Amérique du Nord – 2024 – sujet1 - Correction

Exercice 2 (6 points)

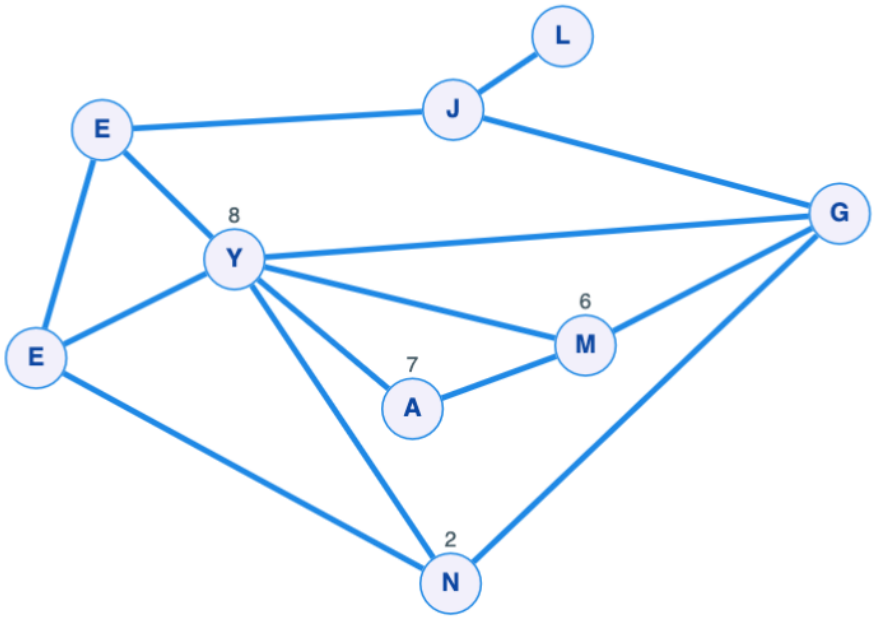
Partie A – Matrice d'adjacence

1. Dessin du graphe

Sommets :
G (Gabriel), J (Jade), Y (Yanis), E (Emma), N (Nina), M (Milo), A (Anas), L (Lou)

Le graphe est **non orienté** (relation d'amitié réciproque).

On relie chaque paire d'amis par une arête.



---

2. Compléter la matrice d'adjacence

Ordre des sommets : `['G', 'J', 'Y', 'E', 'N', 'M', 'A', 'L']`

Matrice :

```
# sommets : G, J, Y, E, N, M, A, L

            matrice_adj = [
            [0, 1, 1, 0, 1, 1, 0, 0],  # G
            [1, 0, 1, 1, 0, 0, 0, 1],  # J
            [1, 1, 0, 1, 1, 1, 1, 0],  # Y
            [0, 1, 1, 0, 1, 0, 0, 0],  # E
            [1, 0, 1, 1, 0, 0, 0, 0],  # N
            [1, 0, 1, 0, 0, 0, 1, 0],  # M
            [0, 0, 1, 0, 0, 1, 0, 0],  # A
            [0, 1, 0, 0, 0, 0, 0, 0]   # L
            ]
```

👉*Commentaire :*

- *Matrice symétrique (graphe non orienté).*
- *Diagonale remplie de 0 (pas de boucle).*

---

3. Résultat de `position`

```
>>> position(sommets, 'G')
0
>>> position(sommets, 'Z')
None
```

👉*Commentaire :*

'G' est en position 0.
'Z' n'est pas dans la liste.

---

4. Fonction `nb_amis` (matrice)

```
def nb_amis(L, m, s):
    pos_s = position(L, s)
    if pos_s == None:
        return None
    amis = 0
    for i in range(len(m)):
        amis += m[pos_s][i]
    return amis
```

👉*Commentaire :*

- *On récupère la ligne correspondant au sommet.*
- *On additionne les 1.*
- *Cela donne le nombre d'amis.*

---

5. Résultat

```
>>> nb_amis(sommets, matrice_adj, 'G')
```

Gabriel est ami avec J, Y, N, M → 4 amis.

Résultat : `4`

Partie B – Dictionnaire

6. Dans un dictionnaire {c : v}

- `c` = **clé**
- `v` = **valeur associée**

---

7. Compléter le dictionnaire

```
graphe = {
'G': ['J', 'Y', 'N', 'M'],
'J': ['G', 'Y', 'E', 'L'],
'Y': ['G', 'J', 'E', 'N', 'M', 'A'],
'E': ['J', 'Y', 'N'],
'N': ['G', 'Y', 'E'],
'M': ['G', 'Y', 'A'],
'A': ['Y', 'M'],
'L': ['J']
}
```

👉 *Commentaire :*

- *Graphe non orienté → relation réciproque.*
- *Si X est dans la liste de Y, alors Y est dans la liste de X.*

---

8. Fonction `nb_amis` (dictionnaire)

```
def nb_amis(d, s):
    return len(d[s])
```

👉 *Commentaire : Nombre d'amis = longueur de la liste associée.*

---

9. Cercle d'amis de Lou (nouvelle situation)

Nouveau graphe donné :

```
graphe = {
'G': ['J', 'N'],
'J': ['G', 'Y', 'E', 'L'],
'Y': ['J', 'E', 'N'],
'E': ['J', 'Y', 'N'],
'N': ['G', 'Y', 'E'],
'M': ['A'],
'A': ['M'],
'L': ['J']
}
```

Depuis L :

L → J
J → G, Y, E
G → N
Y → N
E → N

Tous reliés sauf M et A.

👉 **Cercle d'amis de Lou :**

```
['L', 'J', 'G', 'Y', 'E', 'N']
```

(ordre possible selon parcours)

---

10. Fonction `parcours_en_profondeur`

Version correcte :

```python
def parcours_en_profondeur(d, s, visites=None):
    if visites is None:
        visites = []
    visites.append(s)
    for v in d[s]:
        if v not in visites:
            parcours_en_profondeur(d, v, visites)
    return visites
```

👉*Commentaire :*

- *On évite* `visites = []` *directement en paramètre (problème Python classique).*
- *Parcours récursif.*
- *On explore un voisin avant de revenir en arrière (DFS).*
