BACCALAURÉAT

SESSION 2024

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°24

DURÉE DE L'ÉPREUVE : 1 heure

Le sujet comporte 4 pages numérotées de 1 / 4 à 4 / 4
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (10 points)

Un arbre binaire est soit vide, représenté en Python par la valeur None, soit un nœud représenté par un triplet `(g, x, d)` où x est l'étiquette du nœud et g et d sont les sous-arbres gauche et droit.

On souhaite écrire une fonction `parcours_largeur` qui prend en paramètre un arbre binaire et qui renvoie la liste des étiquettes des nœuds de l'arbre parcourus en largeur.

Exemples :

```
>>> arbre = ( ( (None, 1, None), 2, (None, 3, None) ),
              4,
              ( (None, 5, None), 6, (None, 7, None) ) )
>>> parcours_largeur(arbre)
[4, 2, 6, 1, 3, 5, 7]
```

EXERCICE 2 (10 points)

On considère un tableau non vide de nombres entiers, positifs ou négatifs, et on souhaite déterminer la plus grande somme possible de ses éléments consécutifs.

Par exemple, dans le tableau `[1, -2, 3, 10, -4, 7, 2, -5]`, la plus grande somme est `18` obtenue en additionnant les éléments `3, 10, -4, 7, 2`.

Pour cela, on va résoudre le problème par programmation dynamique. Si on note `tab` le tableau considéré et `i` un indice dans ce tableau, on se ramène à un problème plus simple : déterminer la plus grande somme possible de ses éléments consécutifs se terminant à l'indice `i`.

Si on connait la plus grande somme possible de ses éléments consécutifs se terminant à l'indice `i-1`, on peut déterminer la plus grande somme possible de ses éléments consécutifs se terminant à l'indice `i` :

- soit on obtient une plus grande somme en ajoutant `tab[i]` à cette somme précédente ;
- soit on commence une nouvelle somme à partir de `tab[i]`.

Remarque : les sommes considérées contiennent toujours au moins un terme.

Compléter la fonction `somme_max` ci-dessous qui réalise cet algorithme.

```
def somme_max(tab):
    n = len(tab)
    sommes_max = [0]*n
    sommes_max[0] = tab[0]
    # on calcule la plus grande somme se terminant en i
    for i in range(1,n):
        if ... + ... > ...:
            sommes_max[i] = ...
        else:
            sommes_max[i] = ...
    # on en déduit la plus grande somme de celles-ci
    maximum = 0
    for i in range(1, n):
        if ... > ...:
            maximum = i

    return sommes_max[...]
```

Exemples :

```
>>> somme_max([1, 2, 3, 4, 5])
15
>> somme_max([1, 2, -3, 4, 5])
9
>>> somme_max([1, 2, -2, 4, 5])
10
>>> somme_max([1, -2, 3, 10, -4, 7, 2, -5])
18
```

```
>>> somme_max([-3,-2,-1,-4])
-1
```