4TPM205U: Algorithmique des tableaux: feuille 3

Tableaux

Travaux dirigés

Exercice 1 – Soit la fonction `mystere` suivante :

```
1 def mystere(t,n):
2   m = t[0]
3   for i in range(1,n):
4     if t[i] < m :
5       m = t[i]
6   return m
```

1. Simulez l'exécution de `mystere([3,5,7,1,10,-3,7,2], 8)` en complétant le tableau suivant montrant l'évolution des variables `n`, `i`, `t[i]` et `m` :

`n`		
`i`		
`t[i]`		
`m`		

2. Quel sera en général le résultat de l'appel de `mystere` pour $n > 1$?
3. Quel est le nombre de comparaisons effectuées par la fonction ?

Par la suite, dans tous les exercices manipulant des tableaux, on distinguera la **taille** $n\mathrm{Max}$ d'un tableau et le **nombre d'éléments** n qu'il contient. On entend par taille d'un tableau le nombre de cases de mémoire qui le composent. Un tableau ne peut pas contenir plus d'élements que sa taille. Un tableau pourra par exemple être de taille 10 (*i.e.*, il dispose de 10 cases) et contenir moins de 10 éléments. Les éléments seront systématiquement rangés dans les cases d'indices les plus petits possibles, soit de 0 à $n-1$.
Par exemple, si un tableau a une taille égale à 10 et contient seulement 4 éléments, ils seront rangés dans les cases d'indice 0 à 3.

Pour connaître la taille $n\mathrm{Max}$ d'un tableau on fera appel à la fonction `len`, prédéfinie en Python, qui prend en paramètre un tableau et renvoie sa taille.

Puisque le nombre d'éléments n contenus dans un tableau `t` est inférieur ou égal à sa taille `len(t)`, il sera nécessaire, chaque fois que l'on passe un tableau en paramètre à une fonction, de passer aussi en paramètre le nombre n de ses éléments.

Pour chacune des fonctions de la feuille, préciser le nombre maximal et minimal d'accès (en lecture ou en écriture) à des éléments du tableau qui sont utilisés pour obtenir le résultat, en fonction de `n`.

Sauf indication contraire, dans tous les exercices on considère que `t` est un tableau de nombres **contenant au moins un élément**.

Exercice 2 – Écrire une fonction `moyenne(t,n)` qui calcule la moyenne des valeurs des `n` éléments du tableau `t`.

Exercice 3 – Écrire une fonction `substituer(t,n,x,y)` qui, étant donnés un tableau `t` (contenant `n` éléments) et deux valeurs `x` et `y`, remplace dans `t` toutes les occurences de `x` par `y`. Par exemple si `t=[2,1,2,7,3,5,1]` l'appel `substituer(t,7,1,8)` modifiera `t` et on aura `t=[2,8,2,7,3,5,8]`.

Exercice 4 – Écrire une fonction `recherche(t,n,x)` qui, étant donnés un tableau `t` (contenant `n` éléments) et une valeur `x`, renvoie le premier indice de `t` où se trouve `x` et `-1` si `x` n'est pas parmi les `n` éléments de `t`.

Exercice 5 – Écrire une fonction `indiceMaximum(t,n)` qui renvoie l'indice du plus grand élément du tableau `t` contenant `n` éléments. Si plusieurs indices correspondent au plus grand élément, on renverra le plus petit de ces indices.

Exercice 6 – Écrire une fonction `complementaire(t,n)` qui étant donné un tableau `t` contenant `n` valeurs **booléennes**, transforme le contenu de `t` en remplaçant chaque élément par son complémentaire (rappel : `False` est le complémentaire de `True` et vice versa).

Exercice complémentaire 1 – Écrire une fonction `amplitude(t,n)` qui, en parcourant une seule fois le tableau `t`, calcule et renvoie la différence entre le plus grand et le plus petit élément parmi les `n` premiers éléments du tableau `t`.

Par exemple, si `t=[3,6,2,7,1,4,13,5]`, l'appel `amplitude(t,8)` doit renvoyer `12` (car $12 = 13 - 1$) et l'appel `amplitude(t,4)` doit renvoyer `5` (car $5 = 7 - 2$).

Exercice complémentaire 2 – Écrire une fonction `max2(t,n)` qui calcule et renvoie la *deuxième* plus grande valeur parmi les `n` premiers éléments du tableau `t`.

Par exemple, si `t=[5,3,4,6,1,10,2,10,4]`, `max2(t,5)` doit renvoyer `5`, mais `max2(t,9)` doit renvoyer `10`.

La fonction ne devra parcourir le tableau qu'une seule fois.