

dfs-annexe.zipParcours en profondeur
dans un graphe orienté

Christophe Viroulaud

Terminale - NSI

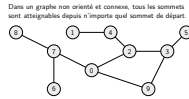
Algo 18

Parcours en profondeur dans un graphe orienté

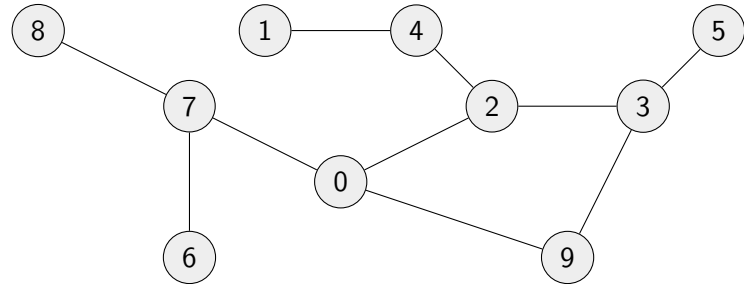
Christophe Viroulaud

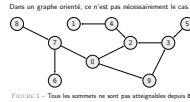
Terminale - NSI

Algo 18Parcours en
profondeur
dans un graphe
orientéReprésentation en
mémoireParcours en
profondeurParcours en
profondeur de tout
le graphe



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.

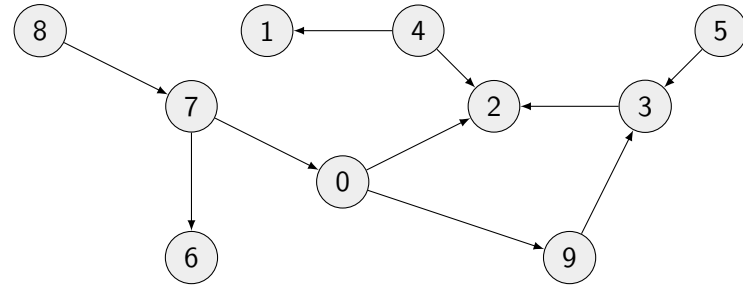


FIGURE 1 – Tous les sommets ne sont pas atteignables depuis 8.

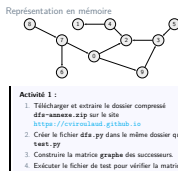
Comment réaliser un parcours en profondeur dans un graphe orienté ?

Comment réaliser un parcours en profondeur dans un graphe orienté ?

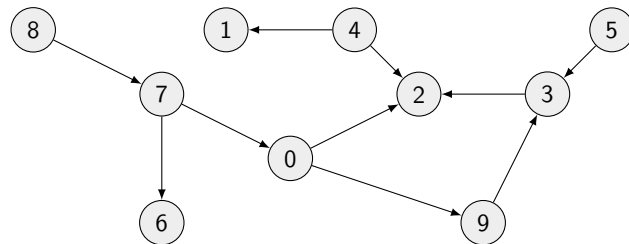
1. Représentation en mémoire
2. Parcours en profondeur
3. Parcours en profondeur de tout le graphe

Sommaire

1. Représentation en mémoire
2. Parcours en profondeur
3. Parcours en profondeur de tout le graphe



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.



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

Avant de regarder la correction



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

```

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]]

```

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]]

```


1. Représentation en mémoire
2. **Parcours en profondeur**
3. Parcours en profondeur de tout le graphe

Sommaire

1. Représentation en mémoire
2. **Parcours en profondeur**
3. Parcours en profondeur de tout le graphe

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.

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) → 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.

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) → 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.



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

Avant de regarder la correction



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

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

Correction

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

Parcours en profondeur dans un graphe orienté

└─ Parcours en profondeur

└─ Correction

Correction

```

1 def dfs(mat: list, dep: int, vis: list) -> None:
2     if not vis[dep]:
3         # marque le sommet visité
4         vis[dep] = True
5         print(dep)
6         for i in range(len(mat[dep])):
7             # c'est un successeur
8             if mat[dep][i] == 1:
9                 dfs(mat, i, vis)

```

```

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

```

Code 1 – Appel de la fonction

Correction

```

1 def dfs(mat: list, dep: int, vis: list) -> None:
2     if not vis[dep]:
3         # marque le sommet visités
4         vis[dep] = True
5         print(dep)
6         for i in range(len(mat[dep])):
7             # c'est un successeur
8             if mat[dep][i] == 1:
9                 dfs(mat, i, vis)

```

```

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

```

Code 1 – Appel de la fonction

- 1. Représentation en mémoire
- 2. Parcours en profondeur
- 3. Parcours en profondeur de tout le graphe

Sommaire

1. Représentation en mémoire
2. Parcours en profondeur
3. Parcours en profondeur de tout le graphe

Parcours en profondeur dans un graphe orienté

└─ Parcours en profondeur de tout le graphe

└─ Parcours en profondeur de tout le graphe

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.

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.

Parcours en
profondeur
dans un graphe
orienté

Représentation en
mémoire

Parcours en
profondeur

Parcours en
profondeur de tout
le graphe

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

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



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

Avant de regarder la correction



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

Parcours en profondeur dans un graphe orienté

└─ Parcours en profondeur de tout le graphe

└─ Correction

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

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

Parcours en
profondeur
dans un graphe
orientéReprésentation en
mémoireParcours en
profondeurParcours en
profondeur de tout
le graphe