Portail Descartes

Introduction à la science info : TD 4

Tableaux

amU Faculté des sciences Aix Marseille Université

Exercice 1 On considère la fonction suivante prenant en paramètre un tableau d'entiers et renvoyant un booléen (`True` ou `False` selon le résultat du test qu'on effectue dans la dernière ligne) :

```
def f(t):
    n = len(t)
    i = n - 1
    while i >= 0 and t[i] > 0:
        i = i - 1
    return i == -1
```

1. Exécuter l'appel `f([2, 5, 8])` en détaillant l'évolution des valeurs de toutes les variables le long de l'exécution.
2. Faites de même avec l'appel `f([1, 2, -1, 5, 2])`.
3. Montrer que la fonction `f` termine.
4. Décrire en quelques mots ce que calcule la fonction `f`.
5. Proposer une réécriture de la fonction `f` utilisant une boucle `for` au lieu d'une boucle `while`.

Exercice 2 Pour rappel, voici la fonction de recherche séquentielle dans un tableau :

```
def linear_search(t, e):
    n = len(t)
    for i in range(n):
        if t[i] == e:
            return i
    return -1
```

1. Modifier la fonction de recherche séquentielle pour qu'elle renvoie `True` ou `False`, selon que l'élément a été trouvé dans le tableau ou non : ainsi, la recherche de l'élément 8 dans le tableau $[3, 5, 8, 2, 8, 1]$ devra renvoyer `True`, alors que la recherche de 9 dans ce même tableau devra renvoyer `False`.
2. Modifier la fonction de recherche séquentielle pour qu'elle renvoie l'indice de la *dernière occurrence* de l'élément recherché : ainsi, la recherche de l'élément 8 dans le tableau $[3, 5, 8, 2, 8, 1]$ renverra désormais 4.
3. Améliorer la fonction de recherche séquentielle dans le cas où le tableau donné en entrée est supposé trié, pour qu'il s'arrête dans son parcours du tableau dès lors qu'il a trouvé l'élément à chercher, ou bien qu'il est sûr que l'élément à chercher ne se trouve pas dans le tableau.
4. Quelles sont les complexités dans le meilleur des cas et dans le pire des cas en fonction de la longueur du tableau en paramètre ?

Exercice 3

1. Écrire une fonction en Python qui calcule la moyenne des valeurs d'un tableau donné en argument **qu'on supposera non vide**. Analyser sa complexité.
2. Écrire une fonction en Python qui calcule la variance des valeurs d'un tableau donné en argument **qu'on supposera non vide**. Analyser sa complexité. Pour rappel, la variance des valeurs $x_1, \ldots, x_n$ est calculée comme $\frac{(x_1-m)^2+(x_2-m)^2+\cdots+(x_n-m)^2}{n}$, où m est la moyenne des valeurs.

Exercice 4 Considérons la fonction suivante, qui prend en argument un tableau `t` d'entiers :

```
def f(t):
    n = len(t)
    j = n
    while j > 0:
        for i in range(n):
            t[i] = t[i] + t[j-1]
        j = j // 2
    return t
```

1. Exécuter la fonction `f` sur le paramètre `[4, 1, 3]` en remplissant une table qui montre l'évolution des valeurs de toutes les variables le long de l'exécution.
2. Expliquer pourquoi la fonction `f` termine.
3. Combien d'opérations sont exécutées par la fonction `f` sur une entrée de longueur n strictement positive ?

Exercice 5

1. Écrire une fonction qui prend en paramètre un tableau `t` et un entier `k` et renvoie la `k`-ième rotation de `t` vers la gauche. Par exemple :

```
>>> rotation([1, 2, 3, 4, 5], 2)
[3, 4, 5, 1, 2]
>>> rotation([1, 2, 3, 4, 5], 0)
[1, 2, 3, 4, 5]
>>> rotation([1, 2, 3, 4, 5], -3)
[3, 4, 5, 1, 2]
```

   Analyser sa complexité.

2. On veut désormais combiner cette opération de rotation, en produisant un tableau qui contient à la suite l'ensemble des rotations possibles du tableau en paramètre. Écrire une fonction qui prend en paramètre un tableau `t` (de longueur n) et renvoie le tableau obtenu en concaténant [1] la k-ième rotation de `t` vers la gauche pour k allant de 1 à $n-1$. Analyser la complexité.

1. Concaténer deux tableaux, c'est les mettre l'un à la suite de l'autre dans un seul tableau.