L2 MIASHS – Informatique S3
Feuille bonus : Introduction aux graphes

2024

Introduction aux graphes

Un graphe est une structure composée de sommets (ou nœuds) et d'arêtes qui relient ces sommets. Les graphes peuvent être représentés de différentes manières :

- **Liste d'adjacence** : Pour chaque sommet, on associe une liste (ou un ensemble) de ses voisins. - **Matrice d'adjacence** : Un tableau à deux dimensions où chaque cellule indique s'il existe une arête entre deux sommets.

Exemple : Considérons un graphe avec quatre sommets $1, 2, 3, 4$, et des arêtes reliant $1-2$, $1-3$, $2-3$, $2-4$. La liste d'adjacence serait :

$$\{1 : [2, 3], 2 : [1, 3, 4], 3 : [1, 2], 4 : [2]\}$$

—

1 Représentation d'un graphe avec des ensembles

Problème : Vous devez représenter un graphe non orienté à l'aide d'ensembles. Un graphe est constitué de sommets et d'arêtes qui relient ces sommets.
Écrivez une fonction `creer_graphe` qui prend une liste de paires de sommets (représentant les arêtes) et renvoie un dictionnaire où les clés sont les sommets et les valeurs sont des ensembles contenant les sommets adjacents.
Exemple d'utilisation :

```
aretes = [(1, 2), (1, 3), (2, 3), (2, 4)]
print(creer_graphe(aretes))
# Résultat : {1: {2, 3}, 2: {1, 3, 4}, 3: {1, 2}, 4: {2}}
```

—

2 Parcours en largeur d'un graphe

Problème : Vous devez parcourir un graphe non orienté en largeur, en partant d'un sommet donné.

Écrivez une fonction `parcours_largeur` qui prend un graphe (sous forme de dictionnaire) et un sommet de départ, et renvoie la liste des sommets parcourus dans l'ordre du parcours en largeur.
Exemple d'utilisation :

```
graphe = {1: {2, 3}, 2: {1, 4}, 3: {1}, 4: {2}}
print(parcours_largeur(graphe, 1))
# Résultat : [1, 2, 3, 4]
```

—

3 Parcours en profondeur d'un graphe

Problème : Vous devez parcourir un graphe non orienté en profondeur, en partant d'un sommet donné.
Écrivez une fonction `parcours_profondeur` qui prend un graphe (sous forme de dictionnaire) et un sommet de départ, et renvoie la liste des sommets parcourus dans l'ordre du parcours en profondeur.
Exemple d'utilisation :

```
graphe = {1: {2, 3}, 2: {1, 4}, 3: {1}, 4: {2}}
print(parcours_profondeur(graphe, 1))
# Résultat : [1, 2, 4, 3]
```

—

4 Détection de cycles dans un graphe

Problème : Vous devez détecter s'il existe un cycle dans un graphe non orienté.
Écrivez une fonction `detecter_cycle` qui prend un graphe et renvoie `True` s'il existe un cycle, sinon `False`.
Exemple d'utilisation :

```
graphe = {1: {2}, 2: {1, 3}, 3: {2, 4}, 4: {3, 1}}
print(detecter_cycle(graphe))
# Résultat : True
```