t = [-183, 12, -3, 1]`. La fonction ne doit pas utiliser un tableau supplémentaire.

Exercice 5 – Soient `t1` et `t2` deux tableaux contenant chacun `n` valeurs **booléennes**. Écrire une fonction `ouLogique(t1,t2,n)` qui crée un tableau de taille `n` dont chaque élément sera calculé en appliquant l'opération *ou logique* au couple d'éléments de `t1` et `t2` de même indice. La fonction renverra le tableau créé.

Exercice 6 – Écrire et tester les fonctions des exercices 2 à 6 de la feuille de TD#3.

Exercice 7 – Écrire une fonction `fibonacciTableau(n)` qui crée et renvoie un tableau de taille `n` contenant les `n` premiers nombres de la suite de Fibonacci (de F_0 à F_{n-1}). Pour rappel, la suite

de Fibonacci est définie par :

$$\begin{cases} F_0 = 1 \\ F_1 = 1 \\ F_k = F_{k-1} + F_{k-2}, \quad \text{pour } k \geq 2 \end{cases}$$

Votre fonction remplira le tableau directement sans utiliser la fonction de calcul de la suite de Fibonacci écrite lors de TD/TM précédents.

Exercice complémentaire 1 – Écrire une fonction `uniformite(n)` qui lance `n` fois un dé et affiche ensuite un histogramme représentant le nombre de tirages de chaque valeur de 1 à 6. Par exemple, votre fonction affichera :

```
nombre de lancers: 15

1: ***
2: *
3: *****
4:
5: **
6: ****
```

si elle a effectué 15 lancers et qu'elle a obtenu trois fois la face 1, une fois la face 2, ...

Exercice complémentaire 2 – Écrire une fonction `estTrie(t,n)` qui renvoie `True` si le tableau `t` contenant `n` éléments est trié (par ordre croissant ou décroissant) et `False` sinon. Attention, il ne s'agit pas d'une croissance ou d'une décroissance stricte ; ainsi, deux éléments consécutifs peuvent avoir la même valeur.

La fonction effectuera un unique parcours du tableau et ne contiendra pas de répétition de code (notamment on n'écrira pas deux versions d'une boucle de parcours du tableau suivant qu'on teste la croissance ou la décroissance).