Graphes abstraits

Introduction :

Les graphes sont des structures de données relationnelles très utiles pour la modélisation de nombreuses problématiques, dans des domaines très variés.
Nous en préciserons d'abord les caractéristiques dans une première partie. Nous étudierons ensuite les parcours et cheminements sur les graphes dans une deuxième partie. Nous nous intéresserons enfin, dans une troisième partie, aux modalités d'implémentation des structures de données de type graphe.

1 Caractérisation des graphes

La notion de graphe est principalement connue sous l'angle de sa représentation graphique, mais c'est aussi un type de structure de données relationnelles.

→ La représentation graphique d'une structure de type graphe fait figurer des éléments reliés entre eux par des liaisons.

a. Éléments constitutifs d'un graphe

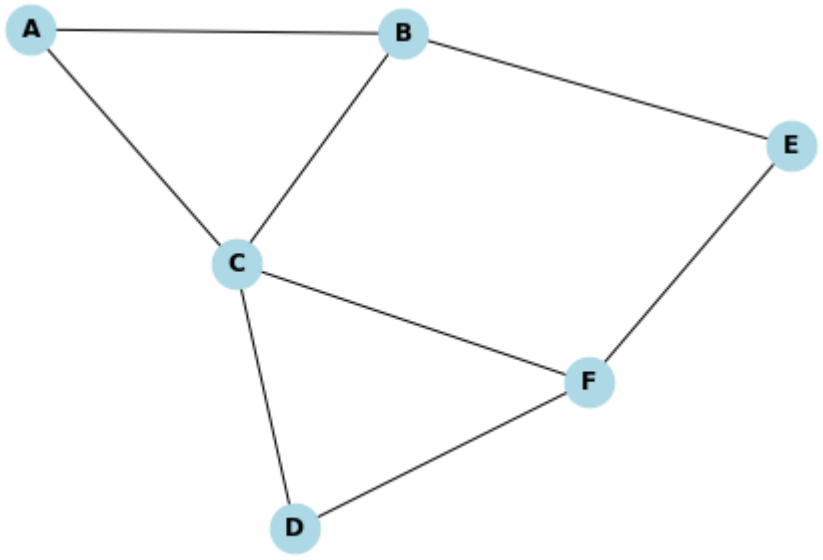
Un graphe se compose de deux types d'éléments complémentaires : les **sommets** et les **arêtes**.

- Les sommets, également appelés « nœuds », sont les éléments individuels.
- Les arêtes sont les connexions reliant deux sommets entre eux, matérialisées graphiquement par une ligne.

Exemple

Le graphe ci-dessous comporte :

- 6 sommets ou nœuds ;
- 8 arêtes.



Les sommets sont A, B, C, D, E et F.
Les arêtes relient les paires de sommets suivantes :

- A et B ;
- A et C ;
- B et C ;
- C et D ;
- B et E ;
- E et F ;
- C et F ;
- D et F.

Tous les sommets ne doivent pas nécessairement être liés entre eux. On remarque dans notre exemple qu'aucune arête ne relie B et F.

À retenir

> Un graphe est défini par deux ensembles complémentaires :
>
> - l'ensemble des sommets ;
> - l'ensemble des arêtes reliant des sommets par paires.

b. Orientation

On distingue deux types de graphes selon leur orientation :

- les graphes **non orientés** ;
- les graphes **orientés**.

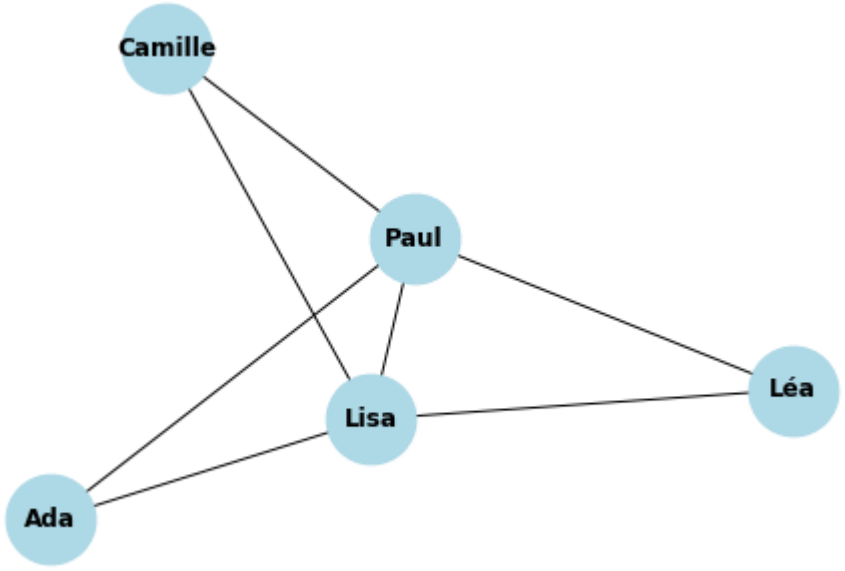
1 Graphe non orienté

À retenir

> Sur un graphe non orienté, les relations sont **mutuelles** : quand une arête relie deux sommets, il n'y a pas de notion de sens.

Ces relations s'illustrent bien avec des réseaux sociaux connus : c'est le cas pour le réseau social Facebook où les relations entre les membres peuvent être représentées par un graphe non orienté.

Exemple



Paul est ami avec Lisa, et Lisa est amie avec Paul. L'arête traduit la symétrie de la relation.

Attention

La nature des relations entre membres des réseaux sociaux varie d'un réseau à l'autre.

2 Graphe orienté

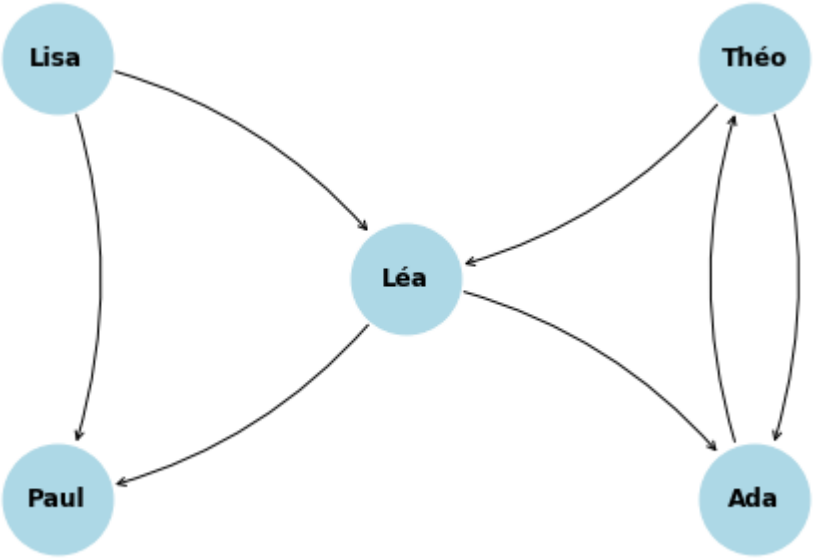
À retenir

Sur un graphe orienté, les sommets ne sont pas reliés par des arêtes mais par des arcs indiquant un sens à la relation. Quand ils sont reliés entre eux, les sommets d'un graphe orienté peuvent donc l'être par un ou deux arcs, selon qu'il y ait ou non réciprocité.

Sur des réseaux sociaux, comme Twitter ou Instagram, la relation n'est pas nécessairement mutuelle. Le suivi unilatéral est possible : un membre choisit de suivre les publications d'un autre, mais il n'y a pas de réciprocité automatique. La personne suivie peut, si elle le souhaite, suivre l'autre personne en retour, mais rien ne l'y oblige.

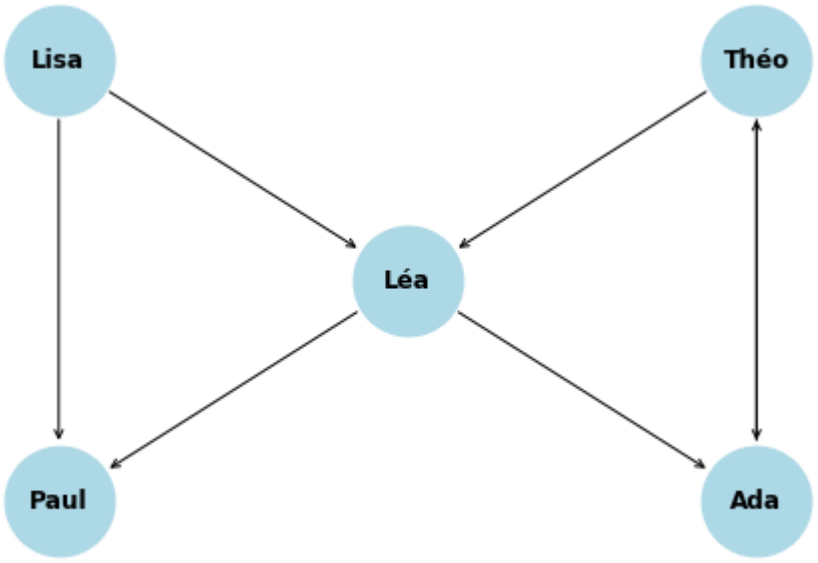
Les relations unilatérales, sur des graphes orientés, sont représentées par des **arcs fléchés**. Une éventuelle réciprocité de la relation est représentée par un arc complémentaire dans l'autre sens.

Exemple



- Lisa suit Paul et Léa. Elle n'est suivie par personne.
- Léa suit Paul et Ada. Elle est suivie par Lisa et par Théo.
- Théo suit Léa et Ada.
- Ada suit Théo.
- Seuls Ada et Théo se suivent mutuellement.

Les graphes orientés sont parfois représentés avec des segments de droite fléchés, bidirectionnels le cas échéant. Le schéma ci-dessous illustre le même graphe que précédemment.

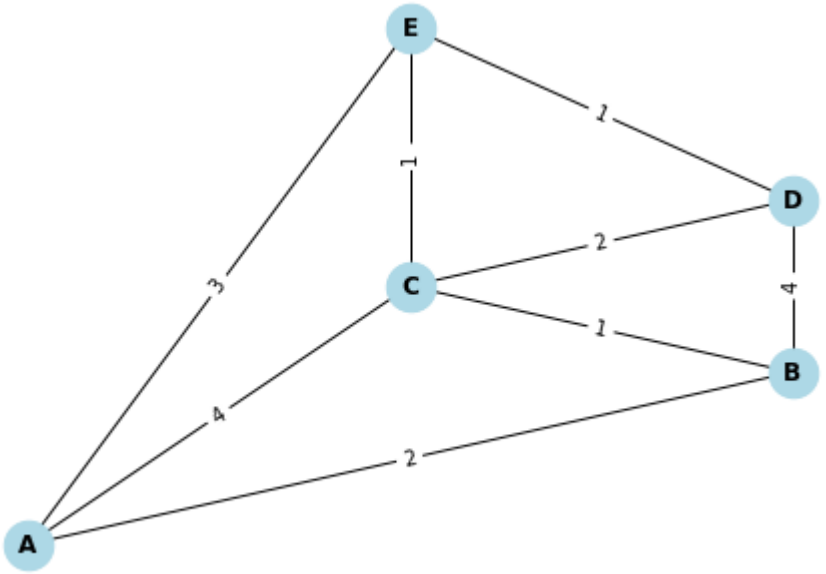


c. Pondération

Les graphes peuvent être ou non pondérés :

- les relations des graphes **non pondérés** ont toutes la même valeur, qui n'a donc pas à être précisée ;
- les relations des graphes **pondérés** pouvant varier, elles doivent être précisées.

→ La valeur affectée à la relation est appelée **pondération** ou **coût**.



À retenir

- Quand la pondération porte sur un graphe non orienté, elle est affectée à chaque arête.
- Quand la pondération porte sur un graphe orienté, une pondération est affectée à chaque arc.

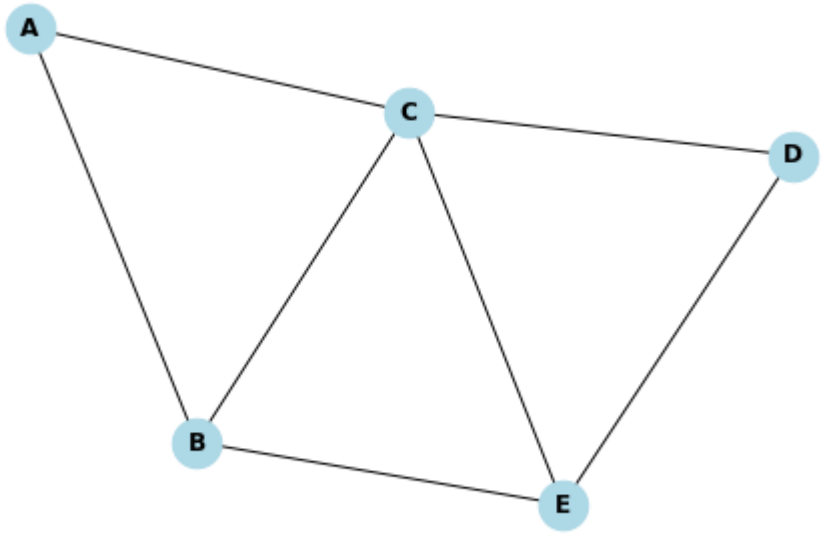
d. Chemin

Un **chemin** est une séquence d'arêtes distinctes (un chemin ne repasse pas par la même arête) permettant de relier deux sommets entre eux. Il peut exister plusieurs chemins reliant une même paire de sommets.

Exemple

Sur le schéma ci-dessous, différentes chaînes plus ou moins longues de segments permettent de rejoindre le sommet D en partant du sommet A :

- A, B, C, D ;
- A, B, C, E, D ;
- A, B, E, D ;
- A, B, E, C, D ;
- A, C, B, E, D ;
- A, C, D ;
- A, C, E, D.



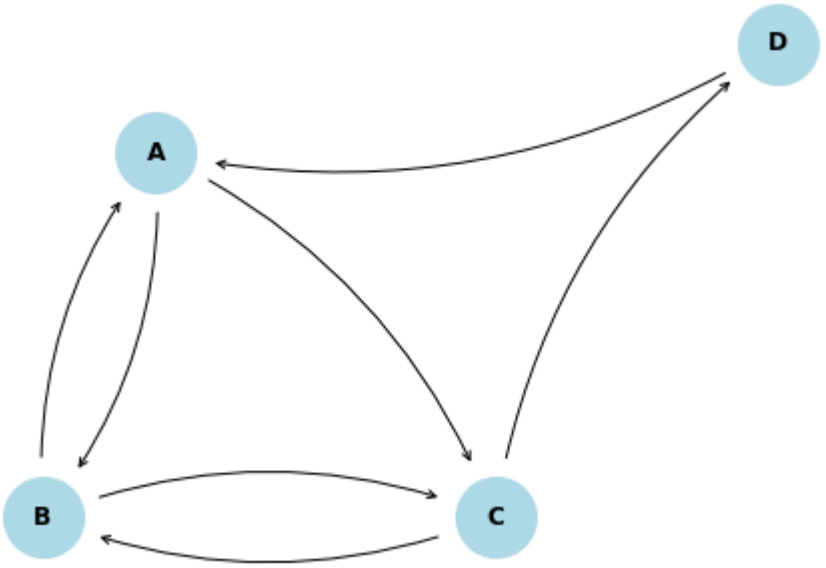
Les graphes ou les portions de graphes peuvent comporter des **cycles**, c'est-à-dire des chaînes de segments permettant de retourner à un point de départ sans passer deux fois par un même sommet.

Quand les graphes sont orientés, les séquences d'arcs, appelées chemins, peuvent présenter un caractère cyclique en formant des **chemins fermés**.

Dans un graphe orienté, les chemins entre un sommet de départ et un sommet d'arrivée ne sont pas nécessairement symétriques.

Exemple

Le schéma ci-après montre qu'il est possible de se rendre en C à partir de A, mais le retour direct de C en A n'est pas possible. Pour se rendre de C en A, il faut emprunter d'autres chemins, en passant soit par D, soit par B.



Cette capacité à distinguer les sens de cheminement entre deux sommets peut s'avérer utile dans différentes applications.

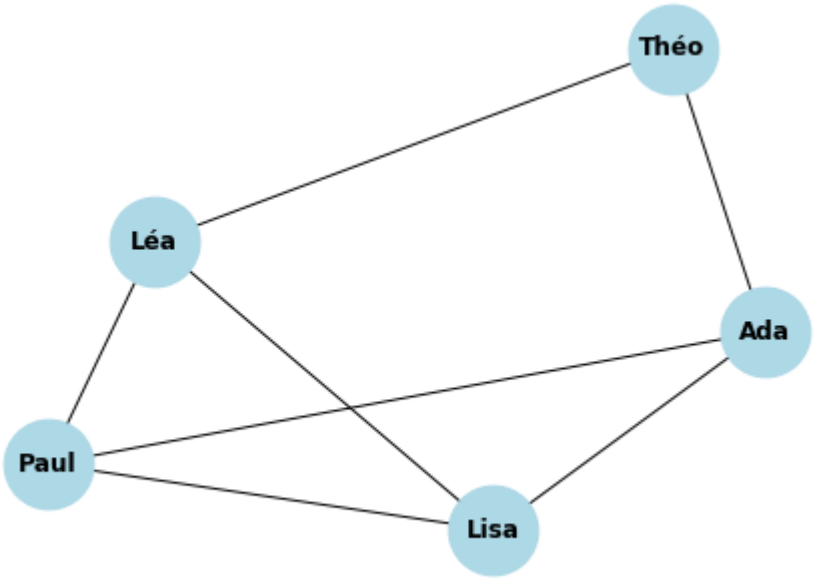
→ Par exemple, quand les logiciels et applications de navigation routière et d'aide à la conduite calculent et proposent un itinéraire, ils doivent prendre en compte le fait que certains axes de circulation sont en sens unique, notamment en centre-ville.

e. Adjacence

Des sommets sont dits **adjacents**, ou **voisins**, s'ils sont reliés entre eux par une arête.

Exemple

L'amitié entre membres d'un réseau social est représentée sur un graphe par l'adjacence des sommets correspondant aux membres de ce réseau.



- Léa est amie avec Paul, Lisa et Théo.
- Théo est ami avec Léa et Ada.

On peut facilement déterminer les ami·e·s commun·e·s de deux personnes en comparant les listes de leurs ami·e·s, représentées par les sommets adjacents à celui considéré. On peut également déterminer quel·le·s ami·e·s de l'un·e ne sont pas ami·e·s de l'autre pour éventuellement les suggérer comme nouveaux·elles ami·e·s possibles.

Exemple

- Lisa et Théo sont tous deux ami·e·s avec Ada et Léa qui sont leurs amies communes.
- Lisa est également amie avec Paul, mais pas avec Théo. Le réseau social pourrait demander à Théo s'il connaît Paul.

Nous avons défini les principales caractéristiques des graphes, lesquels peuvent être ou non orientés, de même qu'ils peuvent être ou non pondérés. Les sommets des graphes peuvent être reliés entre eux par des arêtes ou des arcs selon le type de graphe, et composer des cheminements. Intéressons-nous maintenant à ces cheminements et aux parcours des graphes.

2 | Parcours

Dans cette partie nous nous intéressons aux différentes manières de parcourir les graphes.

a. Définition

Le terme « parcours » est parfois employé comme synonyme de chemin ou de cheminement, mais l'expression « le parcours d'un graphe » désigne l'exploration de celui-ci afin de visiter tous ses sommets.

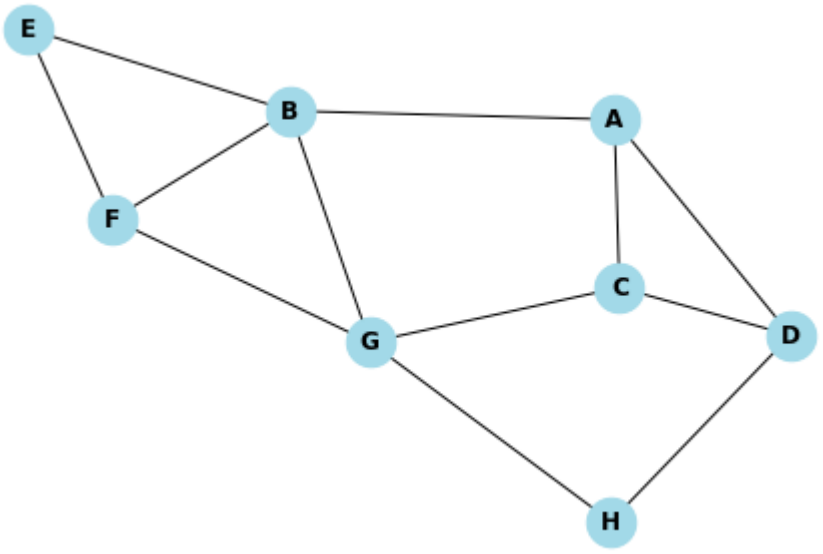
Définition

> **Parcours d'un graphe :**
>
> Le parcours d'un graphe consiste à visiter l'ensemble des sommets d'un graphe à partir d'un point de départ quelconque.

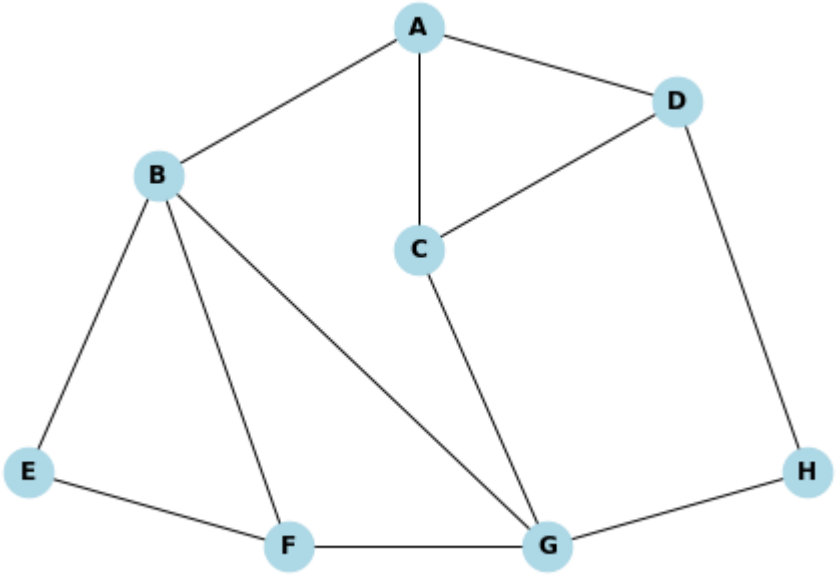
Il existe plusieurs manières de parcourir les graphes. Nous étudions ici les plus courantes que sont :

- le **parcours en largeur** ;
- et le **parcours en profondeur**.

Nous allons effectuer ces deux parcours sur un même graphe.



Un même graphe peut être dessiné de nombreuses manières. Pour faciliter la compréhension des parcours que nous allons présenter, nous redessinons le graphe précédent en lui donnant une orientation plus verticale et en positionnant les sommets par ordre alphabétique.



Si la forme de ce second schéma est différente, les relations entre les sommets sont strictement identiques à la représentation précédente. C'est bien le même graphe.

b. Parcours en largeur

À retenir

> Le parcours en largeur d'un graphe consiste, à partir du sommet de départ, à considérer toutes les adjacences de ce sommet, puis les adjacences des sommets adjacents, et ainsi de suite.
> On garde la trace de chaque sommet visité pour ne pas les visiter plusieurs fois.

Ce parcours s'apparente à une exploration par cercles concentriques depuis le sommet de départ dont on s'éloigne progressivement. On visite d'abord tous les voisins immédiats du sommet initial, puis tous les voisins des voisins initiaux qui n'ont pas déjà été visités, puis leurs voisins qui n'ont pas déjà été visités, etc.

→ La répétition de ce processus permet de découvrir l'intégralité des sommets présents sur le graphe.

1. On commence par le sommet A, en notant chacun des sommets que nous visitons.
2. On commence donc par visiter chacun des voisins immédiats de A, c'est-à-dire B, C et D. Ils sont marqués comme **visités**.
3. On considère ensuite les voisins des voisins B, C et D :
    - B a pour voisins E, F et G, qui n'ont pas été visités. On les visite et on les marque comme visités.
    - C a pour voisins G et D, mais ils ont tous deux déjà été visités.
    - D a pour voisins C et H. C a déjà été visité, on visite H.
    - Depuis H on constate que tous les voisins ont été visités.

→ Le parcours est terminé. Il a été effectué par la visite des sommets suivants, dans cet ordre : A, B, C, D, E, F, G, H.

Les sommets ont été disposés pour être présentés par ordre alphabétique afin de faciliter la compréhension du parcours, mais il n'y a pas de notion d'ordre entre voisins d'un même sommet. Après avoir exploré les voisins initiaux de A que sont B, C et D, on aurait pu poursuivre l'exploration en commençant par les voisins de C, ou de D, au lieu de B.

C. Parcours en profondeur

À retenir

> Contrairement au parcours en largeur, le parcours en profondeur ne considère pas d'emblée l'ensemble des voisins, mais un seul d'entre eux, à partir duquel il cherche à aller le plus loin possible, jusqu'à être entourés de sommets déjà visités.
> Quand cela se produit, on revient sur les sommets découverts mais non visités aux précédentes étapes.

Comme pour le parcours en largeur, on doit garder la trace de chaque sommet visité. On commence arbitrairement par le sommet A pour faciliter la comparaison avec le parcours en largeur que nous venons d'effectuer.

1. Commençant par le sommet A, on identifie les voisins B, C et D.

Astuce

> On choisit arbitrairement de visiter B (comme pour le parcours en largeur, on aurait pu indifféremment choisir un autre des trois sommets voisins de A).

2. Le sommet B a pour voisins non visités E, F et G. On choisit de visiter E.
3. E a pour voisins B et F. B a déjà été visité, on visite F.
4. G a pour voisins B et G. B a déjà été visité, on visite G.
5. G a pour voisins C et H. On choisit arbitrairement de visiter C.
6. C a pour voisins A et D. A a déjà été visité, on visite D.
7. D a pour voisins A et H. A a déjà été visité, on visite H.

→ H a pour voisins D et G. Tous deux ont déjà été visités. Le parcours est terminé. Il a été effectué par la visite des sommets suivants, dans cet ordre : A, B, E, F, G, C, D, H.

En largeur comme en profondeur, les graphes peuvent être parcourus de différentes manières, en fonction de l'ordre dans lequel sont traités les sommets voisins à visiter.

L'ordre des chemins parcourus lors d'un parcours en largeur et d'un parcours en profondeur sont très différents, mais la logique d'exploration du graphe est similaire, à une différence près.

d. Algorithmiques de parcours


Astuce

> L'algorithme des deux modes de parcours est en réalité **identique**. C'est la manière dont on traite les sommets restant à explorer qui fait la différence.


Rappel

> Voir le cours Conteneur de données
>
> - Le parcours en largeur est effectué en recourant à une pile de type FIFO (*First In, First Out*, soit premier arrivé premier sorti), également appelée file d'attente.
> - Le parcours en profondeur est effectué en recourant à une pile de type LIFO (*Last In, First Out*, soit dernier arrivé premier sorti).

La pile (ou file) est initialisée avec le sommet de départ choisi.
Tant que la pile (ou file) n'est pas vide, on la dépile (ou défile) pour traiter le sommet courant, qui est ajouté à la liste des sommets visités.

On ajoute à la pile (ou file) l'ensemble des adjacents du sommet courant qui n'ont pas déjà été visités, c'est-à-dire qui ne font pas partie des sommets déjà visités.

Quand la pile ou la file est vide, l'ensemble du graphe a été parcouru et la liste des sommets constitue son parcours.

e. Recherche de chemins

La recherche de chemins consiste à déterminer les chemins possibles entre deux sommets. Cette recherche est généralement orientée vers une recherche d'efficacité : celle du **meilleur chemin**.

Cette recherche s'inscrit souvent dans le contexte d'un graphe pondéré, et éventuellement orienté. Le critère selon lequel un chemin sera jugé meilleur qu'un autre doit donc être défini.

Les logiciels de navigation routière peuvent évaluer les chemins possibles d'un point de départ à un point d'arrivée selon différents critères :

- rapidité du parcours ;
- nombre de kilomètres parcourus ;
- coût kilométrique.

De la même façon, pour se rendre d'une ville à l'autre, le réseau routier peut proposer différents axes :

- une circulation sur autoroutes ;
- une circulation sur routes principales et voies rapides sans péages ;
- une circulation par de petites routes secondaires.

→ Si le critère choisi est le temps de parcours, le trajet autoroutier constitue souvent le meilleur chemin tant qu'il n'occasionne pas de détours importants.

→ Si le critère choisi est la minimisation du nombre de kilomètres, le meilleur parcours emprunte souvent de petites routes, moins sujettes à contournements que les autoroutes et certaines routes principales.((fleche)) Si le critère choisi est la minimisation du coût kilométrique, la détermination du meilleur itinéraire prendra en compte la présence ou non de péage, mais aussi de la consommation du véhicule, laquelle varie avec sa vitesse et donc les axes empruntés.

Il existe plusieurs algorithmes de détermination du chemin le plus court. Le plus connu est celui de l'informaticien néerlandais Dijkstra, étudié en classe

de seconde dans le cours portant sur le calcul d'itinéraire.

La recherche du meilleur chemin s'applique également au routage dans les réseaux informatiques.
La métrique des protocoles de routage, également appelée coût des routes, dépend du protocole utilisé. Les protocoles de routage interne présentés dans le cours de Terminale sur les réseaux et la sécurité s'appuient sur deux métriques différentes en fonction des protocoles utilisés :

- le nombre de sauts pour le protocole RIP ;
- la bande passante disponible de chaque segment pour le protocole OSPF.

Pour le calcul des routes selon RIP, chaque relation a la même valeur : elles peuvent donc être représentées par un graphe non pondéré.
Le calcul des routes selon OSPF prenant en compte la bande passante disponible des différents segments composant une route, leur représentation nécessite un graphe pondéré.

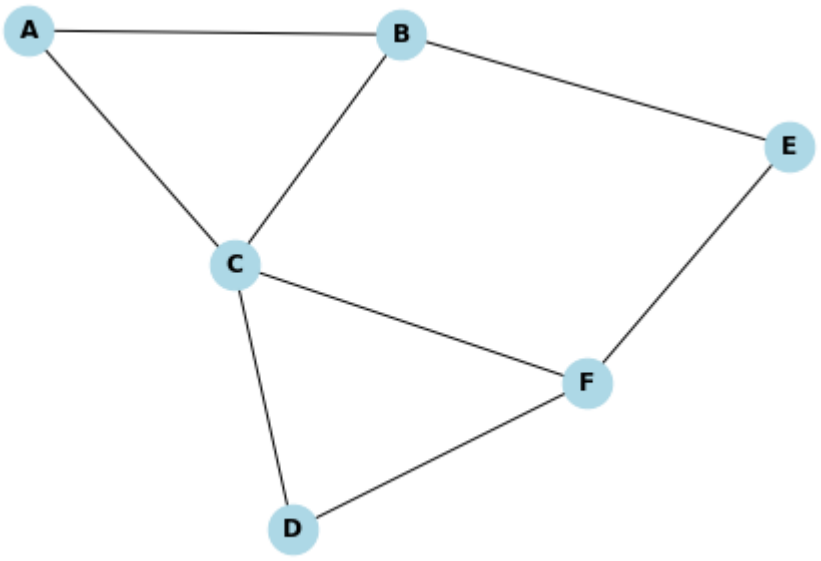
3 | Adjacence

Les adjacences sont une propriété importante des graphes. Elles indiquent la proximité entre des sommets ou nœuds.

Il existe plusieurs manières de représenter et d'implémenter ces adjacences. Nous étudierons les adjacences selon deux représentations :

- les **matrices d'adjacence** ;
- les **listes d'adjacence**.

Nous utiliserons le même graphe ci-dessous pour construire les deux représentations.



a. Matrice d'adjacences

À retenir

> La **représentation par matrice** consiste à matérialiser chacune des adjacences, elles-mêmes matérialisées par l'existence d'une relation entre deux nœuds, elle-même matérialisée par une arête.

Cela revient à remplir un tableau dont les entrées horizontales et verticales sont les nœuds.

	A	B	C	D	E	F
A	0	1	1	0	0	0
B	1	0	1	0	1	1
C	1	1	0	1	0	1
D	0	0	1	0	0	1
E	0	1	0	0	0	1
F	0	0	1	1	1	0

Sur la première ligne, on considère le sommet A. Le graphe montre qu'il est relié par une arête aux sommets B et C. On note donc un 1 dans les colonnes correspondantes. L'absence d'arête est notée par un 0.

Astuce

> Cette matrice est symétrique par rapport à la diagonale, s'agissant d'un graphe non orienté. Elle ne le serait pas nécessairement pour un graphe orienté, sauf dans le cas particulier où il comporterait uniquement des arcs réciproques.

b. Listes d'adjacences

À retenir

> La **représentation par listes d'adjacence** consiste à lister, pour chaque sommet, l'ensemble des sommets qui y sont eux-mêmes reliés par une arête.

A [B, C]
B [A, C, E]
C [A, B, D, F]
D [C, F]
E [B, F]
F [E, C, D]

L'implémentation des listes d'adjacence peut prendre la forme suivante :

```
adjacences = {
        'A': ['B', 'C'],
        'B': ['A', 'C', 'E'],
        'C': ['A', 'B', 'D', 'F'],
        'D': ['C', 'F'],
        'E': ['B', 'F'],
        'F': ['C', 'D', 'E']
}
```

Astuce

> L'expression « listes d'adjacence » pourrait faire penser que celles-ci doivent obligatoirement être implémentées avec des listes, mais ce n'est pas le cas. On pourrait remplacer par des **sets** dans le cas présent, et par des **dictionnaires** pour des graphes pondérés.

```
# Implémentation à base de sets
adjacences = {
        'A': {'B', 'C'},
        'B': {'A', 'C', 'E'},
        'C': {'A', 'B', 'D', 'F'},
        'D': {'C', 'F'},
        'E': {'B', 'F'},
        'F': {'C', 'D', 'E'},

}
```

Si on souhaite n'avoir qu'une seule implémentation pour les graphes pondérés ou non, les graphes non pondérés sont affectés d'une même valeur arbitraire.

```
# Implémentation avec des dictionnaires
adjacences = {
        'A': {'B': 1, 'C': 1},
        'B': {'A': 1, 'C': 1, 'E': 1},
        'C': {'A': 1, 'B': 1, 'D': 1, 'F': 1},
        'D': {'C': 1, 'F': 1},
        'E': {'B': 1, 'F': 1},
        'F': {'C': 1, 'D': 1, 'E': 1},

}
```

Attention

> Dans le cas de graphes orientés, la simple adjacence n'est pas suffisante pour renseigner les propriétés du graphe. Le caractère orienté des arcs

se traduit par l'établissement de deux listes pour les différents sommets :

- listes de prédécesseurs ;
- listes de successeurs.

Les matrices d'adjacence et les listes d'adjacence sont les principales structures de données pour représenter les graphes. D'autres structures sont utilisables mais cela dépasse le cadre de ce cours.

Abordons maintenant l'intérêt d'utiliser plutôt l'une ou l'autre de ces structures.

c. Choix d'une représentation

Le choix d'une représentation dépend du traitement algorithmique que l'on souhaite mettre en place et de la densité du graphe.

Définition

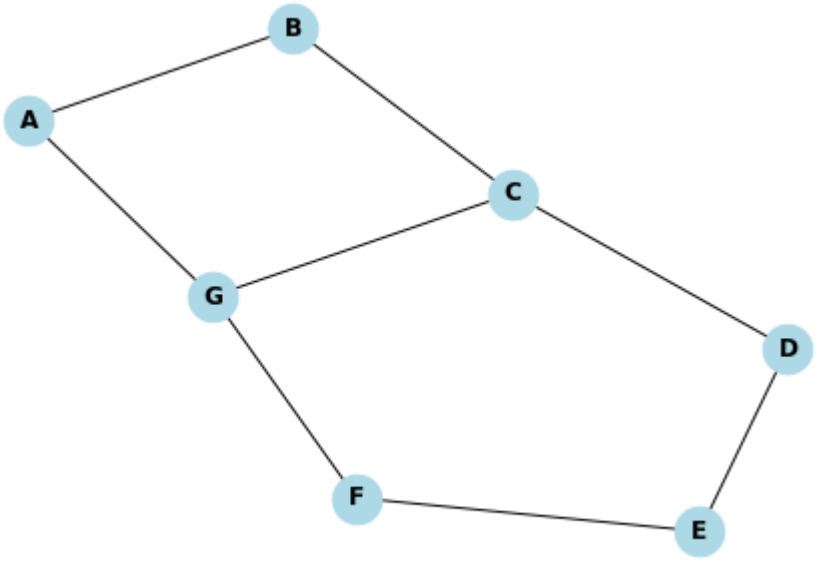
Densité d'un graphe :

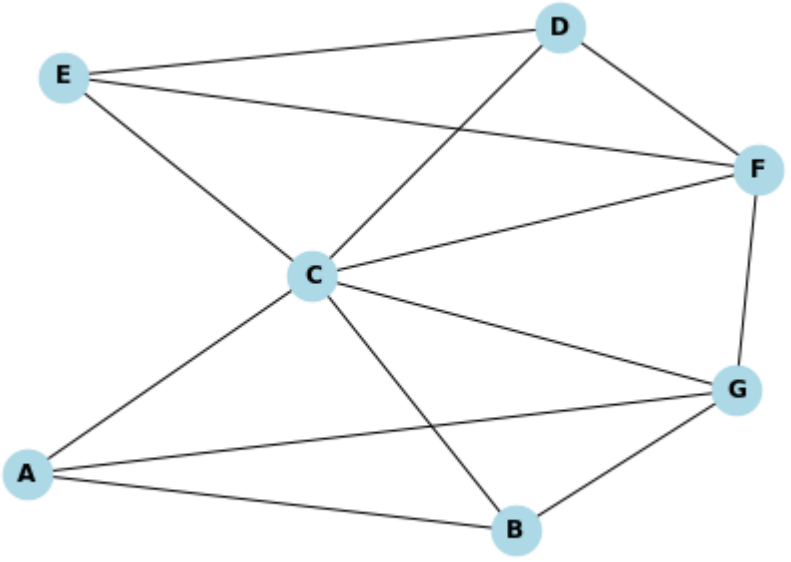
La densité d'un graphe exprime la proportion des relations existantes parmi celles possibles entre tous les sommets.

Un graphe dont les sommets sont très largement reliés entre eux sera considéré comme **dense**. *A contrario* un graphe dont les sommets sont relativement peu reliés est dit **creux**.

Exemple

Les deux graphes ci-après comportent le même nombre de sommets, avec une densité plus ou moins importante de relations.





À retenir

Les listes d'adjacence sont plus indiquées pour des graphes creux.
Les matrices sont préférables pour des graphes denses.

→ Le choix du mode de représentation dépend également des traitements que l'on envisage d'effectuer sur le graphe.

Les graphes peuvent être implémentés selon différents paradigmes. Ils peuvent notamment être implémentés sous forme de classes, pour encapsuler les données qui leur sont liées, et leur appliquer différentes méthodes correspondant aux fonctionnalités requises.

Les traitements sur les graphes peuvent notamment inclure :

- l'ajout, la modification ou la suppression de sommets ;
- l'ajout, modification ou suppression de relations (arcs ou arêtes), et de leurs pondérations éventuelles ;
- la recherche d'adjacences, prenant en compte la notion de prédécesseurs et de successeurs dans le cas de graphes orientés.

> Conclusion :
>
> Nous avons présenté dans ce cours les structures de données relationnelles appelées graphes. Nous avons d'abord caractérisé les graphes, lesquels peuvent être orientés ou non, et pondérés ou non. Nous nous sommes ensuite intéressés aux parcours sur les graphes et à la recherche de chemins optimisés. Nous avons enfin abordé les modalités d'implémentation des graphes, et notamment des adjacences sous forme de listes ou de matrices. Les différents exemples illustrant ce cours ont montré la grande diversité des problématiques qu'il est possible de modéliser à l'aide de graphes, du routage aux itinéraires routiers en passant par les réseaux sociaux.