BACCALAURÉAT

SESSION 2024

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°21

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)

Écrire une fonction `recherche_motif` qui prend en paramètres une chaîne de caractères `motif` non vide et une chaîne de caractères `texte` et qui renvoie la liste des positions de `motif` dans `texte`. Si `motif` n'apparaît pas, la fonction renvoie une liste vide.

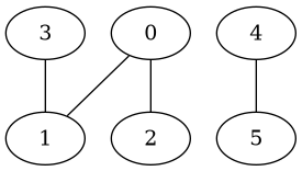
Exemples:

```
>>> recherche_motif("ab", "")
[]
>>> recherche_motif("ab", "cdcdcdcd")
[]
>>> recherche_motif("ab", "abracadabra")
[0, 7]
>>> recherche_motif("ab", "abracadabraab")
[0, 7, 11]
```

EXERCICE 2 (10 points)

Dans cet exercice, on considère un graphe non orienté représenté sous forme de listes d'adjacence. On suppose que les sommets sont numérotés de 0 à $n-1$.

Ainsi, le graphe suivant:



sera représenté par la liste d'adjacence suivante:

```
adj = [[1, 2], [0, 3], [0], [1], [5], [4]]
```

Rappel : cela signifie que les voisins sortants du sommet `i` sont les sommets de la liste `adj[i]`.

On souhaite déterminer les sommets accessibles depuis un sommet donné dans le graphe. Pour cela, on va procéder à un parcours en profondeur du graphe.

Compléter la fonction suivante.

```
def parcours(adj, x, acc):
    '''Réalise un parcours en profondeur récursif
    du graphe donné par les listes d'adjacence adj
    depuis le sommet x en accumulant les sommets
    rencontrés dans acc'''
    if x ...:
        acc.append(x)
        for y in ...:
            parcours(adj, ...)

def accessibles(adj, x):
    '''Renvoie la liste des sommets accessibles dans le
    graphe donné par les listes d'adjacence adj depuis
    le sommet x.'''
    acc = []
    parcours(adj, ...)
    return acc
```

Exemples :

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