	Structures de données	**NSI T^{ale}**
	Les Graphes	Cours/TD

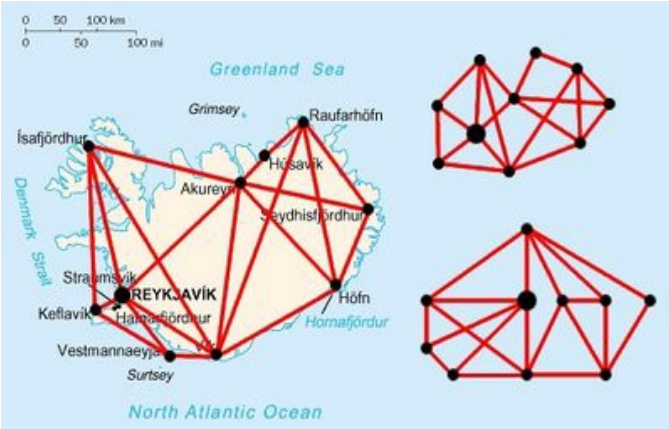
*

1. Introduction

→ Présentation

La **théorie des graphe** est une théorie fondamentale de l'informatique et des mathématiques.
On rencontre des graphes dans tous les domaines des sciences et leurs applications sont innombrables :

- le réseau routier d'un pays et le réseau de transport d'une ville forment un graphe :



- Internet peut être pensé comme un graphe,
- les réseaux sociaux présentent naturellement des graphes entre les personnes,

→ Utilisation des graphes en informatique

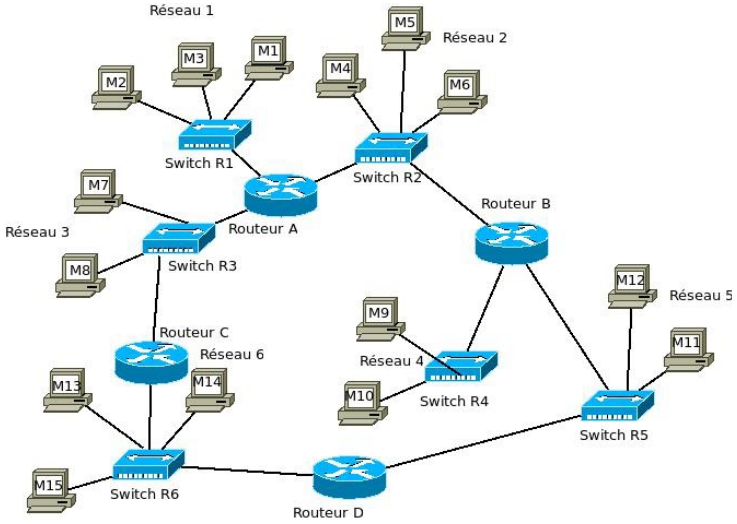
Parmi les problèmes fréquents faisant apparaître des graphes on rencontre :

1. La recherche des chemin. Puis-je passer de l'état A à l'état B ?

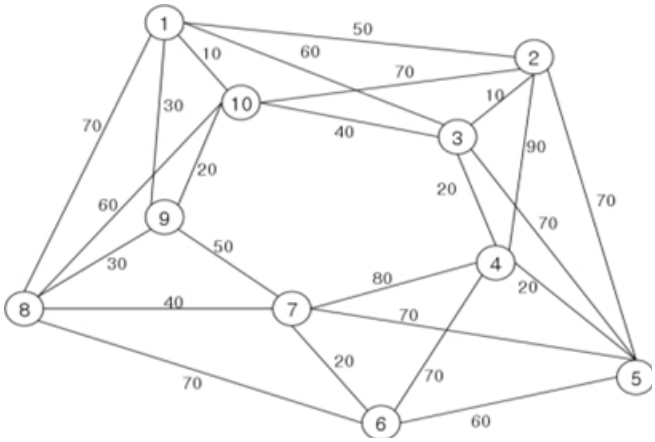
- Comment résoudre un taquin ?

2. L'exploration de graphe.

- Recherche d'une meilleure route dans un réseau.

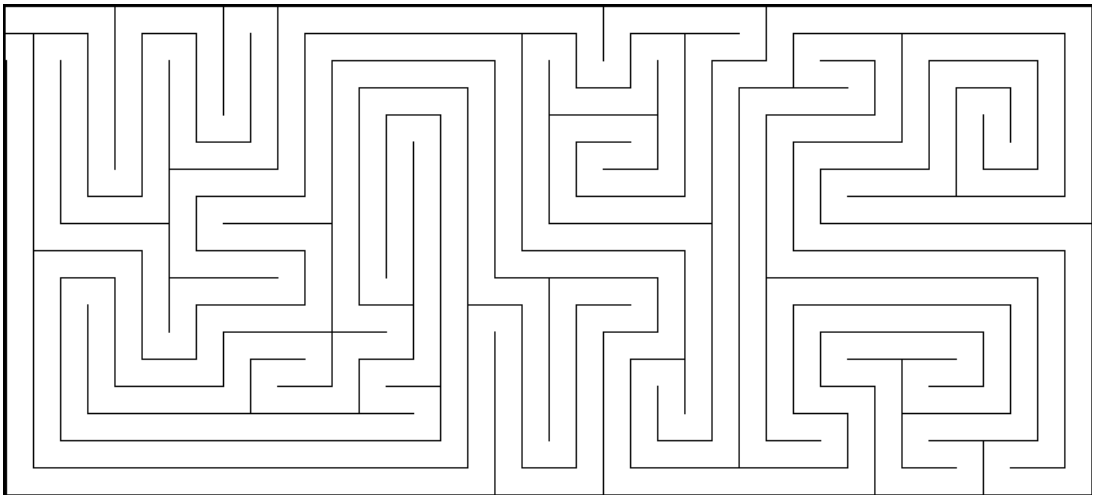


- Recherche d'un trajet entre des villes sur une carte : Quelle est la meilleure route reliant 1 à 5 ?

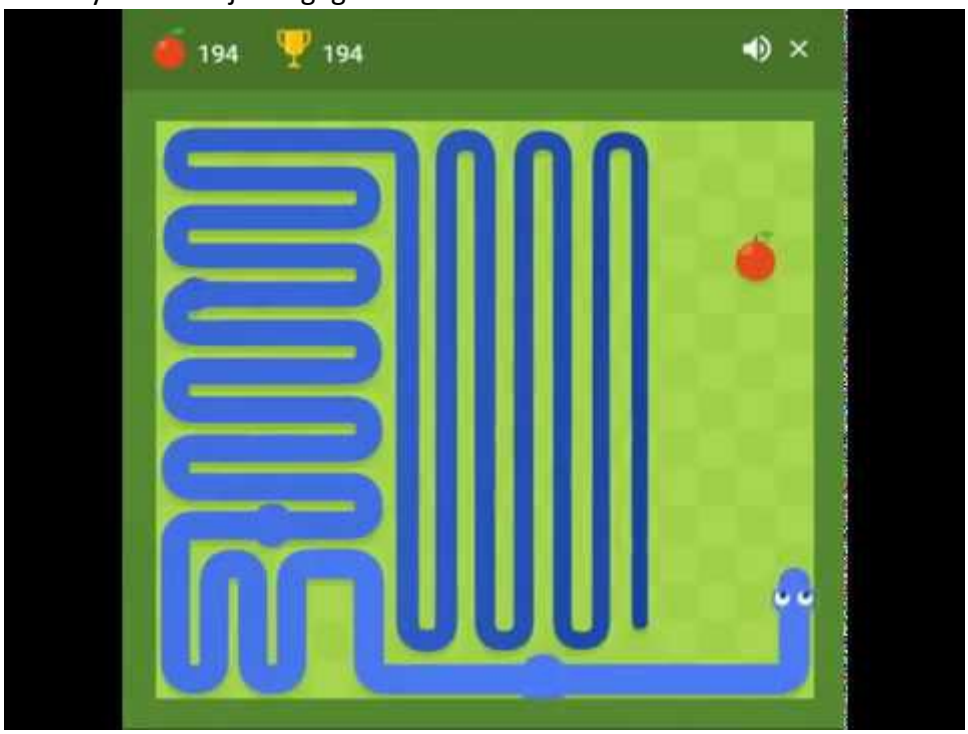


3. La recherche de cycles dans un graphe.

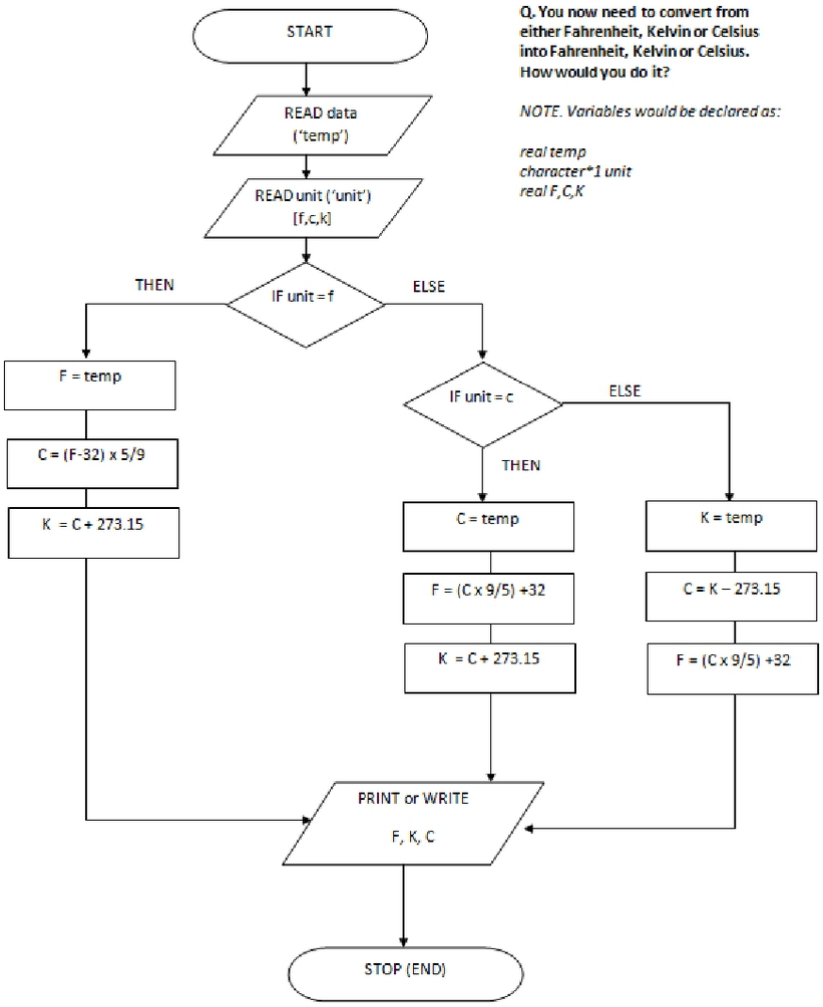
- Comment trouver la sortie sans tourner en rond ?



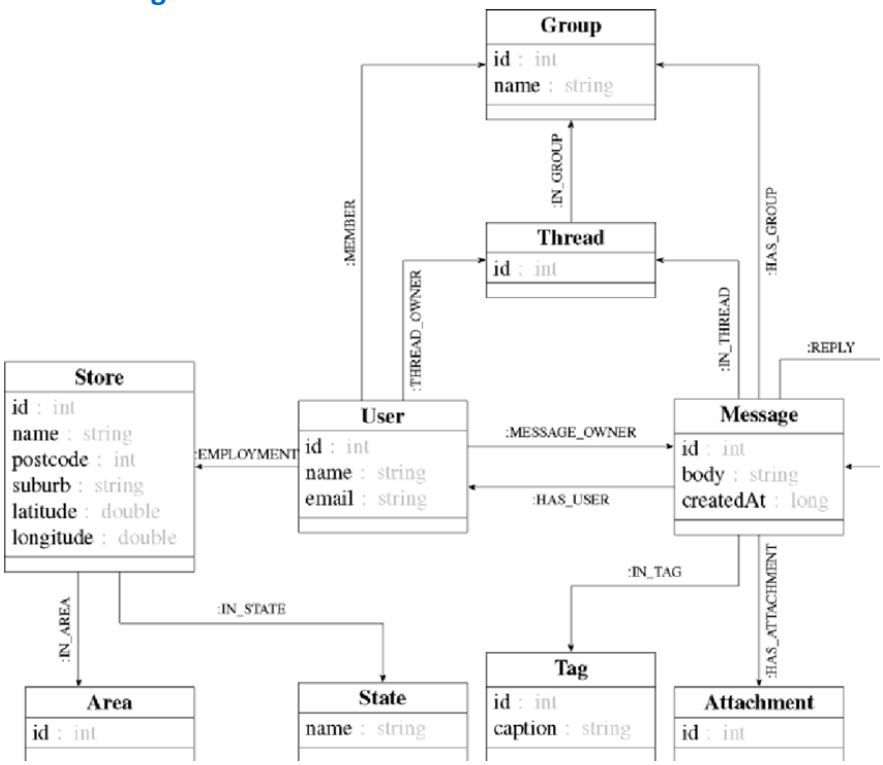
- Existe-t-il un moyen de toujours gagner à Snake ?



4. Les algorithmes peuvent être vus comme des graphes :



5. Mais aussi la manière d'enregistrer les données :



→ Distinction mathématique, informatique

Les définitions des graphes en mathématique et en informatique sont similaires. Les applications diffèrent souvent.
Nous (les informaticiens) cherchons souvent à exposer les solutions de nos problèmes. Un algorithme doit fournir une solution en un temps fini. En mathématiques, on se contente parfois de l'existante d'une solution, sans proposer de moyen de la construire. C'est souvent jugé insatisfaisant mais c'est ainsi.
Autre nuance importante, nous cherchons généralement des algorithmes *rapides*. Un algorithme qui termine mais prend des millions d'années n'a que peu d'intérêt.
Enfin, si la structure nous intéresse d'un point de vue abstrait, son *implémentation* est importante.

→ Objectifs

Nous allons donc :

1. Définir une **structure de donnée** permettant de manipuler les graphes.
2. **Implémenter cette structure**.
3. Résoudre des problèmes utilisant les graphes et donc :
   - proposer des **algorithmes** pour les résoudre,
   - **implémenter** ces algorithmes.

Parmi les problèmes que nous allons aborder :

1. L'exploration de graphe,
2. La recherche de chemin dans un graphe,
3. La recherche de cycles dans un graphe.

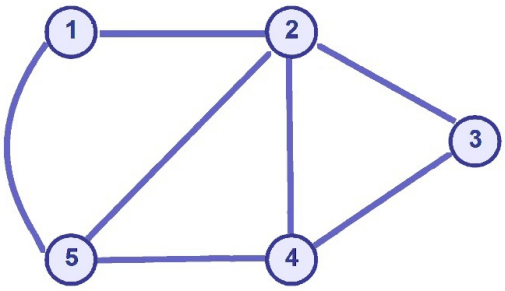
2. Définitions

→ Définition : Graphe simple

Un **graphe non orienté** est un couple (X,E) où X est un ensemble fini de **sommets** et E un ensemble de paires {x, y} de deux éléments de X appelées **arêtes**.

Les termes et notations anglais, que vous rencontrerez souvent sont : sommet : ***vertice*** et arête : ***edge***.
Afin de mieux voir les choses, il est possible de représenter les graphes par leur **diagramme sagittal**.

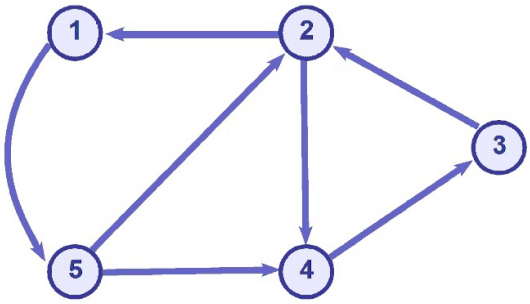
Exemple :
Le graphe G1 = {X = {1,2,3,4,5}, E = {{1,5},{2,1},{2,4},{3,2},{4,3},{5,2},{5,4}}} peut être représenté par le diagramme suivant :



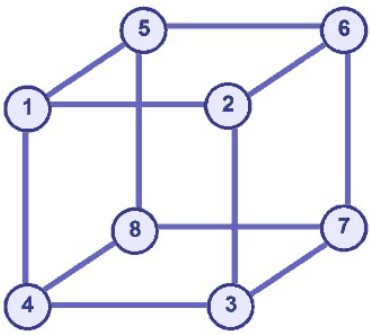
→ Définition : Graphe orienté

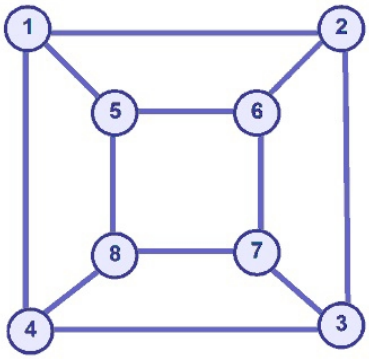
Un **graphe orienté** est un couple (X,E) où X est un ensemble fini de **sommets** et E un ensemble de couples (x, y) d'éléments de X appelées **arcs**.

Exemple :
Le graphe G2 = {X = {1,2,3,4,5}, E = {(1,5),(2,1),(2,4),(3,2),(4,3),(5,2),(5,4)}} peut être représenté par le diagramme suivant :



Attention, il ne faut pas confondre un graphe et sa représentation sagittale (qui n'est pas unique !).

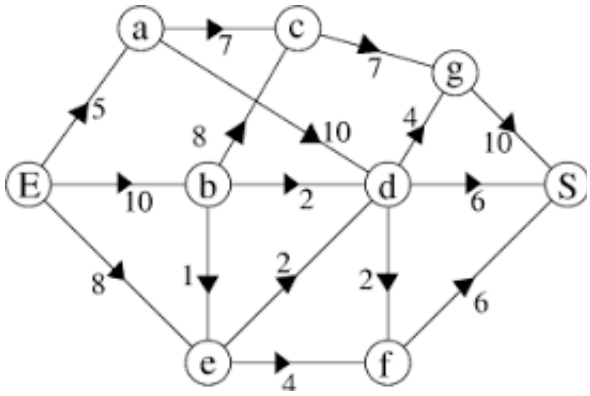




Deux représentations sagittales d'un même graphe.

→ Définition : Graphe pondéré

Dans certains cas, toutes les arêtes ne se valent pas. (Un trajet Lille-Paris ne “coûte” pas autant qu’un “Paris-Lyon”.)
Dans ce cas on attribue aux arrêtes un **poids**, souvent noté w (pour weight).



Ici l’arête *(E, b)* a un poids de 10.
Et le trajet *(E, b, d, S)* pèse 10 + 2 + 6 = 18

→ Définition : Ordre d'un graphe

Soit G = (X,E) un graphe. L’**ordre du graphe** G est le **nombre de ses sommets**.
Nous notons généralement n = |X| et m= |E|.

Par convention, nous noterons toujours X = {1,2,3,...,n}.

→ Définition : Voisinage, degré

Soit **G** un **GRAPHE NON ORIENTÉ** :

- Les deux sommets x et y sont dits **adjacents** ou **voisins** s’il existe une arête {x, y} dans G. x et y sont appelés les **extrémités** de l’arête {x, y}.
- Le **voisinage** du sommet x est **l’ensemble** noté V(x) **de ses sommets adjacents**.
- Nous appelons **degré** de x le **nombre d’arêtes incidentes à x**. C’est donc aussi le **cardinal du voisinage** de x. Nous notons d(x) = |V(x)|, le degré de x.

Exemple :
Dans le graphe G1, le sommet 4 est de degré 3 et a pour voisinage {2,3,5}.Attention,

Soit **G** un **GRAPHE ORIENTÉ** :

- Les deux sommets x et y sont dits **adjacents** s’il existe un arc (x, y) dans G. x est l’**origine** de l’arc et y son **extrémité**. x est un **prédécesseur** de y et y un **successeur** de x.
- Le **voisinage sortant** d’un sommet x, est l’ensemble noté $V^+(x)$ de ses successeurs. Nous appelons **degré sortant** de x le nombre de successeurs de x, on note $d^+(x) = |V^+(x)|$.
- Le **voisinage entrant** d’un sommet x, est l’ensemble noté $V^-(x)$ de ses prédécesseurs. Nous appelons **degré entrant** de x le nombre de prédécesseurs de x, on note $d^-(x) = |V^-(x)|$.

Exemple :
Dans le graphe G2, le sommet 4 est de degré entrant $d^-(4) = 2$ (car $V^-(4) = \{2,5\}$) et de degré sortant $d^+(4) = 1$ (car $V^+(x) = \{3\}$).

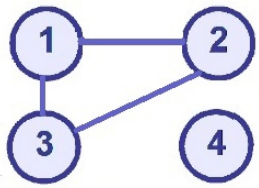
3. Représentations d'un graphe

Il existe plusieurs façons de représenter un graphe en machine.

→ Représentation par liste d'arêtes

La représentation d'un graphe G par sa **liste d'arêtes** (ou d'arcs) consiste à représenter G par un couple : son nombre de sommets et la liste de ses arêtes (ou arcs).

Exemple :



Le graphe non orienté G3 ci-contre est ainsi représenté par

(4,[[1,2],[1,3],[2,3]])

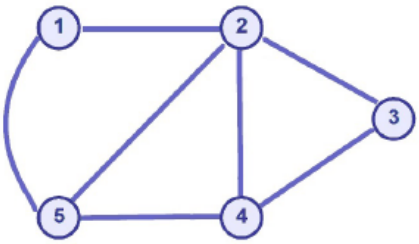
Remarque : Cette représentation par liste d'arêtes (ou d'arcs) de G correspond à sa définition "naturelle", d'après la définition d'un graphe. Cependant, nous verrons qu'elle est peu efficace en pratique. En algorithmique, elle est en général seulement utilisée pour la saisie d'un graphe, puis immédiatement transformée en représentation par matrice ou liste d'adjacence.

→ Représentation par matrice d'adjacence

Nous appelons **matrice d'adjacence** de G, la matrice A = (ai, j) telle que pour tout (i, j) Є X^2 :

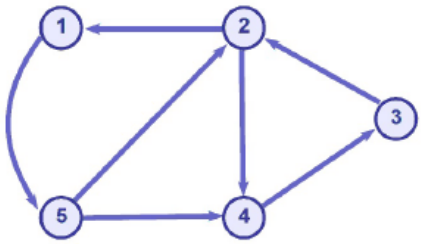
- (*Cas non orienté*) $a_{i,j}$ est égal à 1 si $\{i,j\} \in E$ et 0 sinon. La matrice est donc ***symétrique***.
- (*Cas orienté*) $a_{i,j}$ est égal à 1 si $(i,j) \in E$ et 0 sinon. La matrice n'est donc plus nécessairement symétrique.

Exemple :



La matrice d'adjacence de G_1 est

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



La matrice d'adjacence de G_2 est

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

La lecture de cette matrice s'effectue ainsi :

- la 1ère ligne concerne le sommet 1 : il est relié par une arête avec les sommets 2 et 5
- la 2ème ligne concerne le sommet 2 : il est relié par une arête avec les sommets 1, 3, 4 et 5
- etc

PROPRIÉTÉS

Proposition

Dans le cas d'un **graphe non orienté**,

1. La somme des coefficients de la matrice A est égale à $2 \times m$.
2. La somme des éléments de la ligne i ou de la colonne i est égale à $d(i)$.

Dans le cas d'un **graphe orienté**,

1. La somme des coefficients de la matrice A est égale à m.
2. La somme des éléments de la ligne i est égale à $d^+(i)$.
3. La somme des éléments de la colonne i est égale à $d^-(i)$.

ALGORITHMIQUE

Cette représentation est pratique par exemple pour

- tester l'existence d'un arc entre deux sommets
- ajouter ou retirer un arc

ce qui se fait en temps constant *O(1)*.

Nous pouvons également parcourir ou traiter tous les sommets adjacents d'un sommet (en *O(n)*). Par exemple, pour le calcul du degré d'un sommet i dans un graphe non orienté :

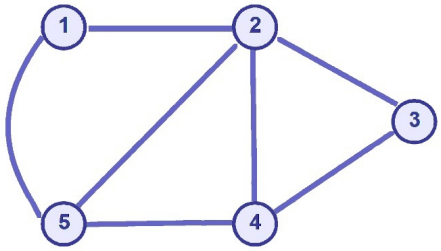
```
Di = 0
Pour j de 1 à n faire
     Si A[i][j] != 0 alors Di = Di + 1
```

En revanche, une consultation de toutes les arêtes (ou arcs) nécessite de parcourir toute la matrice, ce qui donne un algorithme de complexité $O(n^2)$. L'espace mémoire nécessaire est également en $O(n^2)$ quelque soit le nombre d'arêtes (arcs) m. Si m est très petit par rapport à n, ce n'est pas judicieux et la représentation suivante sera plus économe.

→ Représentation par les listes d'adjacence

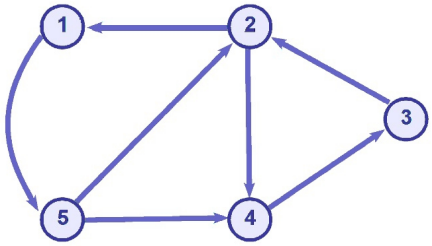
La représentation de *G* par **listes d'adjacence** consiste à représenter *G* par un tableau *L* indexé par *X* tel que pour tout sommet *x*, *L[x]* est la liste de ses successeurs dans *G*.

Exemple :



La représentation de G1 par listes d'adjacence est :

[[2,5], [1,3,4,5], [2,4], [2,3,5], [1,2,4]]



La représentation de G2 par listes d'adjacence est :

[[5], [1,4], [2], [3], [2,4]]

ALGORITHMIQUE

Pour un graphe à *n* sommets et *m* arcs, l'espace mémoire utilisé est en *O(n+m)*. Et le parcours des voisins d'un sommet *i*, comme pour le calcul de son degré, cela s'effectue en *O(d(i))* :

```
Di = 0
Pour y dans L[i] :
     Di += 1
```

Ces performances sont telles que nous n'utiliserons en TP que cette représentation.

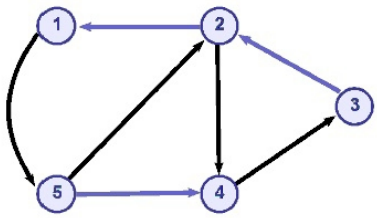
4. Chaînes et cycles d'un graphe non orienté. Connexité.

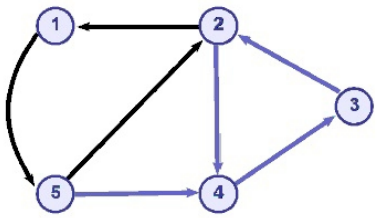
1. On appelle **chaîne** de $G = (X,E)$ toute suite de sommets de G : $c = (x_0, x_1,..., x_p)$ telle que pour tout $i \in \{1,..., p\}$, $\{x_{i-1}, x_i\} \in E$. c est une chaîne de longueur p (nombre d'arêtes) joignant x_0 à x_p.
   Par convention, (x_0) est une chaîne de longueur nulle joignant x_0 à lui-même.
2. Soit x et y deux sommets de G. Nous disons que y est **accessible** à partir de x dans G s'il existe une chaîne de G joignant x à y.
3. $G = (X,E)$ est dit **connexe** si pour tout $(x, y) \in X^2$, y est accessible à partir de x.
4. Nous appelons **cycle** une chaîne dont l'origine est égale à l'extrémité, c'est à dire, une chaîne de la forme $c = (x_0, x_1,..., x_p, x_0)$.

5. Chemins et circuits d'un graphe orienté

1. On appelle **chemin** de $G = (X,E)$ toute suite de sommets de G : $c = (x_0, x_1,..., x_p)$ telle que pour tout $i \in \{1,..., p\}$, $(x_{i-1}, x_i) \in E$. c est une chaîne de longueur p (nombre d'arcs) joignant x_0 à x_p.
2. Soit x et y deux sommets de G. Nous disons que y est **accessible** à partir de x dans G s'il existe un chemin de G joignant x à y.
3. Nous appelons **circuit** un chemin dont l'origine est égale à l'extrémité, c'est à dire, une chaîne de la forme $c = (x_0, x_1,..., x_p, x_0)$.
4. Un chemin est **élémentaire** s'il ne passe pas deux fois par le même sommet.
5. Un chemin est **simple** s'il ne passe pas deux fois par le même arc.

Exemple :





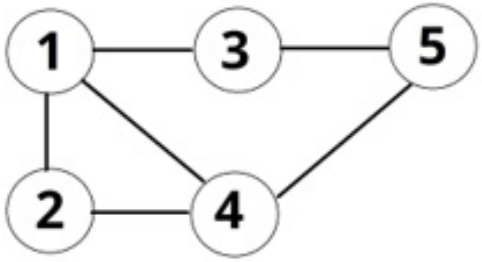
Exemple d'un chemin et d'un circuit dans les graphes G1 et G2.

REMARQUES :

1. Il existe pour les graphes orientés, une notion analogue à la connexité des graphes non orientés, appelée la forte connexité, mais elle se révèle plus complexe !
2. Par abus de langage, nous utiliserons le vocabulaire chemin et cycle que le graphe soit orienté ou non.

6. Exercice

Compléter le tableau suivant, pour donner les différentes représentations des graphes proposés.

Représentation sagittale	Représentation par liste d'arêtes	Représentation par matrice d'adjacence	Représentation par listes d'adjacence
			
	{{1,2}, {1,5}, {2,3}, {2,4}, {2,5}, {4,5}}		
		0 1 0 1 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 1 0 1 0 0 0	
			{{2,6,7}, {3,4}, {5,1}, {6,7}, {1,4}, {7}, {}}