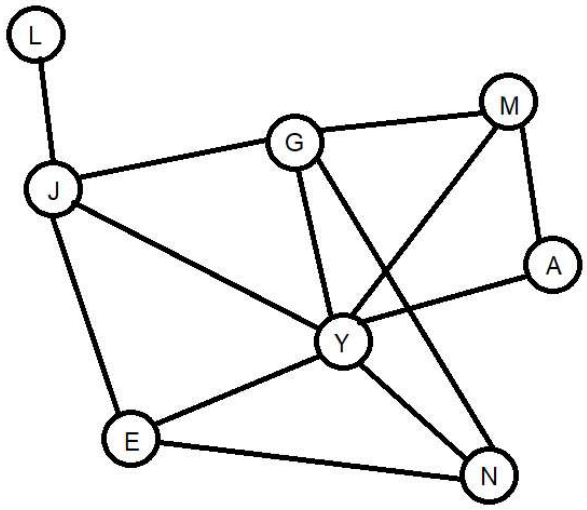
Éléments de correction du devoir surveillé sur les graphes

Exercice 1 :

Sujet Amérique du Nord J1 – 21/05/2024

Partie A : Matrice d'adjacence

1.



2.
```
# 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
```

3.
```
>>> position(sommets,'G')
0
>>> position(sommets,'Z')
None
```

4.
```
1  def nb_amis(L:list,m:list,s:str):
2      pos_s = position(L, s)
3      if pos_s == None:
4          return None
5      amis = 0
6      for i in range(len(m)):
7          amis += m[pos_s][i]  #ou m[i][pos_s]
8      return amis
```

5.
```
>>> nb_amis(sommets,matrice_adj,'G')
4
```

Partie B : Dictionnaire de listes d'adjacence

6. Dans un dictionnaire Python, `c` représente une clé et `v` la valeur associée à cette clé.

7.
```
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']
              }
```

8.
```
def nb_amis(d : dict, s : str) -> int:
    return len(d[s])
```

9. Le cercle d'amis de Lou est : Emma, Gabriel, Jade, Nina et Yanis.

10.
```
1  def parcours_en_profondeur(d,s,visites = []):
2      visites.append(s)
3      for v in d[s]:
4          if v not in visites:
5              parcours_en_profondeur(d,v)
6      return visites
```

Exercice 2

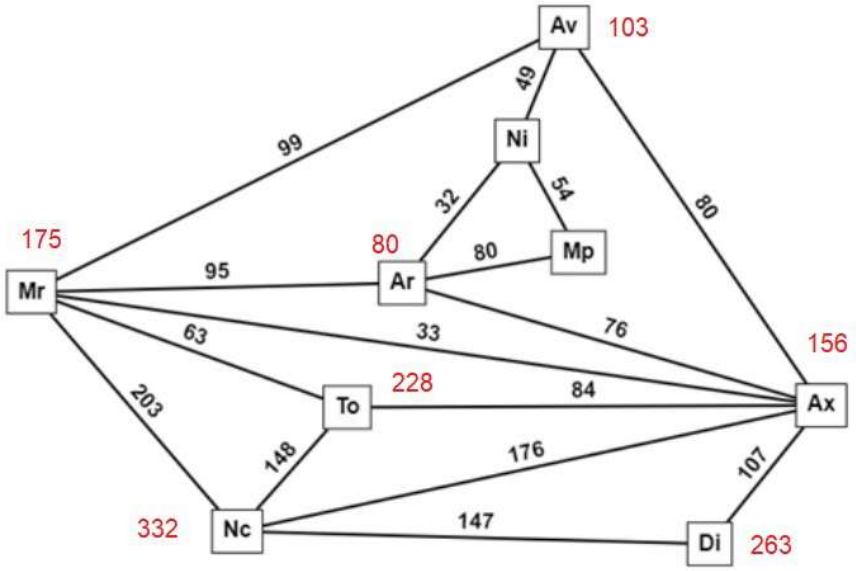
Sujet Polynésie J2 – 21/06/2024

Partie A

1. Mp → Ar → Ax → Nc (332km)

2. Mp → Ar → Mr → Nc

   Mp → Ar → Ax → Nc



Partie B

3.
```
G = {
        'Av': ['Mr', 'Ni', 'Ax'],
        'Ni': ['Av', 'Ar', 'Mp'],
        'Mp': ['Ni', 'Ar'],
        'Ar': ['Mr', 'Ni', 'Mp', 'Ax'],
        'Mr': ['Av', 'Ar', 'Ax', 'To', 'Nc'],
        'Ax': ['Av', 'Ar', 'Mr', 'To', 'Nc', 'Di'],
        'To': ['Mr', 'Ax', 'Nc'],
        'Nc': ['Mr', 'To', 'Ax', 'Di'],
        'Di': ['Nc', 'Ax']
        }
```

4. LIFO : Last In First Out // FIFO : First In First Out

5. L'acronyme utilisé pour désigner la structure de file est FIFO.

6.
```
>>> parcours(G,'Av').
['Av', 'Mr', 'Ni', 'Ax', 'Ar', 'To', 'Nc', 'Mp', 'Di']
```

Remarque : la réponse à cette question dépend complètement du graphe que vous avez donné à la question 3.

7. **proposition A** : parcours en largeur

8.
```
def distance(graphe : dict, sommet : str) -> dict:
    f = creerFile()
    enfiler(f, sommet)
    distances = {sommet: 0}
    visite = [sommet]
    while not estVide(f):
        s = defiler(f)
        for v in graphe[s]:
            if v not in visite:
                visite.append(v)
                distances[v] = distances[s] + 1
                enfiler(f, v)
    return distances
```

9.
```
>>> distance(G,'Av')
{'Av': 0,
 'Mr': 1,
 'Ni': 1,
 'Ax': 1,
 'Ar': 2,
 'To': 2,
 'Nc': 2,
 'Mp': 2,
 'Di': 2}
```

10.
```
def parcours2(G : dict, sommet : str) -> list:
    p = creerPile()
    empiler(p, sommet)
    visite = []
    while not estVide(p):
        s = depiler(p)
        if not (s in visite):
            visite.append(s)
            for v in G[s]:
                empiler(p, v)
    return visite
```

11.
```
>>> parcours2(G,'Av').
['Av', 'Ax', 'Di', 'Nc', 'To', 'Mr', 'Ar', 'Mp', 'Ni']
```

Remarque : la réponse à cette question dépend complètement du graphe que vous avez donné à la question 3.