

Exercices représentation correction

Christophe Viroulaud

Terminale - NSI

Algo 15

Sommaire

1. Exercice 1

2. Exercice 2

3. Exercice 3

4. Exercice 4

Exercice 1

Exercice 2

Exercice 3

Exercice 4

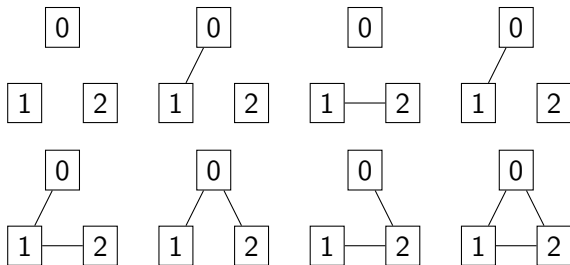
Exercice 1

Exercice 1

Exercice 2

Exercice 3

Exercice 4



Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 2

degrés :

- ▶ A : 4
- ▶ B : 2
- ▶ C : 2
- ▶ D : 3
- ▶ E : 2
- ▶ F : 1

$$\sum_{s \in S} \deg(s) = 14 = 2.A$$

Exercice 1

Exercice 2

Exercice 3

Exercice 4

1. Matrice d'adjacence

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

2. Dictionnaire d'adjacence

```
1 g = {"A": ["B", "C", "D", "F"],
2     "B": ["A", "D"],
3     "C": ["A", "E"],
4     "D": ["A", "B", "E"],
5     "E": ["C", "D"],
6     "F": ["A"]}
```

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 3

Le graphe est complet.

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

Code 1 – Matrice d'adjance

Exercice 1

Exercice 2

Exercice 3

Exercice 4


```
1 def ordre(mat: list) -> int:  
2     return len(mat)
```

```
1 >>> ordre(mat)  
2 5
```

Code 2 – Appel

```
1 def est_complet(mat: list) -> bool:
2     """
3     vérifie si chaque sommet est relié à tous
4     les autres (sauf lui même)
5     """
6     for ligne in range(ordre(mat)):
7         for col in range(ordre(mat)):
8             if (ligne != col and mat[ligne][col] == 0) or \
9                 (ligne == col and mat[ligne][col] == 1):
10                return False
11     return True
```

```
1 >>> est_complet(mat)
2 True
```

Code 3 – Appel

Remarque

Les parenthèses ne sont pas obligatoires ici : la porte **and** est prioritaire devant **or**. Il est conseillé de les ajouter tout de même en cas de doute.

Sommaire

1. Exercice 1
2. Exercice 2
3. Exercice 3
4. Exercice 4

Exercice 1

Exercice 2

Exercice 3

Exercice 4

Exercice 4

- ▶ 0 : $d^+ = 2, d^- = 0$
- ▶ 1 : $d^+ = 1, d^- = 0$
- ▶ 2 : $d^+ = 1, d^- = 1$
- ▶ 3 : $d^+ = 0, d^- = 3$
- ▶ 4 : $d^+ = 2, d^- = 1$
- ▶ 5 : $d^+ = 1, d^- = 2$

1 suivants = [[3, 5], [3], [4], [], [3, 5], [2]]

Exercice 1

Exercice 2

Exercice 3

Exercice 4

```
1 def degres_sortants(liste: list, s: int) -> int:  
2     return len(liste[s])
```

```
1 >>> degres_sortants(suivants, 4)  
2 2
```

Code 4 – Appel

```
1 def degres_entrants(liste: list, s: int) -> int:  
2     deg = 0  
3     for liste_successeurs in liste:  
4         for successeurs in liste_successeurs:  
5             if successeurs == s:  
6                 deg = deg+1  
7     return deg
```

```
1 >>> degres_entrants(suivants, 4)  
2 1
```

Code 5 – Appel

Exercice 1

Exercice 2

Exercice 3

Exercice 4