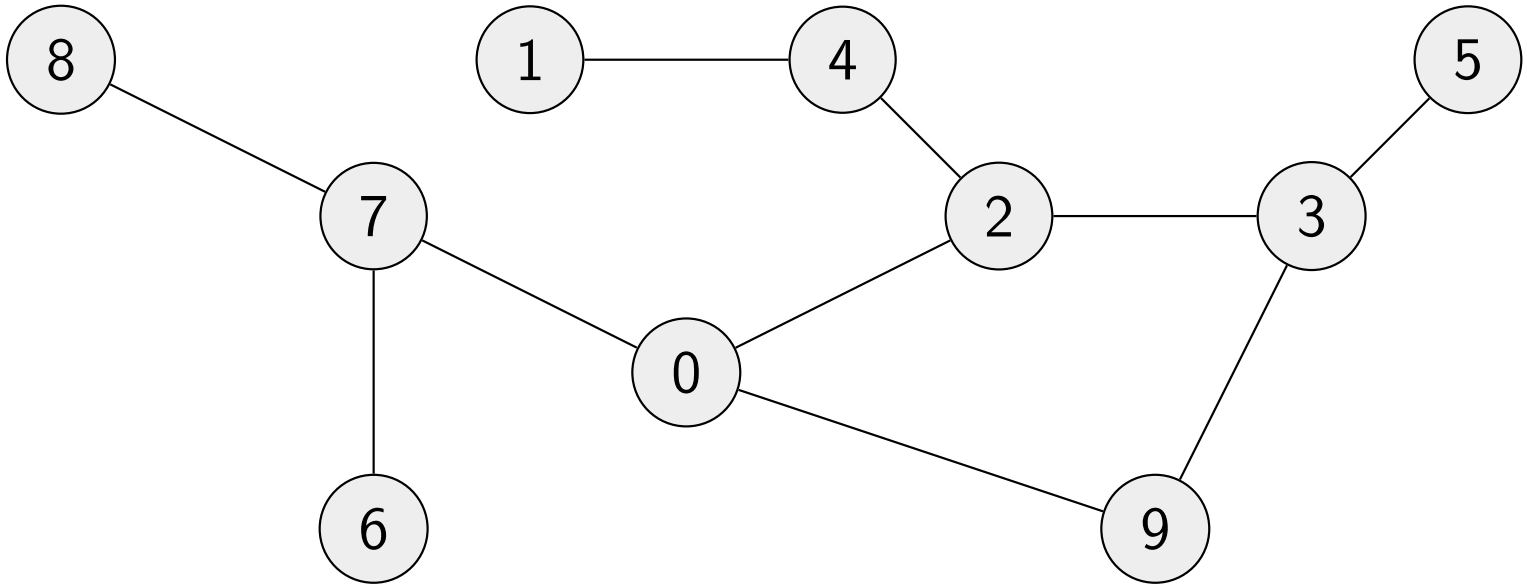
# Parcours en profondeur dans un graphe orienté

Christophe Viroulaud

Terminale - NSI

**Algo 18**

Dans un graphe non orienté et connexe, tous les sommets sont atteignables depuis n'importe quel sommet de départ.



Dans un graphe orienté, ce n'est pas nécessairement le cas.

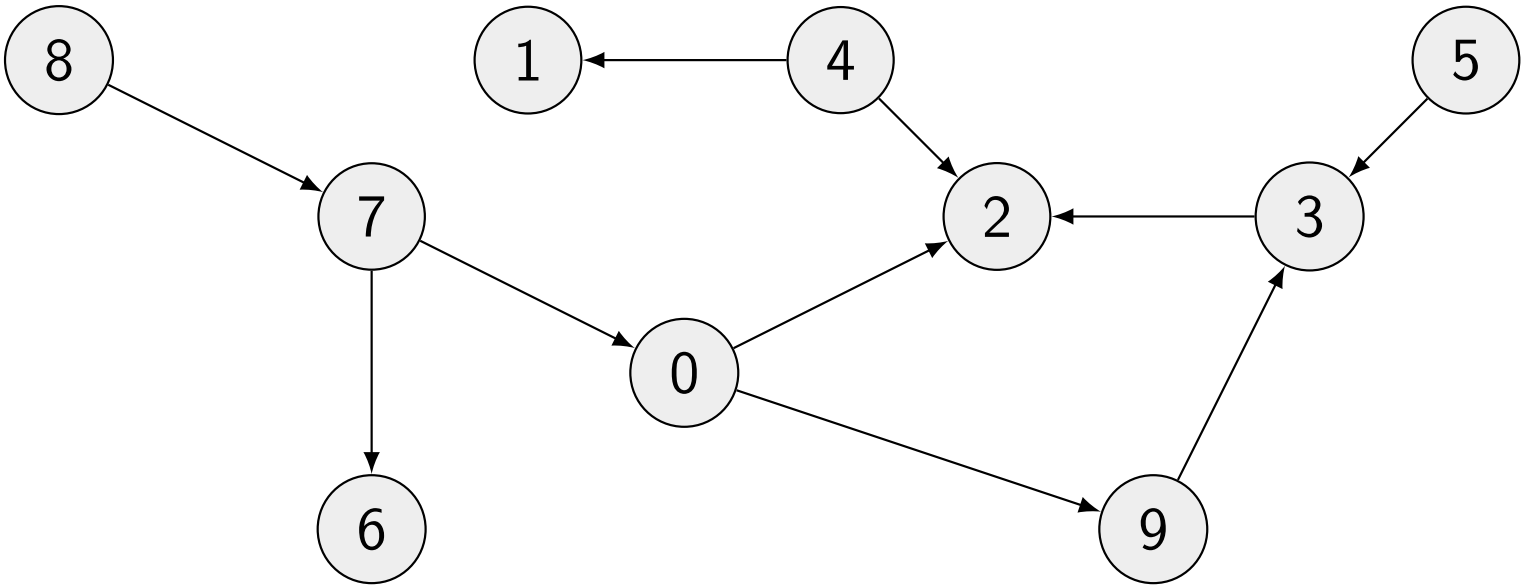
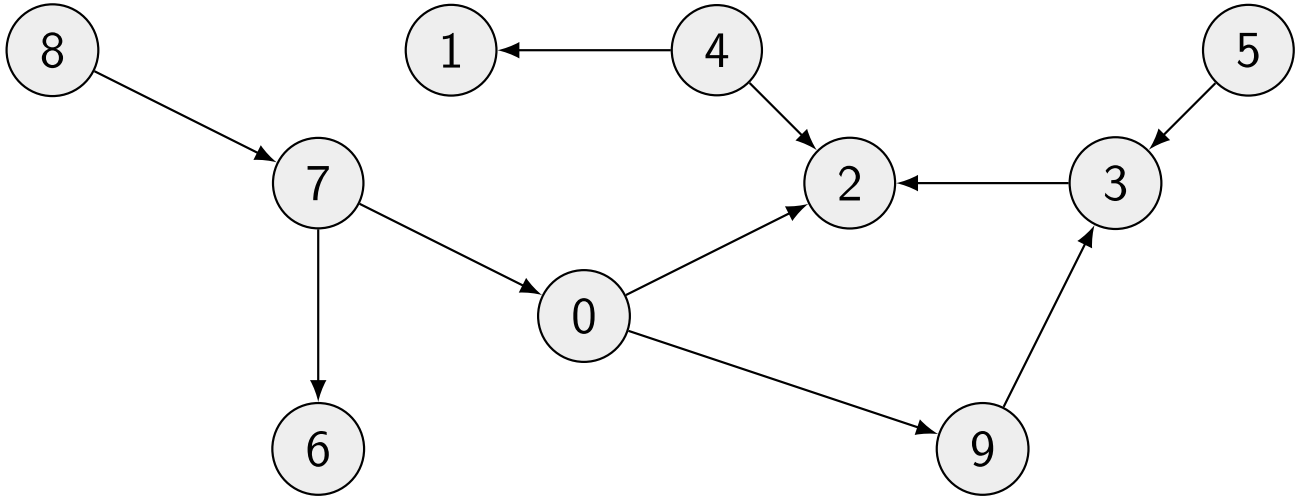



FIGURE 1 – Tous les sommets ne sont pas atteignables depuis 8.

Comment réaliser un parcours en profondeur dans un graphe orienté ?

# Sommaire

# Représentation en mémoire




**Activité 1 :**

1. Télécharger et extraire le dossier compressé **`dfs-annexe.zip`** sur le site https://cviroulaud.github.io
2. Créer le fichier **`dfs.py`** dans le même dossier que **`test.py`**
3. Construire la matrice **`graphe`** des successeurs.
4. Exécuter le fichier de test pour vérifier la matrice.

# Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

# Correction

```
1  graphe = [
2      [0, 0, 1, 0, 0, 0, 0, 0, 0, 1],
3      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
4      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
5      [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
6      [0, 1, 1, 0, 0, 0, 0, 0, 0, 0],
7      [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
8      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
9      [1, 0, 0, 0, 0, 0, 1, 0, 0, 0],
10     [0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
11     [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]]
```

# Sommaire

# Parcours en profondeur



La fonction de parcours en profondeur vue en classe masque un coût : il faut parcourir le tableau `visites` pour vérifier que le sommet n'a pas déjà été traversé.

**À retenir**

Une solution classique consiste à créer un tableau `visites` de booléens. La valeur `False` à l'indice `i` indique que le sommet `i` n'a pas encore été visité.

**Activité 2 :**

1. Construire par compréhension un tableau `visites`, de la taille de l'ordre du graphe et rempli de `False`.
2. Écrire la fonction `dfs(mat: list, dep: int, vis: list)` $\rightarrow$ `None` qui effectue un parcours en profondeur depuis le sommet `dep`. La fonction affichera, dans la console, les sommets traversés au fur et à mesure du parcours.
3. Effectuer un parcours en profondeur depuis le sommet 0.

# Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

# Correction

```
1 visites = [False for _ in range(len(graphe))]
```

# Correction

```python
def dfs(mat: list, dep: int, vis: list) -> None:
    if not vis[dep]:
        # marque le sommet visités
        vis[dep] = True
        print(dep)
        for i in range(len(mat[dep])):
            # c'est un successeur
            if mat[dep][i] == 1:
                dfs(mat, i, vis)
```

```python
visites = [False for _ in range(len(graphe))]
dfs(graphe, 0, visites)
```

Code 1 – Appel de la fonction

# Sommaire

# Parcours en profondeur de tout le graphe

Dans un graphe connexe orienté, tous les sommets ne sont pas forcément atteignables depuis n'importe quel départ. Il faut donc effectuer un parcours depuis chaque sommet.

**Activité 3 :** Écrire la fonction `parcours(mat: list)` $\rightarrow$ `None` qui lance un parcours en profondeur depuis chaque sommet. La fonction construira le tableau `visites` présenté précédemment.

# Avant de regarder la correction

- Prendre le temps de réfléchir,
- Analyser les messages d'erreur,
- Demander au professeur.

# Correction

```
1 def parcours(mat: list) -> None:
2     visites = [False for _ in range(len(mat))]
3     for i in range(len(mat)):
4         # lance un parcours depuis chaque sommet
5         dfs(mat, i, visites)
```

```
1 parcours(graphe)
```

Code 2 – Appel de la fonction