Cours 9 : Notion de graphe

Lycée Fénelon

BCPST1

Notion de graphe
- Définition
- Chemin et connexité
- Matrice d'adjacence d'un graphe

Implémentation des graphes
- Par une liste d'arêtes
- Par une liste d'adjacence
- Par une matrice d'adjacence

Graphe valué
- Définition
- Matrice de valuation
- Longueur d'un chemin

Parcours en largeur d'un graphe
- Principe
- Exemple
- Implémentation à partir d'une liste d'adjacence

Ce cours traite de la notion de graphe et de la façon de les implémenter.

Ce cours traite de la notion de graphe et de la façon de les implémenter.

Definition

Un **graphe (fini)** $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ est la donnée de deux ensembles finis :

- un ensemble fini $\mathscr{S}$ d'éléments, appelés les **sommets** ;
- un sous-ensemble $\mathscr{A}$ de $\mathscr{S} \times \mathscr{S}$, de couples de sommets, appelés les **arêtes**.

Ce cours traite de la notion de graphe et de la façon de les implémenter.

Definition

Un **graphe (fini)** $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ est la donnée de deux ensembles finis :

- un ensemble fini $\mathscr{S}$ d'éléments, appelés les **sommets** ;
- un sous-ensemble $\mathscr{A}$ de $\mathscr{S} \times \mathscr{S}$, de couples de sommets, appelés les **arêtes**.

Le graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ est dit **non orienté** si :

$$\forall\, (a, b) \in \mathscr{S} \times \mathscr{S},\ (a, b) \in \mathscr{A} \implies (b, a) \in \mathscr{A}\ .$$

Sinon il est dit **orienté**.

Définissons la notion de sous-graphe d'un graphe.

Definition

Donnés deux graphes $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{G}' = (\mathscr{S}', \mathscr{A}')$, on dit que $\mathscr{G}'$ est un **sous-graphe** de $\mathscr{G}$ si $\mathscr{S}' \subset \mathscr{S}$ et $\mathscr{A}' \subset \mathscr{A}$.

Définissons la notion de sous-graphe d'un graphe.

Definition

Donnés deux graphes $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{G}' = (\mathscr{S}', \mathscr{A}')$, on dit que $\mathscr{G}'$ est un **sous-graphe** de $\mathscr{G}$ si $\mathscr{S}' \subset \mathscr{S}$ et $\mathscr{A}' \subset \mathscr{A}$.

On définit aussi les notions de successeurs et de prédécesseurs d'un sommet, et dans un graphe non orienté, la notion de sommets adjacents.

Définissons la notion de sous-graphe d'un graphe.

Definition

Donnés deux graphes $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{G}' = (\mathscr{S}', \mathscr{A}')$, on dit que $\mathscr{G}'$ est un **sous-graphe** de $\mathscr{G}$ si $\mathscr{S}' \subset \mathscr{S}$ et $\mathscr{A}' \subset \mathscr{A}$.

On définit aussi les notions de successeurs et de prédécesseurs d'un sommet, et dans un graphe non orienté, la notion de sommets adjacents.

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$:

– si $(a, b) \in \mathscr{A}$ est une arête, les sommets $a \in \mathscr{S}$ et $b \in \mathscr{S}$ sont appelés respectivement origine et l'extrémité de l'arête.

Définissons la notion de sous-graphe d'un graphe.

Definition

Donnés deux graphes $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{G}' = (\mathscr{S}', \mathscr{A}')$, on dit que $\mathscr{G}'$ est un **sous-graphe** de $\mathscr{G}$ si $\mathscr{S}' \subset \mathscr{S}$ et $\mathscr{A}' \subset \mathscr{A}$.

On définit aussi les notions de successeurs et de prédécesseurs d'un sommet, et dans un graphe non orienté, la notion de sommets adjacents.

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$:

– si $(a, b) \in \mathscr{A}$ est une arête, les sommets $a \in \mathscr{S}$ et $b \in \mathscr{S}$ sont appelés respectivement origine et l'extrémité de l'arête. Le sommet b est un **successeur** du sommet a, le sommet a est un **prédécesseur** du sommet b ;

Définissons la notion de sous-graphe d'un graphe.

Definition

Donnés deux graphes $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{G}' = (\mathscr{S}', \mathscr{A}')$, on dit que $\mathscr{G}'$ est un **sous-graphe** de $\mathscr{G}$ si $\mathscr{S}' \subset \mathscr{S}$ et $\mathscr{A}' \subset \mathscr{A}$.

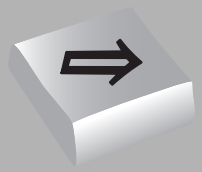
On définit aussi les notions de successeurs et de prédécesseurs d'un sommet, et dans un graphe non orienté, la notion de sommets adjacents.

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$:

– si $(a, b) \in \mathscr{A}$ est une arête, les sommets $a \in \mathscr{S}$ et $b \in \mathscr{S}$ sont appelés respectivement origine et l'extrémité de l'arête. Le sommet b est un **successeur** du sommet a, le sommet a est un **prédécesseur** du sommet b ;

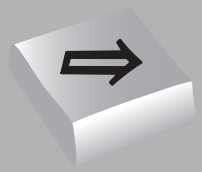
– dans un graphe non orienté, deux sommets $a \in \mathscr{S}$ et $b \in \mathscr{S}$ sont **adjacents** si $(a, b) \in \mathscr{A}$;



Dans la suite, pour un graphe non orienté, on ne donnera qu'une arête sur 2 parmi (a, b) et (b, a). Par exemple, ci-dessus, on s'autorisera à écrire abusivement :

$$\mathscr{A} = \Big\{(A, C); (B, C); (C, D); (A, B); (A, D); (B, D)\Big\}$$

et on dira que le nombre d'arêtes non orientées, noté $\#\mathscr{A}$, est 6.



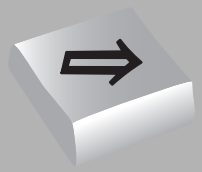
Dans la suite, pour un graphe non orienté, on ne donnera qu'une arête sur 2 parmi (a, b) et (b, a). Par exemple, ci-dessus, on s'autorisera à écrire abusivement :

$$\mathscr{A} = \Big\{(A, C); (B, C); (C, D); (A, B); (A, D); (B, D)\Big\}$$

et on dira que le nombre d'arêtes non orientées, noté $\#\mathscr{A}$, est 6.

- **Représentation graphique d'un graphe.**

On représente graphiquement un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ dans le plan, à l'aide d'un motif, point ou cercle, pour chaque sommet (avec pour label le nom du sommet), et d'un arc reliant les sommets a et b pour toute arête $(a, b) \in \mathscr{A}$.



Dans la suite, pour un graphe non orienté, on ne donnera qu'une arête sur 2 parmi (a, b) et (b, a). Par exemple, ci-dessus, on s'autorisera à écrire abusivement :

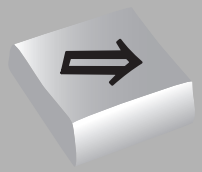
$$\mathscr{A} = \Big\{(A, C); (B, C); (C, D); (A, B); (A, D); (B, D)\Big\}$$

et on dira que le nombre d'arêtes non orientées, noté $\#\mathscr{A}$, est 6.

- **Représentation graphique d'un graphe.**

On représente graphiquement un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ dans le plan, à l'aide d'un motif, point ou cercle, pour chaque sommet (avec pour label le nom du sommet), et d'un arc reliant les sommets a et b pour toute arête $(a, b) \in \mathscr{A}$.

– Pour un graphe orienté, une flèche permet de distinguer l'arête (a, b) de l'arête (b, a).



Dans la suite, pour un graphe non orienté, on ne donnera qu'une arête sur 2 parmi (a, b) et (b, a). Par exemple, ci-dessus, on s'autorisera à écrire abusivement :

$$\mathscr{A} = \Big\{(A, C); (B, C); (C, D); (A, B); (A, D); (B, D)\Big\}$$

et on dira que le nombre d'arêtes non orientées, noté $\#\mathscr{A}$, est 6.

- **Représentation graphique d'un graphe.**

On représente graphiquement un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ dans le plan, à l'aide d'un motif, point ou cercle, pour chaque sommet (avec pour label le nom du sommet), et d'un arc reliant les sommets a et b pour toute arête $(a, b) \in \mathscr{A}$.

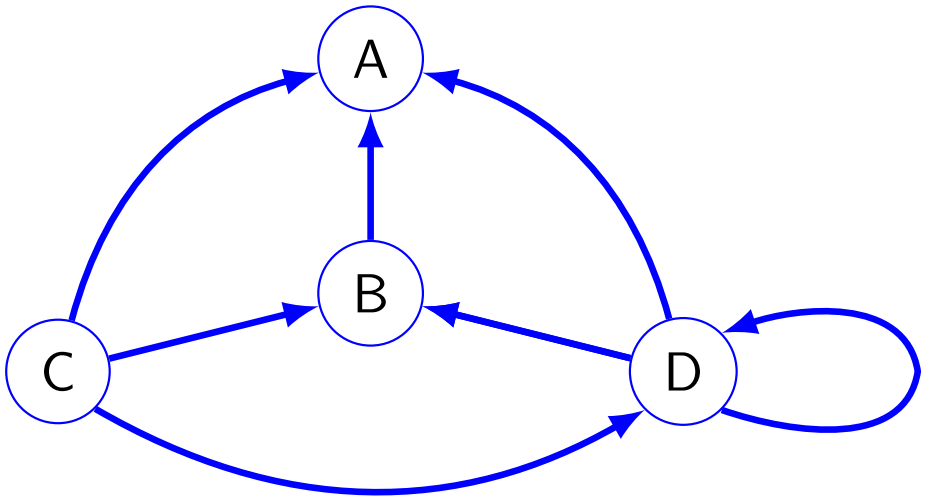
– Pour un graphe orienté, une flèche permet de distinguer l'arête (a, b) de l'arête (b, a).
– Pour un graphe non orienté, on ne distingue pas l'arête (a, b) de l'arête (b, a).

Exemple.

Exemple.

- Sur la figure de gauche on a représenté le graphe orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ avec : $\mathscr{S} = \{A, B, C, D\}$ et

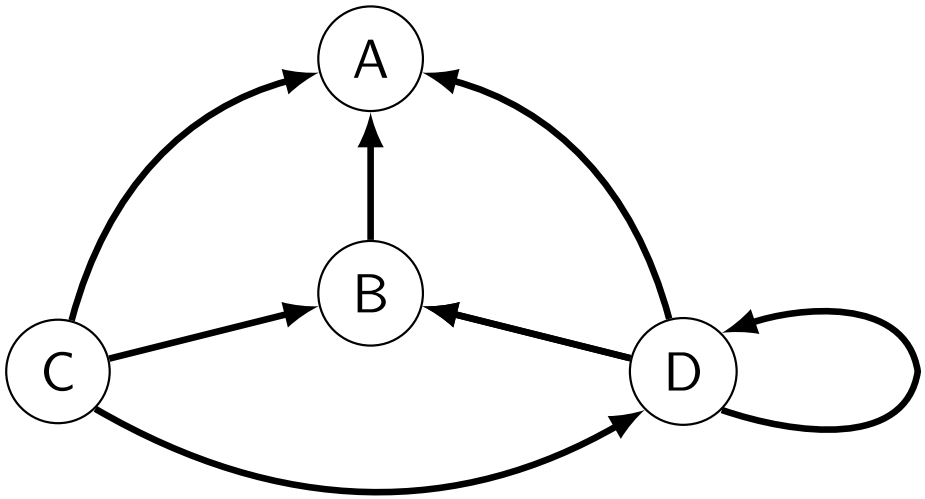
$$\mathscr{A} = \{(C, A); (C, B); (C, D); (B, A); (D, A); (D, B), (D, D)\} .$$

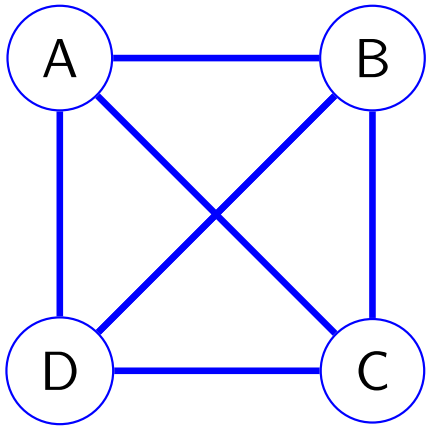


Exemple.

• Sur la figure de gauche on a représenté le graphe orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ avec : $\mathscr{S} = \{A, B, C, D\}$ et

$$\mathscr{A} = \{(C, A); (C, B); (C, D); (B, A); (D, A); (D, B), (D, D)\} .$$





• Sur la figure de droite, on a représenté le graphe non orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ avec : $\mathscr{S} = \{A, B, C, D\}$ et

$$\mathscr{A} = \{(A, C); (C, A); (B, C); (C, B); (D, C); (C, D); (B, A);$$
$$(A, B); (D, A); (A, D); (D, B); (B, D)\}$$

- **Relation binaire associée à un graphe**

Un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ permet de définir une relation binaire $R_{\mathscr{G}}$ sur l'ensemble de ses sommets, c'est-à-dire l'application :

$$R_{\mathscr{G}} : \begin{array}{ccc} \mathscr{S} \times \mathscr{S} & \longrightarrow & \{0,1\} \\ (a,b) & \longmapsto & aR_{\mathscr{G}}b \end{array} \quad \text{où} \quad aR_{\mathscr{G}}b = \begin{cases} 1 & \text{si } (a,b) \in \mathscr{A} \\ 0 & \text{sinon} \end{cases}$$

- **Relation binaire associée à un graphe**

Un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ permet de définir une relation binaire $R_{\mathscr{G}}$ sur l'ensemble de ses sommets, c'est-à-dire l'application :

$$R_{\mathscr{G}} : \begin{array}{ccc} \mathscr{S} \times \mathscr{S} & \longrightarrow & \{0,1\} \\ (a,b) & \longmapsto & aR_{\mathscr{G}}\,b \end{array} \quad \text{où} \quad aR_{\mathscr{G}}\,b = \begin{cases} 1 & \text{si } (a,b) \in \mathscr{A} \\ 0 & \text{sinon} \end{cases}$$

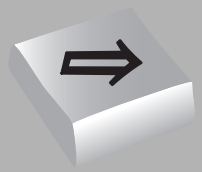
Ainsi un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ est non orienté si et seulement si $R_{\mathscr{G}}$ est symétrique, c'est-à-dire si et seulement si $\forall\,(a,b) \in \mathscr{S} \times \mathscr{S}$, $aR_{\mathscr{G}}\,b = bR_{\mathscr{G}}\,a$.

• Relation binaire associée à un graphe

Un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ permet de définir une relation binaire $R_{\mathscr{G}}$ sur l'ensemble de ses sommets, c'est-à-dire l'application :

$$R_{\mathscr{G}} : \begin{array}{ccc} \mathscr{S} \times \mathscr{S} & \longrightarrow & \{0,1\} \\ (a,b) & \longmapsto & aR_{\mathscr{G}}\,b \end{array} \quad \text{où} \quad aR_{\mathscr{G}}\,b = \begin{cases} 1 & \text{si } (a,b) \in \mathscr{A} \\ 0 & \text{sinon} \end{cases}$$

Ainsi un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ est non orienté si et seulement si $R_{\mathscr{G}}$ est symétrique, c'est-à-dire si et seulement si $\forall\,(a,b) \in \mathscr{S} \times \mathscr{S}$, $aR_{\mathscr{G}}\,b = bR_{\mathscr{G}}\,a$.



Un graphe n'est qu'une manière graphique de définir une relation binaire (c'est-à-dire une relation ayant deux opérandes) sur un ensemble fini (constitué des sommets), un graphe non orienté définissant une relation symétrique.
C'est la raison pour laquelle l'emploi des graphes est naturellement si prépondérant.

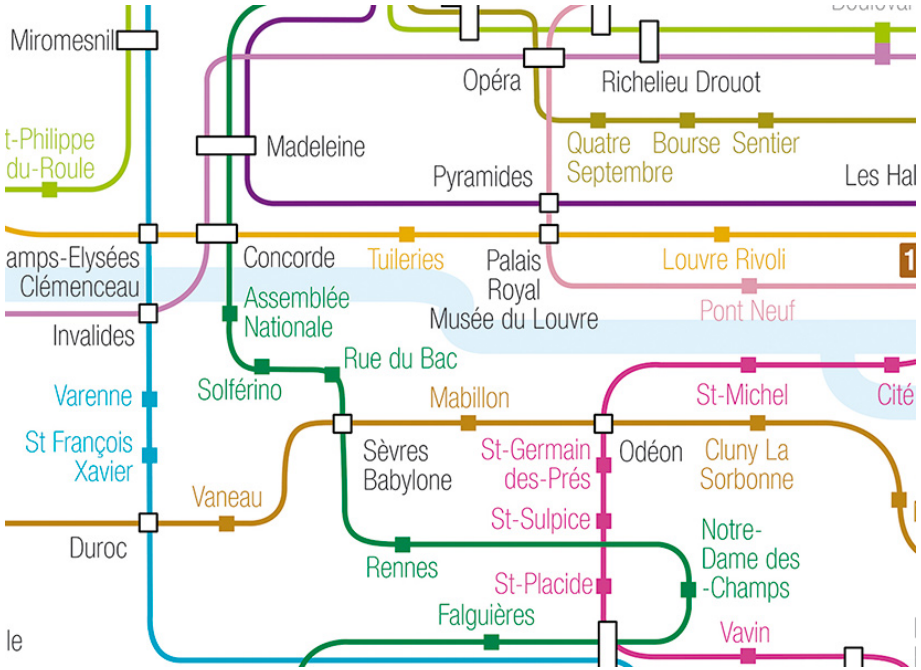
- **Exemples de situation bien modélisées par un graphe**

Énormément de situation peuvent se modéliser à l'aide d'un graphe.

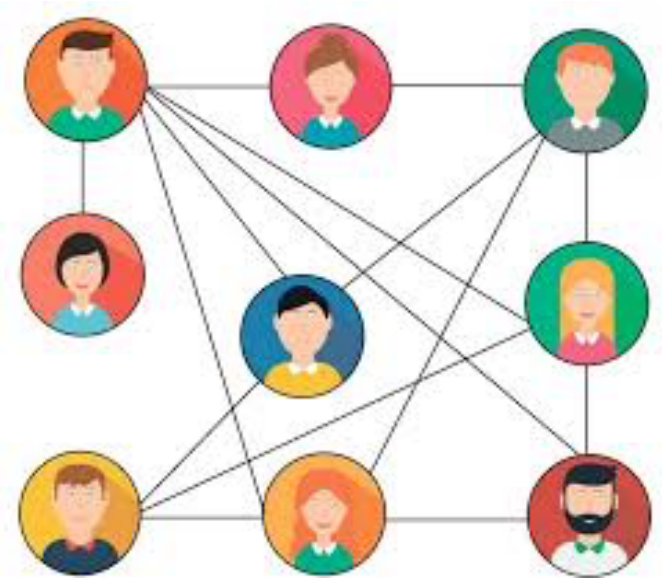
- **Exemples de situation bien modélisées par un graphe**

Énormément de situation peuvent se modéliser à l'aide d'un graphe.

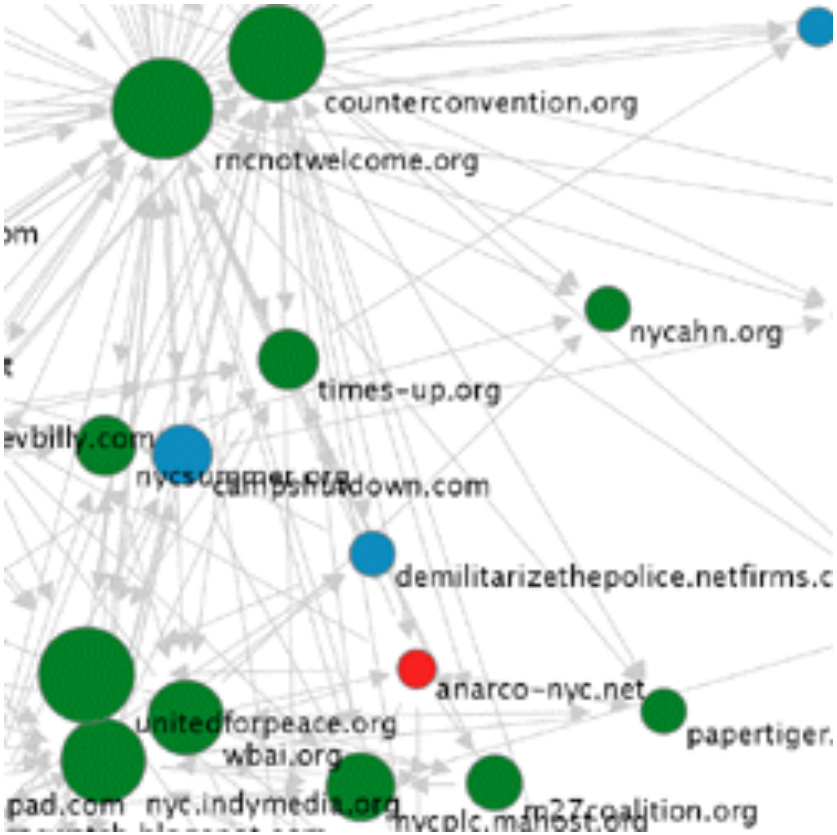
- Réseau de transport :



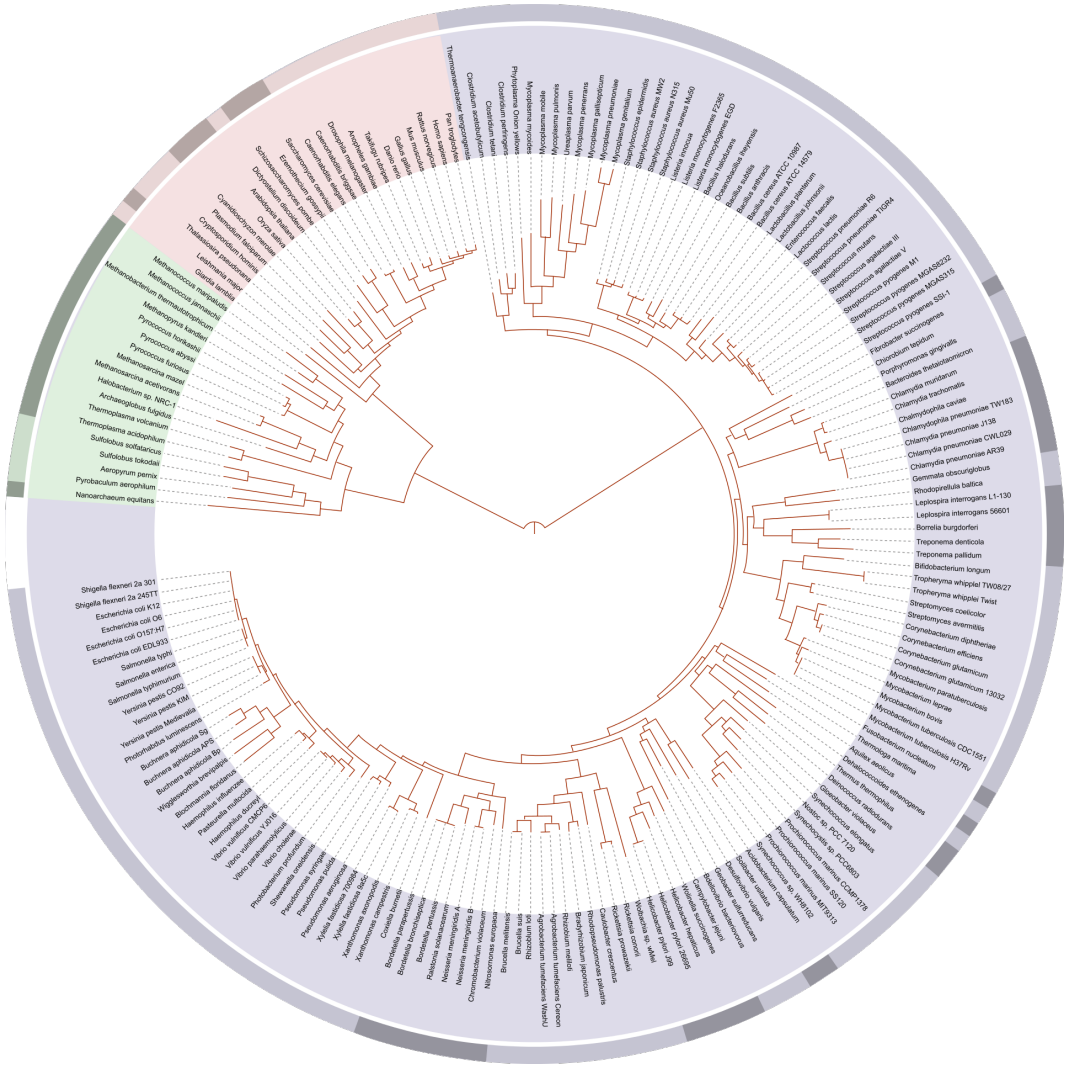
- Réseaux sociaux :



- Graphe du web :



- Arbre Phylogénétique du vivant :



- **Graphe non orienté sous-jacent**

Un graphe orienté contient plus d'information qu'un graphe non orienté, et à tout graphe on peut lui associer un graphe non orienté, dit sous-jacent, ayant même ensemble de sommets, et obtenu en symétrisant sa relation associée, c'est-à-dire en supprimant toutes les flèches des arcs dans sa représentation graphique.

- **Graphe non orienté sous-jacent**

Un graphe orienté contient plus d'information qu'un graphe non orienté, et à tout graphe on peut lui associer un graphe non orienté, dit sous-jacent, ayant même ensemble de sommets, et obtenu en symétrisant sa relation associée, c'est-à-dire en supprimant toutes les flèches des arcs dans sa représentation graphique.

Definition

Donné un **graphe (fini)** $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, son **graphe non orienté sous-jacent** est le graphe $\overline{\mathscr{G}} = (\overline{\mathscr{S}}, \overline{\mathscr{A}})$ défini par :

$$\overline{\mathscr{S}} = \mathscr{S} \quad \text{et} \quad \overline{\mathscr{A}} = \left\{ (a,b) \in \mathscr{S} \times \mathscr{S} \mid (a,b) \in \mathscr{A} \text{ ou } (b,a) \in \mathscr{A} \right\}.$$

- **Graphe non orienté sous-jacent**

Un graphe orienté contient plus d'information qu'un graphe non orienté, et à tout graphe on peut lui associer un graphe non orienté, dit sous-jacent, ayant même ensemble de sommets, et obtenu en symétrisant sa relation associée, c'est-à-dire en supprimant toutes les flèches des arcs dans sa représentation graphique.

Definition

Donné un **graphe (fini)** $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, son **graphe non orienté sous-jacent** est le graphe $\overline{\mathscr{G}} = (\overline{\mathscr{S}}, \overline{\mathscr{A}})$ défini par :

$$\overline{\mathscr{S}} = \mathscr{S} \quad \text{et} \quad \overline{\mathscr{A}} = \left\{ (a,b) \in \mathscr{S} \times \mathscr{S} \mid (a,b) \in \mathscr{A} \text{ ou } (b,a) \in \mathscr{A} \right\}.$$

Évidemment, pour un graphe non orienté $\mathscr{G}$, on a $\overline{\mathscr{G}} = \mathscr{G}$, et cette définition ne présente d'intérêt que pour un graphe orienté.

- **Chemin**

La notion de chemin entre sommets est une notion importante dans les graphes ; elle correspond à l'idée intuitive de chemin reliant deux sommets par une succession d'arêtes.

- **Chemin**

La notion de chemin entre sommets est une notion importante dans les graphes ; elle correspond à l'idée intuitive de chemin reliant deux sommets par une succession d'arêtes.

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, donnés deux sommets $a \in \mathscr{S}$ et $b \in \mathscr{S}$, un **chemin de a à b** est une suite finie de sommets

$$s_0, s_1, \ldots, s_i, s_{i+1}, \ldots, s_{n-1}, s_n$$

vérifiant : $s_0 = a$, $s_n = b$ et

$$\forall\, i \in [\![0, n-1]\!],\ (s_i, s_{i+1}) \in \mathscr{A}\ .$$

L'entier n est la **longueur** du chemin.

- **Chemin**

La notion de chemin entre sommets est une notion importante dans les graphes ; elle correspond à l'idée intuitive de chemin reliant deux sommets par une succession d'arêtes.

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, donnés deux sommets $a \in \mathscr{S}$ et $b \in \mathscr{S}$, un **chemin de a à b** est une suite finie de sommets

$$s_0, s_1, \ldots, s_i, s_{i+1}, \ldots, s_{n-1}, s_n$$

vérifiant : $s_0 = a$, $s_n = b$ et

$$\forall\, i \in [\![0, n-1]\!],\ (s_i, s_{i+1}) \in \mathscr{A}\ .$$

L'entier n est la **longueur** du chemin.

La distance entre deux sommets reliés par un chemin est la longueur minimale d'un chemin les reliant.

- **Cycle**

Un chemin reliant deux mêmes sommets est appelé un cycle.

Definition

Un **cycle** issu de a est un chemin d'un sommet $a \in \mathscr{S}$ à lui-même, et de longueur $\geqslant 1$.

- **Cycle**

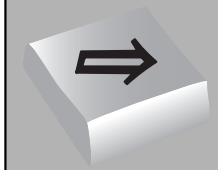
Un chemin reliant deux mêmes sommets est appelé un cycle.

Definition

Un **cycle** issu de a est un chemin d'un sommet $a \in \mathscr{S}$ à lui-même, et de longueur $\geqslant 1$.

- **Composantes connexes**

Pour un graphe non orienté, être relié par un chemin est une relation d'équivalence sur les sommets.



Une relation binaire R sur $\mathscr{S}$ est une **relation d'équivalence** si elle est :

- Réflexive $\quad \forall\, s \in \mathscr{S},\ sRs.$
- Symétrique $\quad \forall\, (s, s') \in \mathscr{S}^2,\ sRs' \implies s'Rs.$
- Transitive $\quad \forall\, (s, s', s'') \in \mathscr{S}^3,\ sRs' \text{ et } s'Rs'' \implies sRs''$

Propriété

Dans un graphe $\mathscr{G}$ non orienté, la relation " être relié par un chemin", est une relation d'équivalence sur l'ensemble $\mathscr{S}$ de ses sommets .

Propriété

Dans un graphe $\mathscr{G}$ non orienté, la relation " être relié par un chemin", est une relation d'équivalence sur l'ensemble $\mathscr{S}$ de ses sommets .

Preuve. Notons la relation R ; elle est bien définie, par aRb si il existe un chemin dans $\mathscr{G}$ de a à b. Il s'agit de montrer qu'elle est réflexive, symétrique et transitive.

Propriété

Dans un graphe $\mathscr{G}$ non orienté, la relation " être relié par un chemin", est une relation d'équivalence sur l'ensemble $\mathscr{S}$ de ses sommets .

Preuve. Notons la relation R ; elle est bien définie, par aRb si il existe un chemin dans $\mathscr{G}$ de a à b. Il s'agit de montrer qu'elle est réflexive, symétrique et transitive.

Réflexivité. Pour un sommet $a \in \mathscr{S}$, la suite a de longueur 0 est un chemin de a à a ; ainsi aRa.

Propriété

Dans un graphe $\mathscr{G}$ non orienté, la relation " être relié par un chemin", est une relation d'équivalence sur l'ensemble $\mathscr{S}$ de ses sommets .

Preuve. Notons la relation R ; elle est bien définie, par aRb si il existe un chemin dans $\mathscr{G}$ de a à b. Il s'agit de montrer qu'elle est réflexive, symétrique et transitive.

Réflexivité. Pour un sommet $a \in \mathscr{S}$, la suite a de longueur 0 est un chemin de a à a ; ainsi aRa.

Symétrie. Soit a, b deux sommets. Si aRb, il existe un chemin $(s_k)_{0\leqslant k\leqslant n} = s_0, \ldots, s_n$ de a à b. Le graphe étant non orienté : $(s_i, s_{i+1}) \in \mathscr{A} \iff (s_{i+1}, s_i) \in \mathscr{A}$. Ainsi la suite de sommets $(s'_k)_{0\leqslant k\leqslant n}$ définie par $s'_k = s_{n-k}$ est un chemin de b à a. Donc bRa.

Propriété

Dans un graphe $\mathscr{G}$ non orienté, la relation " être relié par un chemin", est une relation d'équivalence sur l'ensemble $\mathscr{S}$ de ses sommets .

Preuve. Notons la relation R ; elle est bien définie, par aRb si il existe un chemin dans $\mathscr{G}$ de a à b. Il s'agit de montrer qu'elle est réflexive, symétrique et transitive.

Réflexivité. Pour un sommet $a \in \mathscr{S}$, la suite a de longueur 0 est un chemin de a à a ; ainsi aRa.

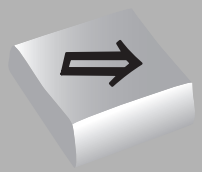
Symétrie. Soit a, b deux sommets. Si aRb, il existe un chemin $(s_k)_{0\leqslant k\leqslant n} = s_0, \ldots, s_n$ de a à b. Le graphe étant non orienté : $(s_i, s_{i+1}) \in \mathscr{A} \iff (s_{i+1}, s_i) \in \mathscr{A}$. Ainsi la suite de sommets $(s'_k)_{0\leqslant k\leqslant n}$ définie par $s'_k = s_{n-k}$ est un chemin de b à a. Donc bRa.

Transitivité. Soit a, b, c trois sommets tels que aRb et bRc, et $(s_k)_{0\leqslant k\leqslant n}$, respectivement $(s'_k)_{0\leqslant k\leqslant m}$, un chemin de a à b, respectivement de b à c. Alors nécessairement $s_0 = a$, $s_n = s'_0 = b$ et $s'_m = c$. Ainsi, la suite de sommets :

$$a = s_0, s_1, \ldots, s_n = s'_0, s'_1, \ldots, s'_m = c$$

est un chemin de a à c (de longueur $n + m$). Donc aRc.

Donnée une relation d'équivalence sur un ensemble $\mathscr{S}$, l'ensemble se partitionne en classes d'équivalences.



Soit S un ensemble et R une relation d'équivalence ; la **classe d'équivalence** de $s \in S$ est $\left[s\right] = \left\{s' \in S \mid sRs'\right\}$.

Les classes d'équivalence forment une partition de S : il existe une famille $(s_i)_{i\in I}$ d'éléments de S tels que :

$$S = \bigcup_{i\in I}\left[s_i\right] \quad \text{et} \quad \forall\,(i,j) \in I^2,\ i \neq j \implies \left[s_i\right] \cap \left[s_j\right] = \varnothing\ .$$

- **Composantes connexes**

- **Composantes connexes**

Donné un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{S}' \subset \mathscr{S}$ un sous-ensemble de sommets, le **sous-graphe** de $\mathscr{G}$ restreint aux sommets $\mathscr{S}'$ est le graphe $(\mathscr{S}', \mathscr{A}')$ avec $\mathscr{A}' = \mathscr{A} \cap (\mathscr{S} \times \mathscr{S})$, c'est-à-dire que c'est le graphe obtenu de $\mathscr{G}$ en ne conservant que les sommets de $\mathscr{S}'$ et les arêtes reliant les sommets de $\mathscr{S}'$.

- **Composantes connexes**

Donné un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{S}' \subset \mathscr{S}$ un sous-ensemble de sommets, le **sous-graphe** de $\mathscr{G}$ restreint aux sommets $\mathscr{S}'$ est le graphe $(\mathscr{S}', \mathscr{A}')$ avec $\mathscr{A}' = \mathscr{A} \cap (\mathscr{S} \times \mathscr{S})$, c'est-à-dire que c'est le graphe obtenu de $\mathscr{G}$ en ne conservant que les sommets de $\mathscr{S}'$ et les arêtes reliant les sommets de $\mathscr{S}'$.

Definition

- **Pour un graphe non orienté** $\mathscr{G}$, ses **composantes connexes**, sont les sous-graphes de $\mathscr{G}$ restreints aux classes d'équivalences des sommets pour la relation être relié par un chemin.

- **Composantes connexes**

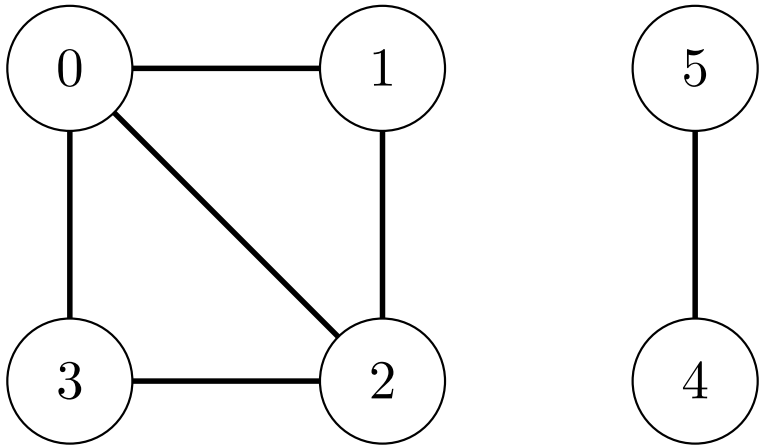
Donné un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et $\mathscr{S}' \subset \mathscr{S}$ un sous-ensemble de sommets, le **sous-graphe** de $\mathscr{G}$ restreint aux sommets $\mathscr{S}'$ est le graphe $(\mathscr{S}', \mathscr{A}')$ avec $\mathscr{A}' = \mathscr{A} \cap (\mathscr{S} \times \mathscr{S})$, c'est-à-dire que c'est le graphe obtenu de $\mathscr{G}$ en ne conservant que les sommets de $\mathscr{S}'$ et les arêtes reliant les sommets de $\mathscr{S}'$.

Definition

- **Pour un graphe non orienté** $\mathscr{G}$, ses **composantes connexes**, sont les sous-graphes de $\mathscr{G}$ restreints aux classes d'équivalences des sommets pour la relation être relié par un chemin.
- **Pour un graphe orienté** $\mathscr{G}$, ses **composantes connexes**, sont les sous-graphes de $\mathscr{G}$ restreints aux classes d'équivalences, de son sous-graphe non orienté sous-jacent $\overline{\mathscr{G}}$, pour la relation être relié par un chemin.

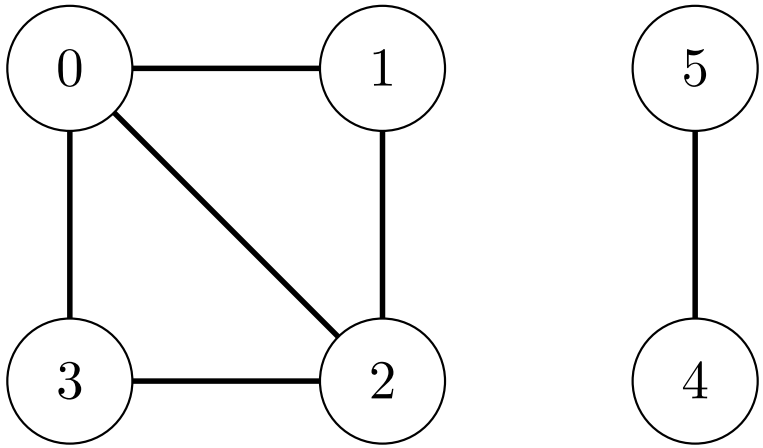
Les composantes connexes sont intuitivement " les morceaux " du graphe. Par exemple, le graphe non orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ défini par :

$$\mathscr{S} = \{0, 1, 2, 3, 4, 5\} \quad \mathscr{A} = \{(0,1); (0,2); (0,3); (1,2); (2,3); (4,5)\}$$



Les composantes connexes sont intuitivement " les morceaux " du graphe. Par exemple, le graphe non orienté $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ défini par :

$$\mathscr{S} = \{0, 1, 2, 3, 4, 5\} \quad \mathscr{A} = \{(0,1); (0,2); (0,3); (1,2); (2,3); (4,5)\}$$



a deux composantes connexes, $\mathscr{G}_1$ (graphe de gauche) et $\mathscr{G}_2$ (graphe de droite) :

$$\mathscr{G}_1 = (\mathscr{S}_1, \mathscr{A}_1) \quad : \quad \mathscr{S}_1 = \{0, 1, 2, 3\} \quad \mathscr{A}_1 = \{(0,1); (0,2); (0,3); (1,2); (2,3)\}$$

$$\mathscr{G}_2 = (\mathscr{S}_2, \mathscr{A}_2) \quad : \quad \mathscr{S}_2 = \{4, 5\} \quad \mathscr{A}_2 = \{(4,5)\}$$

- **Graphe connexe**

La notion de composantes connexes permet de définir la connexité d'un graphe, intuitivement lorsqu'il est constitué d'un seul morceau.

- **Graphe connexe**

La notion de composantes connexes permet de définir la connexité d'un graphe, intuitivement lorsqu'il est constitué d'un seul morceau.

Definition

- **Un graphe non orienté** est dit **connexe** lorsqu'il n'a qu'une seule composante connexe.

- **Graphe connexe**

La notion de composantes connexes permet de définir la connexité d'un graphe, intuitivement lorsqu'il est constitué d'un seul morceau.

Definition

- **Un graphe non orienté** est dit **connexe** lorsqu'il n'a qu'une seule composante connexe.
- **Un graphe orienté** est dit **connexe** lorsque son graphe non orienté sous-jacent est connexe.

- **Graphe connexe**

La notion de composantes connexes permet de définir la connexité d'un graphe, intuitivement lorsqu'il est constitué d'un seul morceau.

Definition

- **Un graphe non orienté** est dit **connexe** lorsqu'il n'a qu'une seule composante connexe.
- **Un graphe orienté** est dit **connexe** lorsque son graphe non orienté sous-jacent est connexe.

Par exemple, le graphe non orienté précédent est non connexe (il a deux composantes connexes),

- **Graphe connexe**

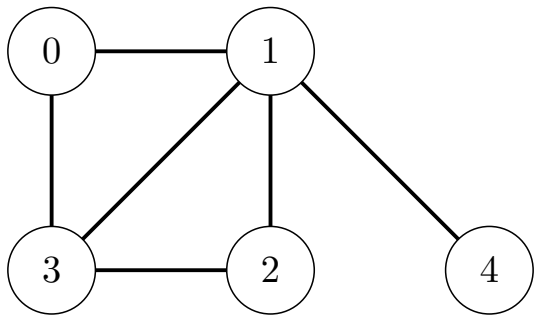
La notion de composantes connexes permet de définir la connexité d'un graphe, intuitivement lorsqu'il est constitué d'un seul morceau.

Definition

- **Un graphe non orienté** est dit **connexe** lorsqu'il n'a qu'une seule composante connexe.
- **Un graphe orienté** est dit **connexe** lorsque son graphe non orienté sous-jacent est connexe.

Par exemple, le graphe non orienté précédent est non connexe (il a deux composantes connexes), alors que le graphe non orienté suivant $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ est connexe.

$$\mathscr{S} = \{0, 1, 2, 3, 4\} \quad \mathscr{A} = \{(0,1); (0,3); (1,2); (1,3); (2,3); (1,4)\}$$



Pour un graphe non orienté, on peut aussi donner la définition alternative.

Propriété

Un graphe non orienté est connexe lorsque chaque couple de ses sommets est relié par un chemin.

Pour un graphe non orienté, on peut aussi donner la définition alternative.

Propriété

Un graphe non orienté est connexe lorsque chaque couple de ses sommets est relié par un chemin.

Preuve. Le graphe est connexe si et seulement si la relation " être relié par un chemin " n'a qu'une seule classe d'équivalence, si et seulement si chaque couple de sommet est en relation, c'est-à-dire est relié par un chemin. □

- **Matrice d'adjacence d'un graphe**

Donné un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, on peut le représenter à l'aide d'une matrice d'adjacence, après avoir numéroté ses sommets de 1 jusqu'à $\mathrm{Card}(\mathscr{S})$.

- **Matrice d'adjacence d'un graphe**

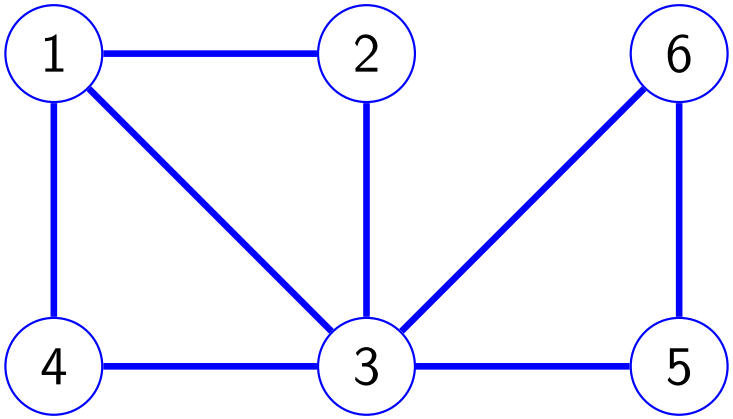
Donné un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$, on peut le représenter à l'aide d'une matrice d'adjacence, après avoir numéroté ses sommets de 1 jusqu'à $\mathrm{Card}(\mathscr{S})$.

Definition

Soit un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ ayant n sommets. Donnée une bijection $s : [\![1, n]\!] \longrightarrow \mathscr{S}$, la **matrice d'adjacence** du graphe est la matrice carrée $A \in \mathscr{M}_n(\mathbb{R})$ constituée de 0 et de 1, et définie par :

$$\forall\, (i,j) \in [\![1,n]\!]^2, \quad A_{i,j} = \begin{cases} 1 & \text{si } (s(i), s(j)) \in \mathscr{A} \\ 0 & \textit{sinon} \end{cases}$$

Exemple. Le graphe non orienté :



a pour matrice d'adjacence :

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

- **Propriétés**

Bien sûr la matrice d'adjacence, n'est pas unique, elle dépend de la " numérotation " des sommets. Mais changer cette numérotation ne fait qu'appliquer une suite d'échanges simultanés des lignes et colonnes $L_i \leftrightarrow L_j$ et $C_i \leftrightarrow C_j$.

- **Propriétés**

Bien sûr la matrice d'adjacence, n'est pas unique, elle dépend de la " numérotation " des sommets. Mais changer cette numérotation ne fait qu'appliquer une suite d'échanges simultanés des lignes et colonnes $L_i \leftrightarrow L_j$ et $C_i \leftrightarrow C_j$.

Propriété

Une matrice d'adjacence d'un graphe $\mathscr{G}$ est symétrique si et seulement si le graphe $\mathscr{G}$ est non orienté.

Preuve. Immédiate par définition. □

D'autres propriétés plus remarquables relient le graphe à sa matrice d'adjacence.

Propriété

Soit $\mathscr{G}$ un graphe ayant n sommets et $A \in \mathscr{M}_n(\mathbb{R})$ une matrice d'adjacence de $\mathscr{G}$. Pour tout $k \in \mathbb{N}^*$ et tout $i, j \in [\![1, n]\!]$, l'élément $A^k_{i,j}$ ligne i colonne j de la matrice A^k est égal au nombre de chemins de longueur k dans $\mathscr{G}$ allant du sommet numéroté i au sommet numéroté j.

D'autres propriétés plus remarquables relient le graphe à sa matrice d'adjacence.

Propriété

Soit $\mathscr{G}$ un graphe ayant n sommets et $A \in \mathscr{M}_n(\mathbb{R})$ une matrice d'adjacence de $\mathscr{G}$. Pour tout $k \in \mathbb{N}^*$ et tout $i, j \in [\![1, n]\!]$, l'élément $A^k_{i,j}$ ligne i colonne j de la matrice A^k est égal au nombre de chemins de longueur k dans $\mathscr{G}$ allant du sommet numéroté i au sommet numéroté j.

Preuve. On démontre la propriété par récurrence sur k. Pour $i \in [\![1, n]\!]$ notons s_k le sommet numéroté k.

D'autres propriétés plus remarquables relient le graphe à sa matrice d'adjacence.

Propriété

Soit $\mathscr{G}$ un graphe ayant n sommets et $A \in \mathscr{M}_n(\mathbb{R})$ une matrice d'adjacence de $\mathscr{G}$. Pour tout $k \in \mathbb{N}^*$ et tout $i, j \in [\![1, n]\!]$, l'élément $A^k_{i,j}$ ligne i colonne j de la matrice A^k est égal au nombre de chemins de longueur k dans $\mathscr{G}$ allant du sommet numéroté i au sommet numéroté j.

Preuve. On démontre la propriété par récurrence sur k. Pour $i \in [\![1, n]\!]$ notons s_k le sommet numéroté k.

Initialisation. Si $k = 0$. Soit $(i, j) \in [\![1, n]\!]^2$. La matrice A^0 est l'identité, et il existe un chemin de longueur 0 de s_i à s_j si et seulement si $i = j$. La proposition de récurrence est vraie.

Hérédité. Supposons la proposition vraie au rang $k \in \mathbb{N}^*$. Soit $(i,j) \in [\![1,n]\!]^2$, on a :

$$A_{i,j}^{k+1} = \sum_{t=1}^{n} A_{i,t}^{k} \times A_{t,j}$$

où $A_{t,j}$ vaut 1 si et seulement si (s_t, s_j) est une arête, et 0 sinon.

Hérédité. Supposons la proposition vraie au rang $k \in \mathbb{N}^*$. Soit $(i,j) \in [\![1,n]\!]^2$, on a :

$$A_{i,j}^{k+1} = \sum_{t=1}^{n} A_{i,t}^{k} \times A_{t,j}$$

où $A_{t,j}$ vaut 1 si et seulement si (s_t, s_j) est une arête, et 0 sinon. Par hypothèse de récurrence $A_{i,t}^k$ est le nombre de chemins de longueur k de s_i à s_t.

Hérédité. Supposons la proposition vraie au rang $k \in \mathbb{N}^*$. Soit $(i,j) \in [\![1,n]\!]^2$, on a :

$$A_{i,j}^{k+1} = \sum_{t=1}^{n} A_{i,t}^{k} \times A_{t,j}$$

où $A_{t,j}$ vaut 1 si et seulement si (s_t, s_j) est une arête, et 0 sinon. Par hypothèse de récurrence $A_{i,t}^{k}$ est le nombre de chemins de longueur k de s_i à s_t.

Or pour chaque sommet $s_t \in \mathscr{S}$ tel que (s_t, s_j) soit une arête, chaque chemin de s_i à s_t de longueur k donne un chemin différent de s_i à s_j de longueur $k+1$.

Hérédité. Supposons la proposition vraie au rang $k \in \mathbb{N}^*$. Soit $(i,j) \in [\![1,n]\!]^2$, on a :

$$A_{i,j}^{k+1} = \sum_{t=1}^{n} A_{i,t}^{k} \times A_{t,j}$$

où $A_{t,j}$ vaut 1 si et seulement si (s_t, s_j) est une arête, et 0 sinon. Par hypothèse de récurrence $A_{i,t}^{k}$ est le nombre de chemins de longueur k de s_i à s_t.

Or pour chaque sommet $s_t \in \mathscr{S}$ tel que (s_t, s_j) soit une arête, chaque chemin de s_i à s_t de longueur k donne un chemin différent de s_i à s_j de longueur $k+1$.

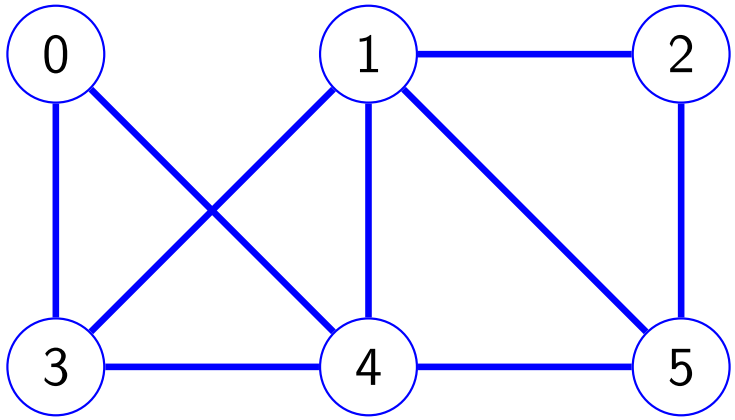
Ainsi $A_{i,j}^{k+1}$ est égal au nombre de chemins de longueur $k+1$ allant de s_i à s_j. La proposition de récurrence demeure vraie au rang $k+1$. □

• Implémentation d'un graphe par une liste d'arêtes

La façon la plus simple de représenter un graphe consiste à représenter les sommets par des entiers, débutant en 0, et de se donner le nombre n de sommets et la liste des arêtes sous forme d'une liste de couples d'entiers-sommets.

• Implémentation d'un graphe par une liste d'arêtes

La façon la plus simple de représenter un graphe consiste à représenter les sommets par des entiers, débutant en 0, et de se donner le nombre n de sommets et la liste des arêtes sous forme d'une liste de couples d'entiers-sommets. Par exemple pour le graphe non orienté :



on peut le représenter en déclarant une liste dont le premier élément est le nombre de sommets, et le second la liste des arêtes, données comme couples de sommets :

```
Graphe = [ 6, [(0,3), (0,4), (1,3), (1,4), (3,4),
               (1,2), (1,5), (2,5), (4,5)] ]
```

(Pour un graphe non orienté on peut ne stocker qu'une arête sur deux).

Cette approche n'est cependant pas très efficace pour implanter des algorithmes sur les graphes.

Cette approche n'est cependant pas très efficace pour implanter des algorithmes sur les graphes.

• **Implémentation d'un graphe par une liste d'adjacence** Une autre approche stocke le nombre de sommets et la liste des sommets adjacents (respectivement successeurs) pour chaque sommet d'un graphe non orienté (respectivement orienté) ;

Cette approche n'est cependant pas très efficace pour implanter des algorithmes sur les graphes.

• **Implémentation d'un graphe par une liste d'adjacence** Une autre approche stocke le nombre de sommets et la liste des sommets adjacents (respectivement successeurs) pour chaque sommet d'un graphe non orienté (respectivement orienté) ; si la liste est `L`, `L[i]` contiendra la liste des sommets adjacents à `i`.

Cette approche n'est cependant pas très efficace pour implanter des algorithmes sur les graphes.

• **Implémentation d'un graphe par une liste d'adjacence** Une autre approche stocke le nombre de sommets et la liste des sommets adjacents (respectivement successeurs) pour chaque sommet d'un graphe non orienté (respectivement orienté) ; si la liste est `L`, `L[i]` contiendra la liste des sommets adjacents à `i`. Pour le graphe non orienté ci-dessus, par exemple, on peut déclarer :

```
Graphe = [ 6, [ [3, 4],
                [2, 3, 4, 5],
                [1, 5],
                [0, 1, 4],
                [0, 1, 3, 5],
                [1, 2, 4]] ]
```

Cette approche n'est cependant pas très efficace pour implanter des algorithmes sur les graphes.

• **Implémentation d'un graphe par une liste d'adjacence** Une autre approche stocke le nombre de sommets et la liste des sommets adjacents (respectivement successeurs) pour chaque sommet d'un graphe non orienté (respectivement orienté) ; si la liste est `L`, `L[i]` contiendra la liste des sommets adjacents à `i`. Pour le graphe non orienté ci-dessus, par exemple, on peut déclarer :

```
Graphe = [ 6, [ [3, 4],
                [2, 3, 4, 5],
                [1, 5],
                [0, 1, 4],
                [0, 1, 3, 5],
                [1, 2, 4]] ]
```

La liste des sommets adjacents (successeurs) s'obtient en $O(1)$; c'est l'avantage de cette implantation.

Cette approche est efficace pour des graphes ayant peu d'arêtes.

- **Implémentation d'un graphe par une matrice d'adjacence**

Pour l'implémentation d'un graphe, un usage répandu, notamment lorsque le graphe comporte beaucoup d'arêtes, est d'utiliser une matrice d'adjacence.

- **Implémentation d'un graphe par une matrice d'adjacence**

Pour l'implémentation d'un graphe, un usage répandu, notamment lorsque le graphe comporte beaucoup d'arêtes, est d'utiliser une matrice d'adjacence. Il faut alors identifier ses sommets avec les entiers 0, 1, 2, etc. qui donneront les indices des lignes et colonnes dans la matrice.

- **Implémentation d'un graphe par une matrice d'adjacence**

Pour l'implémentation d'un graphe, un usage répandu, notamment lorsque le graphe comporte beaucoup d'arêtes, est d'utiliser une matrice d'adjacence. Il faut alors identifier ses sommets avec les entiers 0, 1, 2, etc. qui donneront les indices des lignes et colonnes dans la matrice. Une liste pourra permettre de stocker aux mêmes indices les identifiants des sommets, si nécessaire.

- **Implémentation d'un graphe par une matrice d'adjacence**

Pour l'implémentation d'un graphe, un usage répandu, notamment lorsque le graphe comporte beaucoup d'arêtes, est d'utiliser une matrice d'adjacence. Il faut alors identifier ses sommets avec les entiers 0, 1, 2, etc. qui donneront les indices des lignes et colonnes dans la matrice. Une liste pourra permettre de stocker aux mêmes indices les identifiants des sommets, si nécessaire.

```
from numpy import array

Graphe = array([[0,0,0,1,1,0],
                [0,0,1,1,1,1],
                [0,1,0,0,0,1],
                [1,1,0,0,1,0],
                [1,1,0,1,0,1],
                [0,1,1,0,1,0]])
```

- **Graphe valué**

Un graphe est valué lorsque chaque arête est munie d'une valeur, appelée son poids.

- **Graphe valué**

Un graphe est valué lorsque chaque arête est munie d'une valeur, appelée son poids.

Definition

Un **graphe valué** est la donnée d'un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ et d'une application $p : \mathscr{A} \longrightarrow \mathbb{R}$ appelée valuation.

Pour chaque arête $(s, s') \in \mathscr{A}$, le nombre $p(s, s')$ est appelé son poids.

Une valuation p est dite **strictement positive** si $\forall\, (s, s') \in \mathscr{A}$, $p(s, s') > 0$.

La valuation d'un graphe s'étend en une application : $\mathscr{S} \times \mathscr{S} \longrightarrow \overline{\mathbb{R}}$ en posant $p(s, s') = +\infty$ lorsque $(s, s') \notin \mathscr{A}$.

• Matrice de valuation

On peut décrire un graphe valué à l'aide de sa matrice de valuation.

Definition

Donnés un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ ayant $n = \#\mathscr{S}$ sommets, une valuation p du graphe et une bijection $s : [\![1, n]\!] \longrightarrow \mathscr{S}$, la **matrice de valuation** est la matrice $M \in \mathscr{M}_n(\mathbb{R})$ définie par :

$$\forall\, (i,j) \in [\![1,n]\!]^2,\ M_{i,j} = \begin{cases} p(s(i), s(j)) & \text{si } (s(i), s(j)) \in \mathscr{A} \\ \infty & \text{sinon} \end{cases}$$

- **Matrice de valuation**

On peut décrire un graphe valué à l'aide de sa matrice de valuation.

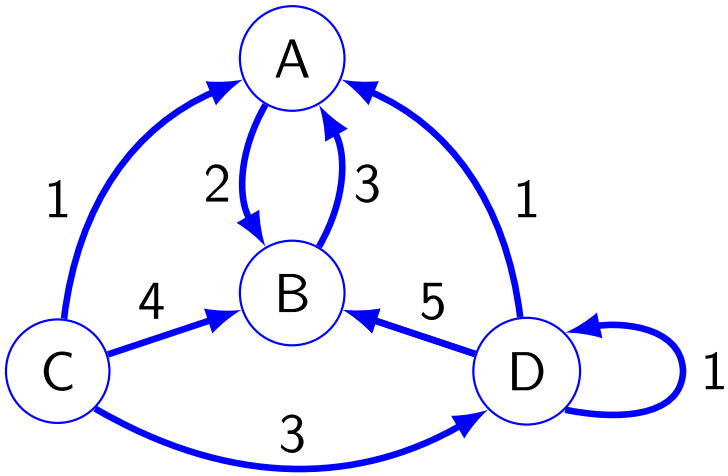
Definition

Donnés un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ ayant $n = \#\mathscr{S}$ sommets, une valuation p du graphe et une bijection $s : [\![1, n]\!] \longrightarrow \mathscr{S}$, la **matrice de valuation** est la matrice $M \in \mathscr{M}_n(\mathbb{R})$ définie par :

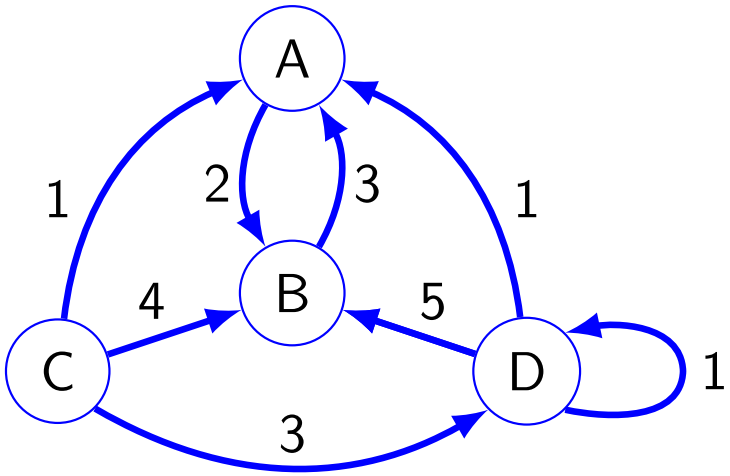
$$\forall\, (i,j) \in [\![1,n]\!]^2,\ M_{i,j} = \begin{cases} p(s(i), s(j)) & \text{si } (s(i), s(j)) \in \mathscr{A} \\ \infty & \text{sinon} \end{cases}$$

C'est une adaptation de la notion de matrice d'adjacence aux graphes valués, en changeant 1 par le poids d'une arête et 0 par $+\infty$.

- **Exemple**. Un graphe valué



- **Exemple**. Un graphe valué et sa matrice de valuation :



	A	B	C	D
A	∞	2	∞	∞
B	3	∞	∞	∞
C	1	4	∞	3
D	1	5	∞	1

- **Longueur d'un chemin. Chemin géodésique**

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ à valuation strictement positive, la **longueur d'un chemin** $\gamma = (s_0, s_1, \ldots, s_n)$ est le réel :

$$\ell(\gamma) = \sum_{k=0}^{n-1} p(s_k, s_{k+1}) \ .$$

- **Longueur d'un chemin. Chemin géodésique**

Definition

Dans un graphe $\mathscr{G} = (\mathscr{S}, \mathscr{A})$ à valuation strictement positive, la **longueur d'un chemin** $\gamma = (s_0, s_1, \ldots, s_n)$ est le réel :

$$\ell(\gamma) = \sum_{k=0}^{n-1} p(s_k, s_{k+1}) \ .$$

Un chemin de s à s' de longueur minimale est un **chemin géodésique** ou **plus court chemin** de s à s'.

Si de plus le graphe est non orienté et sa valuation symétrique (c'est-à-dire $p(s, s') = p(s', s)$), la longueur minimale d'un chemin (géodésique) de s à s', si elle existe, est la **distance** $d(s, s')$ entre les sommets s, s'.

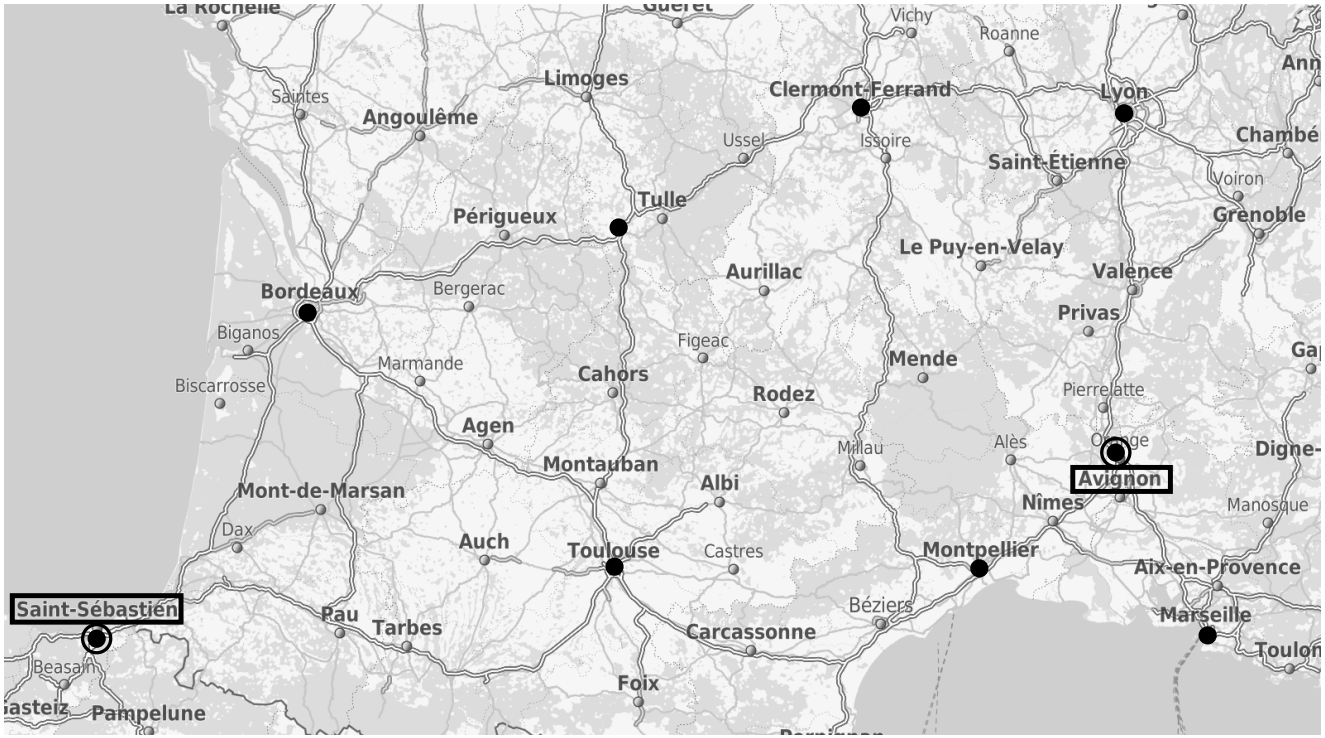
Si de plus le graphe est non orienté et sa valuation symétrique (c'est-à-dire $p(s,s') = p(s',s)$), la longueur minimale d'un chemin (géodésique) de s à s', si elle existe, est la **distance** $d(s,s')$ entre les sommets s, s'.

Si le graphe est de plus connexe, l'application d ainsi définie est une distance sur $\mathscr{S}$, c'est-à-dire :

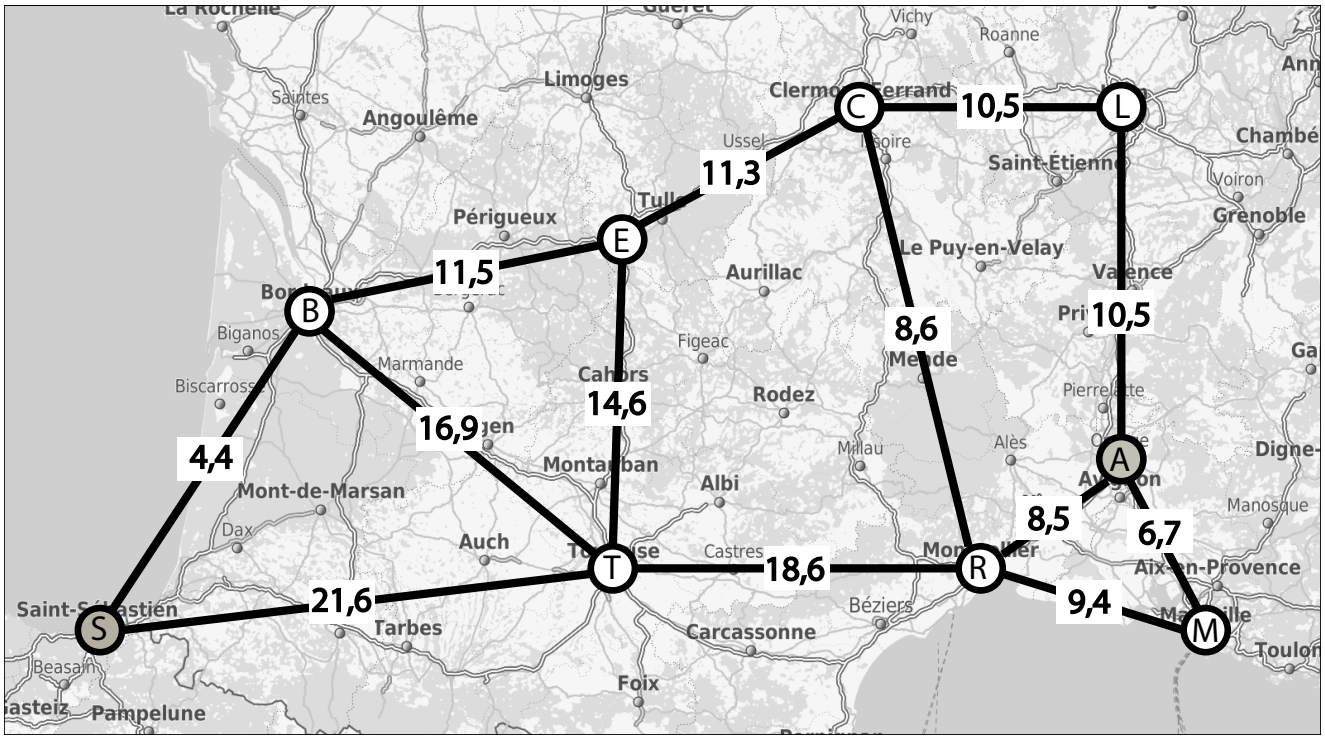
- Séparation. $\forall\, s,s' \in \mathscr{S},\ d(s,s') = 0 \iff s = s'$.
- Symétrie. $\forall\, s,s' \in \mathscr{S},\ d(s,s') = d(s',s)$.
- Inégalité triangulaire. $\forall\, s,s',s'' \in \mathscr{S}$, $d(s,s'') \leqslant d(s,s') + d(s',s'')$.

- **Illustration sur un exemple**

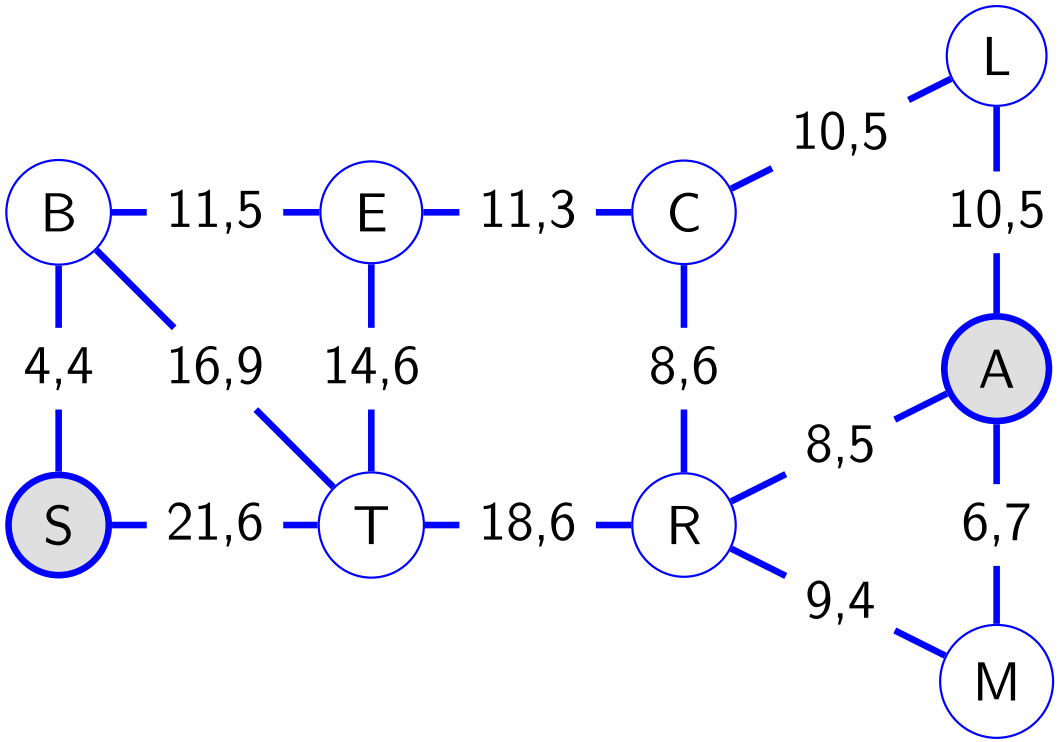
On souhaite effectuer un trajet d'Avignon à la ville de Saint-Sébastien (à la frontière espagnole) par le réseau autoroutier du sud de la France.



Connaissant le coût (en péage et carburant) pour chaque tronçon autoroutier, on souhaite déterminer le parcours autoroutier dont le tarif est le plus économique pour relier Avignon à Saint-Sébastien.



On code ces informations dans un graphe valué non orienté : ses sommets représentent les échangeurs (aux grandes villes Avignon, Marseille, montpellieR, Lyon, Clermont-ferrand, tullE, Bordeaux, Toulouse, Saint-sébastien), et ses arêtes les tronçons autoroutiers les reliant. Les valuations des arêtes sont les coûts des différents tronçons exprimés en €, pour le véhicule utilisé et pour une consommation moyenne.



On consigne toutes ces informations dans la **matrice de valuation du graphe** :

	M	A	L	R	C	E	T	B	S
M	∞	6,7	∞	9,4	∞	∞	∞	∞	∞
A	6,7	∞	10,5	8,5	∞	∞	∞	∞	∞
L	∞	10,5	∞	∞	10,5	∞	∞	∞	∞
R	9,4	8,5	∞	∞	8,6	∞	18,6	∞	∞
C	∞	∞	10,5	8,6	∞	11,3	∞	∞	∞
E	∞	∞	∞	∞	11,3	∞	14,6	11,5	∞
T	∞	∞	∞	18,6	∞	14,6	∞	16,9	21,6
B	∞	∞	∞	∞	∞	11,5	16,9	∞	4,4
S	∞	∞	∞	∞	∞	∞	21,6	4,4	∞

On consigne toutes ces informations dans la **matrice de valuation du graphe** :

	M	A	L	R	C	E	T	B	S
M	∞	6,7	∞	9,4	∞	∞	∞	∞	∞
A	6,7	∞	10,5	8,5	∞	∞	∞	∞	∞
L	∞	10,5	∞	∞	10,5	∞	∞	∞	∞
R	9,4	8,5	∞	∞	8,6	∞	18,6	∞	∞
C	∞	∞	10,5	8,6	∞	11,3	∞	∞	∞
E	∞	∞	∞	∞	11,3	∞	14,6	11,5	∞
T	∞	∞	∞	18,6	∞	14,6	∞	16,9	21,6
B	∞	∞	∞	∞	∞	11,5	16,9	∞	4,4
S	∞	∞	∞	∞	∞	∞	21,6	4,4	∞

C'est une matrice symétrique ; on a compté les mêmes coûts sur chaque tronçon dans les deux directions. Les arêtes absentes sont représentées par un poids ∞.

On consigne toutes ces informations dans la **matrice de valuation du graphe** :

	M	A	L	R	C	E	T	B	S
M	∞	6,7	∞	9,4	∞	∞	∞	∞	∞
A	6,7	∞	10,5	8,5	∞	∞	∞	∞	∞
L	∞	10,5	∞	∞	10,5	∞	∞	∞	∞
R	9,4	8,5	∞	∞	8,6	∞	18,6	∞	∞
C	∞	∞	10,5	8,6	∞	11,3	∞	∞	∞
E	∞	∞	∞	∞	11,3	∞	14,6	11,5	∞
T	∞	∞	∞	18,6	∞	14,6	∞	16,9	21,6
B	∞	∞	∞	∞	∞	11,5	16,9	∞	4,4
S	∞	∞	∞	∞	∞	∞	21,6	4,4	∞

C'est une matrice symétrique ; on a compté les mêmes coûts sur chaque tronçon dans les deux directions. Les arêtes absentes sont représentées par un poids ∞.

Un chemin géodésique reliant A à S dans un graphe fournirait le trajet le plus économique. L'algorithme de Dijkstra permet de trouver de manière efficace un tel chemin (vu en 2ème année.)

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc...

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :

– la liste `M` du marquage des sommets dejà parcourus ;

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets déjà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets déjà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.
– La liste `F` file d'attente des sommets à parcourir ;

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :

– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.

– La liste `F` file d'attente des sommets à parcourir ; initialement elle contient le sommet initial s_0.

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.
– La liste `F` file d'attente des sommets à parcourir ; initialement elle contient le sommet initial s_0. On retire de la file d'attente son premier sommet s ; on ajoute à la fin de `F` tous les sommets successeurs de s non marqués, et on les marque dans `M`.

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.
– La liste `F` file d'attente des sommets à parcourir ; initialement elle contient le sommet initial s_0. On retire de la file d'attente son premier sommet s ; on ajoute à la fin de `F` tous les sommets successeurs de s non marqués, et on les marque dans `M`. On poursuit ainsi tant que `F` est non vide.

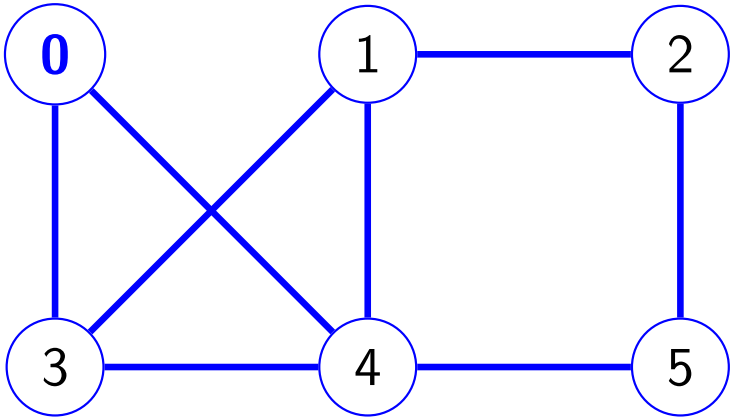
On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.
– La liste `F` file d'attente des sommets à parcourir ; initialement elle contient le sommet initial s_0. On retire de la file d'attente son premier sommet s ; on ajoute à la fin de `F` tous les sommets successeurs de s non marqués, et on les marque dans `M`. On poursuit ainsi tant que `F` est non vide.
À la fin la liste `M` permettra de récupérer tous les sommets marqués, c'est à dire parcourus.

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.
– La liste `F` file d'attente des sommets à parcourir ; initialement elle contient le sommet initial s_0. On retire de la file d'attente son premier sommet s ; on ajoute à la fin de `F` tous les sommets successeurs de s non marqués, et on les marque dans `M`. On poursuit ainsi tant que `F` est non vide.
À la fin la liste `M` permettra de récupérer tous les sommets marqués, c'est à dire parcourus.Dans un graphe non orienté, on obtiendra tous les sommets dans la même composante connexe que s_0.

On souhaite parcourir tous les sommets d'un graphe reliés à un sommet initial. Pour cela on peut effectuer un **Parcours en largeur**, où les sommets sont parcourus de proches en proches, à partir d'un sommet on parcourt tous ses sommets adjacents, puis leur sommets adjacents non deja parcourus, etc... le principe est le suivant : On considère deux listes :
– la liste `M` du marquage des sommets dejà parcourus ; elle contient autant d'élément qu'il n'y a de sommets, et permet de marquer les sommets dejà parcourus. Initialement elle ne contient que des 0 ; sauf pour le sommet s_0, qu'on marque en 1.
– La liste `F` file d'attente des sommets à parcourir ; initialement elle contient le sommet initial s_0. On retire de la file d'attente son premier sommet s ; on ajoute à la fin de `F` tous les sommets successeurs de s non marqués, et on les marque dans `M`. On poursuit ainsi tant que `F` est non vide.
À la fin la liste `M` permettra de récupérer tous les sommets marqués, c'est à dire parcourus.Dans un graphe non orienté, on obtiendra tous les sommets dans la même composante connexe que s_0. Dans un graphe orienté, on obtiendra tous les sommets pour lesquels il existe un chemin au départ de s_0 les reliant.

- **Exemple :**

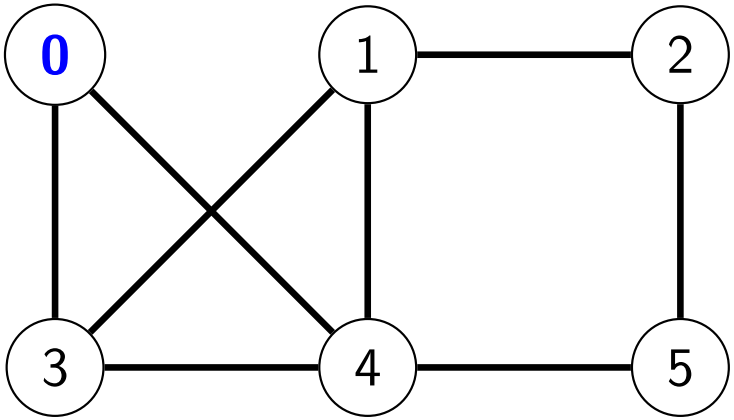
Au départ du sommet 0 :



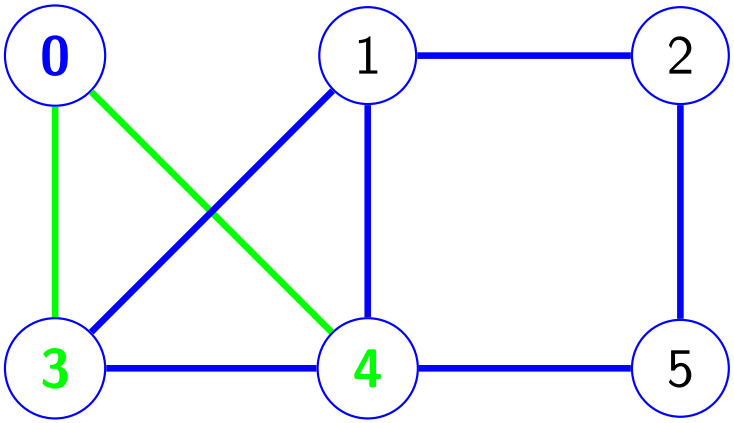
`M = [1,0,0,0,0,0] ; F = [0]`

- **Exemple :**

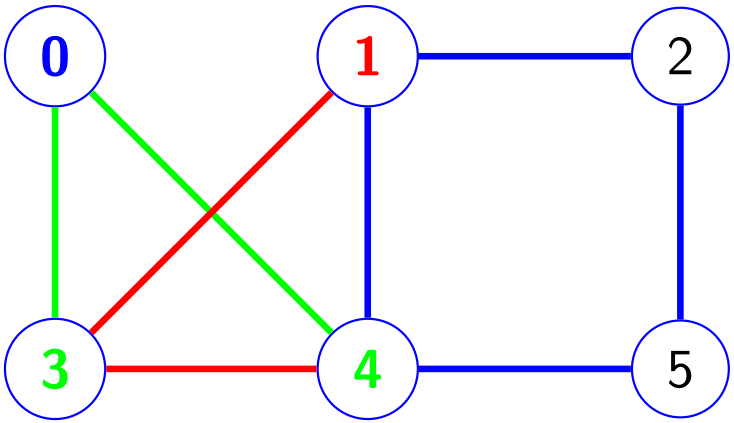
Au départ du sommet 0 :



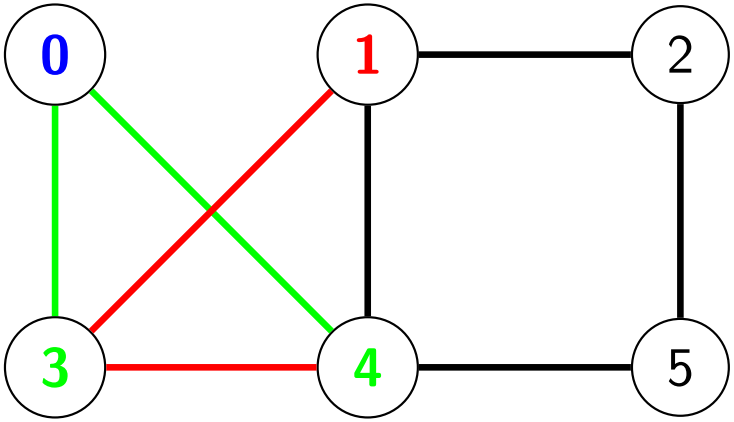

```
M = [1,0,0,0,0,0] ; F = [0]
```



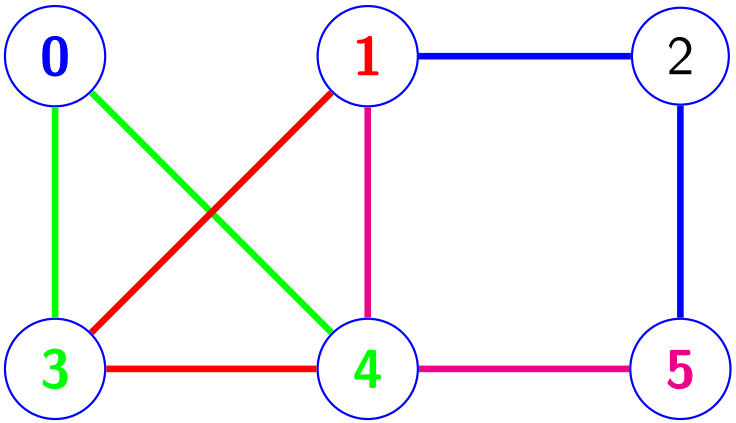

```
M = [1,0,0,1,1,0]; F = [3,4]
```



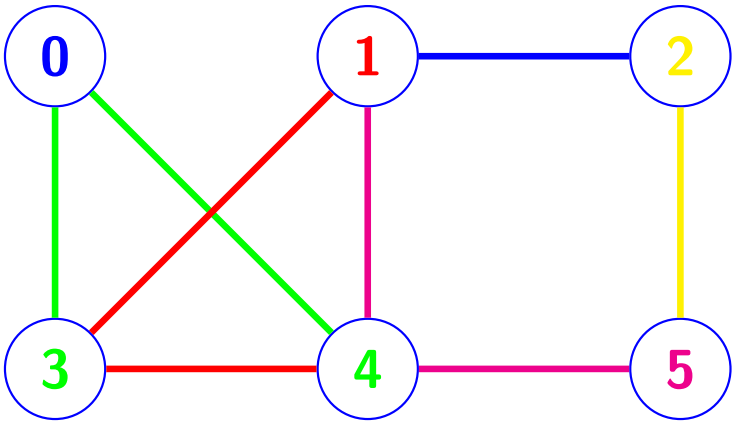

```
M = [1,1,0,1,1,0]; F = [4,1]
```



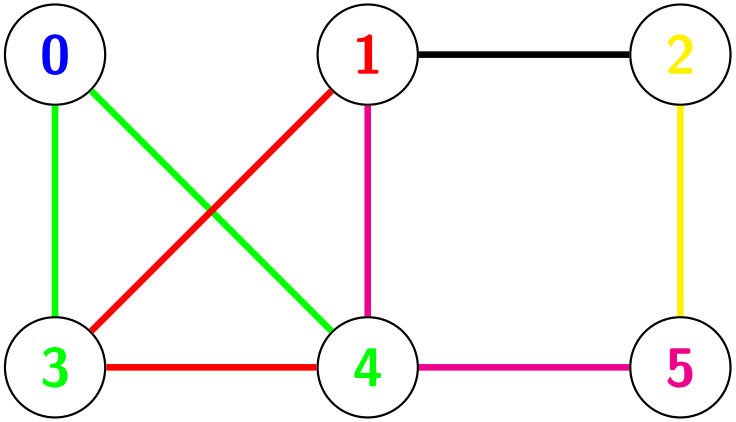

```
M = [1,1,0,1,1,0]; F = [4,1]
```



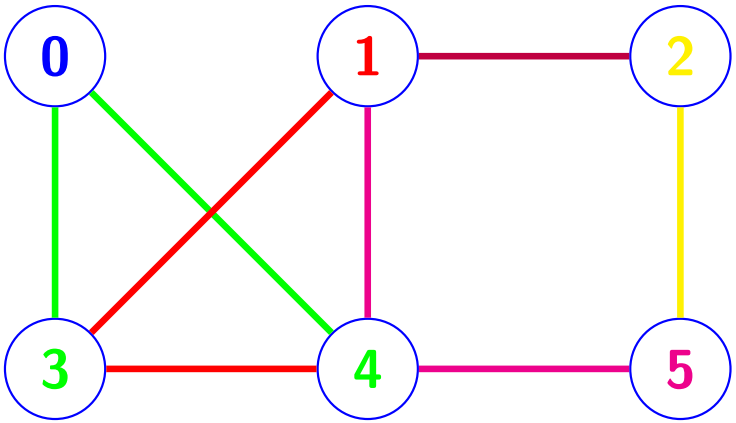

```
M = [1,1,0,1,1,1] ; F = [1,5]
```



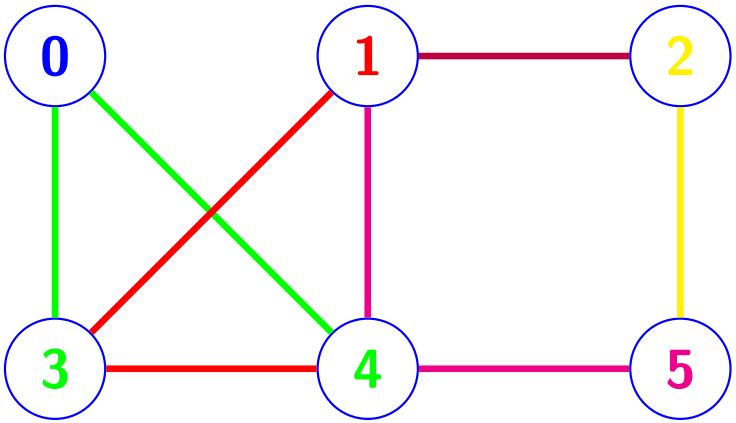

```
M = [1,1,1,1,1,1]; F = [5,2]
```




```
M = [1,1,1,1,1,1]; F = [5,2]
```




```
M = [1,1,1,1,1,1] ; F = [2]
```




```
M = [1,1,1,1,1,1] ; F = []
```

```
def parcours_largeur(Liste,s0):
    n = Liste[0]            # Nbre sommets
    Ad = Liste[1]           # Ad[i] : successeurs de i
    M = [0] * n             # tableau Marquage
    file = [s0] # File
    M[s0] = 1   # Marquage sommet initial
    while file != []:
        s = file.pop(0)  # Retrait 1er élt file
        for v in Ad[s]:  # Pour tous ses successeurs
            if M[v] == 0: # si non marqué
                file.append(v)      # ajout fin de file
                M[v] = 1            # et marquage
    parcourus = []          # liste sommets parcourus
    for i in range(n):
        if M[i] == 1:
            parcourus.append(i)
    return parcourus
```

Cette implémentation est améliorable : utiliser une liste pour la file rend couteux le retrait du premier élément de la file.