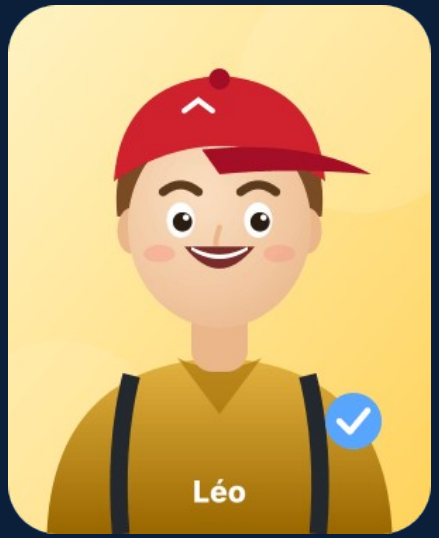
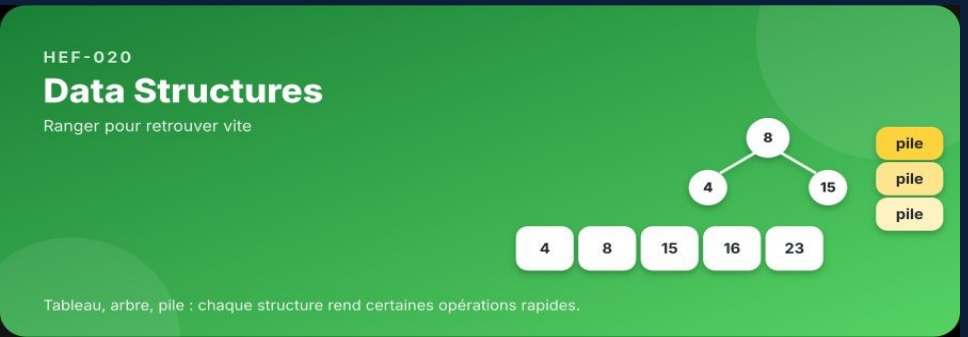
HEF-020 · MISSION 08 / 10

Graphes : modéliser les relations

Réseaux sociaux, cartes routières, dépendances de cours : tout ce qui est « relié à » est un graphe.



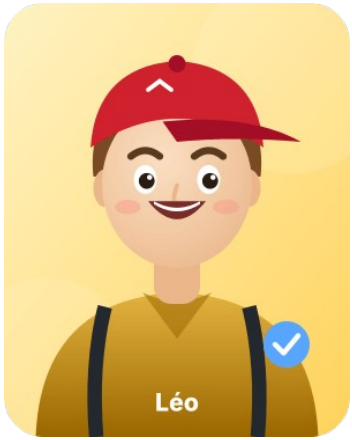


Badge : Cartographe

75 min · débutant

L'histoire

HashBox veut ajouter un réseau de **mentorat** : chaque membre peut suivre d'autres membres, et on veut proposer « des gens que tu pourrais connaître ». Il faut aussi ordonner les cours selon leurs **prérequis**.



« Ce sont des problèmes de relations, dit Sam. Des listes ne suffisent pas, des arbres non plus. Il te faut des graphes. »

Tu apprends à modéliser n'importe quel réseau de relations.

Ta mission

À la fin, je serai capable de…

1. définir sommet, arête, orienté/non orienté, pondéré
2. modéliser un problème réel sous forme de graphe
3. représenter un graphe avec une liste d'adjacence
4. représenter un graphe avec une matrice d'adjacence
5. comparer les deux représentations

Défi d'échauffement

Défi d'échauffement

Dessine le réseau de **cinq personnes** (toi inclus) : trace un trait entre deux personnes qui se connaissent. Ensuite, réponds : qui a le plus de relations ? Existe-t-il quelqu'un d'isolé ? Comment ferais-tu pour qu'un ordinateur retienne ton dessin ?

Le concept

Vocabulaire

Un **graphe** est un ensemble de **sommets** (nœuds) reliés par des **arêtes**.

Type	Signification	Exemple
Non orienté	La relation est réciproque	« est ami avec »
Orienté	La relation a un sens	« suit », « est prérequis de »
Pondéré	Chaque arête a un coût	distance, durée

Le **degré** d'un sommet est le nombre d'arêtes qui y touchent. Un **chemin** est une suite de sommets reliés. Un **cycle** est un chemin qui revient à son point de départ.

Un arbre est un graphe particulier : connexe et sans cycle.

Le concept

Modéliser

La première compétence : transformer un problème en graphe.

Problème	Sommets	Arêtes
Carte routière	villes	routes (pondérées par la distance)
Réseau social	personnes	amitiés ou abonnements
Prérequis de cours	cours	« A doit précéder B » (orienté)
Pages web	pages	liens

Liste d'adjacence

Pour chaque sommet, on garde la liste de ses voisins.

Le concept

Liste d'adjacence (suite)

```
graphe = {
    "Ada": ["Bob", "Cléo"],
    "Bob": ["Ada"],
    "Cléo": ["Ada", "Dan"],
    "Dan": ["Cléo"],
}
```

Matrice d'adjacence

Un tableau carré : la case `[i][j]` vaut 1 s'il existe une arête de i vers j, 0 sinon.

```
        Ada Bob Cléo Dan
Ada      0   1    1    0
Bob      1   0    0    0
Cléo     1   0    0    1
Dan      0   0    1    0
```

Le concept

Quelle représentation ?

Critère	Liste d'adjacence	Matrice
Mémoire	proportionnelle aux arêtes	proportionnelle au carré des sommets
Voisins d'un sommet	immédiat	parcourir une ligne
« A est-il relié à B ? »	parcourir la liste de A	immédiat
Graphe clairsemé (peu d'arêtes)	idéal	gaspillage

En pratique, la **liste d'adjacence** est le choix par défaut.

Exemple pas à pas

Problème : à partir d'une liste d'amitiés, construire le graphe, puis proposer des « amis d'amis ».

```
def construire(amities):
    g = {}
    for a, b in amities:
        g.setdefault(a, set()).add(b)
        g.setdefault(b, set()).add(a)
    return g

def amis_d_amis(g, personne):
    proposes = set()
    for ami in g[personne]:
        proposes |= g[ami]
    return sorted(proposes - g[personne] - {personne})

g = construire([("Ada", "Bob"), ("Ada", "Cléo"), ("Cléo", "Dan"), ("Bob", "Eve")])
print(sorted(g["Ada"]))        # ['Bob', 'Cléo']
print(amis_d_amis(g, "Ada"))   # ['Dan', 'Eve']
```

Exemple pas à pas (suite)

Idée clé : on a choisi des ensembles comme voisinage, car on teste souvent « est-il déjà mon ami ? » et on veut éviter les doublons.

À toi, avec aide

À toi, avec aide

Voici les prérequis de cours (A → B signifie « A doit précéder B ») :

```
HEF-000 → HEF-010
HEF-010 → HEF-020
HEF-010 → HEF-030
HEF-020 → HEF-030
```

1. Construis le graphe **orienté** en liste d'adjacence.
2. Écris `prerequis_directs(graphe, cours)` qui renvoie les cours qui précèdent directement un cours.
3. Quel est le degré sortant de HEF-010 ?

À toi, avec aide · réponse

Indices progressifs

Indice 1

Dans un graphe orienté, on n'ajoute l'arête que dans un sens.

Indice 2

Pour les prérequis directs, cherche les sommets dont la liste contient le cours.

Indice 3

Le degré sortant est la taille de la liste de HEF-010.

À toi, avec aide · réponse

Réponse

```
graphe = {
    "HEF-000": ["HEF-010"],
    "HEF-010": ["HEF-020", "HEF-030"],
    "HEF-020": ["HEF-030"],
    "HEF-030": [],
}

def prerequis_directs(graphe, cours):
    return sorted(c for c, suivants in graphe.items() if cours in suivants)

print(prerequis_directs(graphe, "HEF-030"))   # ['HEF-010', 'HEF-020']
print(len(graphe["HEF-010"]))                 # 2
```

À toi, en solo

Écris `en_matrice(graphe)` qui convertit une liste d'adjacence (dictionnaire) en matrice, et `en_liste(sommets, matrice)` qui fait l'inverse.
Teste que `en_liste(*en_matrice(g))` redonne `g` (listes triées).
Puis calcule, pour 1 000 sommets et 3 000 arêtes, la mémoire (nombre de cases) des deux représentations.

À toi, en solo · réponse

Pistes de correction

```
def en_matrice(graphe):
    sommets = sorted(graphe)
    indice = {s: i for i, s in enumerate(sommets)}
    m = [[0] * len(sommets) for _ in sommets]
    for s, voisins in graphe.items():
        for v in voisins:
            m[indice[s]][indice[v]] = 1
    return sommets, m

def en_liste(sommets, matrice):
    return {s: [sommets[j] for j, x in enumerate(matrice[i]) if x] for i, s in enu
   merate(sommets)}

g = {"a": ["b", "c"], "b": ["a"], "c": ["a"]}
assert {k: sorted(v) for k, v in en_liste(*en_matrice(g)).items()} == g
print("matrice :", 1000 * 1000, "cases ; liste :", 3000 * 2, "entrées")
```

À toi, en solo · réponse

Pistes de correction (suite)

La matrice coûte 1 million de cases pour seulement 3 000 arêtes (liste : 6 000 entrées pour un graphe non orienté).

Pièges classiques

Piège	Pourquoi c'est un problème	Comment l'éviter
Oublier l'arête retour	Un graphe non orienté devient orienté	Ajouter les deux sens
Sommet absent du dictionnaire	`KeyError`	Inclure tous les sommets, même isolés
Matrice pour un graphe clairsemé	Mémoire gaspillée	Liste d'adjacence
Mal choisir les sommets	Modèle illisible	Se demander : qu'est-ce qui est relié à quoi ?

Mini-boss

Mini-boss

Modélise le **réseau de mentorat** de HashBox.

Données : liste de couples `(mentor, élève)`.

1. Construis un graphe orienté.
2. Calcule le **degré sortant** (nombre d'élèves) de chaque mentor et le **degré entrant** (nombre de mentors) de chaque élève.
3. Trouve les mentors sans élève, et les élèves sans mentor.
4. Trouve le mentor qui a le plus d'élèves.
5. Vérifie qu'il n'y a pas de couple « A mentor de B et B mentor de A ».

Mini-boss · réponse

Pistes de correction

```
def analyser(couples):
    g, entrants = {}, {}
    for m, e in couples:
        g.setdefault(m, set()).add(e)
        g.setdefault(e, set())
        entrants.setdefault(e, set()).add(m)
        entrants.setdefault(m, set())
    sortant = {s: len(v) for s, v in g.items()}
    entrant = {s: len(v) for s, v in entrants.items()}
    sans_eleve = sorted(s for s, d in sortant.items() if d == 0)
    sans_mentor = sorted(s for s, d in entrant.items() if d == 0)
    meilleur = max(sortant, key=sortant.get)
    reciproques = sorted((a, b) for a in g for b in g[a] if a in g[b] and a < b)
    return sortant, entrant, sans_eleve, sans_mentor, meilleur, reciproques

couples = [("Ada", "Bob"), ("Ada", "Cléo"), ("Bob", "Dan")]
print(analyser(couples))
```

Quiz éclair

Quiz éclair

1. Dans un graphe, une arête relie :

A. deux graphes
B. un sommet à lui-même uniquement
C. deux sommets

2. Pour un graphe avec peu d'arêtes, la meilleure représentation est :

A. la liste d'adjacence
B. la matrice
C. aucune

3. « A est prérequis de B » se modélise par un graphe :

A. non orienté seulement

Quiz éclair

Quiz éclair (suite)

B. orienté
C. sans sommets

4. 🔁 *Révision mission 07* : Un tas-min a pour racine :

A. le plus grand
B. un élément au hasard
C. le plus petit élément

5. 🔁 *Révision mission 06* : Un parcours en ordre d'un BST donne :

A. les valeurs triées
B. les valeurs inversées
C. les feuilles

Quiz éclair · réponse

Réponses

1. **C**.
2. **A**.
3. **B**.
4. **C**.
5. **A**.

Je réfléchis

Je réfléchis

1. Quel réseau de ta vie (amis, collègues, trajets) aimerais-tu modéliser ?
2. Qu'est-ce qui rend un problème « de graphe » ?

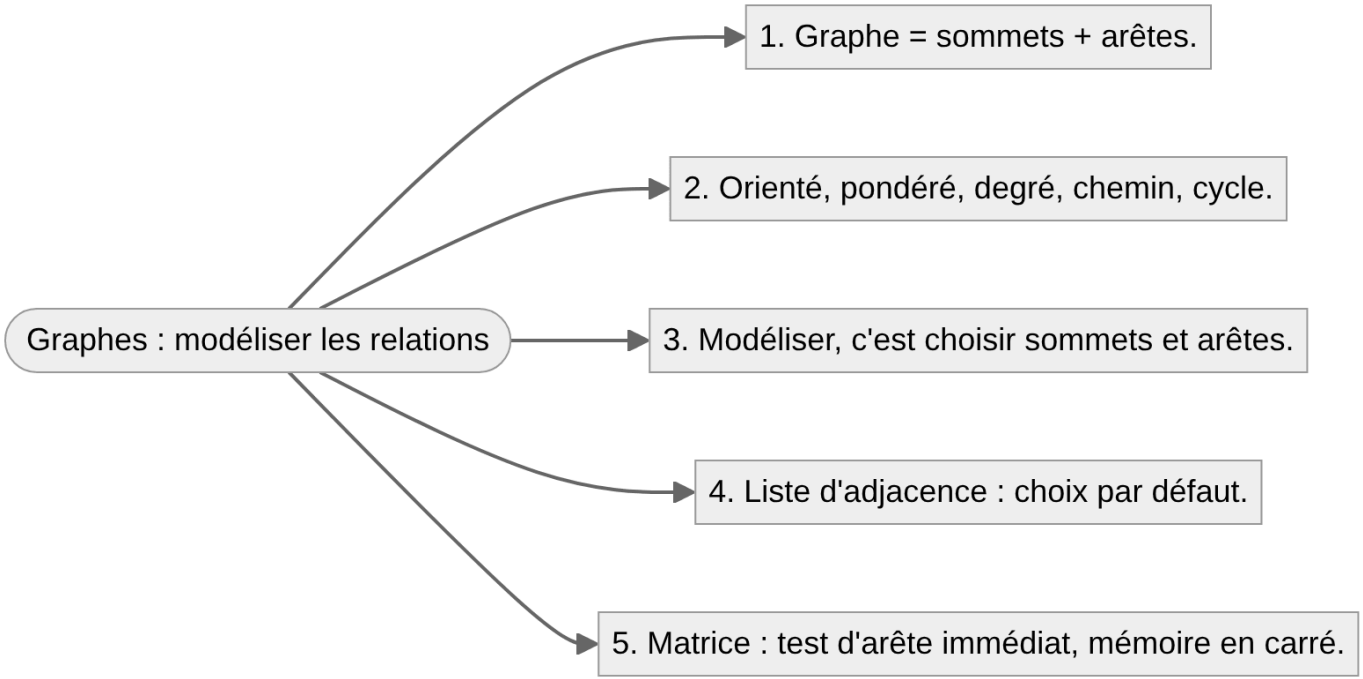
À retenir

À retenir

1. Graphe = sommets + arêtes.
2. Orienté, pondéré, degré, chemin, cycle.
3. Modéliser, c'est choisir sommets et arêtes.
4. Liste d'adjacence : choix par défaut.
5. Matrice : test d'arête immédiat, mémoire en carré.

À retenir

Le schéma de la mission



MISSION ACCOMPLIE

+100 XP

Badge débloqué : Cartographe

Prochaine mission : Explorer un graphe : DFS et BFS

