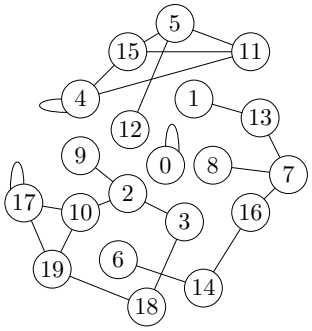



FIGURE 1

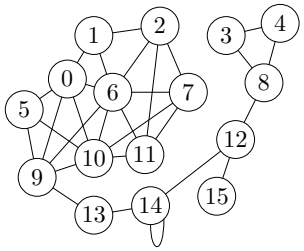



FIGURE 2

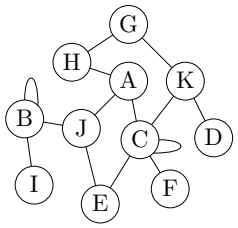



FIGURE 3

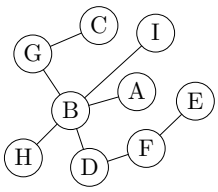



FIGURE 4

## Exercice 1.

Cet exercice est à traiter sur feuille.

1. Pour le graphe de la figure 3, lors d'un parcours en largeur ayant pour sommet initial A, quel est l'ordre dans lequel les sommets sont visités ?
2. Même question pour le parcours en profondeur.

## Exercice 2. Composantes connexes

Soit $G = (S, A)$ un graphe non orienté. Une ***composante connexe*** est un ensemble de sommets non vide $S' \subset S$ tel que :

- ⋆ Pour tout $s_1 \in S'$ et tout $s_2 \in S'$, il existe une chaîne de $s_1$ à $s_2$ dans $G$.
- ⋆ Pour tout $s_1 \in S'$ et tout $s_2 \in S \setminus S'$, il n'existe pas de chaîne de $s_1$ à $s_2$ dans $G$.

On admet que l'ensemble des sommets peut toujours être partitionné comme une union disjointe de composantes connexes :

$$S = \bigcup_{i=1}^{k} S_i \qquad \text{où} \qquad \begin{cases} \text{Pour tout } i,\ S_i \text{ est une composante connexe.} \\ \text{Pour tout } i \neq j : S_i \cap S_j = \varnothing. \end{cases}$$

On admet également que pour tout sommet $s \in S$, il existe une unique composante connexe $S' \subset S$ telle que $s \in S'$.

1. Déterminer les différentes composantes connexes du graphe de la figure 1.

Dans la suite, on suppose que l'ensemble des sommets est de la forme $S = [\![0; n-1]\!]$ avec $n \in \mathbb{N}$. En Python, une composante connexe sera représentée par une liste de type `list[int]` et l'ensemble des composantes connexes par une liste de type `list[list[int]]` (chaque sous-liste est l'une des composantes connexes).

2. À l'aide de parcours en profondeur, proposer un pseudo-code pour déterminer les composantes connexes de $G$.

Récupérez le fichier annexe sur la page du cours (il contient la représentation Python des quatre graphes ci-dessus) :

`https://informatique-lhp.fr/itc-mpsi.html`

3. Dans cette question, on suppose que $G$ est décrit par sa matrice d'adjacence « `M: list[list[int]]` ».
   (a) Écrire une fonction **récursive** qui prend en entrée `M` ainsi qu'un sommet initial $s \in S$ et renvoie la liste des sommets visités lors d'un parcours en profondeur à partir de $s$. Il s'agit donc d'implémenter l'algorithme récursif décrit en cours.
   (b) En suivant la procédure de la question 2, écrire une fonction qui prend en entrée `M` et renvoie la liste des composantes connexes de $G$.
   (c) En déduire une fonction qui indique si $G$ est connexe.

# Exercice 3. Prédécesseurs dans un parcours de graphe

Dans tout l'exercice, on s'intéresse à $G = (S, A)$ un graphe non orienté et on lance un parcours en largeur de $G$ (en réalité, on pourrait tout autant utiliser un parcours en profondeur). Soient $s$ et $t$ deux sommets. On dit que $s$ est le ***prédécesseur*** de $t$ si $s$ est le voisin de $t$ qui a permis de découvrir $t$ lors du parcours. Ainsi, à chaque fois qu'un sommet $s \in S$ est visité, $s$ devient le prédécesseur de tous ses voisins qui n'avaient pas encore été découverts.

1. Dans cette question, $G$ est le graphe de la figure 3 et on s'intéresse au parcours en largeur trouvé dans l'exercice 1. Indiquer le prédécesseur de chacun des sommets de $G$ pour ce parcours.

En Python, les graphes seront décrits par listes d'adjacence. Ainsi, si on note `t` le type utilisé pour représenter les sommets, alors on dispose d'un dictionnaire « `d_G: dict[t: list[t]]` » tel que `d_G[s]` est la liste des voisins de `s` dans $G$. Étant donné $G$, on lance un parcours en largeur à partir d'un sommet initial $s_0$ et on note « `P: dict[t: t]` » le dictionnaire qui associe à chaque sommet son prédécesseur dans le parcours.

2. Écrire une fonction qui prend en entrée $G$ ainsi que $s_0$, et renvoie `P`.

Dans la suite, on va voir plusieurs applications de la notion de prédécesseur.

**Application 1 : recherche d'une chaîne entre deux sommets.**

3. (a) Soient $s$ et $t$ deux sommets de $G$. En utilisant la notion de prédécesseurs, expliquer comment construire une chaîne élémentaire entre $s$ et $t$ (c'est à dire une chaîne dont tous les sommets sont distincts deux à deux).
   (b) Dans le cas où le parcours utilisé est un parcours en largeur, que peut-on dire de la chaîne construite ?
   (c) Écrire une fonction qui prend en entrée $G$, $s$, $t$ et renvoie une chaîne entre $s$ et $t$. Votre fonction renverra `None` s'il n'y a pas de chaîne entre $s$ et $t$.

**Application 2 : construction d'un arbre couvrant.** On souhaite construire un ***arbre couvrant*** de $G$, c'est à dire un graphe non orienté $G' = (S', A')$ avec $S' = S$ et $A' \subset A$ qui soit connexe et sans cycle.

4. (a) Proposer un arbre couvrant pour le graphe de la figure 3 et pour le graphe de la figure 4.
   (b) On suppose que $G$ n'est pas connexe. Déterminer tous les arbres couvrants de $G$.
5. On suppose que $G$ est connexe.
   (a) En utilisant la notion de prédécesseurs, expliquer comment construire un arbre couvrant pour $G$.
   (b) Qu'obtient-on dans le cas du parcours en largeur trouvé dans l'exercice 1 ?
   (c) Écrire une fonction qui prend en entrée $G$ ainsi qu'un sommet $s_0$, et renvoie un dictionnaire « `d_AC: dict[t: list[t]]` » contenant les listes d'adjacence de l'arbre couvrant obtenu à partir d'un parcours en largeur ayant pour sommet initial $s_0$.

**Application 3 : détection de cycles.** On appelle ***cycle élémentaire*** un cycle dont tous les sommets sont distincts (mis à part le premier et le dernier sommets). On admet que $G$ admet un cycle si et seulement s'il admet un cycle élémentaire. Notre but est de déterminer si $G$ contient un cycle et si c'est le cas d'en construire un cycle élémentaire. Pour cela, on commence par construire un graphe $G' = (S', A')$ avec $S' = S$ et $A' \subset A$ tel que :

→ $G'$ ne contient pas de cycle.

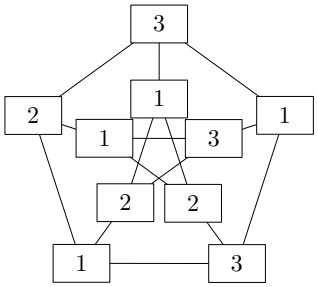
→ Les composantes connexes de $G$ et de $G'$ sont identiques.

6. (a) En s'inspirant de l'exercice 2 et des question précédentes, décrire une procédure pour construire la graphe $G'$.
   (b) Supposons que le graphe $G'$ ait été construit. Expliquer comment construire un cycle élémentaire de $G$ (si un tel cycle existe).
   (c) Écrire une fonction qui prend en entrée `d_G` et renvoie un cycle élémentaire de $G$. Si $G$ ne contient pas de cycle, votre fonction renverra `None`.

# Exercice 4. Coloration d'un graphe

Le ***nombre chromatique*** d'un graphe non orienté $G$ noté $\chi(G)$ est le nombre minimal de couleurs nécessaires pour colorer les sommets de $G$ en faisant en sorte que deux sommets adjacents ne soient pas de la même couleur. Par exemple, le nombre chromatique du graphe ci-contre est 3 (cette figure présente une coloration possible où les trois couleurs sont remplacées par les entiers 1,2 et 3).

1. Écrire une fonction qui prend en entrée un graphe $G$ et renvoie le nombre chromatique de $G$. On pourra procéder par recherche exhaustive.




**Remarque.** Le problème de coloration d'un graphe est un problème NP-complet. Le fait de trouver un algorithme de résolution efficace est donc une question ouverte.