BACCALAURÉAT

SESSION 2025

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°47

DURÉE DE L'ÉPREUVE : 1 heure

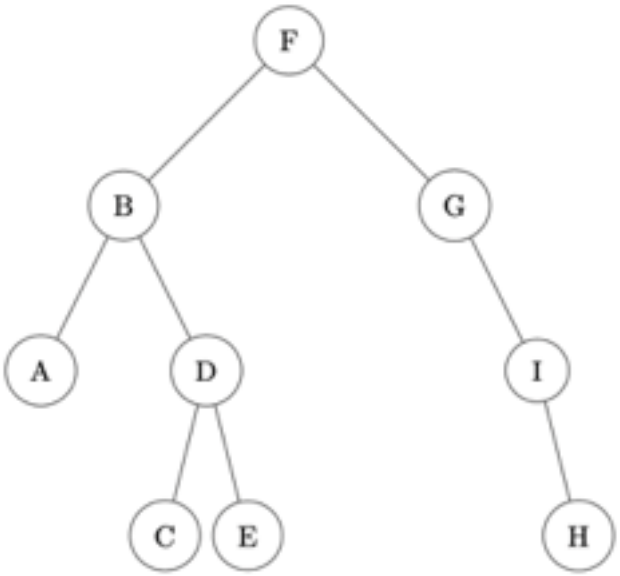
Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (10 points)

Dans cet exercice, un arbre binaire de caractères non vide est stocké sous la forme d'un dictionnaire où les clefs sont les caractères des nœuds de l'arbre et les valeurs, pour chaque clef, la liste des caractères des fils gauche et droit du nœud. On utilise la valeur `''` pour représenter un fils vide.

Par exemple, l'arbre



est stocké dans

```
a = {'F':['B','G'], 'B':['A','D'], 'A':['',''], 'D':['C','E'], \
     'C':['',''], 'E':['',''], 'G':['','I'], 'I':['','H'], \
     'H':['','']}
```

Écrire une fonction récursive `taille` prenant en paramètres un arbre binaire `arbre` non vide sous la forme d'un dictionnaire et un caractère `lettre` qui est la valeur du sommet de l'arbre, et qui renvoie la taille de l'arbre à savoir le nombre total de nœuds.

On observe que, par exemple, `arbre[lettre][0]`, respectivement `arbre[lettre][1]`, permet d'atteindre la clé du sous-arbre gauche, respectivement droit, de l'arbre `arbre` de sommet `lettre`.

Exemples :

```
>>> taille(a, 'F')
9
>>> taille(a, 'B')
5
>>> taille(a, 'I')
2
```

EXERCICE 2 (10 points)

On considère l'algorithme de tri de tableau suivant : à chaque étape, on parcourt le sous-tableau des éléments non rangés et on place le plus petit élément en première position de ce sous-tableau.

Exemple avec le tableau : `t = [41, 55, 21, 18, 12, 6, 25]`

- Étape 1 : on parcourt tous les éléments du tableau, on permute le plus petit élément avec le premier.

  Le tableau devient `t = [`**`6`**`, 55, 21, 18, 12, 41, 25]`

- Étape 2 : on parcourt tous les éléments **sauf le premier**, on permute le plus petit élément trouvé avec le second.

  Le tableau devient : `t = [`**`6, 12`**`, 21, 18, 55, 41, 25]`

Et ainsi de suite.

Le programme ci-dessous implémente cet algorithme.

```
def echange(tab, i, j):
    '''Echange les éléments d'indice i et j dans le tableau tab.'''
    temp = ...
    tab[i] = ...
    tab[j] = ...

def tri_selection(tab):
    '''Trie le tableau tab dans l'ordre croissant
    par la méthode du tri par sélection.'''
    N = len(tab)
    for k in range(...):
        imin = ...
        for i in range(..., N):
            if tab[i] < ...:
                imin = i
        echange(tab, ..., ...)
```

Compléter ce code de façon à obtenir :

```
>>> tab = [41, 55, 21, 18, 12, 6, 25]
>>> tri_selection(tab)
>>> tab
[6, 12, 18, 21, 25, 41, 55]
```