



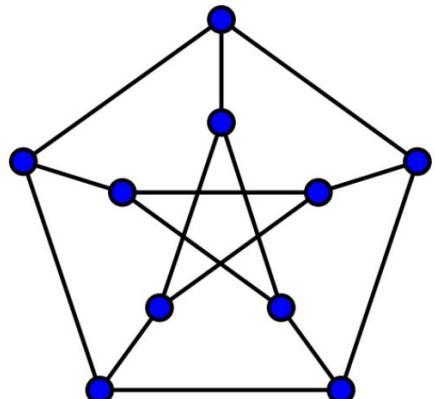
Université Sidi Mohamed Ben Abdellah

Ecole Nationale des Sciences Appliquées Fès ENSAF

Département Génie Industriel

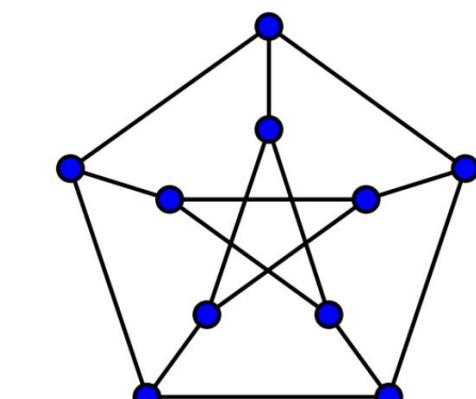
THEORIE DES GRAPHES

Filières : GI/GMSA S2



Pr : Mhamed SAYYOURI

Année universitaire 2023-2024



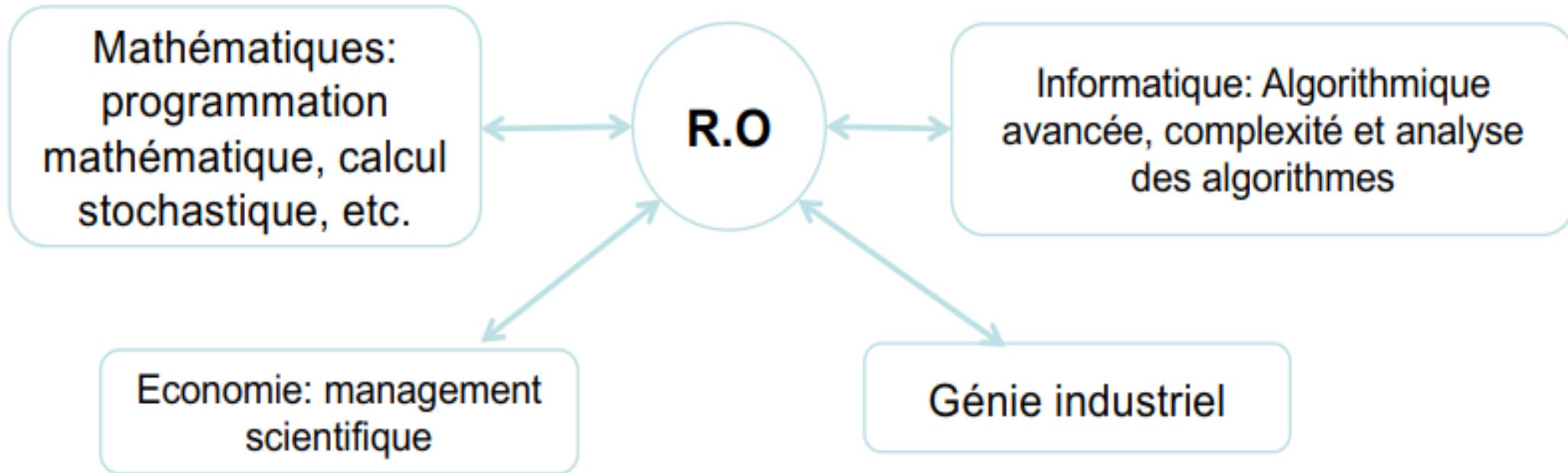
□ Informations générales sur le module : Recherche Opérationnelle

- 3 éléments du module :

- Théorie des graphes (5 semaines)
- Programmation linéaire (6 semaines)
- Files d'attentes (3 semaines)

- La **recherche opérationnelle** est une **approche quantitative** permettant de produire de meilleures décisions:
 - Elle fournit des outils pour **rationaliser, simuler et optimiser** le fonctionnement de plusieurs systèmes (administratifs, de production,...).
 - Elle propose des modèles pour analyser des situations complexes et permet aux **décideurs** de faire **des choix efficaces et robustes**.
- **Mots clés**
 - **Modélisation** : Simplification de la réalité pour pouvoir en apprêhender certains aspects
 - **Optimisation** : Identification d'une configuration qui soit meilleure que toute autre suivant un critère spécifique
 - **Simulation** : Représentation artificielle d'un fonctionnement réel

- Elle est le « carrefour » de différentes sciences et technologies



- Champs d'application :
 - Systèmes administratifs (ex : emplois du temps) ;
 - Ateliers de production (ex : ordonnancement) ;
 - Systèmes physiques (ex : verres de spin) ;
 - Systèmes de transport (ex : tournées de distribution) ;
 - Systèmes informatiques (ex : localisation de fichiers) ;
 - Systèmes biologiques (ex : alignements de séquence) ;

- **L'application de la recherche opérationnelle à un problème réel consiste à :**
 - **Elaborer** un modèle
 - **Développer** un algorithme de résolution exacte ou approchée ;
 - **Evaluer** la qualité des solutions produites par l'algorithme dans l'environnement réel du problème.
- **Les problèmes de la recherche opérationnelle peuvent se classer en :**
 - **Problèmes combinatoires** : problèmes d'affectations, problèmes de transports, optimisation dans les graphes, problèmes d'ordonnancement,
 - **Problèmes stochastiques** (ou intervient le hasard) : filles d'attente; stratégies dans l'incertain, ...
 - **Problèmes d'aide à la décision** : définition de politiques d'approvisionnement, de vente,

■ Les outils de la recherche opérationnelle

- Complexité et approximation;
- **Théorie des graphes** ;
- Probabilités et Processus stochastiques ;
- **Files d'attente** ;
- Programmation mathématique (continue et discrète);
- **Programmation linéaire** (continue, entière et mixte);
- Programmation dynamique ;
- Méta-heuristiques ;
- Méthodes énumératives ;
-

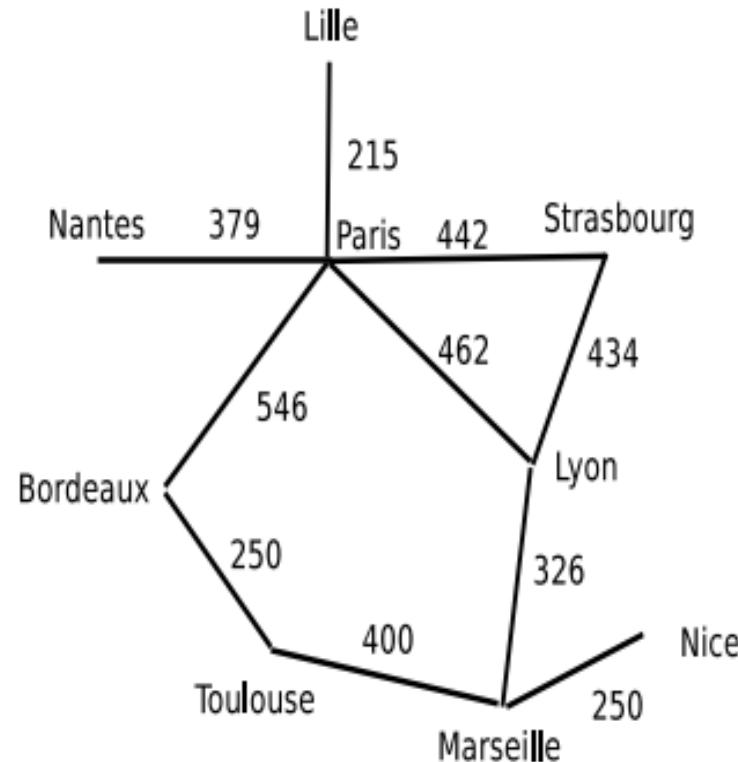
- Introduction
- Concepts fondamentaux des graphes
- Cheminement dans les graphes
- Principaux algorithmes de la théorie des graphes
 - Algorithme de coloration
 - Algorithme d'ordonnancement d'un graphe (MPM)
 - Algorithme de parcours de graphes
 - Algorithme d'optimisation pour des graphes pondérés
 - Algorithme du plus court chemin
 - Algorithme de flot maximum dans les réseaux

□ Introduction

- La théorie des graphes ou théorie mathématique de l'interconnection est la discipline **mathématique** et **informatique** qui étudie les graphes, lesquels sont des modèles abstraits de dessins de réseaux reliant des objets.
- Elle constitue un domaine des mathématiques qui s'est développé aussi au sein de disciplines diverses telles que:
 - la **chimie** (modélisation de structures),
 - la **biologie** (génome),
 - les **sciences sociales** (modélisation de relations),
 - la cartographie (Réseau routier, réseau internet),
 - L'économie (Planning de livraisons, gestion de flots, ordonnancement)
 - et dans **beaucoup d'applications industrielles**.
- Elle constitue un outil efficace pour résoudre **des problèmes discrets de la recherche opérationnelle**.

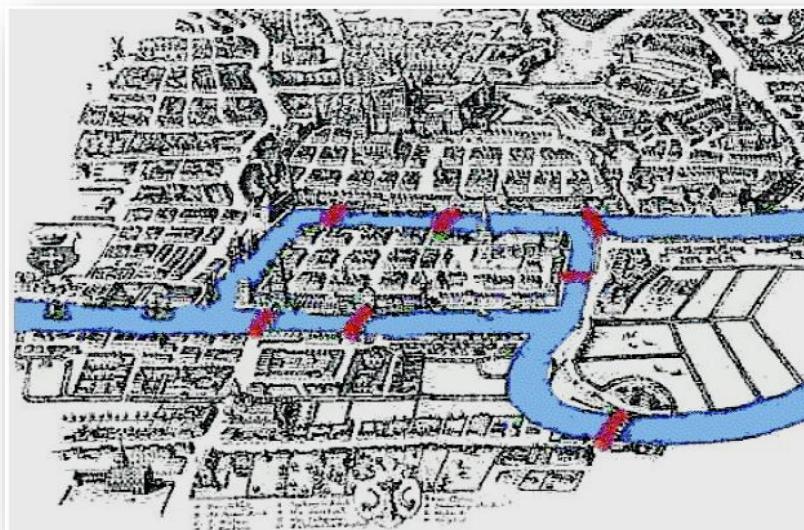
■ Introduction

- Un graphe permet de **représenter simplement la structure et les cheminements** d'un ensemble comprenant un grand nombre de situations, en exprimant les relations de dépendances entre ses éléments. On peut citer les réseaux de communication, les réseaux routiers et ferroviaires, les diagrammes de succession de tâches dans la gestion d'un projet ... etc.
- Le graphe est aussi une structure de donnée **puissante en informatique**. Mais, au-delà de la représentation de données, les graphes servent aussi et surtout pour **proposer des solutions à certains problèmes**.

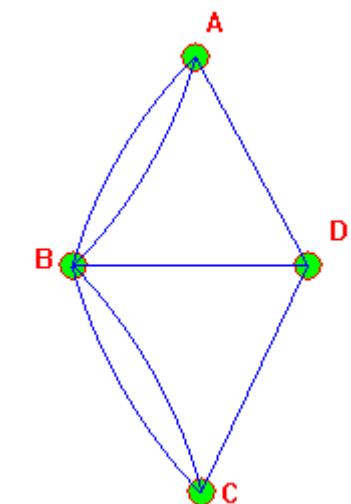
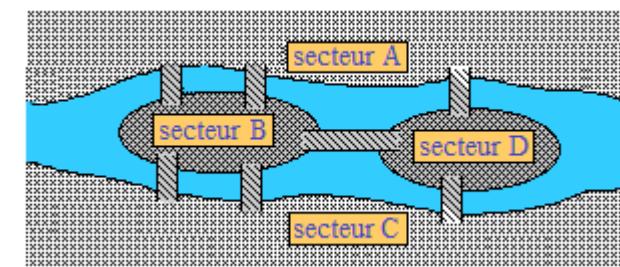


□ Origines

- Leonhard Euler (Académie de Saint Pétersbourg en 1735) a été le premier à soulever un problème de théorie des graphes.
- **Problème des sept ponts de Königsberg** : faire une promenade à partir d'un point et revenant à ce point en passant une seule fois par chaque pont : **Circuit Eulerien**.



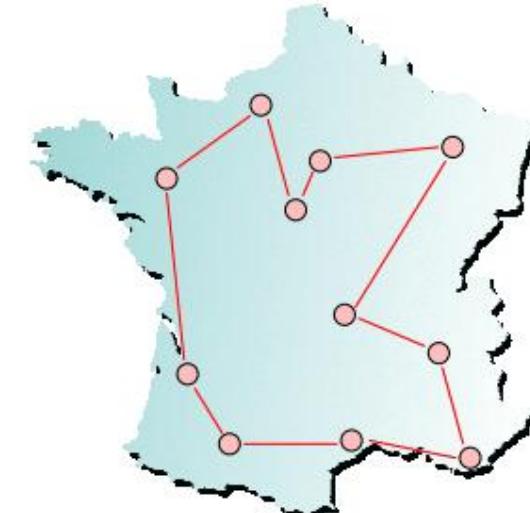
Le problème des ponts de Konigsberg
(Leonard Euler)



- Euler représenta cette situation à l'aide d'un graphe et démontra que ce problème n'admet pas de solution.

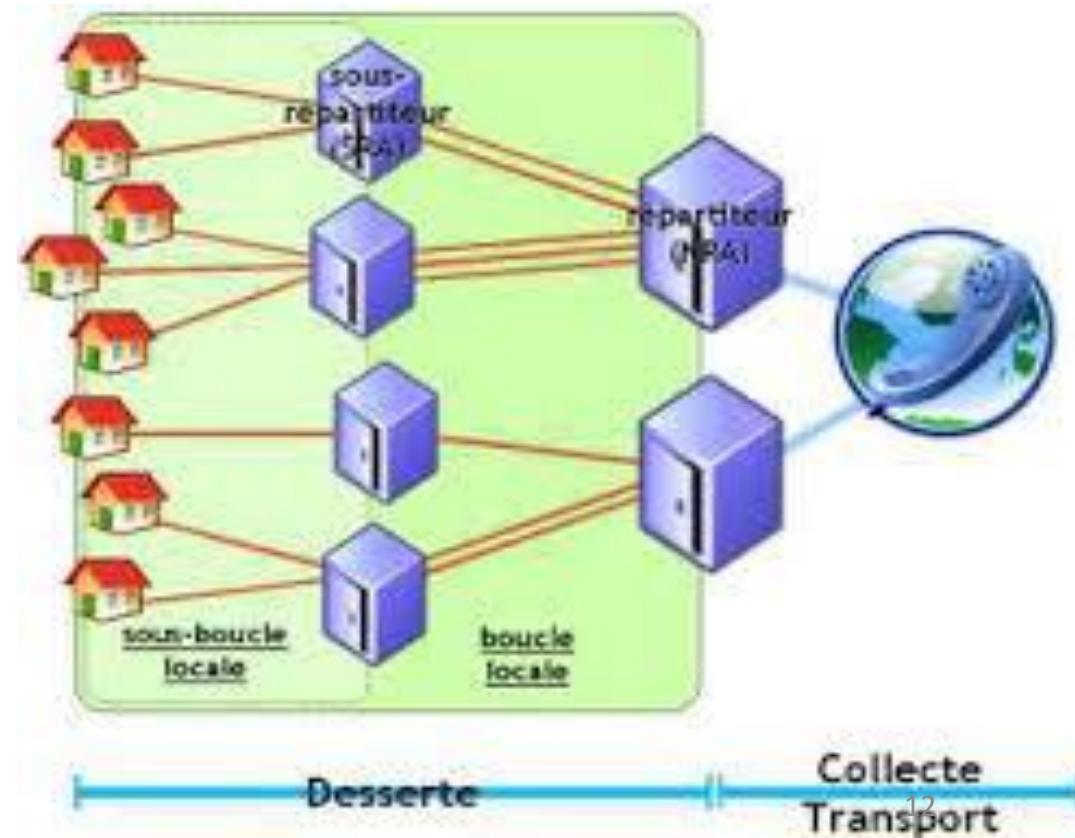
□ Exemples

- Problème du voyageur de commerce (Travelling salesman problem TSP): Un voyageur de commerce ayant n villes à visiter souhaite établir une tournée qui lui permette de passer une fois et une seule dans chaque ville pour finalement revenir à son point de départ, ceci en **minimisant le chemin parcouru** .
- Le problème de voyageur de commerce : **calculer un plus court circuit qui passe une et une seule fois par toutes les villes.**



❑ Exemples

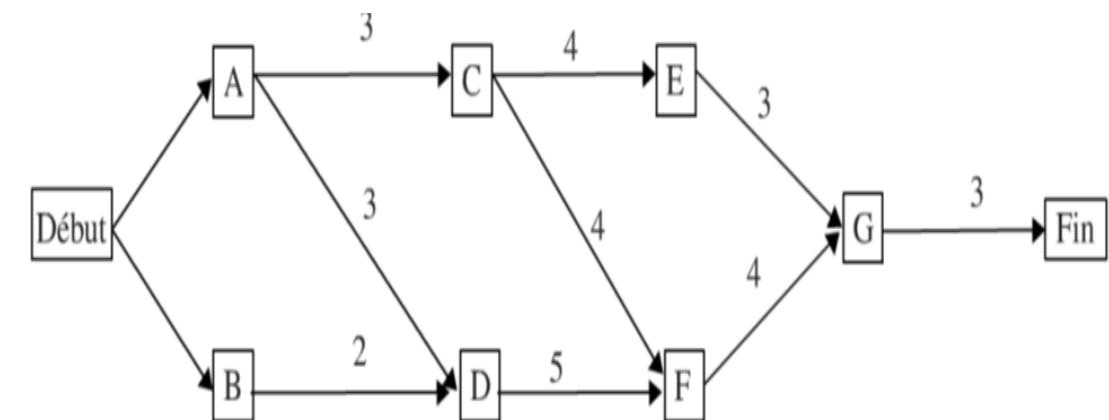
- **Fiabilité dans les réseaux :** Un réseau est un système de communication dans lequel des sites communiquent entre eux soit directement, soit par l'intermédiaire d'autres sites, en envoyant des messages qui circulent le long de lignes de communication. Dans ce réseau, il peut se produire qu'une ligne de communication soit coupée.
- Le problème posé est alors le suivant : **quel est le nombre maximum de pannes auquel cet acheminement peut résister ?**



❑ Exemples

- **Problème d'ordonnancement:** Un projet complexe est en général décomposé en tâches élémentaires ayant chacune une durée donnée. Ces tâches étant reliées les unes aux autres par des contraintes de précédence.
- Il s'agit de déterminer un calendrier d'exécution des tâches qui respecte les contraintes de précédence et permette l'achèvement du projet dans une durée optimale.

Tâche	Durée (en jours)	tâches antérieures
A	3	Aucune
B	2	Aucune
C	4	A
D	5	A, B
E	3	C
F	4	C, D
G	3	E, F



Chapitre 1 :

Concepts fondamentaux des graphes

□ Définition d'un Graphe

- Un **graphe** $G = (S, L)$ est défini par un modèle mathématique dans lequel un ensemble d'objets, représentés par des points appelés *sommets*, sont reliés entre eux par des liens, représentés par des lignes ou des traits appelés *arcs* ou *arêtes*.

- Un ensemble de sommets S . $S = \{S_1, S_2, \dots, S_n\}$
- Un ensemble d'arêtes L . $L = \{L_1, L_2, \dots, L_p\}$

- **Exemple**

Voici un graphe G

Le graphe G peut-être défini par son nombre S de sommets et son nombre L d'arêtes : $G = (S, L)$ où

$S = \{A, B, C, D, E, F\}$ et $L = \{a, b, c, d, e, f, g\}$.

$$a = (A, B)$$

$$b = (B, C)$$

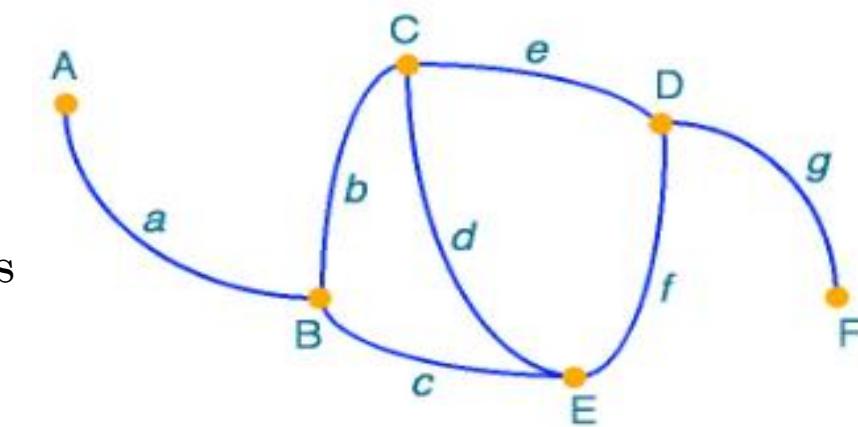
$$c = (B, E)$$

$$d = (C, E)$$

$$e = (C, D)$$

$$f = (E, D)$$

$$g = (D, F)$$

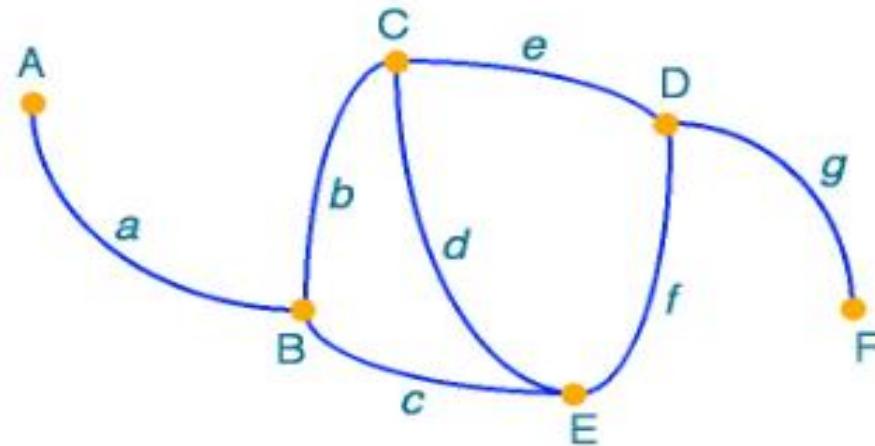


□ Représentation d'un Graphe

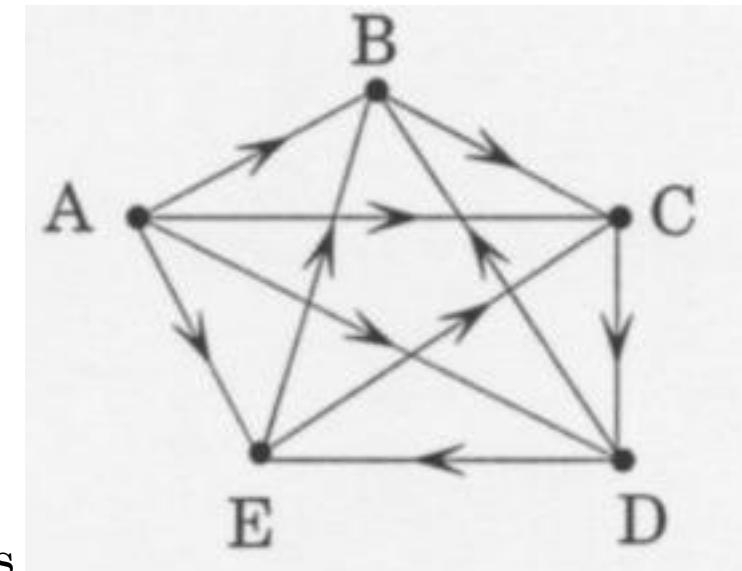
Un petit dessin vaut mieux qu'un grand discours Napoléon

- Un graphe est représenté par un schéma dans le plan où les sommets sont représentés par des points et les arcs par des branches (orientées ou non) reliant deux sommets.

■ Graphe non orienté :



■ Graphe orienté : Graphe dans lequel chacune des arêtes reliant deux sommets est orientée (à un Sens).



- des fois l'orientation n'est pas intéressante
- Il faut juste savoir quelles paires de noeuds sont connectés ou pas

□ Représentation d'un Graphe

▪ Exemples

- **En logistique** : La carte routière d'un pays : **nœuds = villes et arêtes = liens entre les villes**
→ étudier et optimiser le transport
- **En informatique** : Un parc informatique : **nœuds = ordinateurs et arêtes = liaisons entre les ordinateurs** → optimiser l'utilisation du matériel et la circulation de l'information
- **En télécom** : Un réseau téléphonique : **nœuds = postes téléphoniques et arêtes = supports de transmission** → optimiser la qualité de transmission et l'utilisation du matériel
- **En mécatronique** : Le réseau de commande d'un véhicule : **nœuds = parties opératives et de commande et arêtes = liaisons entre les différents composants** → optimiser l'utilisation de l'espace et du matériel, ainsi que la transmission de l'information

□ Représentation d'un Graphe

▪ Exercice 1:

Un site internet est composé de cinq pages notées A, B, C, D et E. En clic, on peut passer d'une page à certaines autres selon les possibilités suivantes:

- De la page A, on peut passer en un clic aux pages C et E.
- De la page B, on peut passer aux pages A et D.
- Depuis la page C, on peut accéder à la page B ou rester sur C.
- Quand on est sur la page D, on peut seulement aller sur la page C et de la page E, on ne peut aller que sur la page A.

Essayez de modéliser cette situation avec un graphe?

Exercice 2.

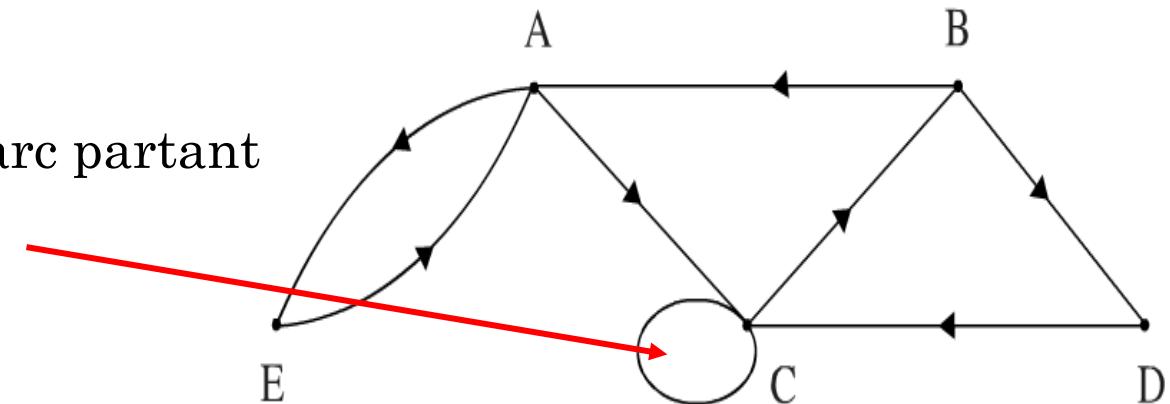
Construire un graphe orienté dont les sommets sont les entiers compris entre 1 et 6 et dont les arcs représentent la relation « être diviseur de ».

□ Représentation d'un Graphe

Exercice 1.

Une boucle d'un graphe est une arêtes ou arc partant d'un sommets et allant vers lui-même

Exercice 2 .



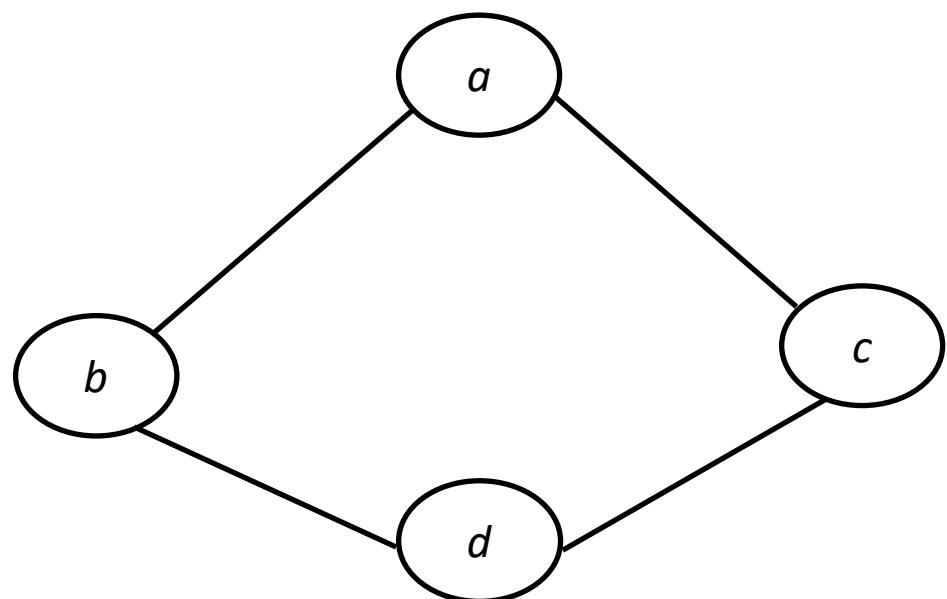
□ Définitions et notations

Etant donnée un graphe $G = (S, A)$, $x \in S$ et $y \in S$

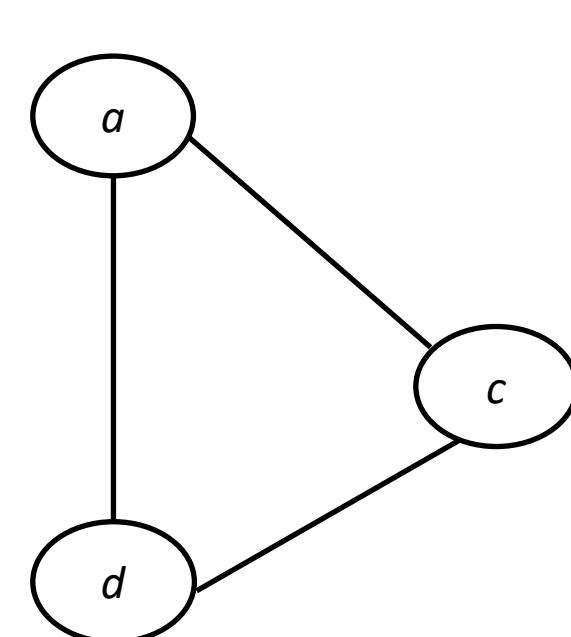
- si $(x, y) \in A$, alors y est un **successeur** de x
- si $(y, x) \in A$, alors y est un **prédecesseur** de x
- x et y sont **adjacents** si y est un **prédecesseur** et/ou un **successeur** de x
- l'**ensemble des successeurs** de x est noté $\Gamma^+(x)$ avec $\Gamma^+(x) = \{y \mid (x, y) \in A\}$
- l'**ensemble des prédecesseurs** de x est noté $\Gamma^-(x)$ avec $\Gamma^-(x) = \{y \mid (y, x) \in A\}$
- l'**ensemble des voisins** de x est noté $\Gamma(x)$ avec $\Gamma(x) = \Gamma^+(x) \cup \Gamma^-(x)$
- le **degré intérieur** de x est noté $d^-(x)$ avec $d^-(x) = |\Gamma^-(x)|$
- le **degré extérieur** de x est noté $d^+(x)$ avec $d^+(x) = |\Gamma^+(x)|$
- le **degré de x** est noté $d(x)$ avec $d(x) = |\Gamma(x)|$

□ Ordre d'un Graphe

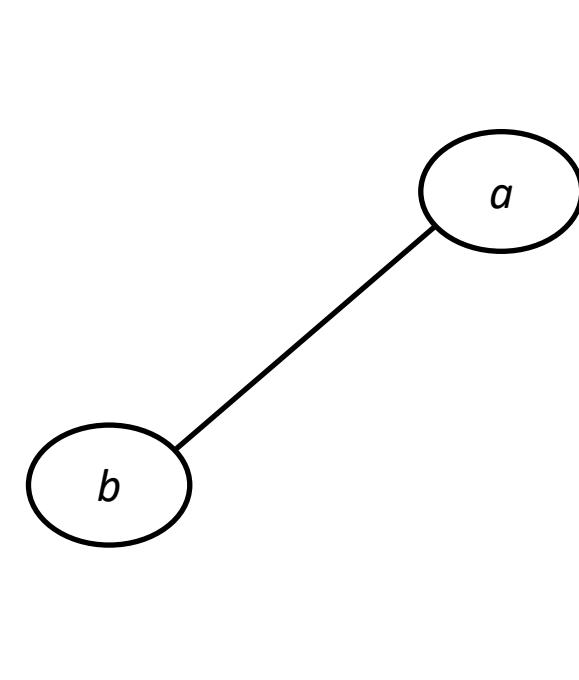
L'ordre du graphe est le nombre de sommets du graphe. On a alors : $\text{card } \{\text{Sommets}\}$



Graphe d'ordre 4



Graphe d'ordre 3



Graphe d'ordre 2

□ Degré d'un sommet

Degré d'un sommet : nombre d'arêtes reliées à ce sommet

■ Exemple

Le degré de chaque sommet de ce graphe:

$$d(1) = 3$$

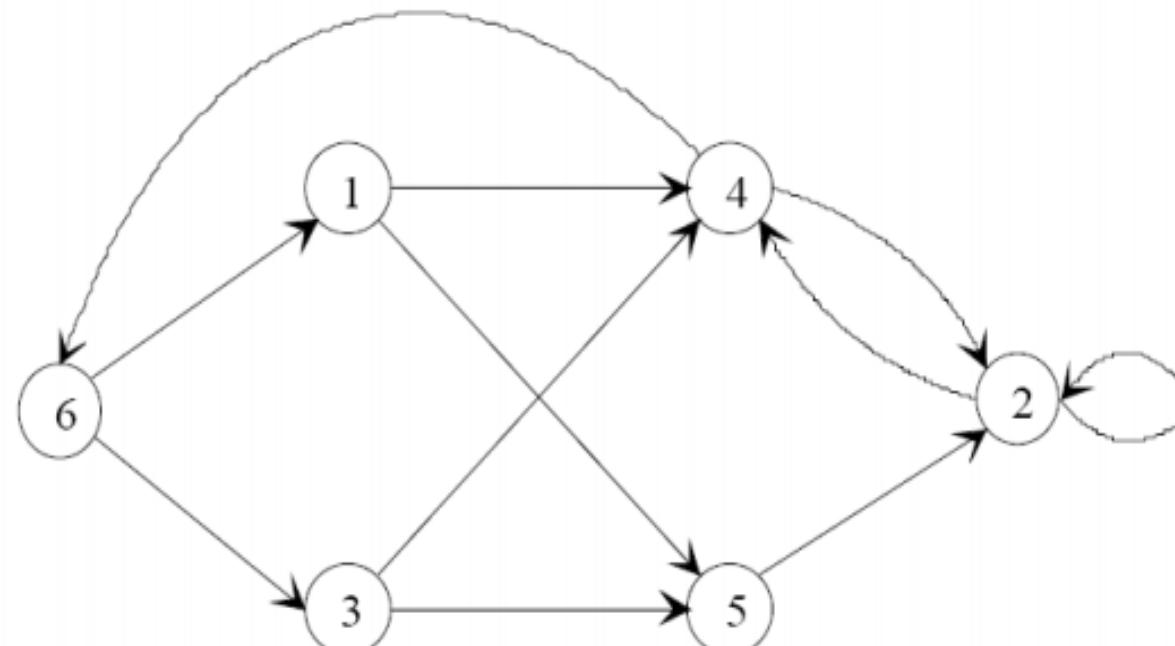
$$d(2) = 5$$

$$d(3) = 3$$

$$d(4) = 5$$

$$d(5) = 3$$

$$d(6) = 3$$



$$\Gamma+(1) = \{4, 5\}$$

$$\Gamma^-(1) = \{6\}$$

$$\Gamma(1) = \{6, 5, 4\}$$

$$d^-(1) = 1$$

$$d^+(1) = 2$$

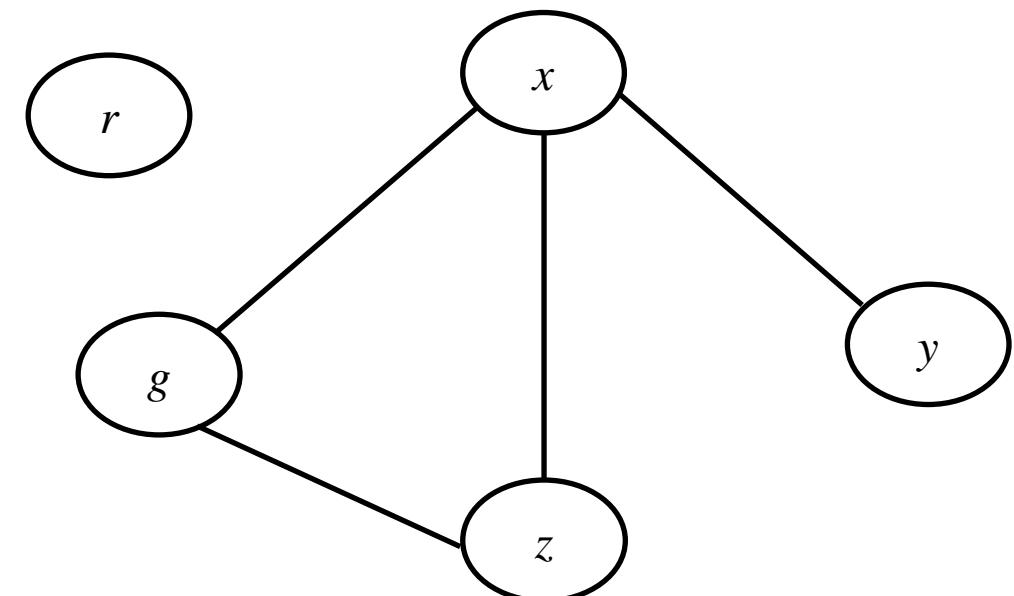
Si un sommet comporte une **boucle**, cette boucle compte double car ses deux extrémités sont incidentes au sommet.

❑ Notion d'adjacence

- On dit que deux sommets x et y sont **adjacents** si x et y sont liés par au moins une arête r .
- Pour les graphes orientés on parle aussi d'**extrémité initiale** et **extrémité finale**.

les sommets adjacents de x , y , z , g et r ?

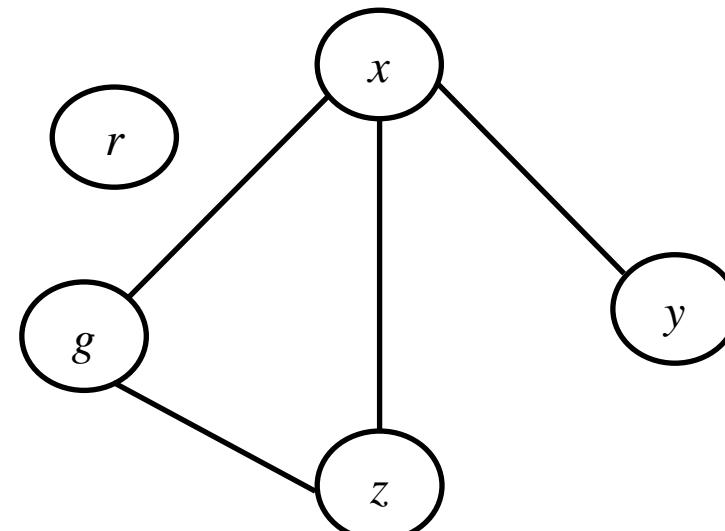
- les sommets adjacents de x : y , z , g
- les sommets adjacents de r :
- les sommets adjacents de z : x , g



□ Relations entre degrés des sommets

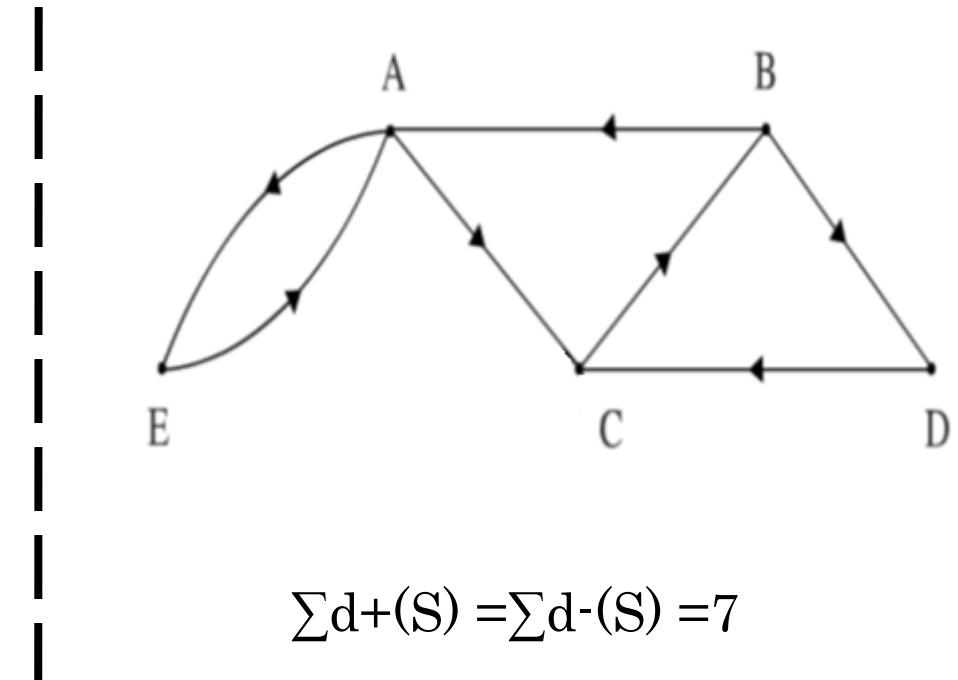
- Pour le cas des graphes non orientés, la somme des degrés des sommets d'un graphe est égale à deux fois son nombre d'arêtes. $\sum \text{degré} (\text{sommets}) = 2 * \sum \text{arêtes}$
- Pour le cas des graphes orientés sans boucles le degré intérieur des sommets =le degré extérieur des sommets =le nombre des arrêtes

▪ Exemple



$$\sum \text{degré} (\text{Sommet}) = 2 * \sum \text{arêtes}$$

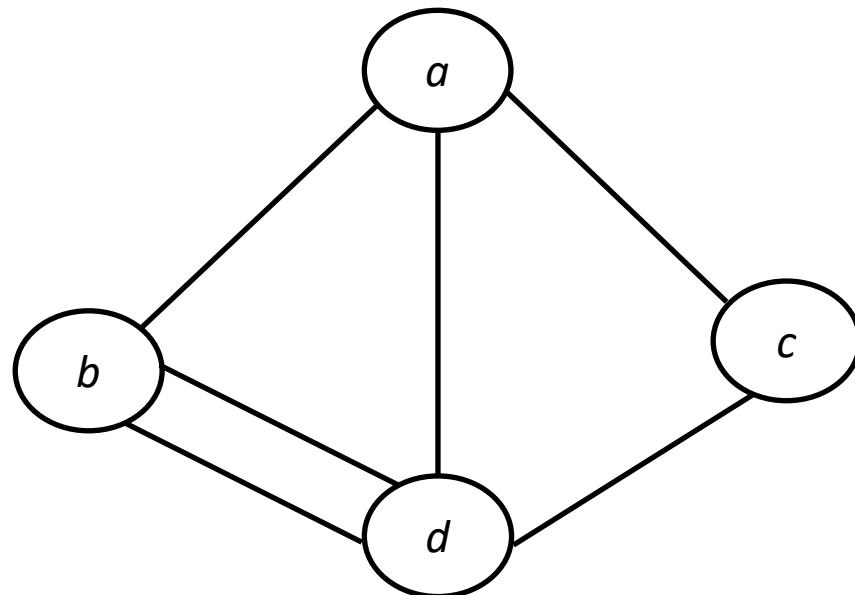
$$3+0+2+1+2 = 2 * 4$$



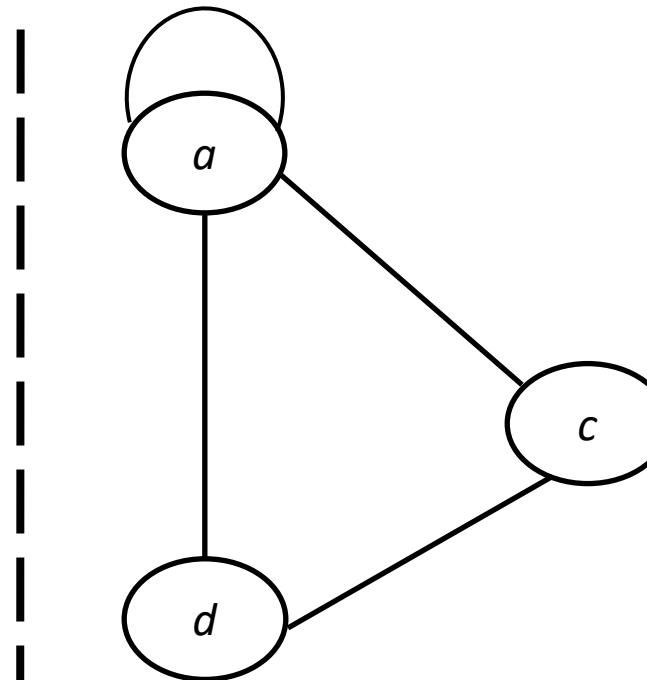
$$\sum d^+(S) = \sum d^-(S) = 7$$

□ Graphe Simple

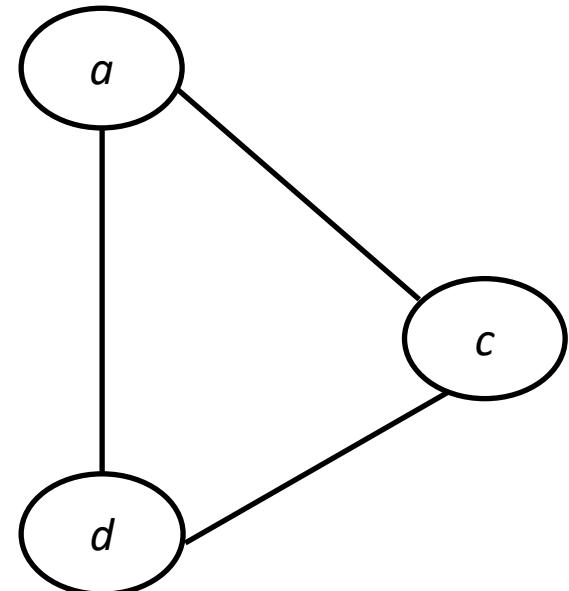
- Un graphe G est dit simple, si et seulement s'il ne contient ni **boucle** ni arêtes **parallèles**.



Ce graphe n'est pas simple car il contient
2 arêtes parallèles



Ce graphe n'est pas simple car il
contient une boucle

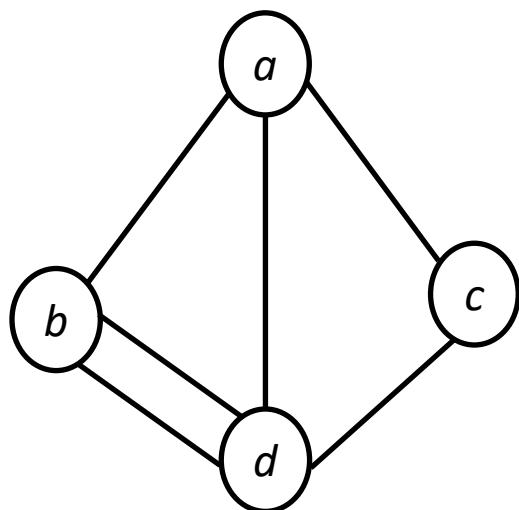


Graphe simple

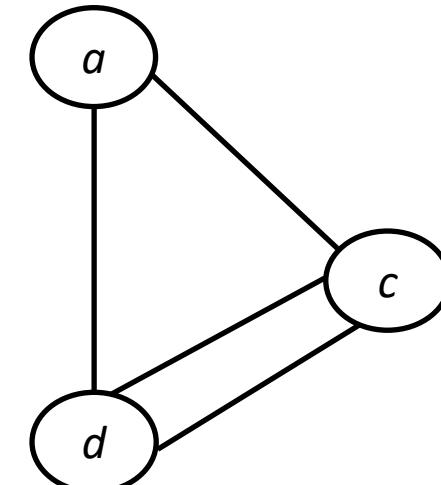
Propriété : Dans un graphe non orienté simple d'ordre n on a pour tout sommet x , $d(x) \leq (n-1)$

□ Graphe Complet

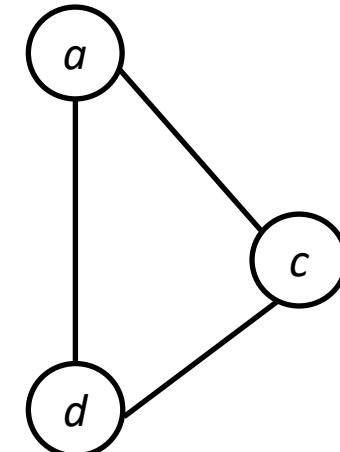
- Un graphe G est dit **complet** si et seulement s'il est **simple** et si chacun de ses sommets est lié à tous ses autres sommets.



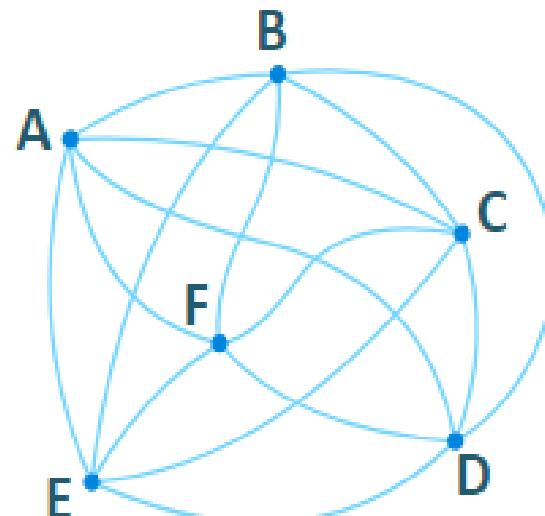
Ce graphe n'est pas complet les sommets b et c ne sont pas liés



Ce graphe n'est pas complet car il n'est pas simple



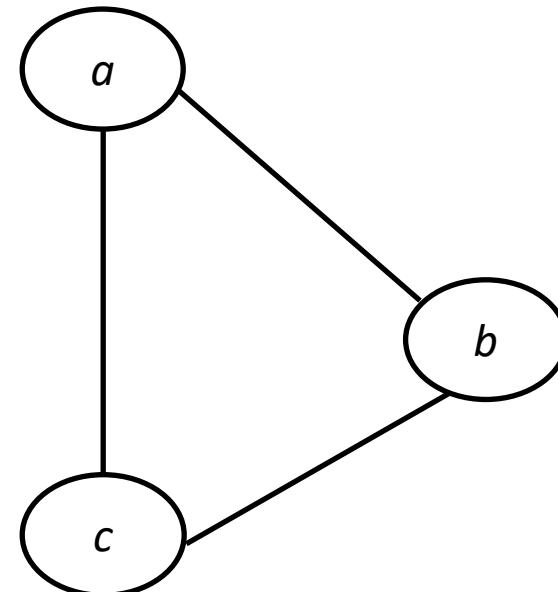
Graphe complet



???

□ Graphe Régulier

- Un graphe G est dit **régulier** si et seulement si tous ses sommets ont le même **degré**.
- **Exemple**



Degré (a) = Degré (b) = Degré (c) = 2

→ **Graphe régulier**

□ Graphe Symétrique

Cette notion est spécifique aux graphes orientés.

- $G=(X, U)$ est **symétrique** ssi $\forall x, y \in X, (x, y) \in U \Rightarrow (y, x) \in U$

□ Graphe Antisymétrique

Cette notion est spécifique aux graphes orientés.

- $G=(X, U)$ est **antisymétrique** ssi $\forall x, y \in X, (x, y) \in U \Rightarrow (y, x)$ n'appartient pas U

□ Graphe Transitif

Cette notion est spécifique aux graphes orientés.

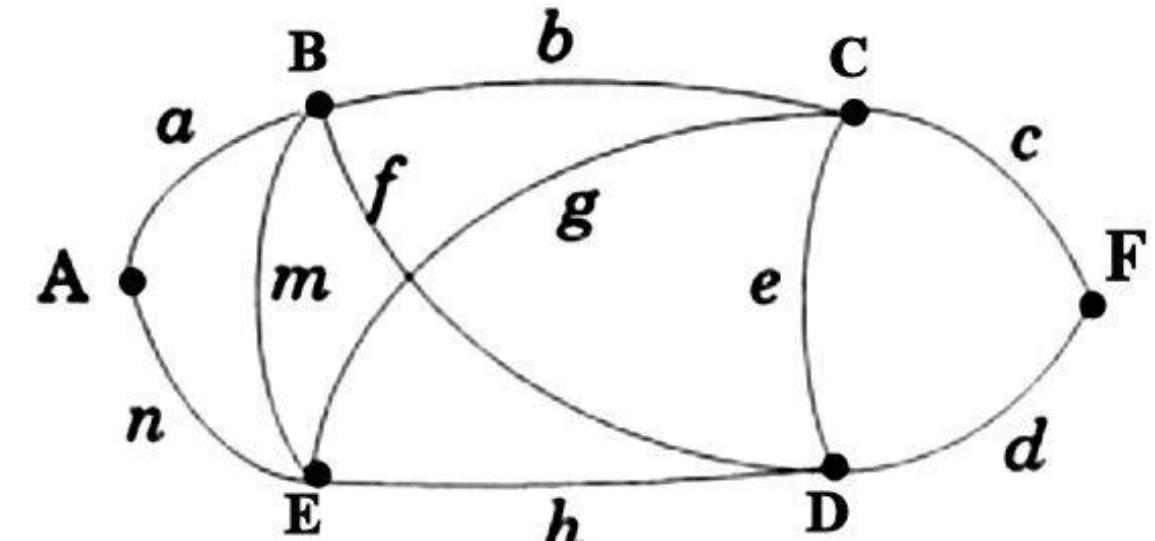
- $G=(X, U)$ est **transitif** ssi $\forall x, y, z \in X, (x, y) \in U \text{ et } (y, z) \in U \Rightarrow (x, z) \in U$

□ Sous-graphe

- Un graphe $G_1 = (S_1, L_1)$ est un *sous-graphe* du graphe $G = (S, L)$ si et seulement si $S_1 \subset S$ et $L_1 \subset L$.
- Exemple

Soit le graphe G défini par $G = (S, L)$,
où $L = \{a, b, c, d, e, g, f, g, h, m, n\}$
et $S = \{A, B, C, D, E, F\}$.

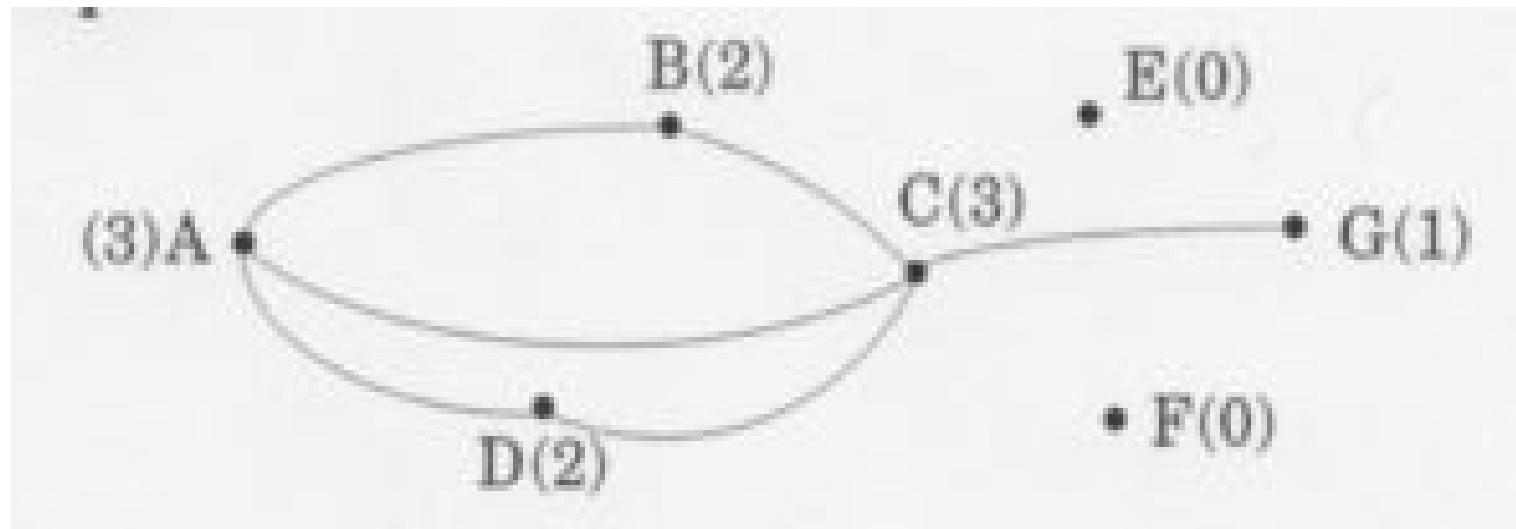
Le graphe $G_1 = (S_1, L_1)$, tel que $L_1 = \{a, b, g, n\}$
et $S_1 = \{A, B, C, E\}$, est un sous-graphe de G .



Rq: Un sous-graphe est obtenu en supprimant des sommets et les arêtes (ou arcs) qui lui sont incidentes.

□ Graphe Ouvert

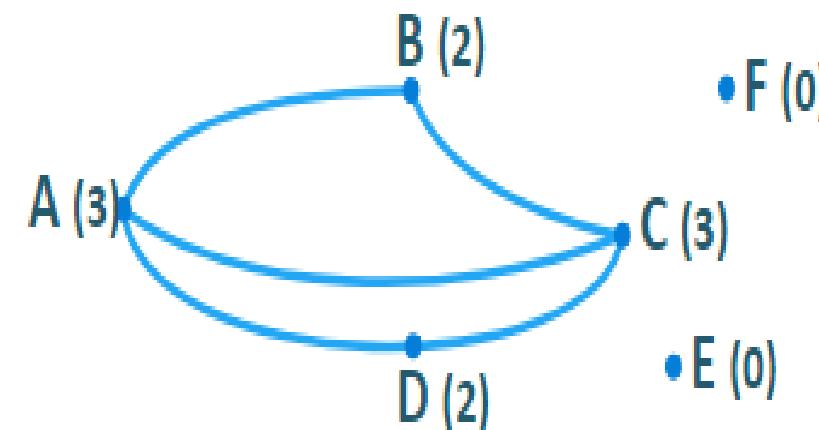
- Un graphe ouvert est un graphe dans lequel il existe au moins un sommet de degré égal à un.
- Exemple
 - Ce graphe est un graphe **ouvert**, car le sommet **G** est de degré égal à 1 :



□ Graphe fermé

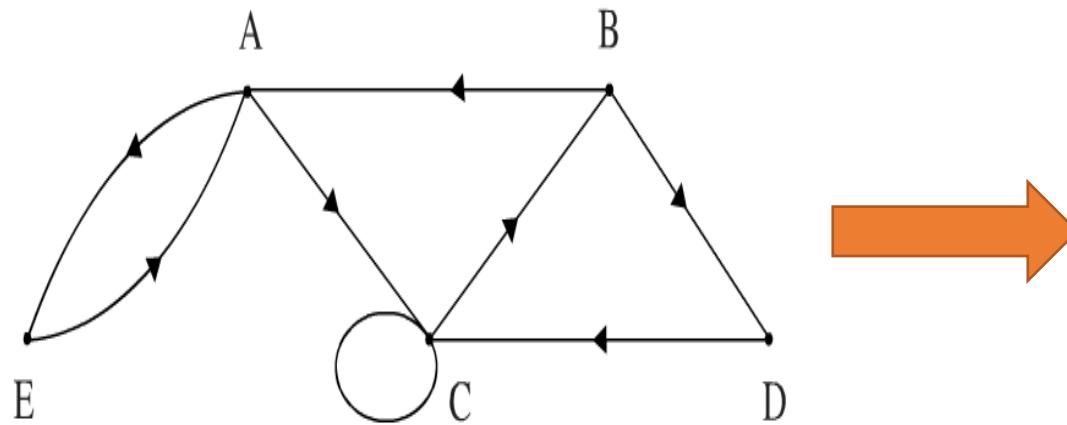
- Un graphe fermé est un graphe dans lequel tous les sommets sont, soit de **degré** supérieur ou égal à 2, soit de **degré** égal à 0.
- **Exemple**

Ce graphe est un graphe **fermé**. Le **degré** de chaque sommet est indiqué ici entre parenthèses.



□ Représentation d'un Graphe (Tableau)

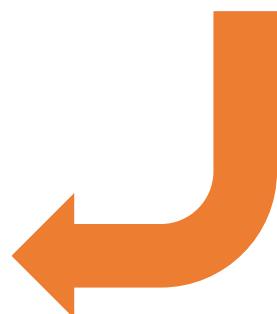
- Exemple 1:



Sommet	Successseurs
A	C, E
B	A, D
C	B, C
D	C
E	A

Sommet	Prédécesseurs
A	B, E
B	C
C	A, C, D
D	B
E	A

		Sommets d'arrivés				
		A	B	C	D	E
Sommet de départs	A	0	0	1	0	1
	B	1	0	0	1	0
	C	0	1	1	0	0
	D	0	0	1	0	0
	E	1	0	0	0	0



□ Représentation d'un Graphe (Matrice)

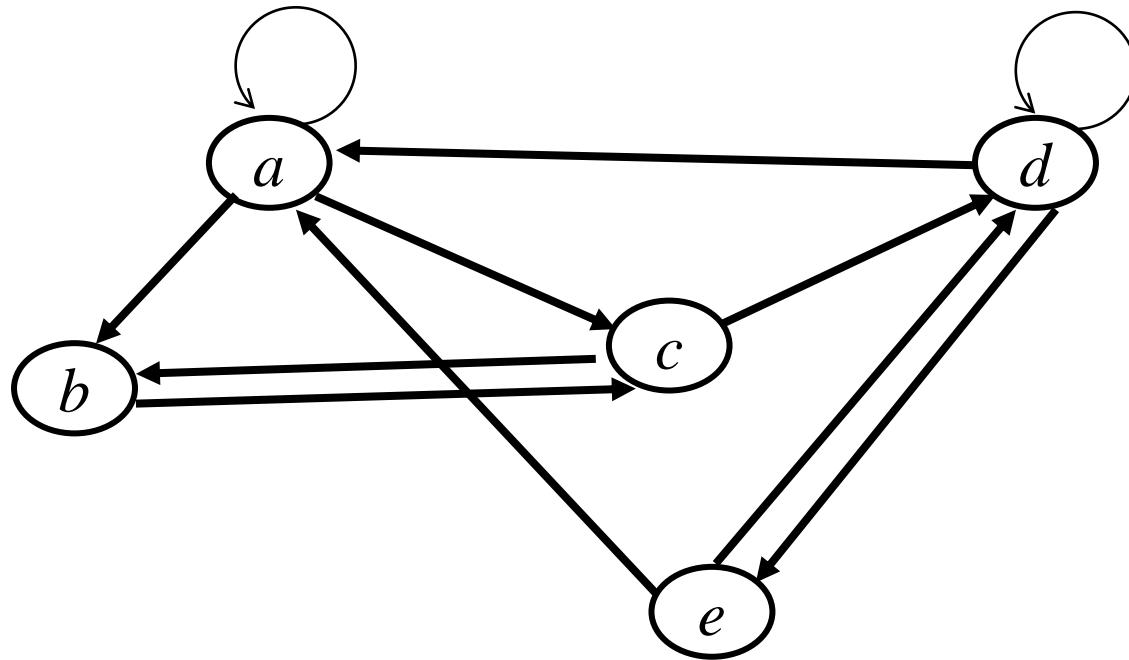
- Exemple 1:

		Sommets d'arrivés				
		A	B	C	D	E
Sommet de départs	A	0	0	1	0	1
	B	1	0	0	1	0
	C	0	1	1	0	0
	D	0	0	1	0	0
	E	1	0	0	0	0



Matrice d'adjacence

$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & \begin{pmatrix} 0 & 0 & 1 & 0 & 1 \end{pmatrix} \\ B & \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \end{pmatrix} \\ C & \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \end{pmatrix} \\ D & \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \end{pmatrix} \\ E & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

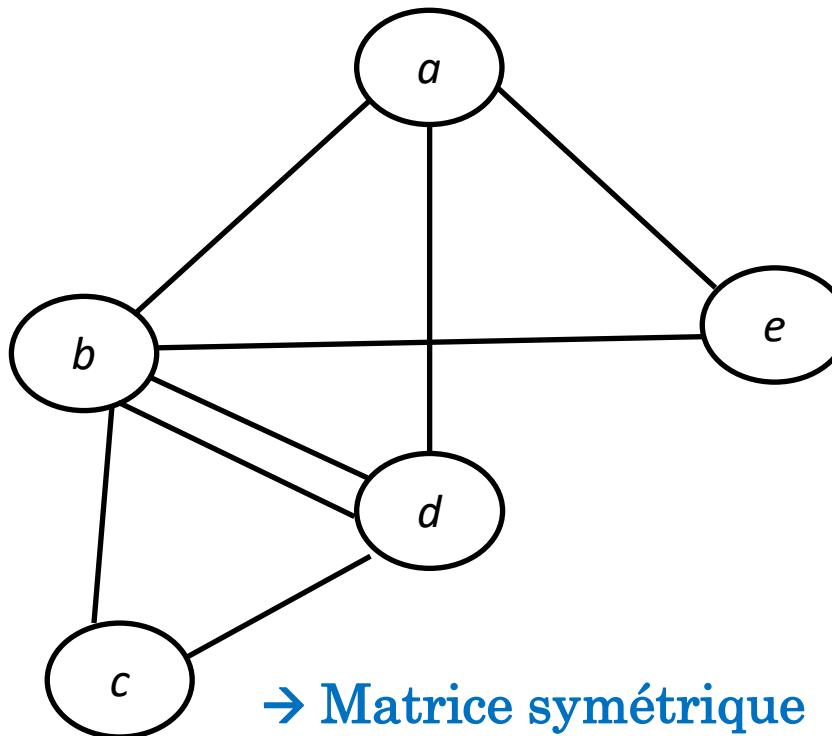
Matrice d'adjacence / Graphes orientés

Matrice
d'adjacence

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

- Matrice non symétrique
- $\sum A_{ij} = m$ avec m est le nombre des arêtes ($m = \text{Card (arêtes)}$)

Matrice d'adjacence / Graphes non orientés



→ Matrice symétrique

→ $\sum A_{ij} = 2m - N$ Avec m étant le nombre d'arêtes du graphe (y compris les boucles) et N le nombre de boucles

→ La somme des nombres d'une même ligne (ou d'une même colonne) donne le degré du sommet correspondant.

Matrice
d'adjacence

A =

	a	b	c	d	e	
a	0	1	0	1	1	3
b	1	0	1	2	1	5
c	0	1	0	1	0	2
d	1	2	1	0	0	4
e	1	1	0	0	0	2

Degré de chaque sommet

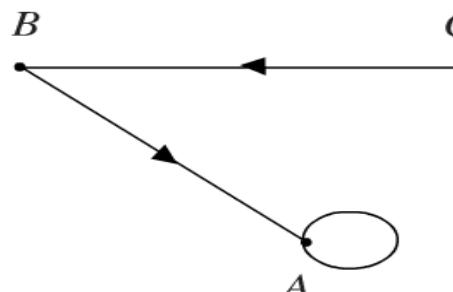
Exercice 3

1) Compléter chaque tableau en ajoutant les prédecesseurs, puis construire le graphe associé

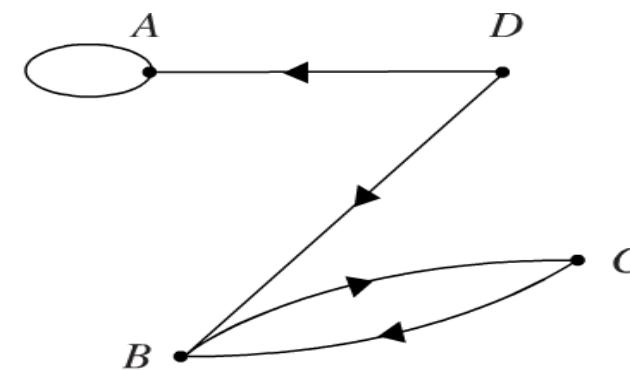
Sommet	A	B	C	D
Successeurs	B, C	B	B, D	C
Prédecesseurs				

Sommet	A	B	C	D
Successeurs	C	A,C	D	B
Prédecesseurs				

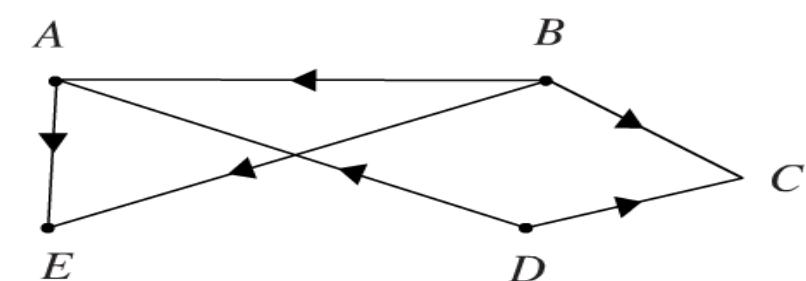
2) Pour chacun des graphes suivants, faire le tableau des successeurs et prédecesseurs puis écrire la matrice d'adjacence.



Graphe 1



Graphe 2



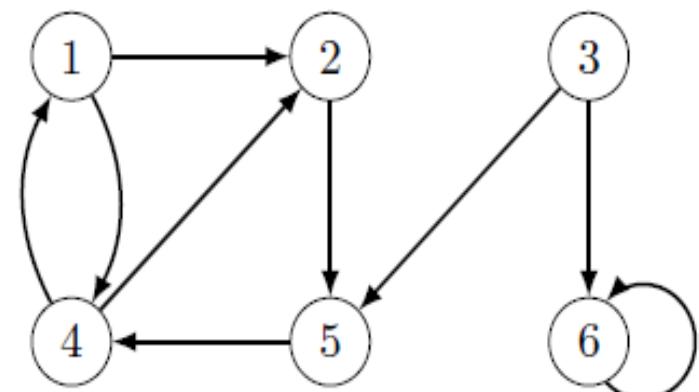
Graphe 3

Chapitre 2 :

Cheminement dans les graphes

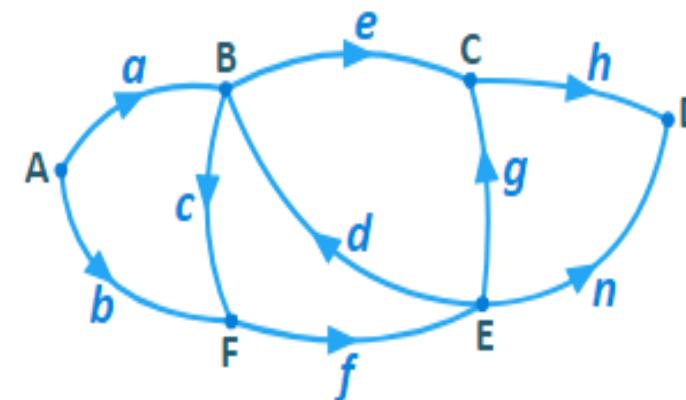
□ Chemin (cas de graphe orienté)

- Un **chemin** conduisant du sommet a au sommet b est une suite ayant pour éléments alternativement des sommets et des arcs, commençant et se terminant par un sommet, et telle que chaque arc est encadré à gauche par son sommet origine et à droite par son sommet destination.
- Un chemin est **simple** s'il ne contient pas deux fois le même arc.
- Un chemin est **élémentaire** s'il ne contient pas deux fois le même sommet.
- **Exemple**
 - Un chemin **simple** dans ce graphe est (1; 2; 5; 4)
 - Un chemin **n'est pas simple** dans ce graphe est (1; 4; 1; 4)
 - Un chemin **élémentaire** dans ce graphe est (1; 4; 2; 5)
 - Un chemin **non élémentaire** dans ce graphe est (3; 6; 6)



□ Chemin (cas de graphe orienté)

- Le nombre d'arcs d'un chemin détermine la **longueur du chemin**.
- **Exemple**
 - Dans le graphe orienté ci-dessous, le chemin formé dans l'ordre par les arcs a, c, f, d, c, f, g , et h est un **chemin de longueur 8**.



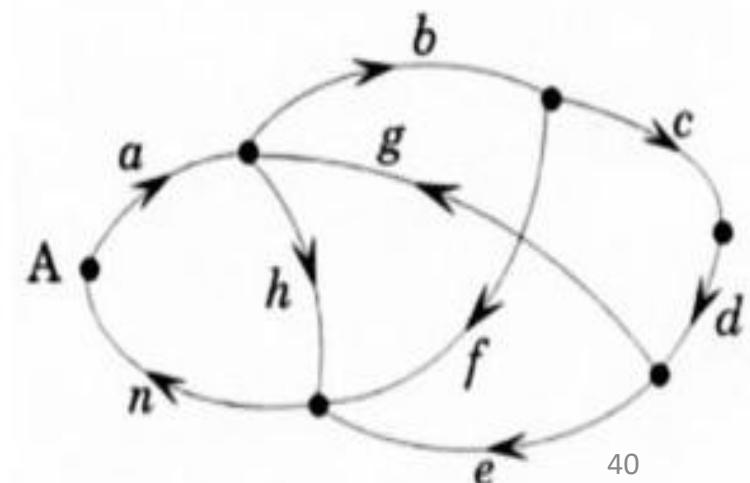
- Dans le graphe orienté ci-dessus, est ce que le chemin, constitué dans l'ordre des arcs a, c, f, d, e et h est un **chemin simple** ? Un **chemin élémentaire** ?

❑ Circuit (Cas de Graphe Orienté)

- **Circuit** est un chemin dont les sommets de départ et de fin sont les mêmes.
- La longueur d'un circuit est **le nombre d'arcs** qui constituent ce circuit.
- **Circuit Simple** qui n'utilise pas deux fois le même arc.
- **Circuit élémentaire** qui ne passe pas deux fois par le même sommet.
- **Exemple**

Dans ce graphe orienté, le chemin constitué, dans l'ordre, des arcs:

- *a, b, c, d, e* et *n* est un **circuit**. La longueur de ce circuit est 6.
- *a, b, c, d, e* et *n* est un **circuit Simple**.
- *a, b, c, d, g, b, f* et *n* est un **circuit non simple**.
- *a, b, c, d, e* et *n* est un **circuit élémentaire**.

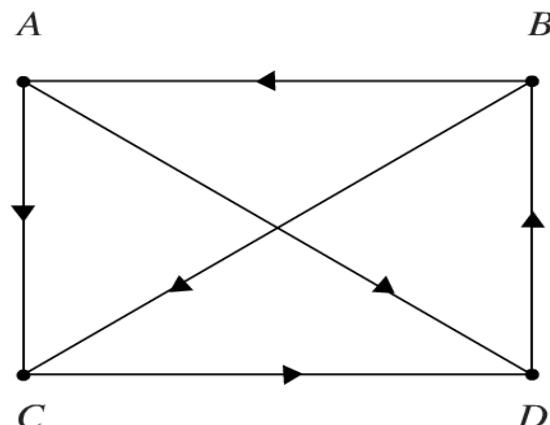


□ Nombre de chemins de longueur donnée

- Propriété :

Soit M la matrice d'adjacence d'un graphe orienté à n sommets a_1, a_2, \dots, a_n . Soit p un entier et $M^p = (m_{i,j})$ la puissance d'exposant p de la matrice M . Alors $m_{i,j}$ est le nombre de chemins de longueur p allant du sommet a_i au sommet a_j

- Exemple:



M est la matrice
d'adjacence de ce graphe



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

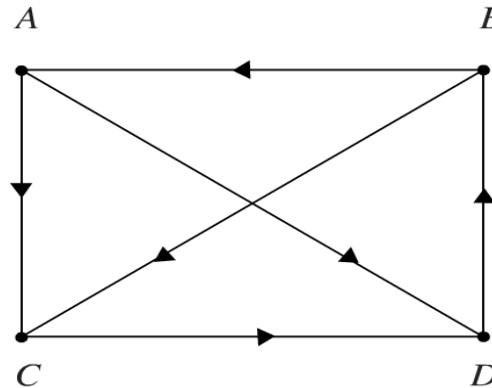
Prenons $p = 2$



$$M^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

□ Nombre de chemins de longueur donnée

- Exemple:



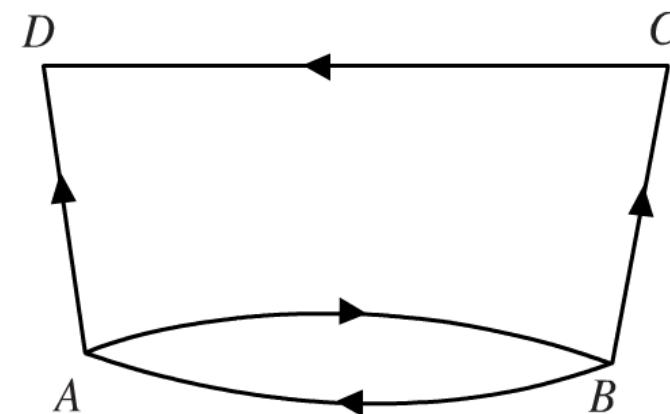
$$M^2 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

- Considérons la deuxième ligne de la matrice M^2 :

- le troisième coefficient est 1, donc il y a un chemin **de longueur 2** qui relie B à C : **le chemin (B, A, C)**.
- le dernier coefficient est 2, attestant l'existence de deux chemins **de longueur 2** reliant B à D : **les chemins (B, A, D) et (B, C, D)**.

Exercice 4

- a) Ecrire la matrice d'adjacence M du graphe.
- b) Calculer M^2 . Expliquer la signification des quatre nombres de la deuxième et troisième ligne de la matrice M^2 ?
- c) Combien y a-t-il de chemins de longueur 2 dans ce graphe ? les citer?

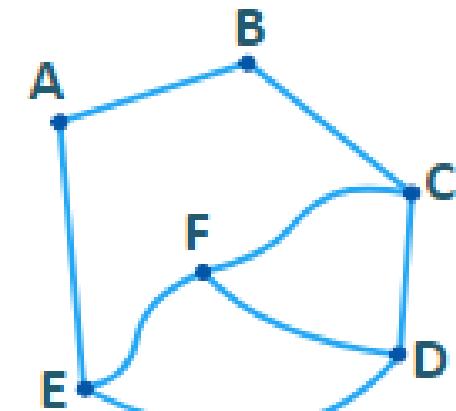


❑ Chaine (Cas de Graphe non Orienté)

- Une chaine est une suite des sommets qui sont reliés par des **arêtes**
- La longueur d'une chaine est égale le nombre d'arêtes de cette chaine
- **Chaine simple** qui n'utilise pas deux fois la même **arête**.
- **Chaine élémentaire** qui ne passe pas deux fois par le même **sommet**.
- **Exemple**

Dans ce graphe non orienté, la chaine constituée, dans l'ordre, des **arêtes** :

- A-E-F-C-B-A est une **chaine simple** de longueur est 5.
- A-B-C-F-D est une **chaine élémentaire**.

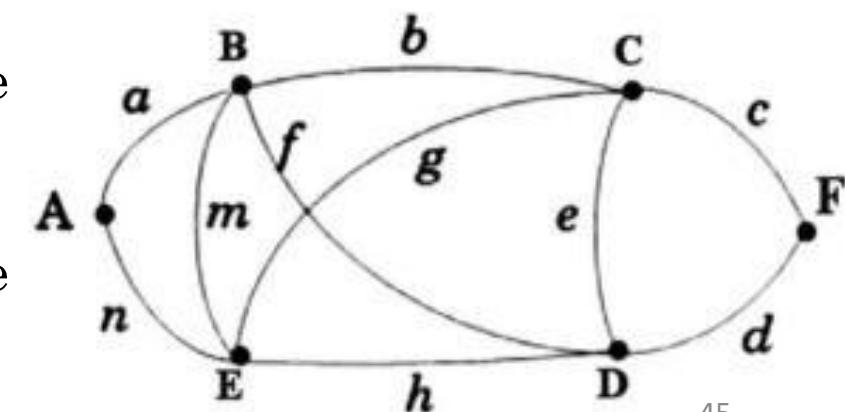


❑ Cycle (Cas de Graphe non Orienté)

- Dans un graphe non orienté, un **cycle** est une chaîne qui commence et se termine au même sommet.
- La longueur d'un cycle est le nombre d'arêtes qui constituent ce cycle.
- **Cycle simple** qui n'utilise pas deux fois la même **arête**.
- **Cycle élémentaire** qui ne passe pas deux fois par le même **sommet**.
- **Exemple**

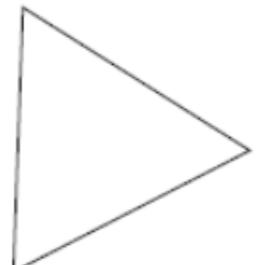
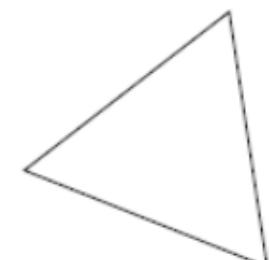
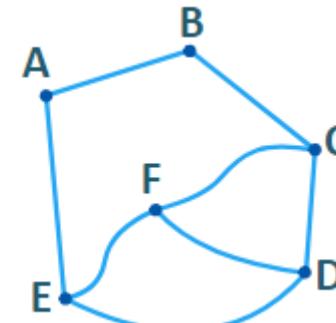
Dans ce graphe non orienté, le cycle constituée, dans l'ordre, des **arêtes** :

- **a, b, c, d, f, m et n** est un **cycle simple** qui commence et se termine au sommet A.
- **a, b, e, h et n** est un **cycle élémentaire** qui commence et se termine au sommet A.



□ Graphe Connexe

- Graphe dans lequel on peut relier, directement ou non, n'importe quel sommet à n'importe quel autre sommet du graphe par une chaîne d'arêtes.
- La connexité est **un concept non orienté**.
- **Exemple :**
 - Le graphe ci-dessous est un **graphe connexe** :
 - À partir de chacun des sommets de ce graphe, on peut se rendre à n'importe quel autre sommet de ce graphe.
- Si G n'est pas connexe, les sous-graphes connexes maximaux au sens de l'inclusion sont appelés **composantes connexes**.



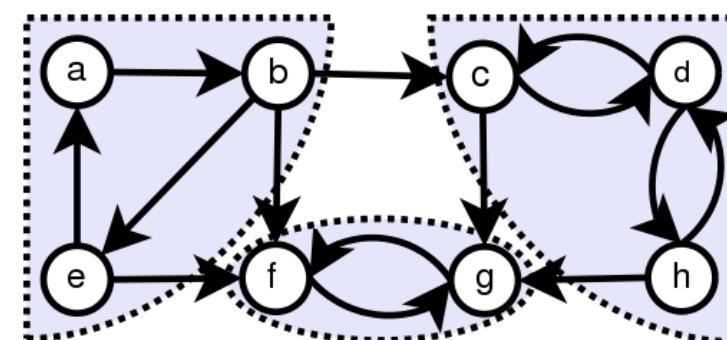
Non connexe (2 composantes connexes)

❑ Forte connexité:

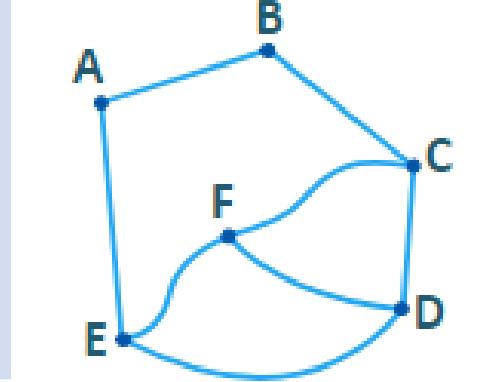
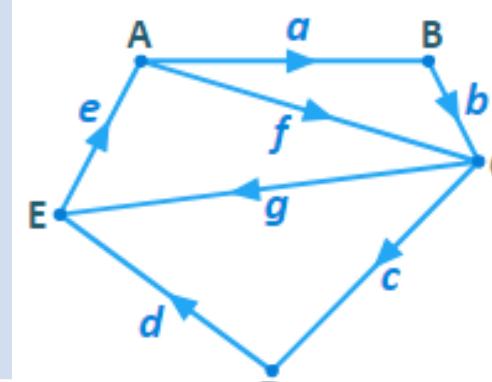
- La forte connexité est une propriété plus forte qui concerne **les graphes orientés**
- $G = (X, U)$ est fortement connexe si pour toute paire de sommets (i, j) il existe un chemin de i à j et un autre de j à i .
- Si G n'est pas fortement connexe, ses sous-graphes fortement connexes maximaux sont appelés **composantes fortement connexes (CFC)**.
- Désirable pour les graphes de systèmes dynamiques, sinon il peut exister des états inaccessibles ou le système peut se bloquer dans des un sous-ensemble d'états.
- **Exemple :**

Le graphe suivant n'est pas fortement connexe

Décomposition en composantes fortement connexes.



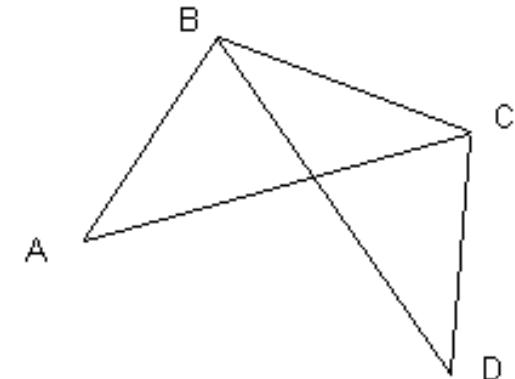
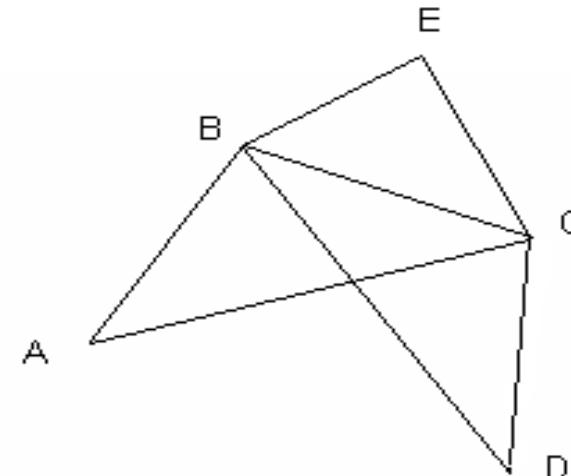
❑ Récapitulatif du vocabulaire (Orienté - non Orienté)

Graphe non Orienté	Graphe Orienté
	
Sommet	Sommet
Arête	Arc
Adjacent	Successeur, Prédécesseur
Chaîne	Chemin
Cycle	Circuit
Connexité	Forte connexité

□ Chaine et Cycle Eulérien

- **Chaine Eulérienne** : Chaine simple qui passe par toutes les arêtes d'un **graphe non orienté**.
- **Cycle Eulérien** : Cycle simple qui passe par toutes les arêtes d'un **graphe non orienté**.
- **Un graphe eulérien** est un graphe possédant un cycle eulérien.
- **Exemple 1**
 - Dans ce **graphe non orienté**, (BACBDC) est une chaîne Eulérienne
- **Exemple 2**

Existe-t-il un cycle eulérien ??



CDABCABEC est un cycle eulérien

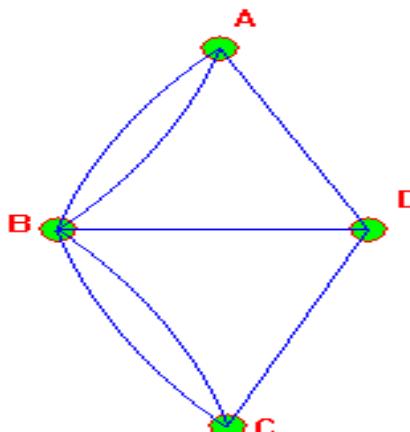
□ Chaine et Cycle Eulérien

■ Théorème

Soit $G=(V,E)$ un graphe **non orienté connexe**.

- Le graphe G admet un **cycle eulérien** si et seulement si tous ses sommets ont un degré pair.
- Le graphe G admet une **chaîne eulérienne** qui n'est pas un cycle si et seulement si tous ses sommets ont un degré pair sauf exactement deux d'entre eux. Ces deux sommets particuliers seront les extrémités de cette chaîne.

Exemple: Problème des sept ponts de Königsberg



$$d(A)=3$$

$$d(B)=5$$

$$d(C)=3$$

$$d(D)=3$$

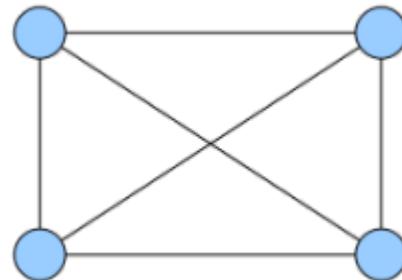
- Les 4 sommets de ce graphe ont des degrés impairs, il n'existe donc **ni cycle eulérien, ni chaîne eulérienne**.
- Le problème des ponts de Königsberg n'a donc pas de solution.

□ Chaine et Cycle Eulérien

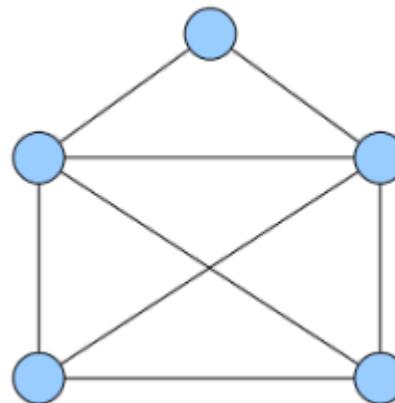
Exemple 2 : Problème du dessin d'enveloppes

La question est simple : peut-on dessiner une enveloppe ouverte (*resp.* fermée) sans lever le crayon et sans repasser sur une ligne déjà dessinée ?

Cela revient à se demander si l'on peut trouver **une chaîne eulérienne** dans les graphes suivants :



Dans ce cas la réponse est non car tous les sommets ont un degré impair.



Dans le second cas, deux sommets exactement ont un degré impair, il existe donc une chaîne eulérienne ayant ces deux sommets pour extrémités

- Pour des graphes plus complexes que les précédents, il va nous falloir des algorithmes afin d'expliciter les éventuels parcours eulériens.
- Commençons par le cas des cycles eulériens.

□ Chaine et Cycle Eulérien

Algorithme de détermination d'un cycle eulérien

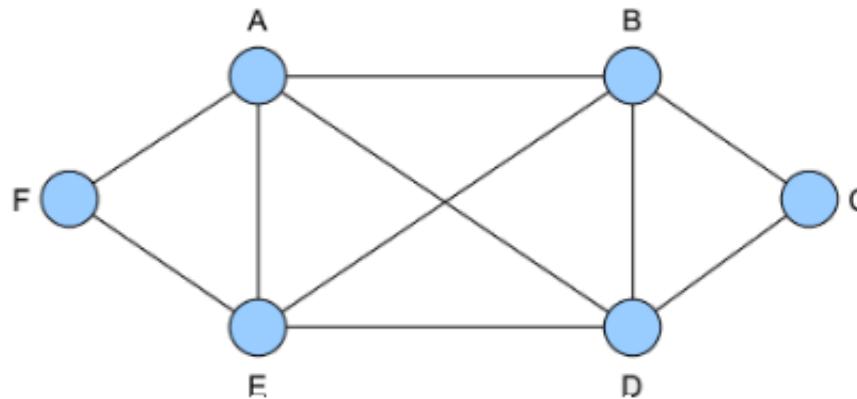
Soit $G=(V,E)$ un graphe **non orienté connexe** possédant un **cycle eulérien**.

1. Choisir un sommet arbitrairement.
2. Construire un cycle simple (quel qu'il soit) ayant pour origine et extrémité ce sommet (c'est toujours possible).
3. Si ce cycle est eulérien, la recherche est terminée.
4. Sinon, considérer le sous-graphe de G obtenu en supprimant les arêtes du cycle précédent.
5. À partir d'un sommet commun au sous-graphe restant et au cycle, construire de nouveau un cycle simple.
6. Insérer ce second cycle dans le premier.
7. Recommencer à partir de l'étape 3.

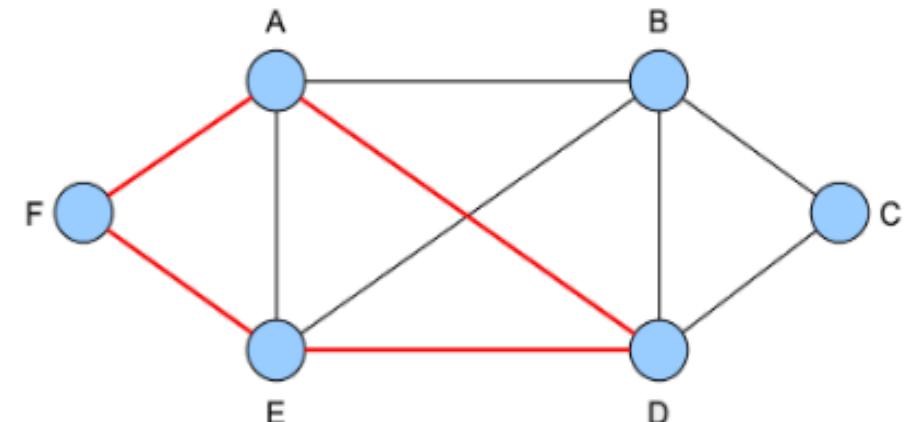
□ Chaine et Cycle Eulérien

Exemple : Détermination d'un cycle eulérien

Considérons le graphe non orienté dont la représentation sagittale est la suivante :



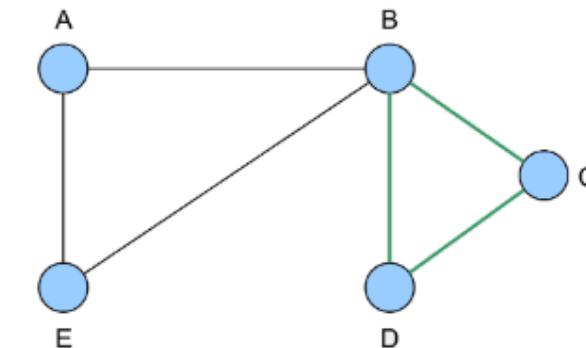
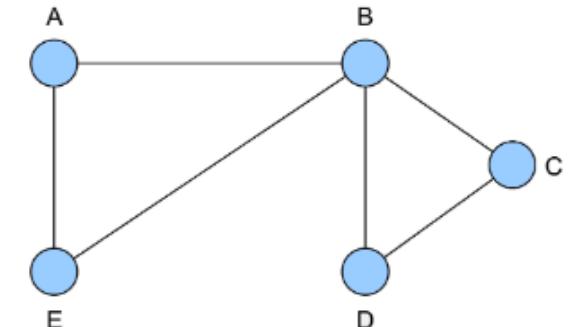
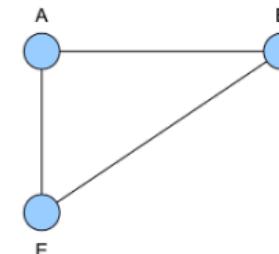
- Ce graphe est **eulérien** car il est connexe, et tous les degrés de ses sommets sont pairs.
- On commence donc par choisir un sommet, par exemple A. On construit alors un cycle simple d'origine et extrémité A, comme (A,D,E,F,A) , représenté en rouge ici :



□ Chaine et Cycle Eulérien

Exemple : Détermination d'un cycle eulérien

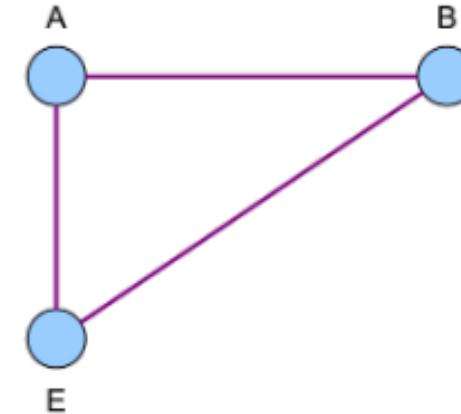
- On considère ensuite le sous-graphe de G obtenu en supprimant ce cycle :
- A partir du sommet D, on peut construire un second cycle simple, par exemple (D,C,B,D), ici en vert :
- On supprime ce nouveau cycle :



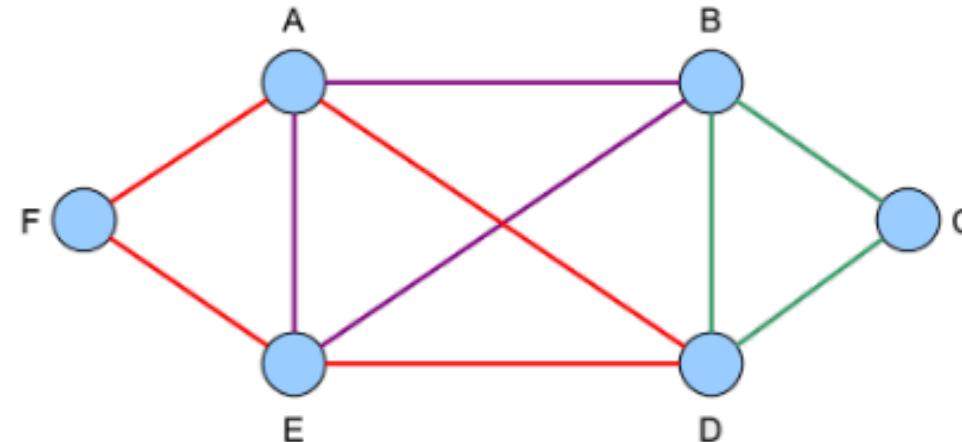
Chaine et Cycle Eulérien

Exemple : Détermination d'un cycle eulérien

- La construction du prochain (et dernier) cycle simple est triviale, il s'agit de (E,A,B,E), mis en évidence en mauve :



- Il ne reste plus qu'à insérer les deux derniers cycles dans le premier, il vient (A,D,C,B,D,E,A,B,E,F,A) :



□ Chaine et Cycle Eulérien

Algorithme de détermination d'une chaîne eulérienne

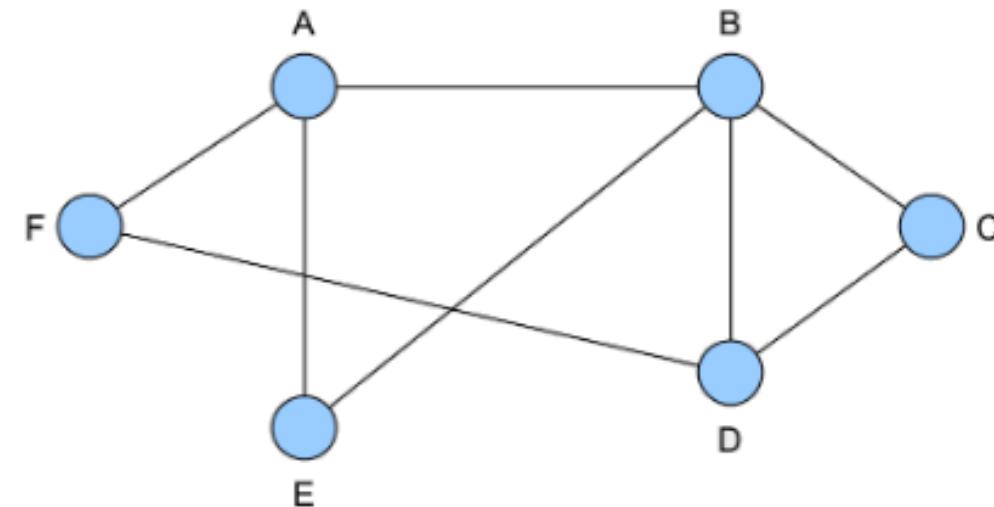
Soit $G=(V,E)$ un graphe **non orienté connexe** possédant une **chaîne eulérienne**.

1. Choisir arbitrairement l'un des deux sommets de degré impair.
2. Construire une chaîne simple (quelle qu'elle soit) ayant pour origine ce sommet et pour extrémité l'autre sommet de degré impair (c'est toujours possible).
3. Si cette chaîne est eulérienne, la recherche est terminée.
4. Sinon, considérer le sous-graphe de G obtenu en supprimant les arêtes de la chaîne précédente.
5. À partir d'un sommet commun au sous-graphe restant et à la chaîne, construire un cycle simple.
6. Insérer ce cycle dans la chaîne.
7. Recommencer à partir de l'étape 3.

□ Chaine et Cycle Eulérien

Exemple : Détermination d'une chaîne eulérienne

Considérons le graphe non orienté dont la représentation sagittale est la suivante :

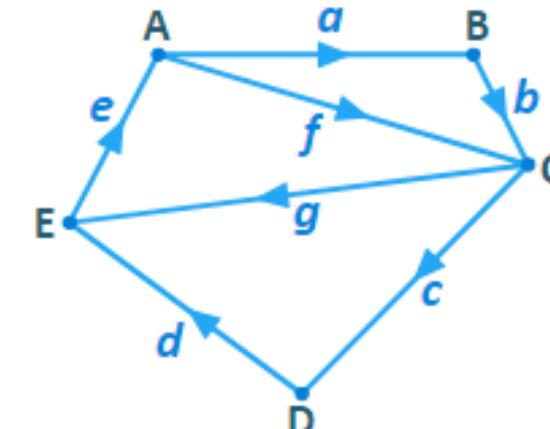


Appliquer l'algorithme précédent pour trouver une chaîne eulérienne pour :

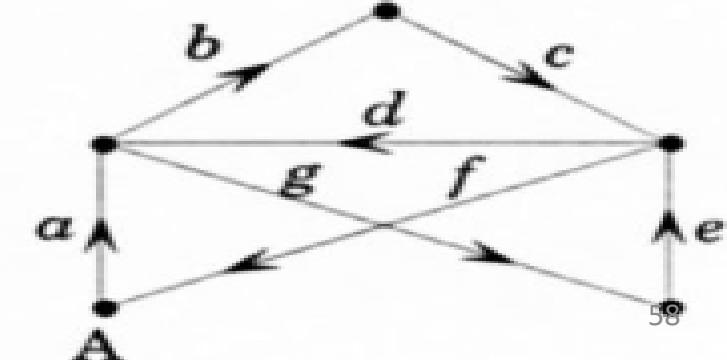
□ Chemin et Circuit Eulérien

- **Chemin Eulérien:** Chemin simple qui passe par tous les arcs d'un **graphe orienté**
- **Circuit Eulérien:** Circuit simple qui passe par tous les arcs d'un **graphe orienté**.
- Un **graphe eulérien** est un graphe possédant un circuit eulérien.
- **Exemple**

➤ Dans ce **graphe orienté**, le chemin formé des arcs (a, b ,c , d , e , f , g) est un **chemin eulérien** de longueur 7.



➤ Dans ce **graphe orienté**, le circuit constitué, dans l'ordre, des arcs (a, b, c, d, g, e , f) est un **circuit eulérien**



❑ Chemin et Circuit Eulérien

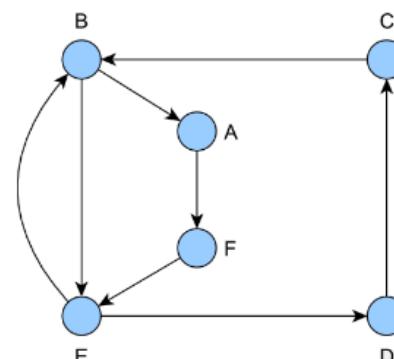
Théorème

Soit $G=(V,E)$ un graphe orienté connexe.

- Le graphe G admet un **circuit eulérien** si et seulement si **tous ses sommets ont un degré entrant égal à leur degré sortant**.
- Le graphe G admet une **chemin eulérien** qui n'est pas un circuit si et seulement si **tous ses sommets ont un degré entrant égal à leur degré sortant, sauf deux d'entre eux x et y qui vérifieront $d+(x)=d-(x)+1$ et $d+(y)=d-(y)-1$**
- Ce chemin aura pour origine x et pour extrémité y .

■ Exemple : Un graphe eulérien

Ce graphe est **eulérien** car il est connexe, et tous ses sommets ont un degré entrant égal à leur degré sortant.



❑ Chemin et Circuit Eulérien

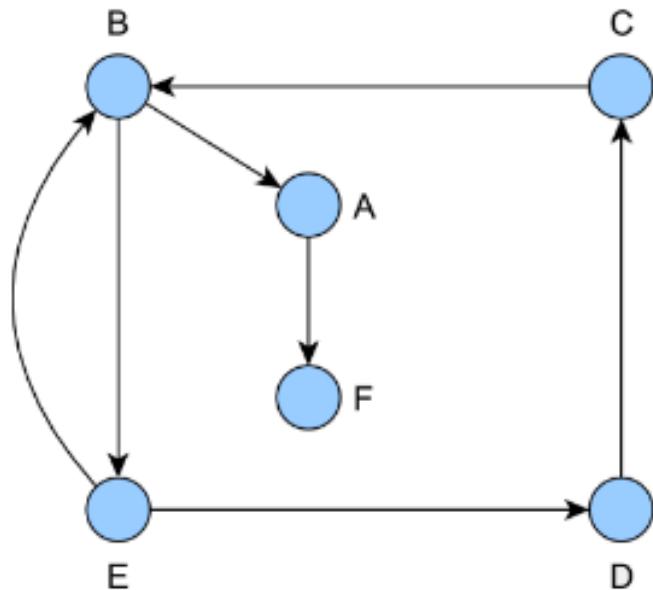
Exemple: Un graphe possédant un chemin eulérien :

Ce graphe est **eulérien** car il est connexe, et tous ses sommets ont un degré entrant égal à leur degré sortant à part les sommets E et F qui vérifient les conditions du théorème.

$$d+(E)=2, d-(E)=1, d+(F)=0, d-(F)=1$$

$$\text{Donc } d+(E)=d-(E)+1 \text{ et } d+(F)=d-(F)-1$$

Il existe donc un chemin eulérien d'origine E et d'extrémité F.



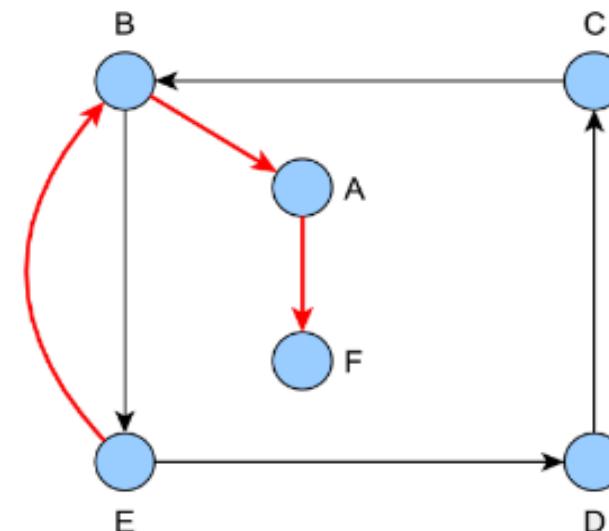
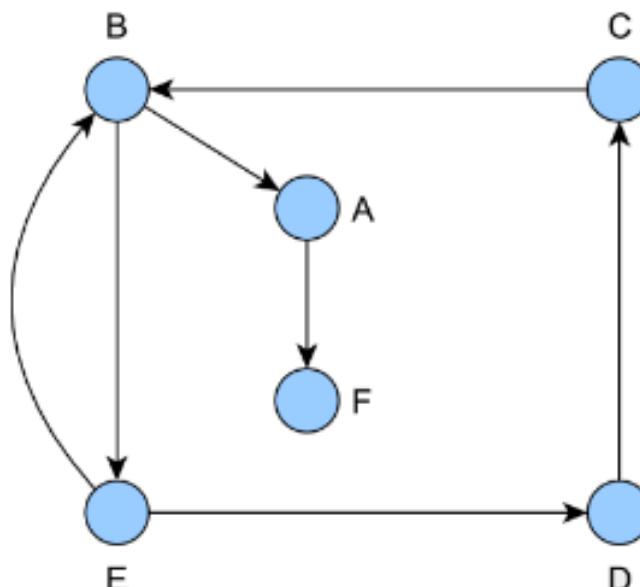
- Les méthodes pour déterminer un chemin ou un circuit eulérien sont **exactement les mêmes que dans le cas non orienté**, en tenant bien sûr compte de l'orientation.
- Il faut juste bien penser que dans la recherche d'un chemin eulérien, l'origine du chemin doit être le sommet dont le degré sortant est supérieur au degré entrant.

❑ Chemin et Circuit Eulérien

Exemple : Détermination d'un chemin eulérien

Reprendons le graphe de l'exemple précédent :

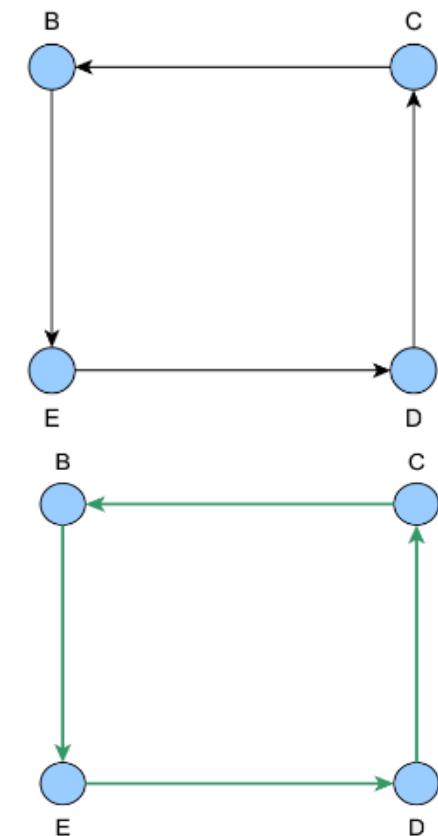
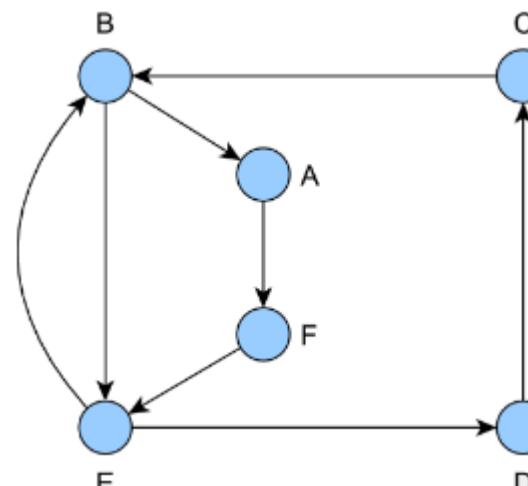
- Nous avions établi qu'il existe un chemin eulérien d'origine E et d'extrémité F.
- Pour en déterminer un, on commence par construire un chemin simple d'origine E et d'extrémité F, comme (E,B,A,F), représenté en rouge ici :



❑ Chemin et Circuit Eulérien

Exemple : Détermination d'un chemin eulérien

- On considère ensuite le sous-graphe de G obtenu en supprimant ce chemin :
- La construction du (seul) circuit simple est triviale, il s'agit de (B,E,D,C,B) ,
ici en vert :
- Il ne reste plus qu'à insérer ce dernier circuit
dans le chemin, il vient (E,B,E,D,C,B,A,F) :

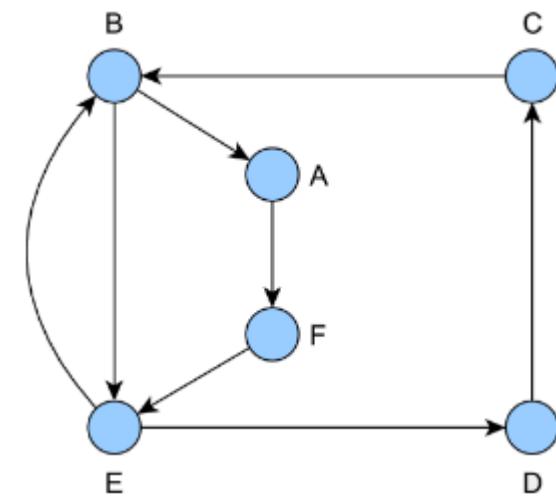


❑ Chemin et Circuit Eulérien

Exemple : Détermination d'un circuit eulérien

Reprendons le graphe de l'exemple précédent :

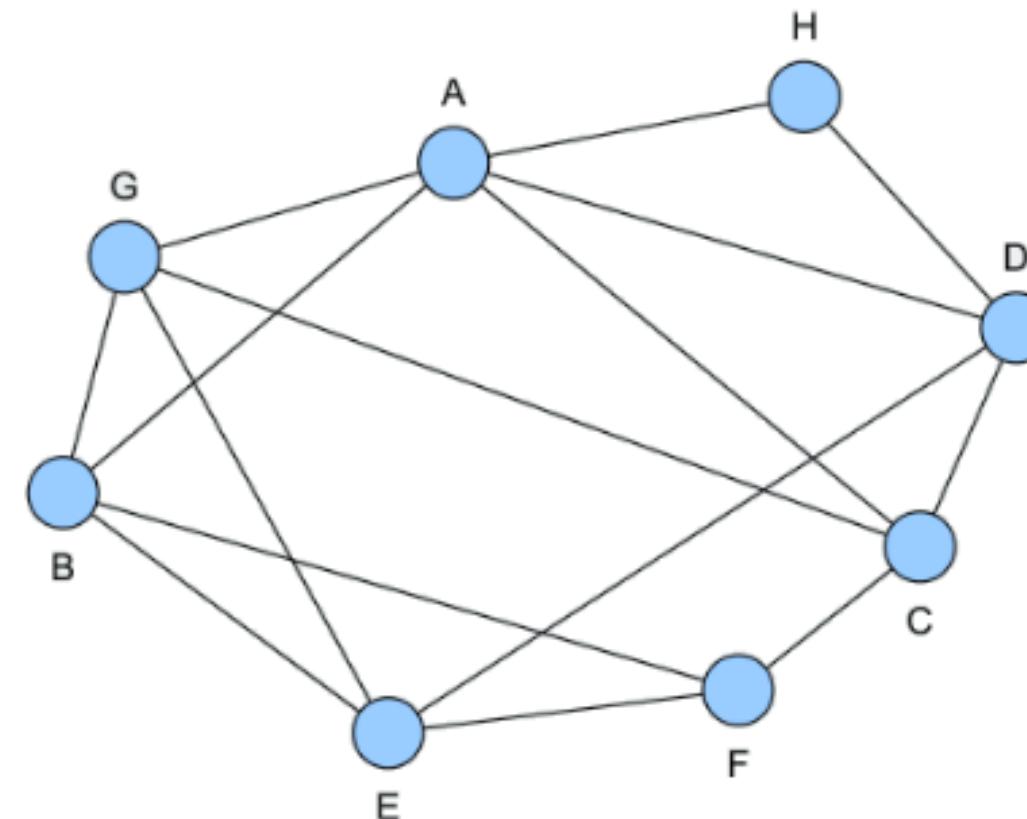
Ce graphe est **eulérien** car il est connexe, et tous ses sommets ont un degré entrant égal à leur degré sortant.



Chemin et Circuit Eulérien

Exercice 5:

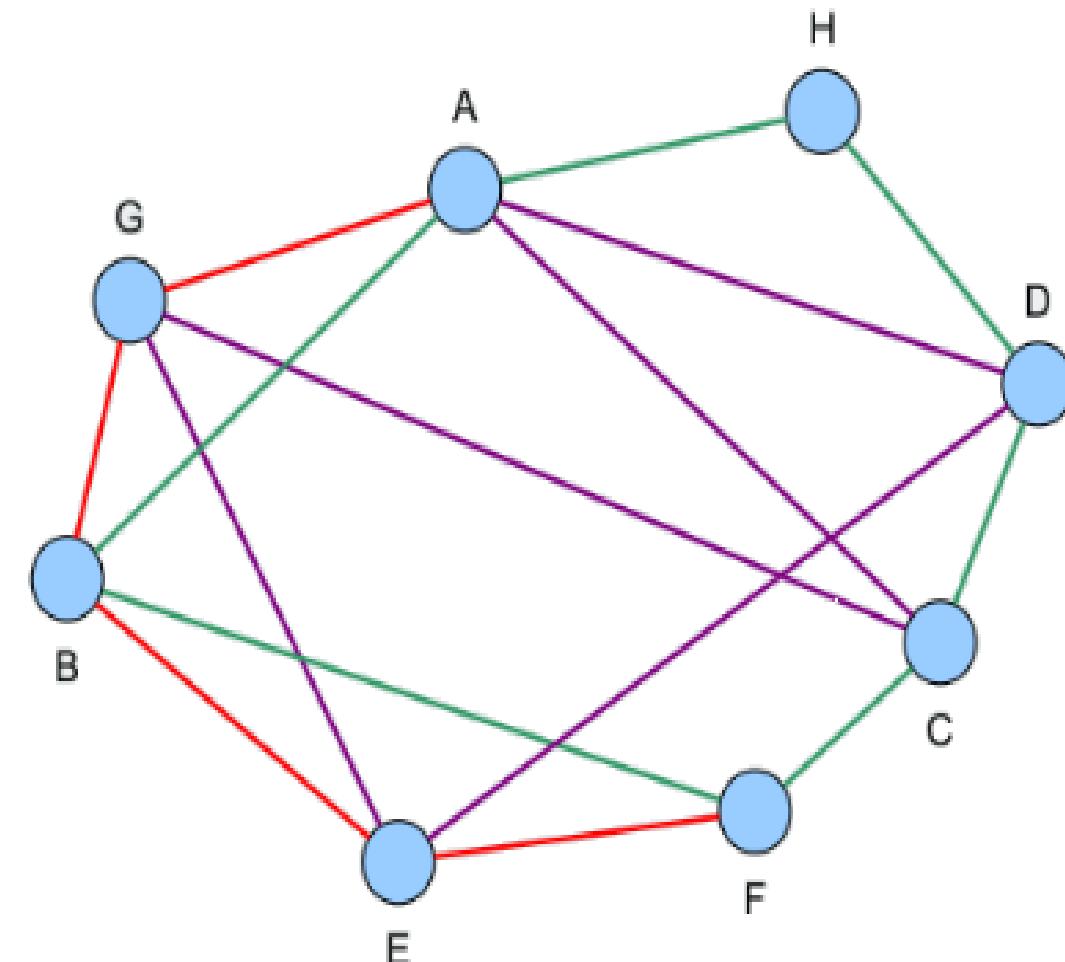
Le graphe suivant admet-il une chaîne eulérienne (*resp.* un cycle eulérien) ? Si oui la (*resp.* le) déterminer.



❑ Chemin et Circuit Eulérien

Solution :

- Tous les sommets n'ont pas un degré pair donc ce graphe ne possède pas de cycle eulérien.
- Exactement deux sommets ont un degré impair, A et F, donc ce graphe possède par contre une chaîne eulérienne.
- En voici une :
(A,G,B,F,C,D,H,A,B,E,D,A,C,G,E,F) :

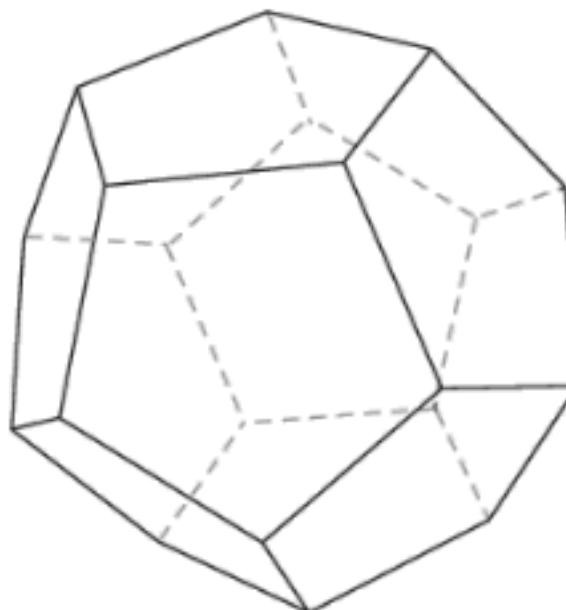


□ Graphes Hamiltoniens

- Origine :

Commençons là aussi par remonter le temps en nous penchant sur une question formulée en 1859 par **Sir William Rowan Hamilton** (1805-1865).

On considère 20 villes du globe terrestre positionnées sur les sommets d'un dodécaèdre régulier :



Un dodécaèdre régulier

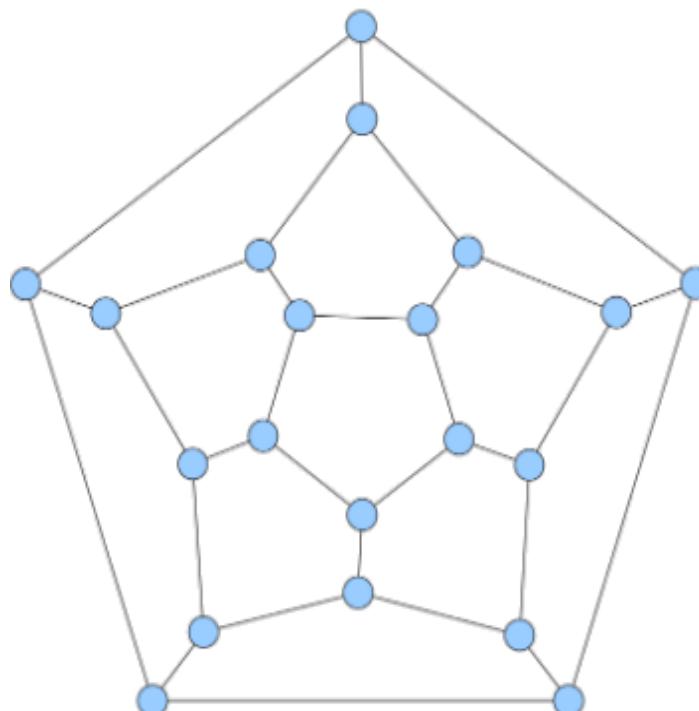
Rappelons qu'un dodécaèdre régulier est un polyèdre comportant 12 faces convexes pentagonales identiques, équilatérales et équiangles, 20 sommets et 30 arêtes.

La question est la suivante : peut-on effectuer un parcours en passant **une et une seule fois** par chacune de ces villes et en revenant à son point de départ ?

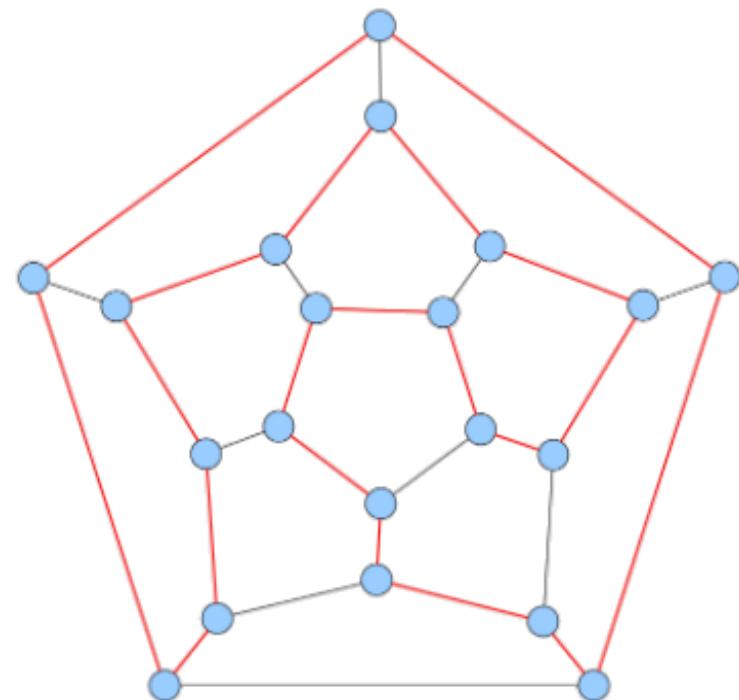
□ Graphes Hamiltoniens

On peut associer à ce solide un **graphe planaire**, dont les sommets et les arêtes correspondent à ceux du dodécaèdre :

Nous pouvons alors reformuler notre question : peut-on trouver un **cycle passant une et une seule fois par tous les sommets** de ce graphe ? Un tel parcours sera qualifié d'**Hamiltonien**.

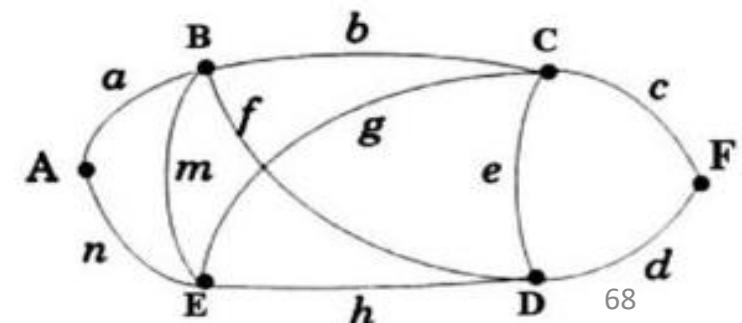
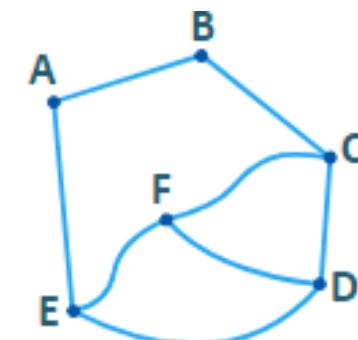


Dans ce cas, le cycle est assez facile à déterminer :



❑ Chaine et Cycle Hamiltonien

- **Chaine Hamiltonienne** : Chaine qui passe une et une seule fois par **tous** les sommets d'un **graphe non orienté**.
- **Cycle Hamiltonien** : Cycle qui passe une et une seule fois par **tous** les sommets d'un **graphe non orienté**.
- **Exemple**
 - Dans ce **graphe non orienté**, la chaine A-B-C-F-E-D est **une chaine Hamiltonienne**.
 - Dans ce **graphe non orienté**, le cycle constituée dans l'ordre des arêtes a, b, c, d, h et n est **un cycle Hamiltonien** qui commence et se termine au sommet A.

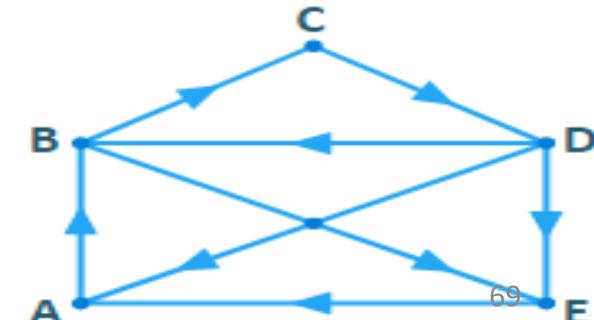
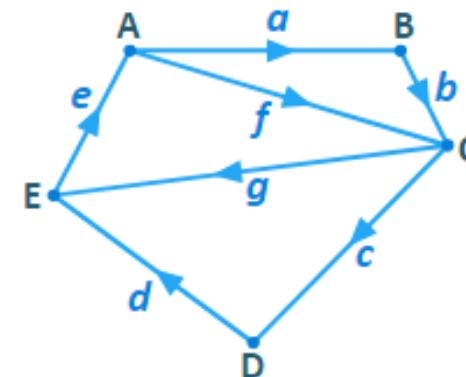


□ Chemin et Circuit Hamiltonien (Graphe orienté)

- **Chemin Hamiltonien** : Chemin qui passe une et une seule fois par **tous** les sommets d'un graphe
- **Circuit Hamiltonien** : Circuit qui passe une et une seule fois par **tous** les sommets d'un **graphe**
- Un **graphe Hamiltonien** est un graphe non orienté possédant un cycle Hamiltonien ou un graphe orienté possédant un circuit Hamiltonien.
- **Exemple**

➤ Dans ce **graphe orienté**, le chemin reliant dans l'ordre les sommets A, B, C, D et E est un **chemin Hamiltonien** de longueur 5. Il est formé des arcs *a*, *b*, *c* et *d*.

➤ Dans ce **graphe orienté**, le circuit qui passe, dans l'ordre, par les sommets A, B, C, D, E et A est un **circuit Hamiltonien**



□ Graphes Hamiltoniens

- Conditions suffisantes d'existence

Si l'on ne connaît pas encore de conditions nécessaires pour affirmer qu'un graphe est Hamiltonien ou non, il existe quelques **conditions suffisantes**. Elles ne sont cependant **pas très restrictives**.

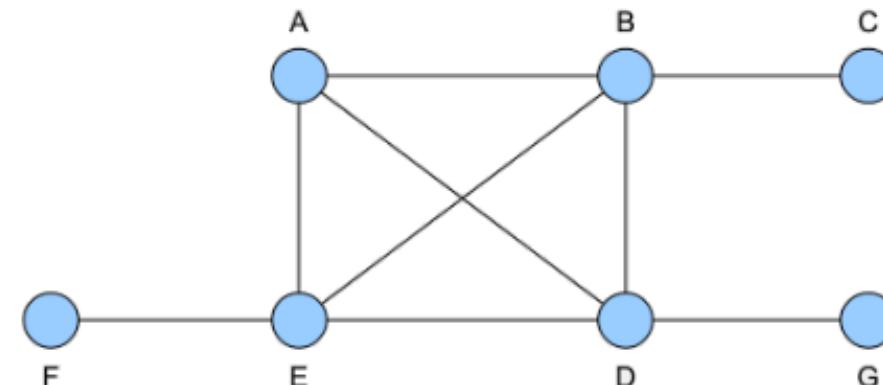
- Propriété

Soit $G=(V,E)$ un graphe **non orienté**.

Si G possède un sommet de degré 1 il ne peut pas être Hamiltonien.

- Exemple

Ce graphe n'est pas Hamiltonien car le sommet F est de degré 1.



□ Graphes Hamiltoniens

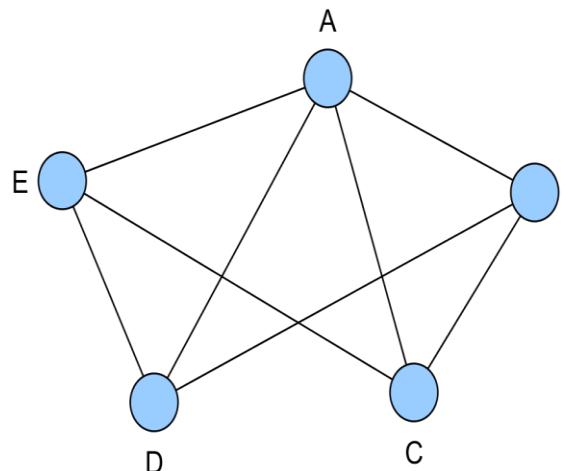
- Condition de Dirac (Gabriel Andrew Dirac 1925-1984).

Soit $G=(V,E)$ un graphe **non orienté** d'ordre n , avec $n \geq 3$.

Si pour tout sommet x de G on a $d(x) \geq n/2$ alors G est Hamiltonien.

Exemple

Considérons le graphe non orienté dont la représentation sagittale est la suivante :



- Ce graphe est d'ordre 5 et chacun de ses sommet a un degré au moins égal à 3. Il est donc Hamiltonien.
- Voici un exemple de cycle Hamiltonien : (A,B,C,E,D,A)

N.B : Cette condition suffisante mais absolument pas nécessaire est en pratique peu vérifiée.¹

□ Graphes Hamiltoniens

La condition précédente va permettre de prouver que **les graphes complets sont Hamiltoniens**

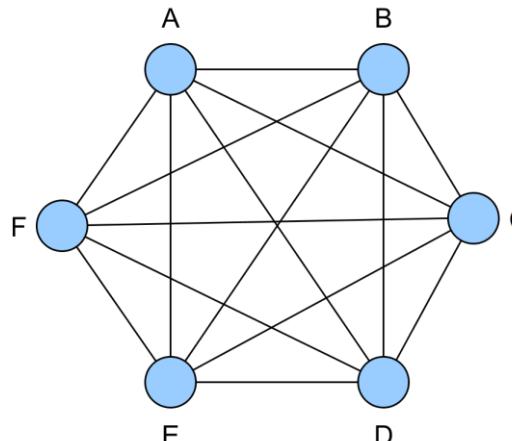
- **Propriété**

Soit $G=(V,E)$ un graphe **non orienté** d'ordre n , avec $n \geq 3$.

Si le graphe G est complet alors il est Hamiltonien.

- **Exemple**

Considérons le graphe non orienté complet dont la représentation sagittale est la suivante :



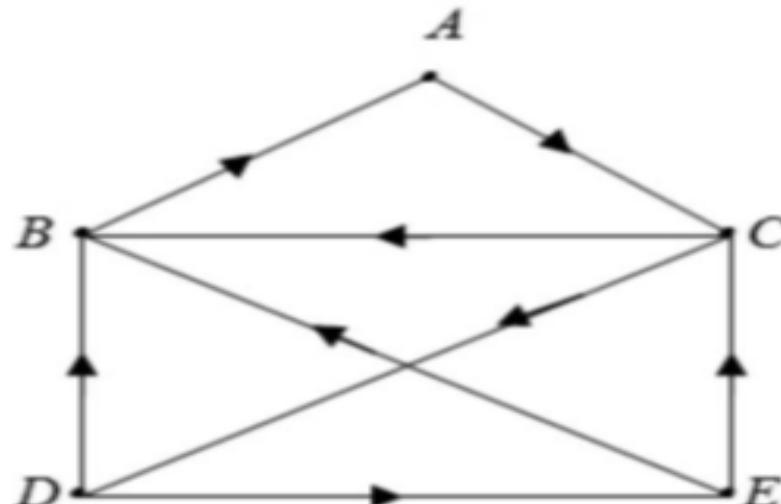
Puisque $n=6$ et $d(S)=5 \geq 6/2$

Don G est Hamiltonian

(ADEBCFA) est un cycle Hamiltonien

■ Exercice 6

- a) Ecrire la matrice d'adjacence M du graphe.
- b) Calculer M^4
- c) Combien y a-t-il de chemins de longueur 4 qui partent de A ? Citer ces chemins, parmi eux, y a-t-il des chemins Hamiltoniens ?



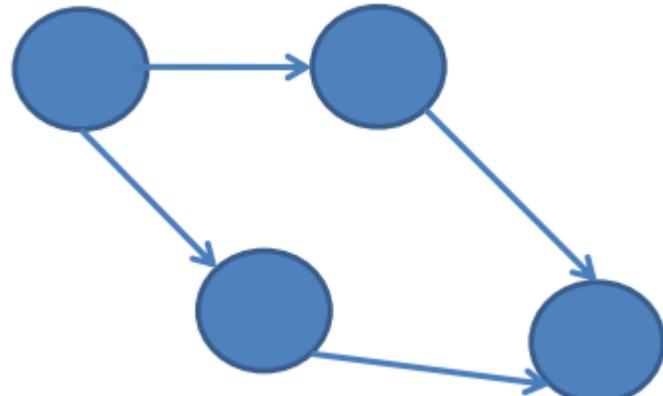
M1 =

1	1	2	0	0
1	1	0	0	1
1	2	1	2	0
1	2	1	1	1
1	2	2	1	0

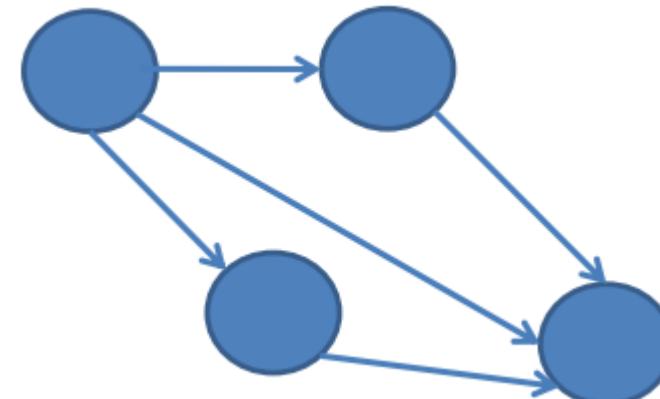
□ Fermeture transitive d'un graphe

- Faire la fermeture transitive d'un graphe consiste **à rajouter tous les arcs (ai, aj)** dès qu'il existe un chemin allant du sommet **ai** au sommet **aj**. C'est-à-dire que dès qu'il y a un chemin entre deux sommets de G, on y ajoute un arc (arête) entre ses deux sommets (s'il n'y a pas) pour obtenir \hat{G} .
- **Exemple 1 :**

Graphe G



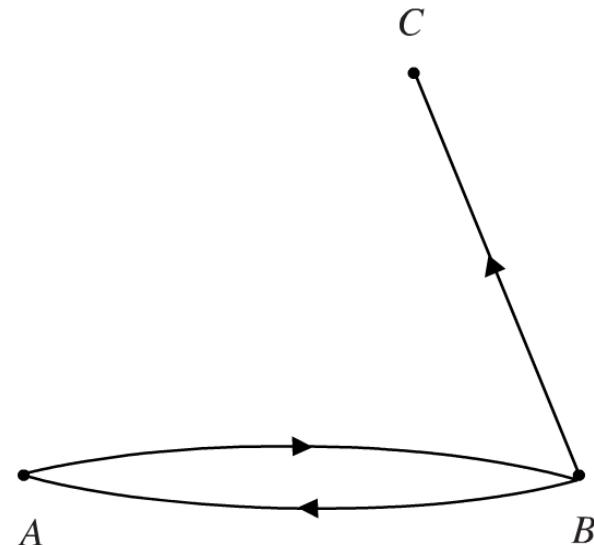
Sa fermeture transitive \hat{G} :



□ Fermeture transitive d'un graphe

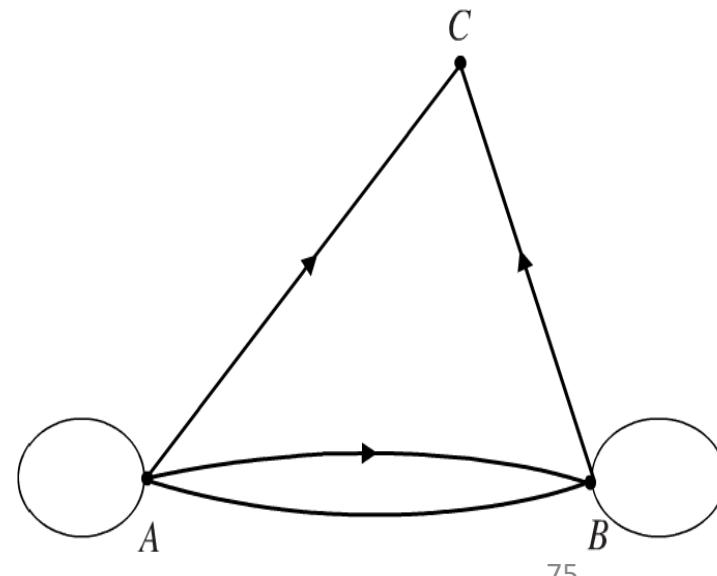
▪ Exemple 2 :

- Sur ce graphe présenté, il existe un chemin allant de A à C en passant par B donc on rajoute l'arc (A, C). De même, on constate qu'il faut ajouter les boucles (A, A) et (B, B).



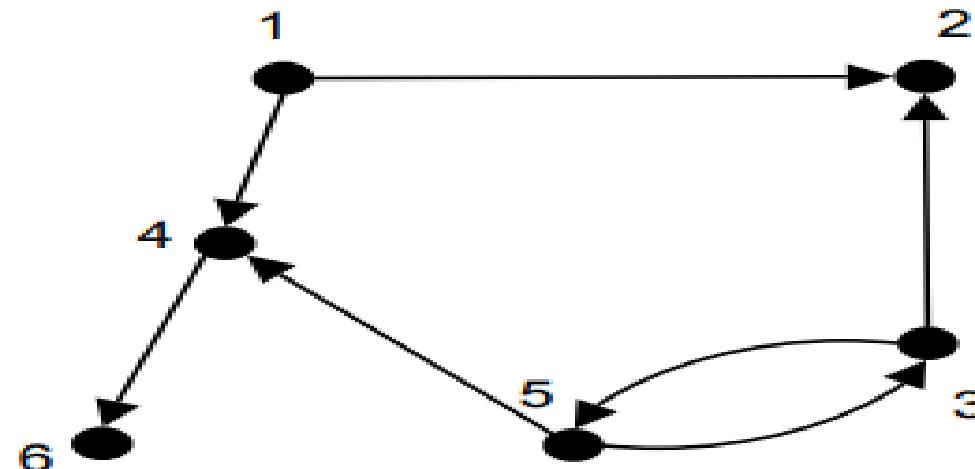
- On obtient ce nouveau graphe, sur lequel il ne reste aucun chemin sans l'arc « direct » correspondant

la fermeture transitive



□ Fermeture transitive d'un graphe

- Exemple 3 : Quels arcs doit-on rajouter pour faire la fermeture transitive du graphe?



Remarque: il peut être difficile de faire la fermeture transitive d'un graphe sans oublier d'arcs. Mais en utilisant la matrice d'adjacence d'un graphe, et au prix de quelques calculs matriciels, on peut déterminer la matrice d'adjacence de la fermeture transitive.

□ Fermeture transitive d'un graphe

- Propriété :

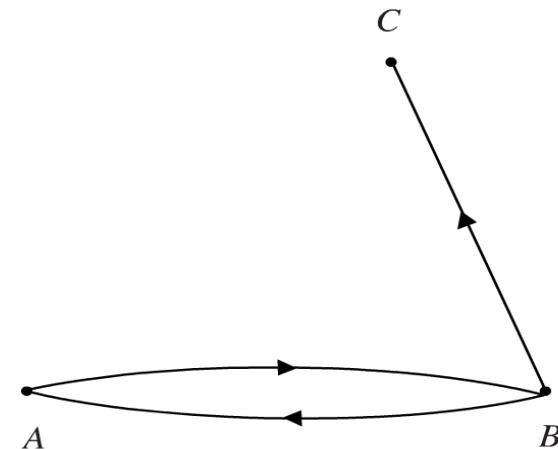
Soit M la matrice d'adjacence d'un graphe à n sommets et soit \tilde{M} la matrice d'adjacence de la fermeture transitive du graphe . Alors $\tilde{M} = M \oplus M^{[2]} \oplus \dots \oplus M^{[n-1]}$.

- Exemple :

- Reprenons l'exemple précédent

- la matrice d'adjacence est $M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$.

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



- Le graphe possède $n = 3$ sommets donc d'après la propriété précédente , la matrice d'adjacence de la fermeture transitive est $\tilde{M} = M \oplus M^{[3-1]}$.

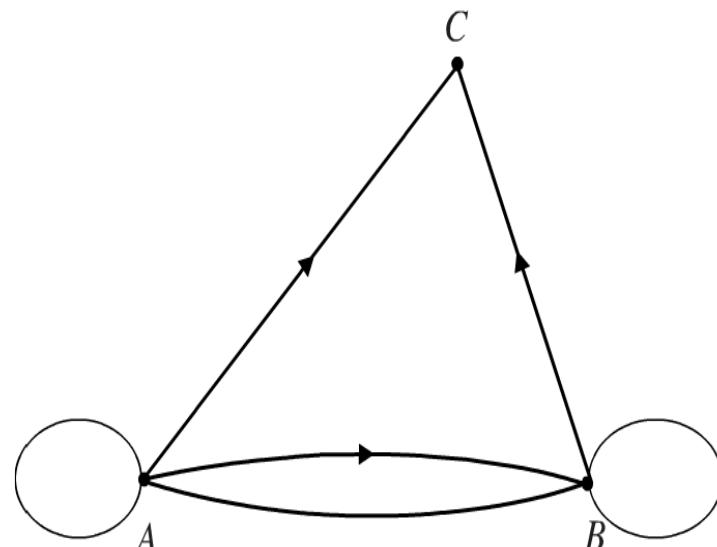
□ Fermeture transitive d'un graphe

- Les calculs donnent les résultats suivants:

$$M^2 = M^{[2]} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \rightarrow$$

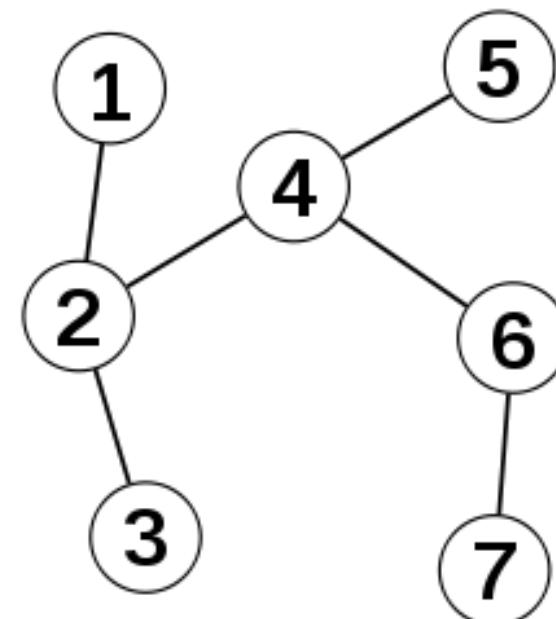
$$\hat{M} = M \oplus M^{[2]} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

→ la fermeture transitive



□ Arbre

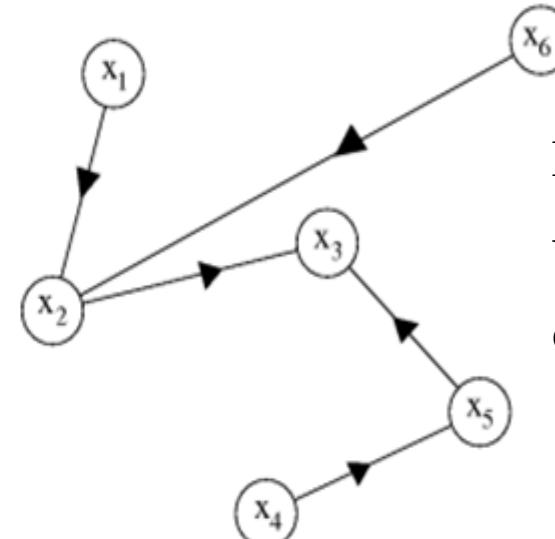
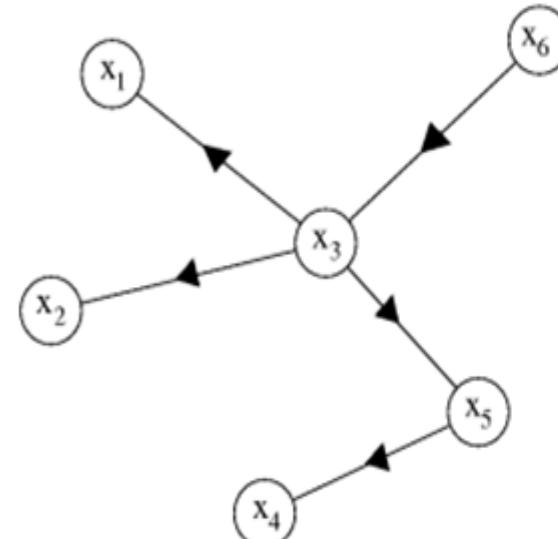
- Un arbre (concept non orienté) est un graphe connexe sans cycle.
- C'est le moyen le plus économique de raccorder un ensemble de nœuds : si on enlève une arête, il n'est plus connexe et, si on ajoute une arête, on crée un cycle et un seul. On a $m = n - 1$ pour un arbre.
- La plupart des réseaux d'eau ou d'électricité sont des arbres.
- On distingue deux types de sommets dans un arbre :
 - Les feuilles dont le degré est 1 ;
 - Les sommets internes dont le degré est supérieur à 1.
- L'arbre suivante avec 4 feuilles (les sommets 1, 3, 5, 7) et 3 sommets internes (les sommets n 2, 4 et 6).



❑ Arborescence

- **Une arborescence** est un graphe orienté qui possède un sommet de niveau 0, appelé racine, à partir duquel on peut atteindre tout autre sommet par un chemin unique.
- Une anti-arborescence est un arbre dont tous les nœuds sont ancêtres d'un même nœud appelé anti-racine. Ces concepts servent à modéliser des réseaux orientés (réseau fluvial par exemple), des processus d'assemblage et de désassemblage, etc.
- **Exemple:**

Le graphe suivant est une arborescence dont la racine est **x6**.

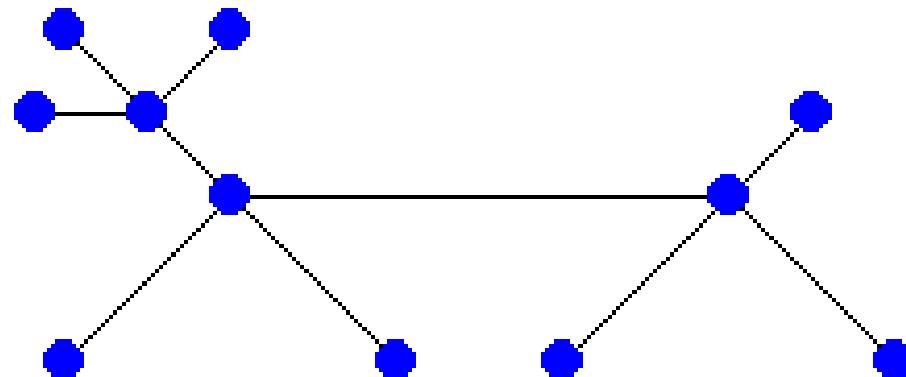


Le graphe suivant est une anti-arborescence dont l'anti-racine est **x3**.

□ Caractérisations des arbres

Propriété : Soit G un graphe non orienté d'ordre n . Les propositions suivantes sont équivalentes :

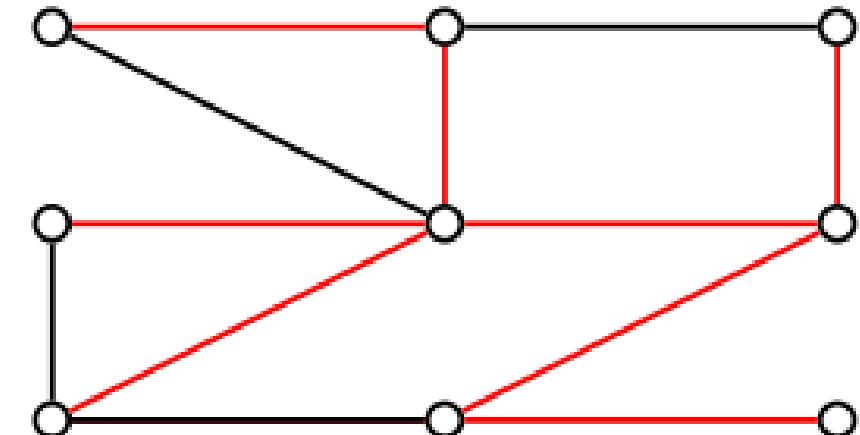
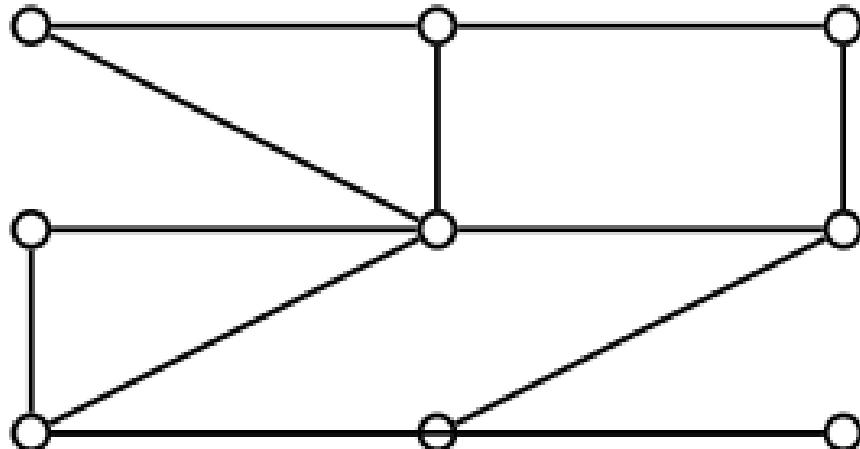
- G est un arbre
- G est connexe et a $(n - 1)$ arêtes
- G est sans cycle et a $(n - 1)$ arêtes
- G est sans boucle et pour tous sommets x, y , $x \neq y$, il existe une unique chaîne reliant x à y .
- G est connexe et si on lui retire une arête il n'est plus connexe.
- G est sans cycle et si on lui rajoute une arête on forme un cycle.



□ Arbre couvrant

Définition : Soit G un graphe non orienté.

Un arbre couvrant de G est un graphe partiel de G qui est un arbre.



Remarque :

- l'existence d'un arbre couvrant de G implique que G est connexe.
- Pour un graphe non connexe, on parle de forêt comme un ensemble d'arbres couvrants de ses composantes connexes

□ Arbre couvrant

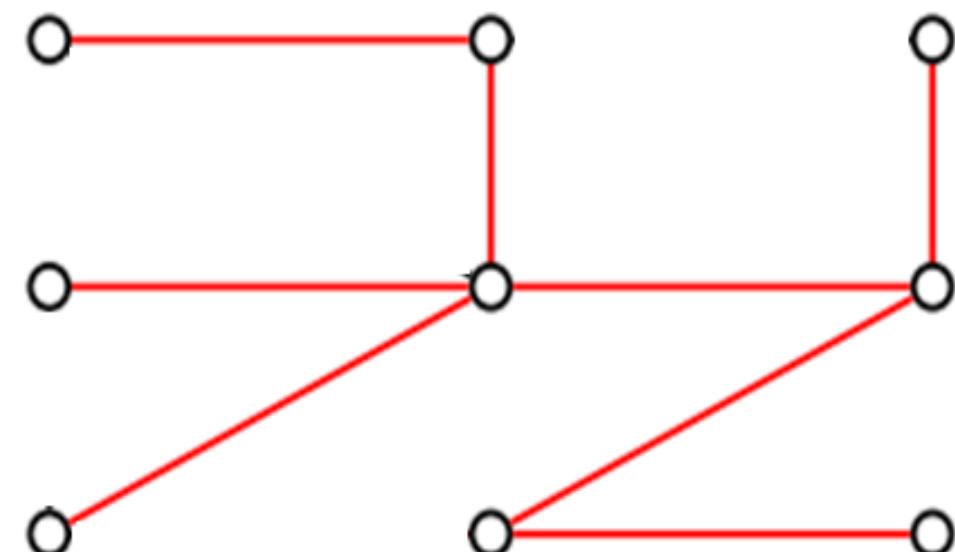
▪ Caractérisations

Propriété 1

Un graphe partiel d'un graphe connexe G est **un arbre couvrant de G** si et seulement s'il est **connexe et minimal** avec cette propriété relativement à la suppression d'arête i.e. s'il perd une arête il n'est plus connexe.

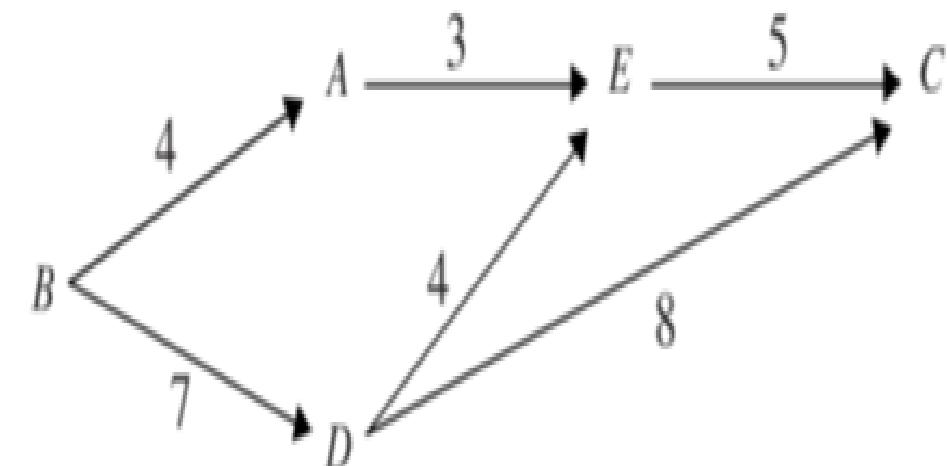
Propriété 2

Un graphe partiel d'un graphe connexe G est **un arbre couvrant de G** si et seulement s'il est **acyclique et maximal** avec cette propriété relativement à l'ajout d'arête i.e. si on lui ajoute une arête on forme un cycle



□ Graphe valué (pondéré)

- Sur certains graphes, il peut être nécessaire d'attribuer une valeur à chacun des arcs : on obtient alors un graphe **pondéré ou valué**.
- La valeur d'un chemin **est la somme des valeurs des arcs qui constituent le chemin**.
- Il est alors possible de chercher le **chemin de valeur minimale ou maximale reliant un sommet à un autre**.
- **Exemple :**
 - Sur le graphe valué de cette figure, pour relier B à C, il y a trois chemins possibles:



Graphe valué (pondéré)

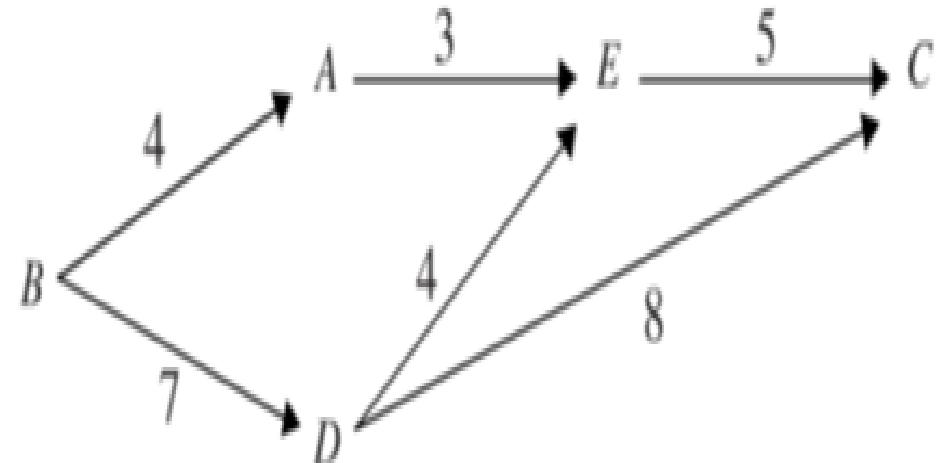
■ Exemple :

- le chemin (B, A, E, C) de valeur $4 + 3 + 5 = 12$.
- le chemin (B, D, E, C) de valeur $7 + 4 + 5 = 16$.
- le chemin (B, D, C) de valeur $7 + 8 = 15$.



Le chemin minimal est (B, A, E, C),
le chemin maximal est (B, D, E, C).

- Selon la nature du problème, les valeurs d'un graphe peuvent représenter une longueur, un coût, un poids, une distance ...



□ Représentation matricielle d'un graphe valué (pondéré)

■ Définition

Soit $G=(V,E)$ un graphe **valué orienté ou non** d'ordre n dont on a numéroté les n sommets.

On peut le représenter par sa **matrice de valuation**, qui est une matrice carrée d'ordre n , le coefficient de cette matrice situé à l'intersection de la i -ème ligne et de la j -ème colonne vaut

$$m_{ij} = \begin{cases} v(i, j) & \text{si } (i, j) \in E \\ +\infty & \text{sinon} \end{cases}$$

- Contrairement à la matrice d'adjacence, on ne peut pas utiliser la valeur 0 pour indiquer qu'il n'y a pas d'arc (*resp.* d'arête) entre deux sommets. En effet, cette valeur signifiera au contraire qu'il y a un arc (*resp.* une arête) et que la valuation de celui-ci (*resp.* celle-ci) vaut 0.
- Notre but est de déterminer des plus courts chemins, nous utiliserons donc la valeur $+\infty$ dans ce cas, afin de ne pas interférer avec notre recherche.

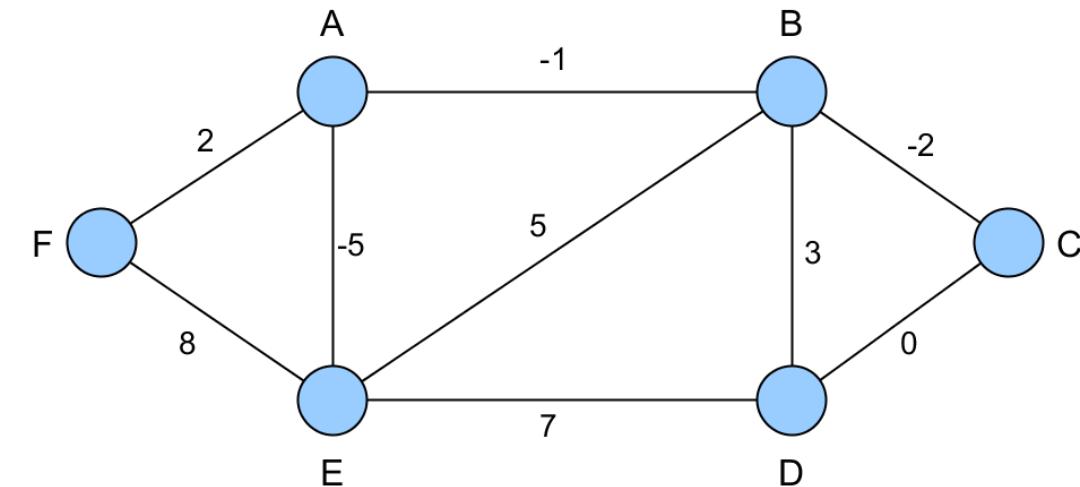
□ Représentation matricielle d'un graphe valué (pondéré)

Exemple : Matrice de valuation d'un graphe non orienté

Considérons le graphe valué non orienté suivant :

- Voici sa matrice de valuation

$$\begin{pmatrix} +\infty & -1 & +\infty & +\infty & -5 & 2 \\ -1 & +\infty & -2 & 3 & 5 & +\infty \\ +\infty & -2 & +\infty & 0 & +\infty & +\infty \\ +\infty & 3 & 0 & +\infty & 7 & +\infty \\ -5 & 5 & +\infty & 7 & +\infty & 8 \\ 2 & +\infty & +\infty & +\infty & 8 & +\infty \end{pmatrix}$$



- On remarquera que la matrice de valuation d'un graphe non orienté est **symétrique**.

□ Niveau des sommets d'un graphe sans circuit

Les graphes sans circuit et sans boucle peuvent être ordonnés par niveaux. La définition suivante va préciser la notion de niveau d'un sommet d'un graphe orienté défini sur un ensemble S de sommets.

■ Définition

- On appelle **sommet de niveau 0** tout sommets qui n'a pas de prédécesseur.
- Si on note S_0 l'ensemble des sommets de niveau 0, alors on appelle sommet de niveau 1 tout sommet qui n'a pas de prédécesseur dans $(S - S_0)$ (ensemble S privé des éléments de S_0).
- On définit ensuite les sommets de niveau suivant et ainsi de suite.....

□ Niveau des sommets d'un graphe sans circuit

- **Exemple:**
- Sur l'ensemble $S = \{A; B; C; D; E\}$, on définit un graphe par sa matrice d'adjacence M .
- Cherchons les sommets de niveau 0.

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

Sommets	A	B	C	D	E
Prédécesseurs	B	--	A, E	B	A, D

- B est le seul sommet qui n'a pas de prédécesseur , il est donc le seul sommet de niveau 0. $S_0 = \{B\}$
- On va maintenant chercher les sommets de niveau 1: ce sont les sommets qui n'ont pas de prédécesseurs dans $S - S_0 = \{A; C; D; E\}$.

□ Niveau des sommets d'un graphe sans circuit

- Exemple:

Sommets	A	C	D	E
Prédécesseurs	--	A, E	--	A, D

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

- A et D n'ont pas de prédécesseurs dans $S - S_0$, ils sont donc de niveau 1. $S_1 = \{A, D\}$
- Les sommets de niveau 2 sont ceux qui n'ont pas de prédécesseurs dans $\{S - S_0\} - S_1 = \{C, E\}$.
- E n'a plus de prédécesseur alors que C a le sommet E comme prédécesseur.
- Donc E est de niveau 2, et enfin C est de niveau 3.

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

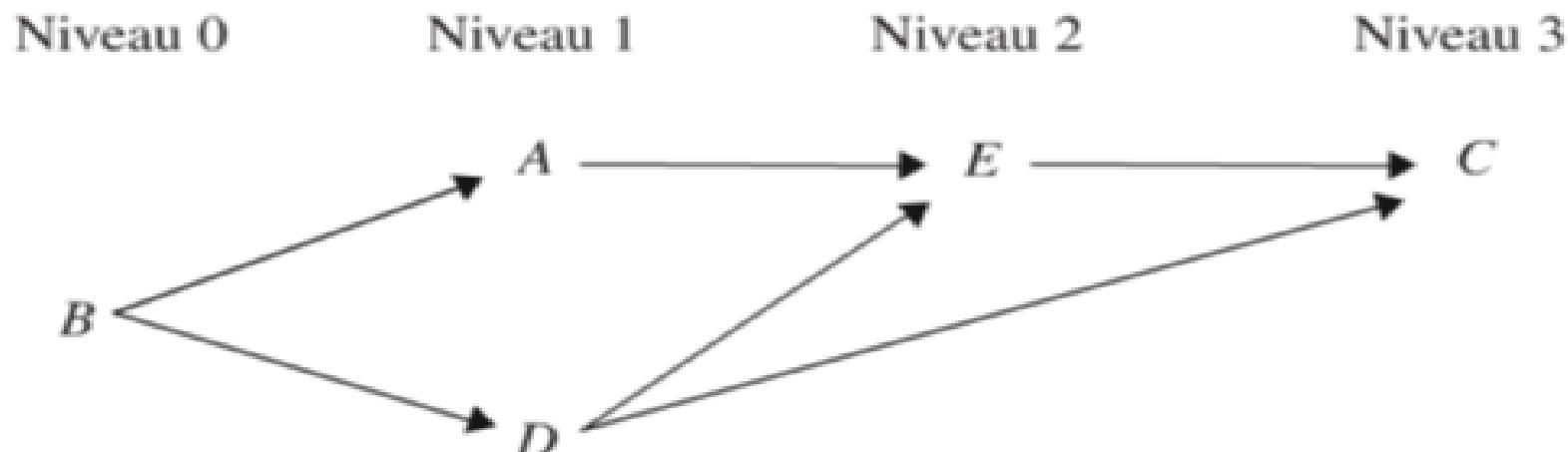
□ Niveau des sommets d'un graphe sans circuit

- Exemple:

- les niveaux des sommets sont résumés dans ce tableau

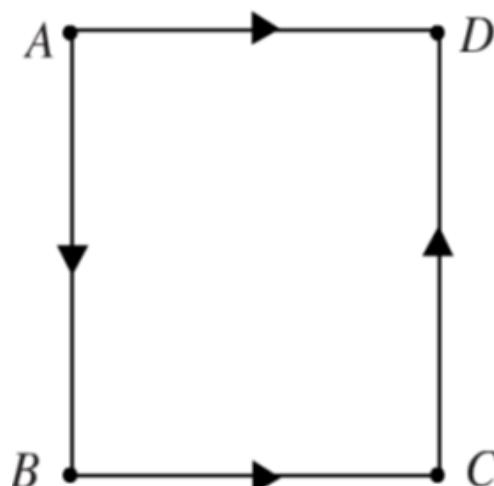
Niveaux	0	1	2	3
Sommets	B	A, D	E	C

- On peut faire le graphe en alignant verticalement les sommets de même niveau



□ Exercice 7

- a) Ecrire la matrice d'adjacence M du graphe suivant.
- b) Quels arcs doit-on rajouter pour faire la fermeture transitive de ce graphe.
- c) Calculer la matrice de la fermeture transitive

 $M =$

$$\begin{array}{rrrr} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}$$

 $\gg M1=M+M^2+M^3$ $M1 =$

$$\begin{array}{rrrr} 0 & 1 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{array}$$

■ Exercice 8:

Un graphe est défini par le tableau suivant. Déterminer le niveau de chaque sommet et représenter le graphe en tenant compte des niveaux.

Sommets	A	B	C	D	E	F	G	H
Successseurs	E, G	---	A, H	F	---	G	---	B, D

■ Exercice 9 :

Un graphe est défini par le tableau suivant. Déterminer les prédecesseurs de chaque sommet. Pourquoi peut-on affirmer que le graphe est une arborescence?

Sommets	A	B	C	D	E	F	G	H	I	J	K	L
Successseurs	B,L	D,K	---	---	A,J	C,G,I	---	---	---	F	---	H

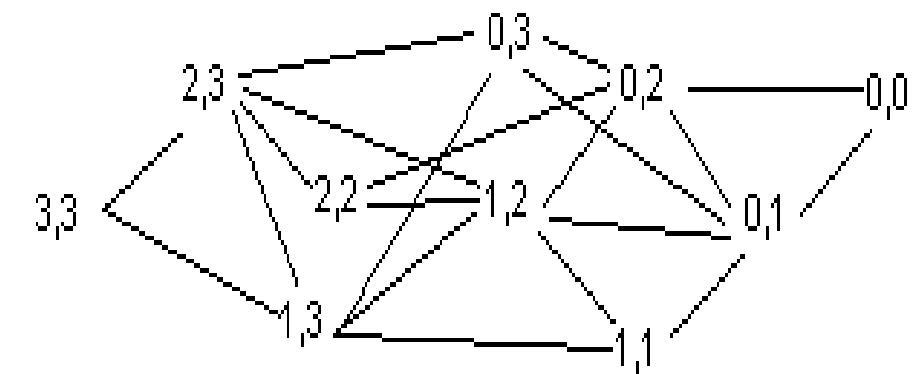
□ Exercice 10 : jeu des allumettes

Deux joueurs disposent de deux boites de trois allumettes, à tour de rôle chaque joueur peut enlever une ou deux allumettes dans une des boites. Le joueur qui retire la dernière allumette perd la partie.

- Modéliser le jeu à l'aide d'un graphe.
- Que doit jouer le premier joueur pour gagner à coup sûr ?

Indication: Chaque état est représenté par un sommet (x, y) indiquant le nombre d'allumettes de chaque boîte.

Le joueur qui atteint la configuration $0,0$ perd la partie. Pour gagner, on doit donc atteindre la configuration $0,1$. On peut vérifier qu'en jouant $1,3$ au premier coup, quelle que soit la réponse de l'adversaire, on peut atteindre ensuite $0,1$. Le coup gagnant au départ est donc « enlever 2 allumettes dans un tas »



Les principaux algorithmes de la théorie des graphes

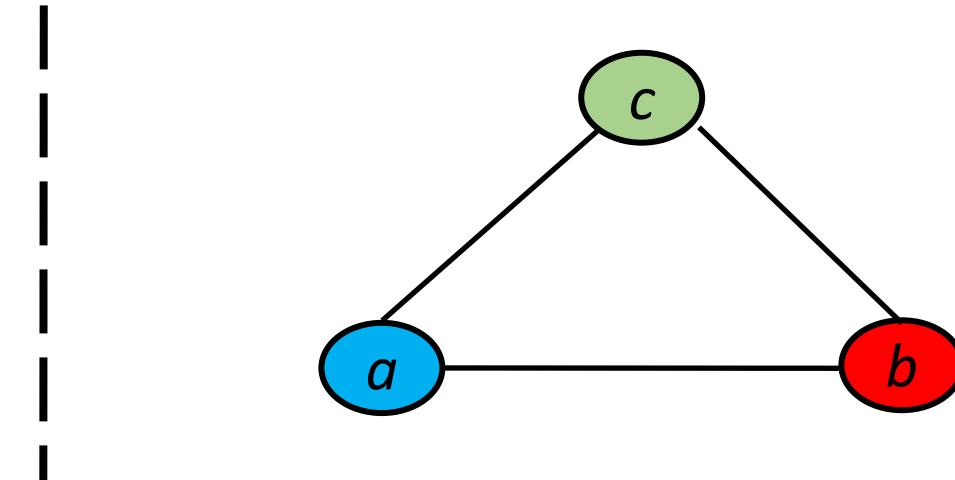
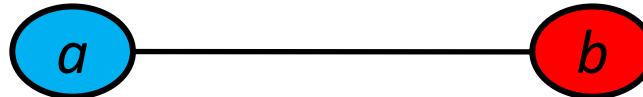
Algorithme de coloration Welsch & Powell

□ Principe

Le principe est de colorier les sommets d'un graphe, de telle façon à attribuer:

- Pour deux sommets adjacents deux couleurs différentes
- Un nombre de couleur minimal

▪ Exemples



- Il est clair que l'on ne pourra colorer que des graphes simples. En effet, lorsqu'un graphe comporte une boucle, cela signifie qu'un sommet est adjacent à lui-même. Il faudrait donc lui attribuer deux couleurs, ce qui est contraire à la définition.

□ Nombre chromatique

- **Définition :**

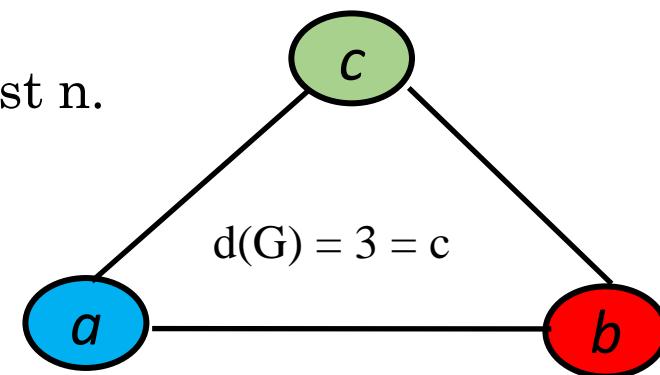
Le nombre chromatique est le **plus petit nombre** de couleurs nécessaires pour colorier le graphe. On le note c .

- La recherche du nombre chromatique d'un graphe est un **problème réellement difficile**. Il n'existe par exemple aucune formule pour le calculer explicitement en fonction du nombre d'arêtes ou de sommets, sauf dans des cas très spécifiques comme celui des **graphes complets**.

- **Propriété 1 :**

Le **nombre chromatique** d'un graphe non orienté **complet** à n sommets est n .

- Cette propriété est immédiate, puisque dans un graphe complet l'on ne peut pas colorer deux sommets de la même couleur car ils sont nécessairement adjacents. Il faut donc utiliser **une couleur par sommet**.



□ Nombre chromatique

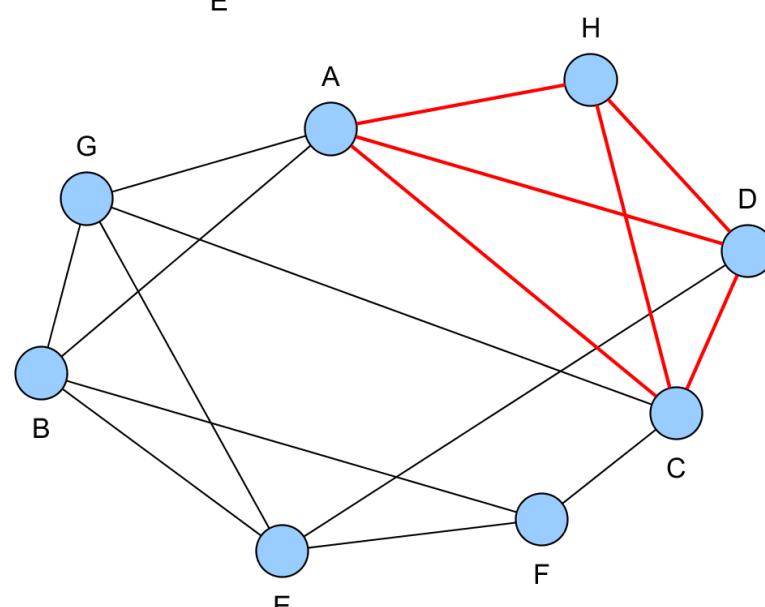
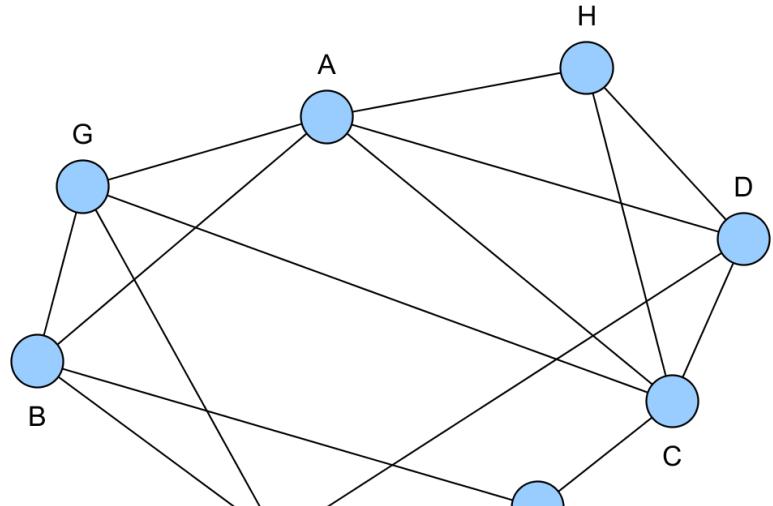
- Si l'on ne peut pas dans le cas général calculer le nombre chromatique d'un graphe, on peut toutefois en donner un **encadrement**.
- Pour le **minorer**, commençons par remarquer que le nombre chromatique d'un graphe est supérieur ou égal à celui de chacun de ses sous-graphes. En pratique, on utilisera ce résultat en recherchant des **sous-graphes complets**, puisque l'on connaît leur nombre chromatique.
- **Propriété 2:**
 $n \leq c$, où **n** est l'ordre du sous-graphe complet d'ordre le plus élevé
- Pour **majorer** le nombre chromatique, il va falloir considérer les **degrés des sommets**, et en particulier **le plus grand** d'entre eux.
- **Propriété 3:**
 $c \leq d + 1$, où **d** est le degré maximal des sommets du graphe.

□ Nombre chromatique

- Exemple : Encadrement du nombre chromatique d'un graphe

Considérons le graphe G non orienté suivant

- Pour minorer le nombre chromatique de G, on cherche donc un sous-graphe complet d'ordre maximum. On trouve A,C,D,H représenté ci-dessous en rouge, et qui est d'ordre 4 : On a ainsi $4 \leq c$.
- Pour majorer le nombre chromatique de G, il faut calculer le degré maximum de ses sommets. Il s'agit de 5, degré des sommets A et C. On a donc $c \leq 6$.
- Finalement, on obtient l'encadrement suivant : $4 \leq c \leq 6$.



❑ Nombre chromatique

- Théorème des quatre couleurs (Francis Guthrie 1852)

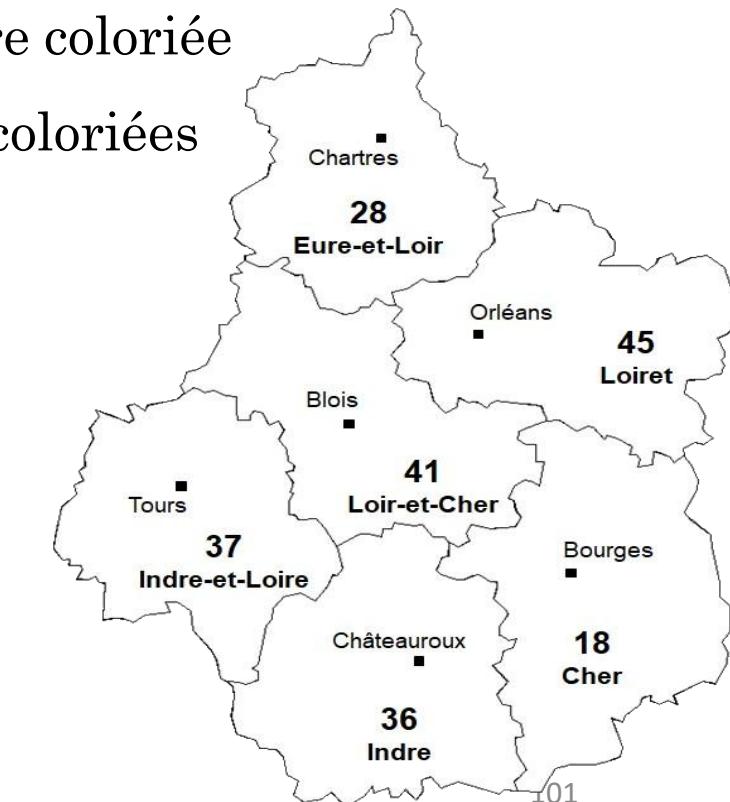
Le nombre chromatique d'un graphe planaire est au plus égal à 4.

- Théorème des quatre couleurs, formulation originelle

Toute carte de géographie dont les régions sont contiguës peut être coloriée avec au plus 4 couleurs sans que deux pays limitrophes ne soient coloriées avec la même couleur.

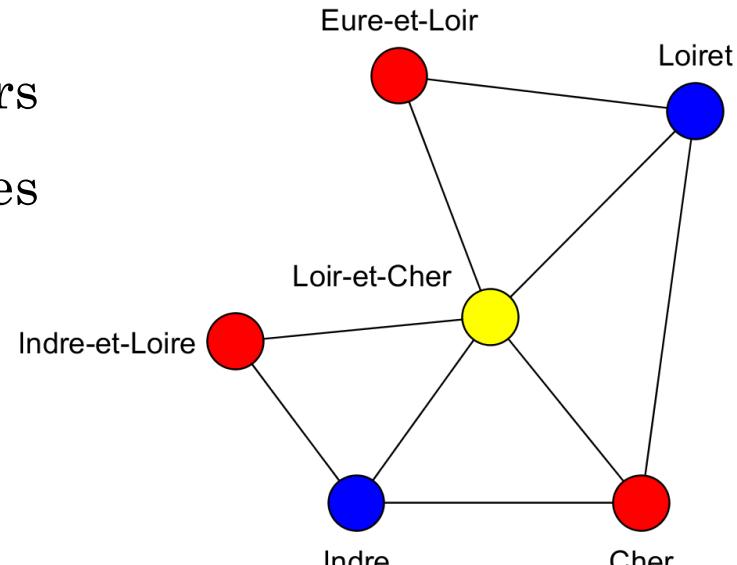
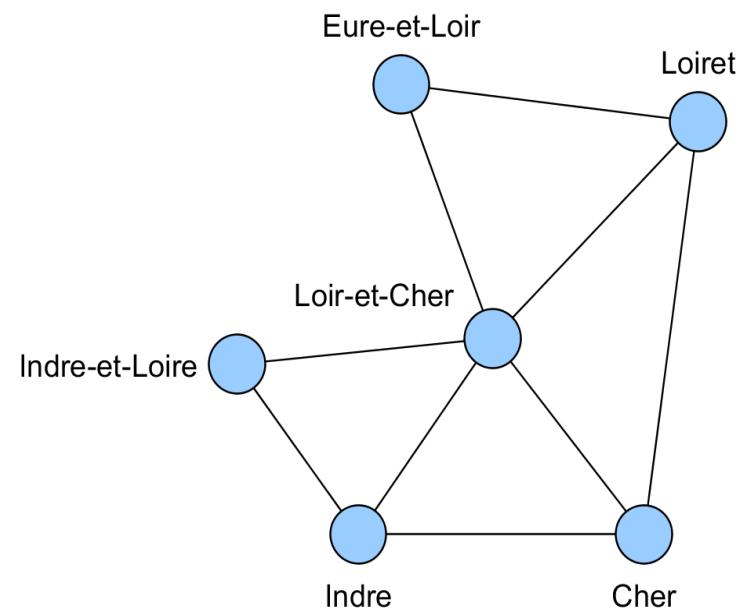
- Exemple : Coloration de la carte de la région Centre-Val de Loire

- On peut lui associer un graphe, en représentant chacun des départements (plus généralement chacune des régions d'une carte) par un sommet, et chacune des frontières par une arête :



❑ Nombre chromatique

- Par construction ce graphe est nécessairement planaire et pourra donc être coloré par au plus 4 couleurs. On constate cependant que 3 couleurs suffisent :
- On ne pourra pas utiliser moins de 3 couleurs car ce graphe comporte des sous-graphes complets d'ordre 3.



□ Algorithme de Coloration de Welsch Powell

Soit $G=(V,E)$ un graphe **non orienté**.

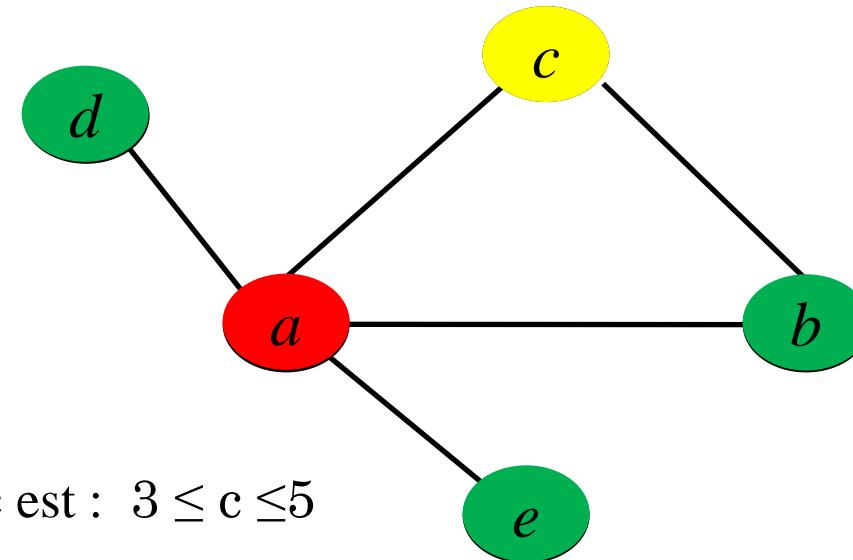
Etapes de l'algorithme :

1. Calculer le degré de chaque sommet.
2. Trier les sommets par ordre décroissant de leur degré : $d(x_1) \geq d(x_2) \geq \dots \geq d(x_n)$.
3. Choisir une couleur pour le premier sommet x_1 .
4. Parcourir la liste des sommets triés puis colorer de cette couleur le premier sommet non adjacent à x_1 (s'il existe).
5. Continuer la liste et colorer de même le prochain sommet non adjacent ni au premier ni au second.
6. Faire de même jusqu'à épuisement de la liste.
7. Prendre une seconde couleur pour le premier sommet non coloré de la liste et recommencer les étapes précédentes.
8. Recommencer jusqu'à avoir coloré tous les sommets.

□ Algorithme de Coloration de Welsch Powell

- Exemple 2: Application de l'algorithme de Welsh & Powell

x	d(x)	Couleur
a	4	C1
b	2	C2
c	2	C3
d	1	C2
e	1	C2



Le nombre chromatique c est : $3 \leq c \leq 5$

- Il est à noter que le nombre de couleurs minimal obtenu par l'algorithme de coloration n'est pas nécessairement optimal. Ce dernier donne une coloration parmi plusieurs possibles.

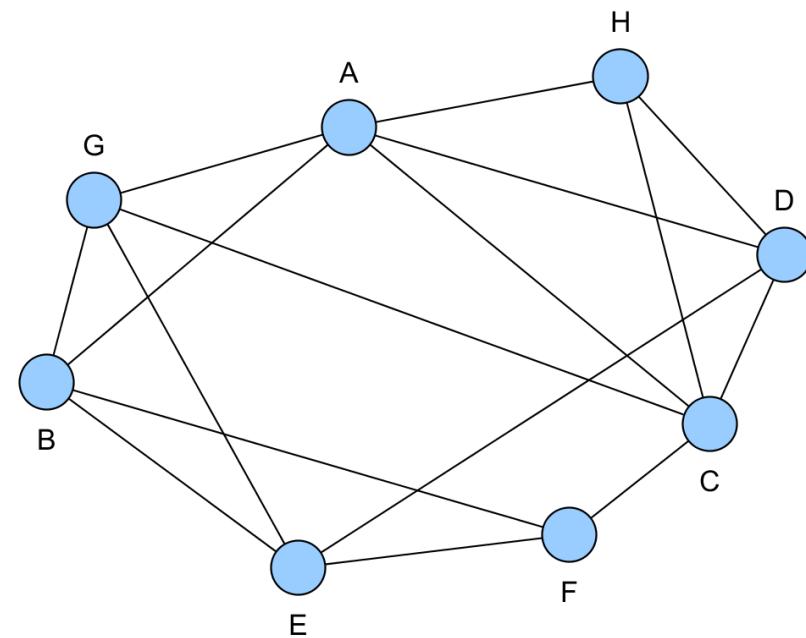
□ Algorithme de Coloration de Welsch Powell

- Exemple 2: Application de l'algorithme de Welsh & Powell

Considérons de nouveau ce graphe :

1.Calcul du degré de chaque sommet :

x	A	B	C	D	E	F	G	H
d(x)	5	4	5	4	4	3	4	3



2.Tri des sommets par ordre décroissant de leur degré :

x	A	C	B	D	E	G	F	H
d(x)	5	5	4	4	4	4	3	3

3.On choisit une couleur pour le premier sommet de cette liste triée. Colorons ainsi le sommet A en rouge par exemple.

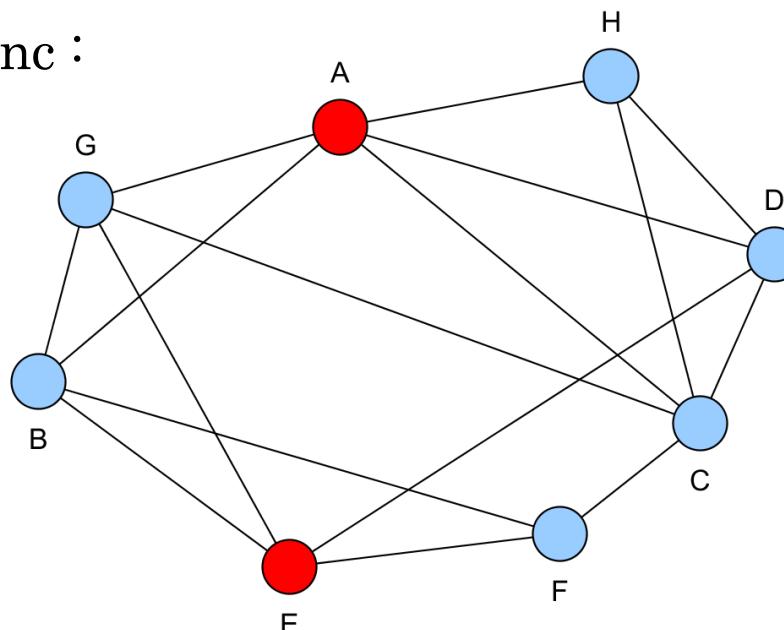
□ Algorithme de Coloration de Welsch Powell

- Exemple 2: Application de l'algorithme de Welsh & Powell

4. On parcourt ensuite la liste dans l'ordre. On constate que les sommets C, B, D, G et H sont adjacents au sommet A donc on ne les colore pas encore. Le premier sommet non adjacent à A est E, on le colore donc aussi en rouge.

5. Les trois derniers sommets de la liste sont adjacents soit à A soit à E donc on ne les colore pas.

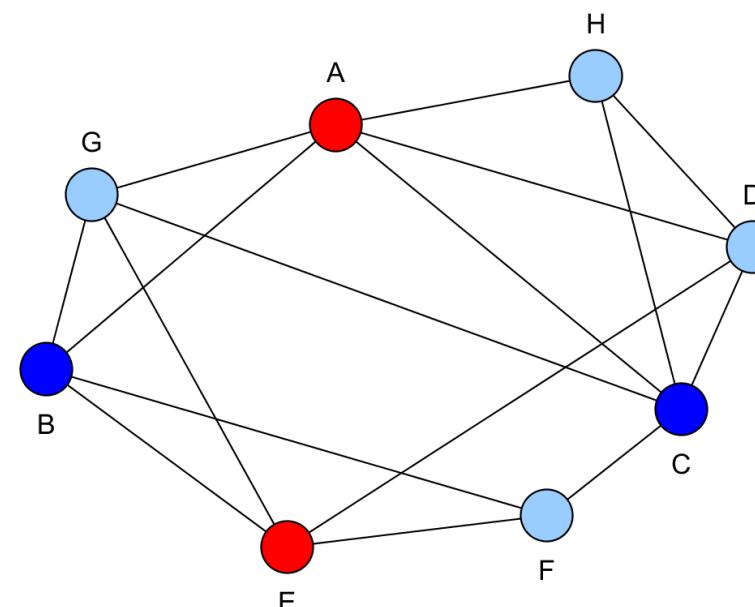
Pour l'instant notre coloration est donc :



□ Algorithme de Coloration de Welsch Powell

- Exemple 2: Application de l'algorithme de Welsh & Powell

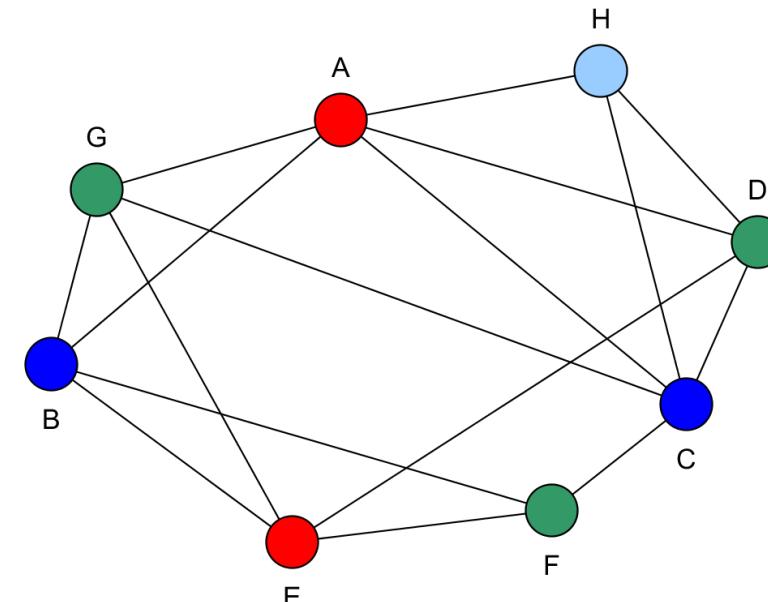
6. On choisit une seconde couleur, le bleu, pour le premier sommet non coloré de la liste, *i.e.* le sommet C. On continue à parcourir la liste, on colore le sommet B aussi en bleu car il n'est pas adjacent à C. Tous les autres sommets de la liste sont adjacents soit à C soit à B donc on ne les colore pas. Notre coloration devient donc :



□ Algorithme de Coloration de Welsch Powell

- Exemple 2: Application de l'algorithme de Welsh & Powell

7. On réitère le procédé en colorant d'une troisième couleur, le vert, le premier sommet non encore coloré, c'est-à-dire le sommet D. Le sommet G n'est pas adjacent à D donc on le colore aussi en vert. On continue de parcourir la liste, et l'on colore aussi F en vert car il n'est ni adjacent à D ou à G. Le dernier sommet de la liste est adjacent à un sommet déjà coloré en vert, on ne le colore pas encore. Il vient :

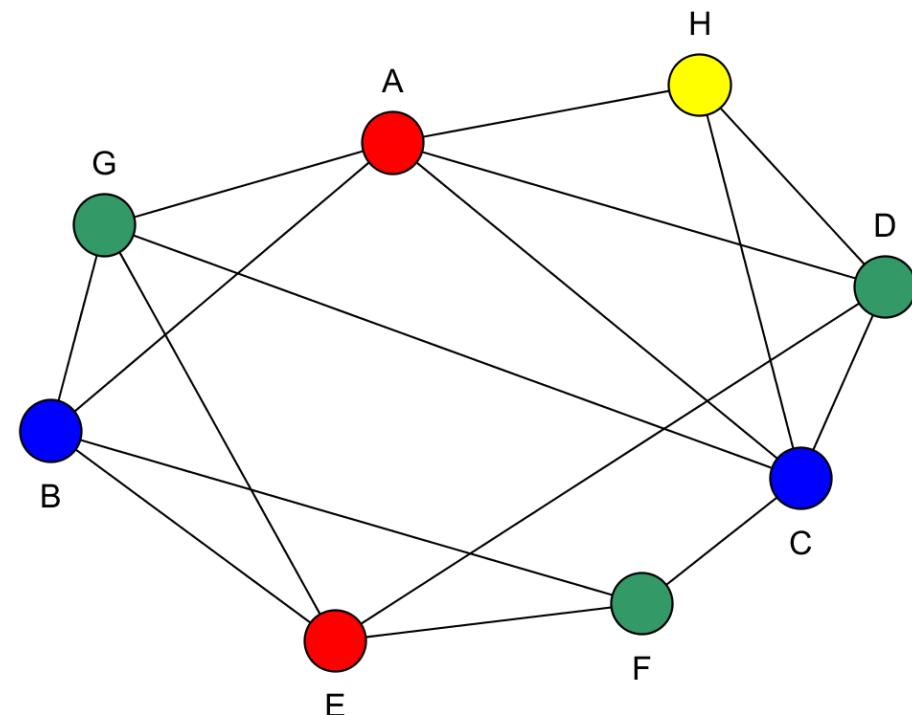


□ Algorithme de Coloration de Welsch Powell

- Exemple 2: Application de l'algorithme de Welsh & Powell

7. On colore enfin le dernier sommet non coloré, *i.e.* H, avec une autre couleur, par exemple le jaune. Notre coloration finale est alors :

- On a ainsi coloré ce graphe avec 4 couleurs. Cette coloration est optimale car rappelons que nous avions encadré le nombre chromatique de ce graphe dans un exemple précédent et que nous avions obtenu : $4 \leq c \leq 6$. On ne peut donc pas faire mieux que 4 couleurs.



□ Algorithme de Coloration de Welsch Powell

▪ Application

L'une des applications les plus classiques de la coloration d'un graphe est la résolution de **problèmes d'incompatibilité**. Ces incompatibilités peuvent être de natures diverses :

- Incompatibilités **horaires**. Exemple : planification d'une session d'examens avec le moins de plages horaires possibles sachant que certains étudiants doivent passer plusieurs examens.
- Incompatibilités **affectives**. Exemple : répartir des invités à un mariage en un nombre minimal de tables tout en tenant compte des vieilles inimitiés.
- Incompatibilités **chimiques**. Exemple : organiser en un minimum de voyages le transport de produits chimiques dont certains pour des raisons de sécurité ne peuvent transiter ensemble.
- Incompatibilités **géographiques**. Exemple : répartir des pays en un nombre minimal de groupes tout en respectant des contraintes d'entente.
- Etc.

■ Exercice 11

A, B, C, D, E, F, G et H désignent huit poissons. Dans le tableau ci-dessous, une croix signifie que les poissons ne peuvent pas cohabiter dans un même aquarium:

	A	B	C	D	E	F	G	H
A		X	X	X			X	X
B	X				X	X	X	
C	X			X		X	X	X
D	X		X		X			X
E		X		X		X	X	
F		X	X		X			
G	X	X	X		X			
H	X		X	X				

1. Formuler ce problème comme un problème de coloration dans un graphe où les poissons représenteront les sommets du graphe.
2. Quel nombre minimum d'aquariums faut-il ?

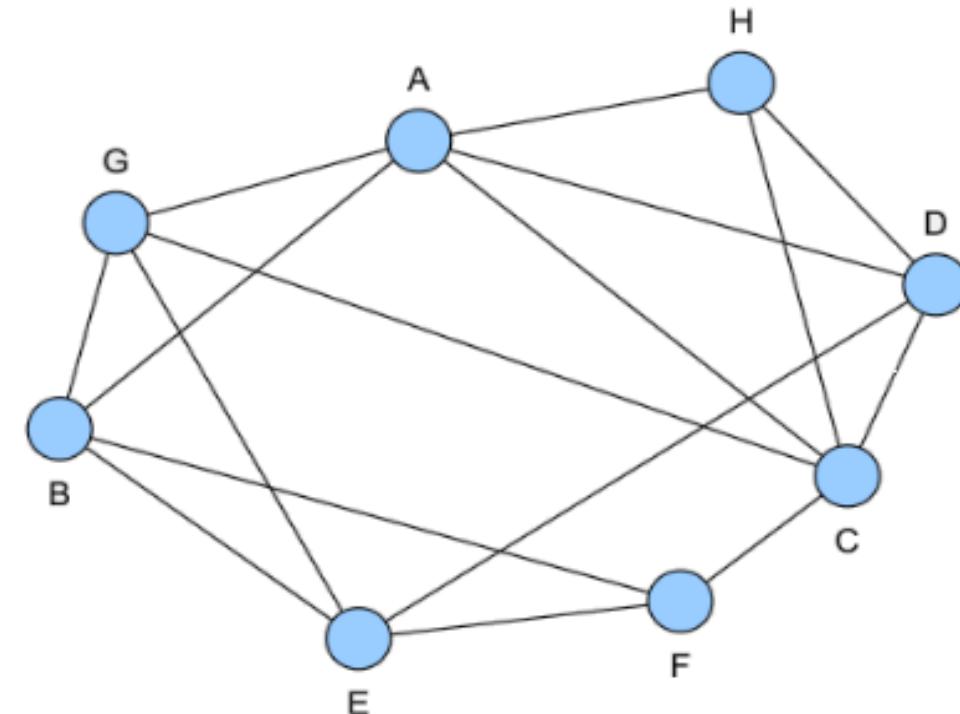
Algorithme de coloration Welsch & Powell

- Solution : Exercice 11

La question est donc de déterminer le nombre minimum d'aquariums nécessaire pour loger tous ces poissons.

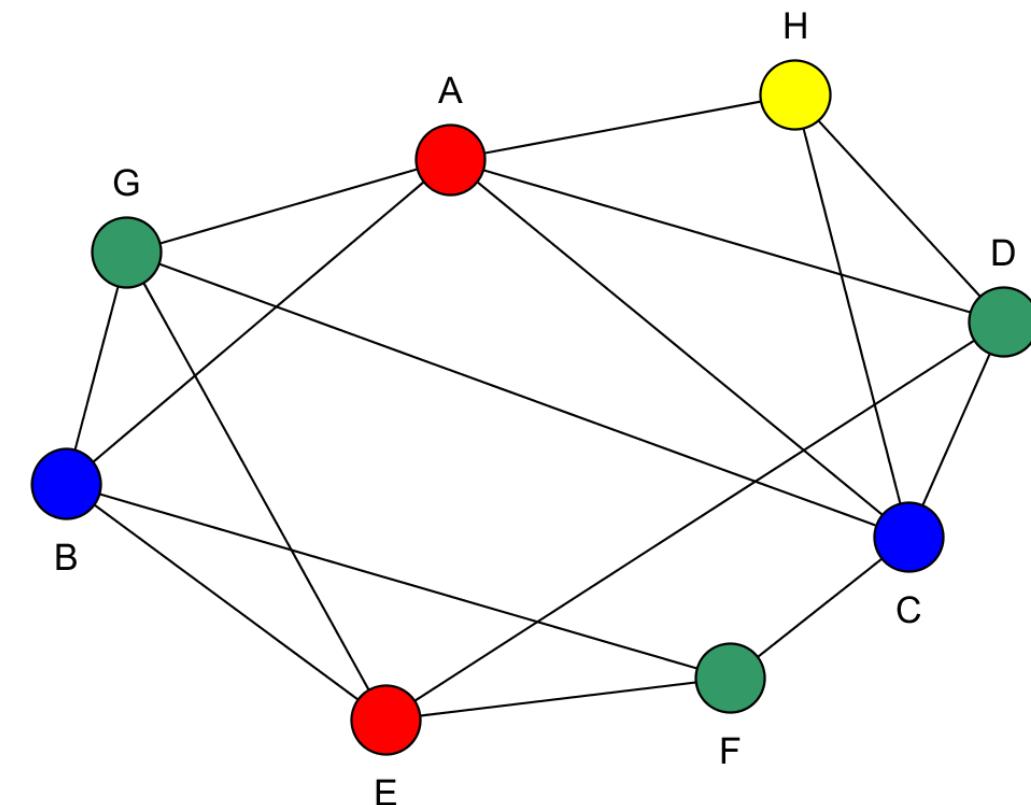
On commence par associer à ce problème un graphe résumant les incompatibilités : un sommet par poisson, et une arête entre deux sommets indique que les poissons correspondants ne peuvent pas cohabiter. On obtient :

	A	B	C	D	E	F	G	H
A		X	X	X			X	X
B	X				X	X	X	
C	X			X		X	X	X
D	X		X		X			X
E		X		X		X	X	
F		X	X		X			
G	X	X	X		X			
H	X		X	X				



- Solution : Exercice 11

- Il nous faut ensuite colorer les sommets de ce graphe.
- Ce graphe est déjà traité et on avait alors établi que son nombre chromatique valait 4 et que l'on pouvait le colorer comme suit :
- Il faudra donc utiliser 4 aquariums, et mettre ensemble les poissons A et E, les poissons B et C, les poissons D, F et G, et isoler le poisson H.



■ Exercice 12

Une université doit préparer un planning des examens. On suppose qu'il y a 7 examens à planifier correspondant aux modules de 1 à 7, et que les paires de cours suivants ont des étudiants communs: (1 et 2), (1 et 3), (1 et 4), (2 et 3), (2 et 4), (2 et 5), (2 et 7), (3 et 4), (3 et 6), (3 et 7), (4 et 5), (4 et 6), (5 et 6), (5 et 7), (6 et 7).

Un étudiant ne peut pas passer plus d'un examen par jour.

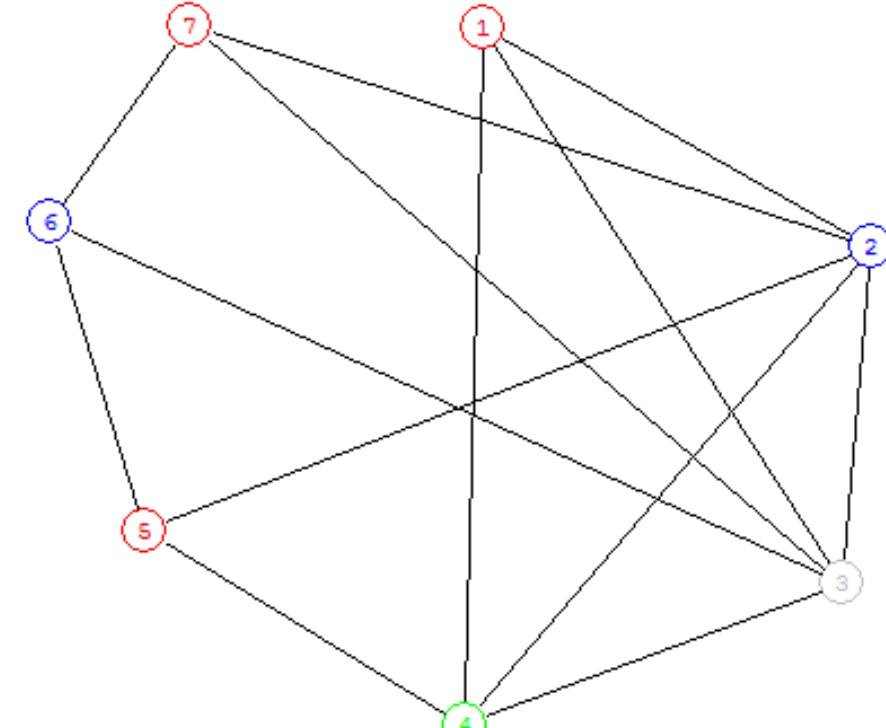
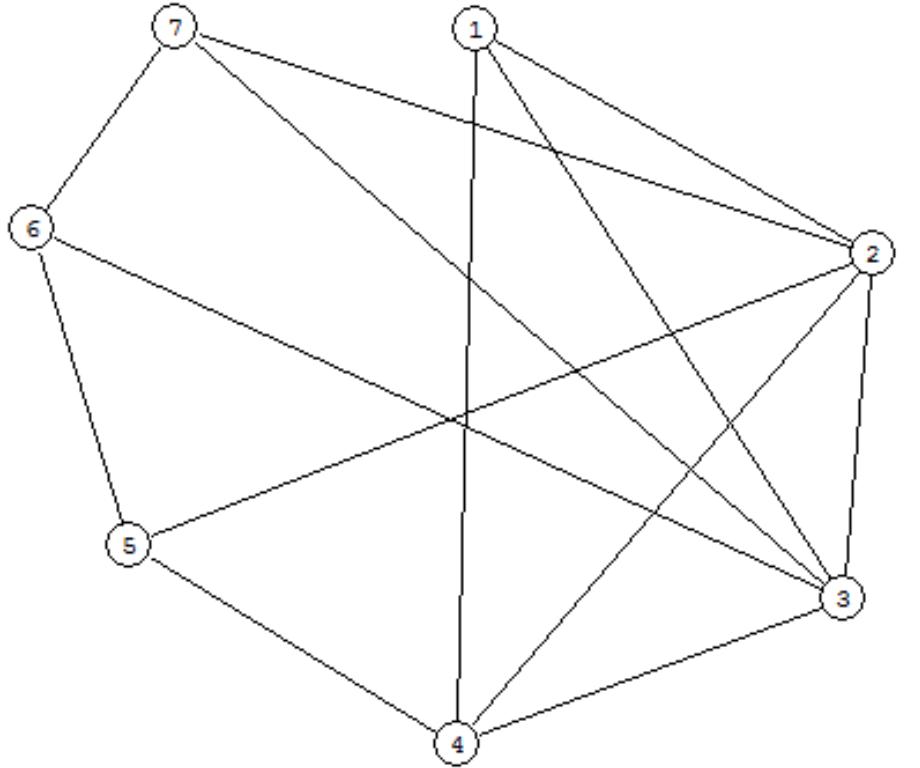
Q1 - présenter ce problème sous forme d'un graphe

Q2 - Que peut-on dire des cours {1,2,3,4}?

Q3 - Quelle est la durée minimale pour organiser ces épreuves?

Algorithme de coloration Welsch & Powell

- Solution : Exercice 12

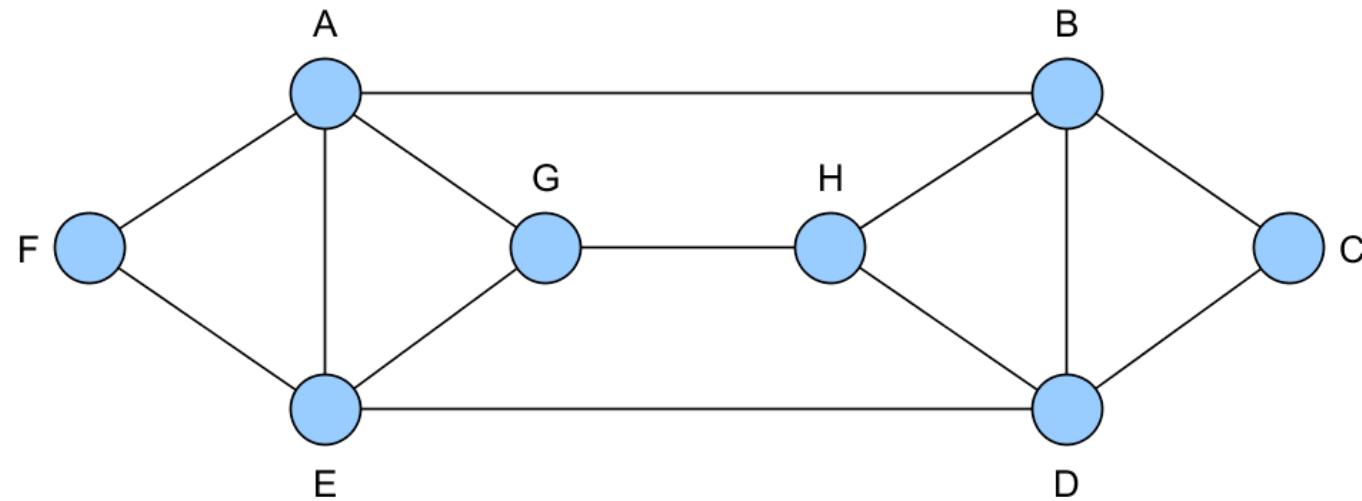


La durée minimale pour organiser ces épreuves est 4 jours :

J1 : 1-7-5 , J2 : 2-6 , J3 : 4 , J4 : 3

■ Exercice 13

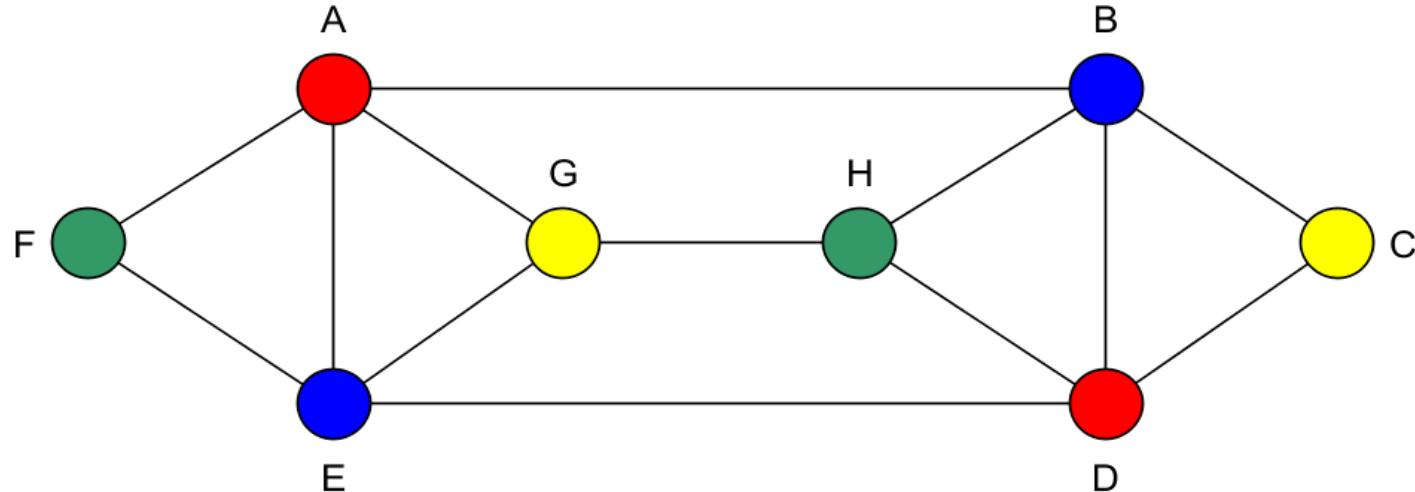
Considérons le graphe non orienté donc la représentation sagittale est la suivante :



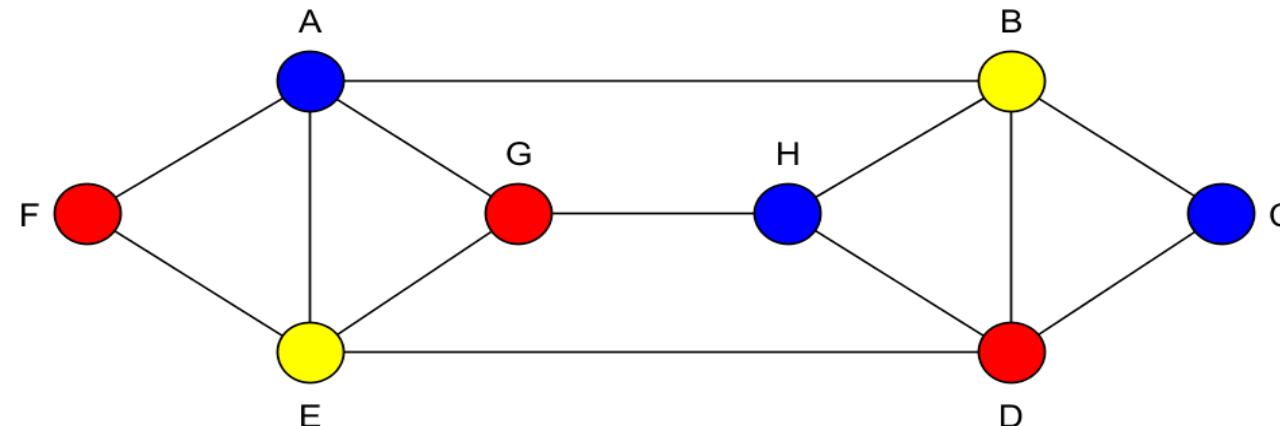
1. Appliquer sur ce graphe l'algorithme de Welsh et Powell. Combien de couleurs utilise-t-on ?
2. Trouver "à la main" une meilleure coloration de ce graphe, ce qui prouvera que l'algorithme de Welsh et Powell n'est pas toujours optimal

■ Corrigé:

1. On obtient la coloration suivante qui utilise 4 couleurs :



2. On trouve facilement une coloration utilisant seulement 3 couleurs :



Ordonnancement d'un graphe

Méthode MPM

□ Ordonnancement d'un graphe

- Faire l'ordonnancement d'un projet **consiste à organiser ce projet en respectant les contraintes d'antériorité des tâches tout en minimisant la durée totale de réalisation.**
- De façon plus précise, la problématique est donc la suivante : étant donné un **projet** à réaliser constitué de différentes **tâches** ayant chacune une certaine **durée**, dans quel **ordre** et à quel **moment** les exécuter afin de **minimiser la durée totale du projet**.
- Le but est donc d'obtenir un **calendrier d'exécution**.
- Les **données** du problème sont :
 - La **liste des tâches**.
 - La **durée** de chaque tâche.
 - Les **contraintes d'antériorité** entre les différentes tâches.

□ Ordonnancement d'un graphe

- Exemple : Les données d'un problème d'ordonnancement
- Il existe plusieurs algorithmes possibles pour résoudre les problèmes d'ordonnancement.
- Les deux plus fréquemment utilisés sont :
 - la méthode **PERT** (en anglais program evaluation and review technique) (méthode américaine).
 - la méthode **MPM** (La méthode des potentiels métra) (méthode française) que nous exposerons dans ce cours.

Tâches	Tâches antérieures	Durées
A	-	10
B	C, E	11
C	F	3
D	F, H	8
E	A, I	6
F	A	5
G	D	7
H	A	4
I	-	9

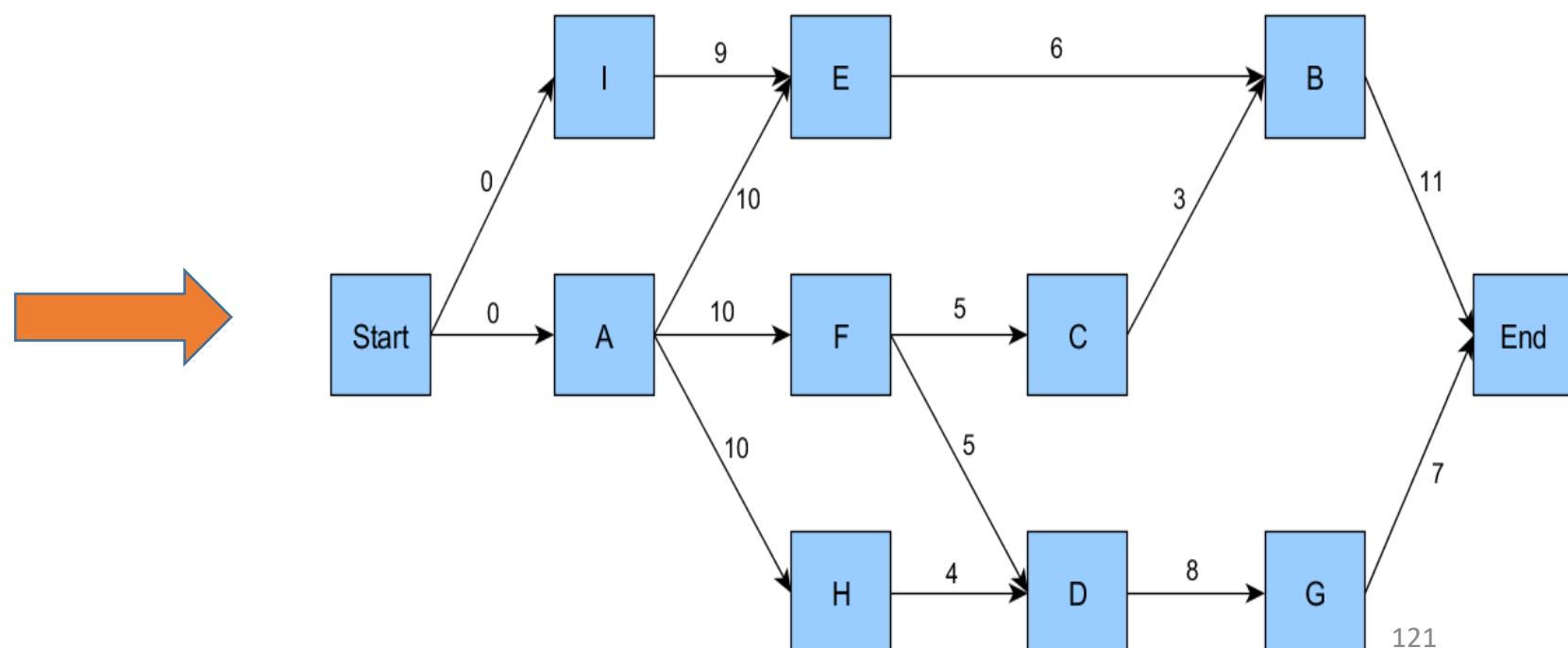
❑ Ordonnancement d'un graphe

- Définition

Un **graphe M.P.M**, aussi appelé **graphe étapes**, est un graphe où chaque tâche est représentée par un sommet, et où des arcs valués modélisent les contraintes d'antériorité ainsi que les durées des tâches.

- **Exemple:** Graphe M.P.M associé au problème d'ordonnancement énoncé dans l'exemple précédent est le suivant :

Tâches	Tâches antérieures	Durées
A	-	10
B	C, E	11
C	F	3
D	F, H	8
E	A, I	6
F	A	5
G	D	7
H	A	4
I	-	9



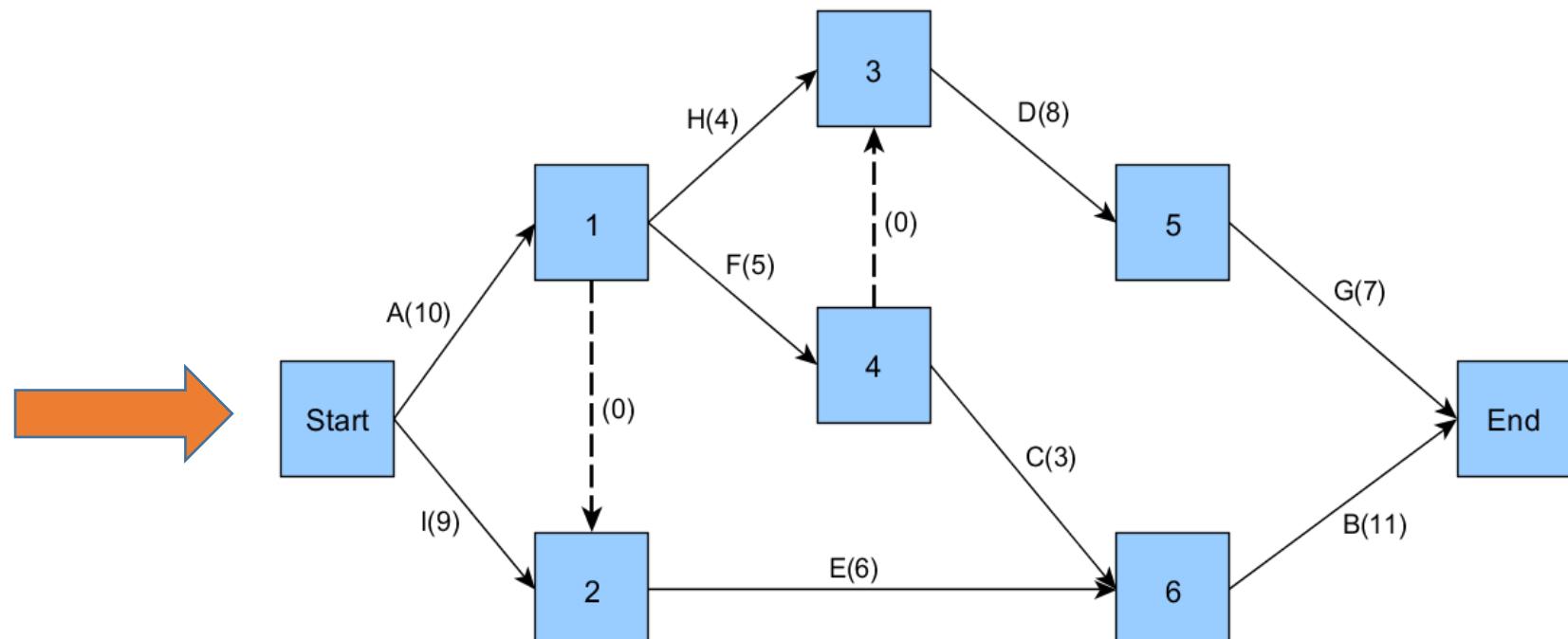
❑ Ordonnancement d'un graphe

- Définition

Un **graphe P.E.R.T**, aussi appelé **graphe étapes**, est un graphe où chaque tâche est représentée par un arc valué avec sa durée, et où les sommets représentent les états d'avancement du projet.

- Exemple : Le graphe P.E.R.T associé au problème d'ordonnancement énoncé dans l'exemple précédent est le suivant :

Tâches	Tâches antérieures	Durées
A	-	10
B	C, E	11
C	F	3
D	F, H	8
E	A, I	6
F	A	5
G	D	7
H	A	4
I	-	9



□ Ordonnancement d'un graphe

- **Exemple:** Pour illustrer la méthode MPM, nous allons réaliser l'ordonnancement d'un projet fictif, comprenant des tâches que nous noterons A, B, C, D, E, F et G. les contraintes d'antériorité et les durées de ces tâches sont consignées dans ce tableau.

Tache	Durée (en jours)	tâches antérieures
A	3	Aucune
B	2	Aucune
C	4	A
D	5	A, B
E	3	C
F	4	C, D
G	3	E, F

- Une entreprise a procédé à la définition d'un certain nombre de tâches à effectuer. On a rempli la colonne « antériorité » avec les tâches qui doivent être exécutées avant celle considérée.

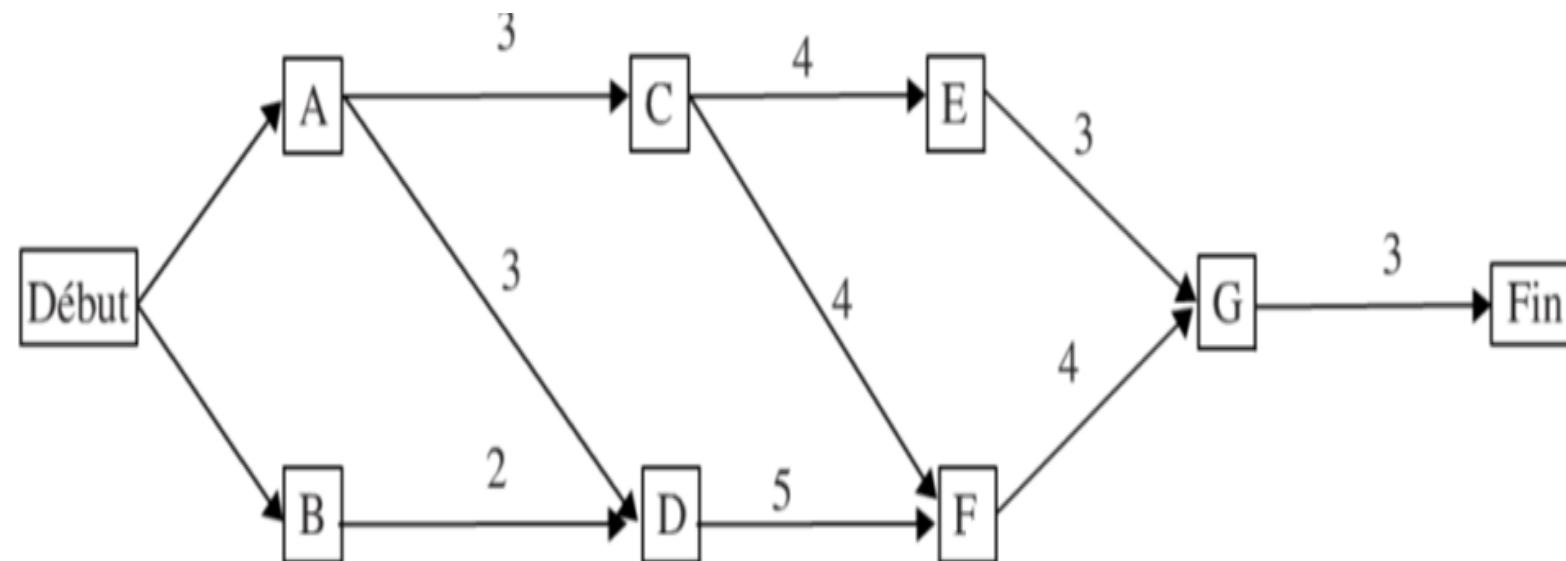
❑ Ordonnancement d'un graphe

- Etape 1: définir le niveau de chaque tâche. Il est simple de voir que A et B sont de niveau 0, C et D de niveau 1, E et F de niveau 2 et G de niveau 3

Niveaux	0	1	2	3
Sommets	A, B	C, D	E, F	G

- Le graphe ordonné par niveaux est celui de cette figure

- On peut remarquer que l'on a rajouté deux tâches fictives: **Début et fin**; et que l'on a pondéré le graphe en ajoutant les durées de chaque tâche.



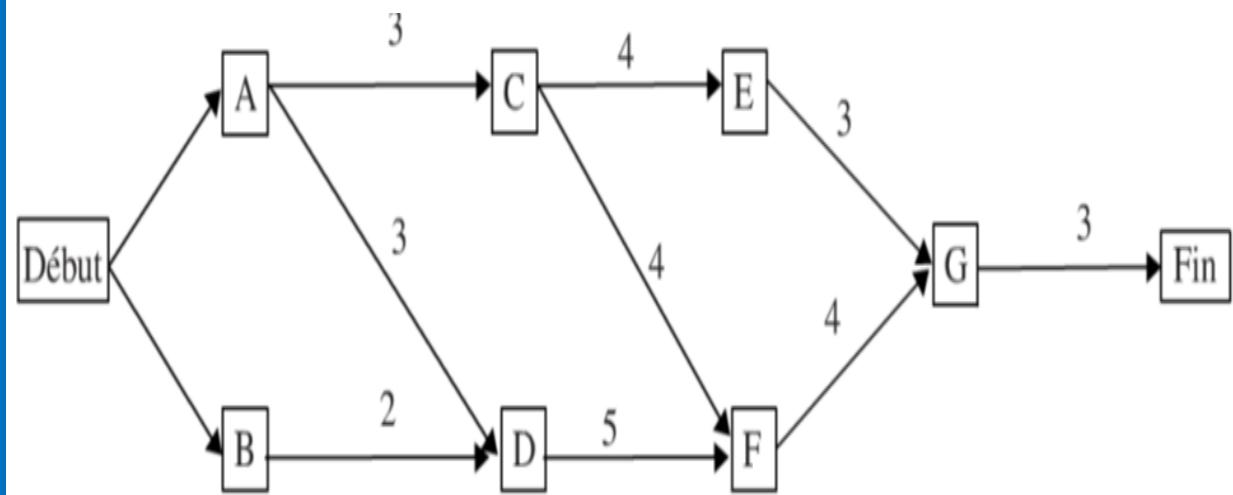
❑ Ordonnancement d'un graphe - date au plus tôt

- Il est plutôt simple de calculer la durée minimale du projet. Mais on peut se poser quelques questions:
 - A quel moment peut-on commencer une tâche ?
 - Est-ce qu'on retarder le moment de démarrer certaines tâches sans que cela ait d'impact sur la durée minimale du projet?
- Pour répondre à ces questions, nous allons définir, pour chaque tâches, les notions de **date au plus tôt** et de **date au plus tard**.
- **Date au plus tôt**
- **Définition**

La **date au plus tôt** d'une tâche est la date minimale à laquelle on peut commencer la tâche, car toutes antérieures sont terminées.

❑ Ordonnancement d'un graphe - date au plus tôt

- Nous noterons $t(x_j)$ la date au plus tôt d'une tâche x_j , $t(x_j)$ est le plus grand des nombres $t(x_i) + d(x_i)$ où x_i est une des tâches qui précédent immédiatement la tâche x_j et $d(x_i)$ à la durée d'une tâche x_i .
- **Exemple :** Calculons les dates au plus tôt des tâches du projet:



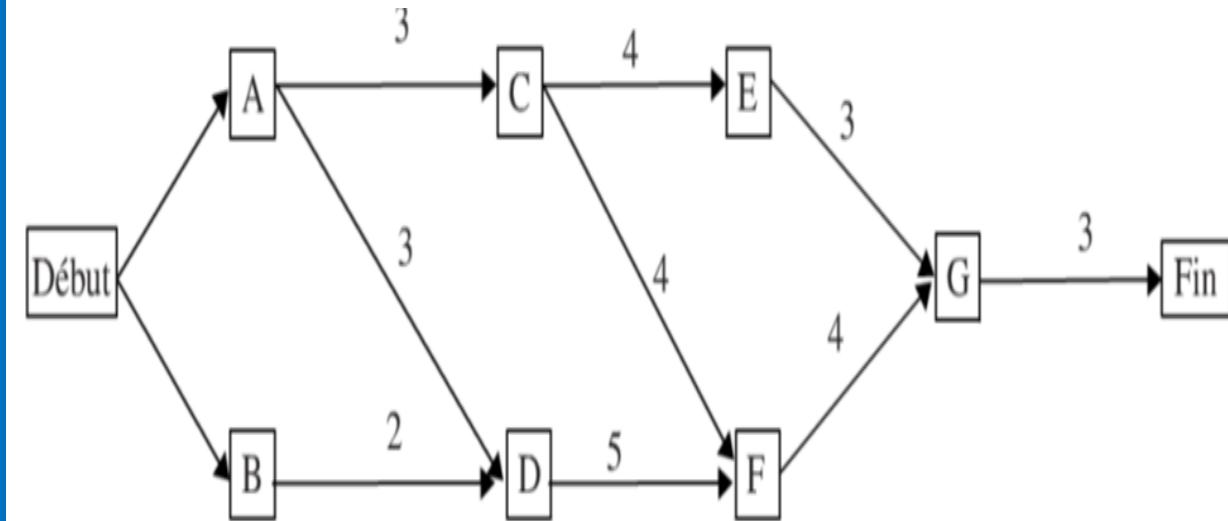
- $t(A) = t(B) = 0$
- $t(C) = t(A) + d(A) = 0 + 3 = 3$

Le sommet D a deux prédecesseurs : A et B. on calcule

- $t(A) + d(A) = 0 + 3 = 3$
- $t(B) + d(B) = 0 + 2 = 2.$

$t(D)$ est le plus grand de ces deux nombres donc $t(D) = 3.$

□ Ordonnancement d'un graphe - date au plus tôt



- Pour E on a $t(E) = t(C) + d(C) = 3 + 4 = 7$
- Pour F a deux prédecesseurs : C et D. On calcule :
 $t(C) + d(C) = 7$ et
 $t(D) + d(D) = 3 + 5 = 8.$
Le plus grand de ces deux nombres est 8 donc $t(F) = 8$.
- Pour G, on calcule
 $t(E) + d(E) = 7 + 5 = 10$ et
 $t(F) + d(F) = 8 + 4 = 12$
donc $t(G) = 12$.
- Pour finir, $t(\text{Fin}) = t(G) + d(G) = 12 + 3 = 15.$
- On a ainsi calculé la date au plus tôt de la fin du projet qui est de **15 jours**

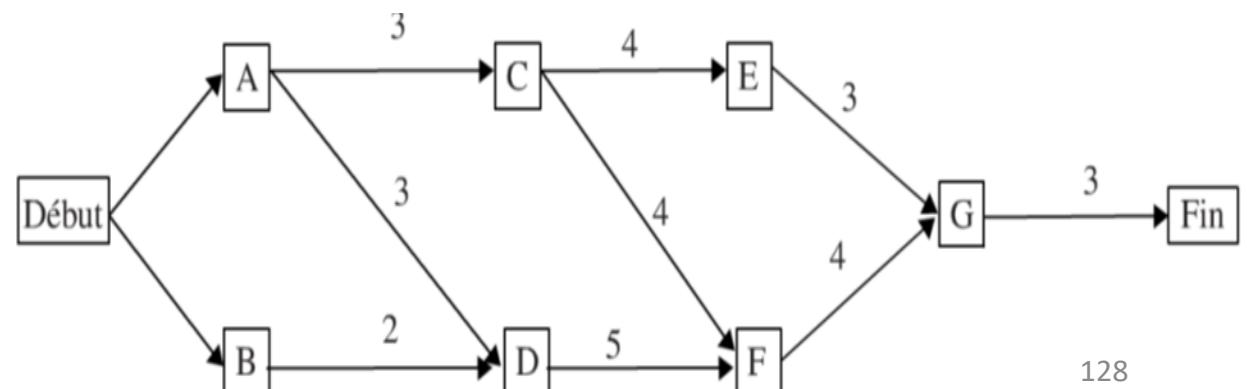
❑ Ordonnancement d'un graphe - date au plus tard

- Date au plus tard
- Définition

La **date au plus tard** d'une tâche est la date maximale à laquelle on peut commencer la tâche sans que cela ne repousse la date de fin du projet.

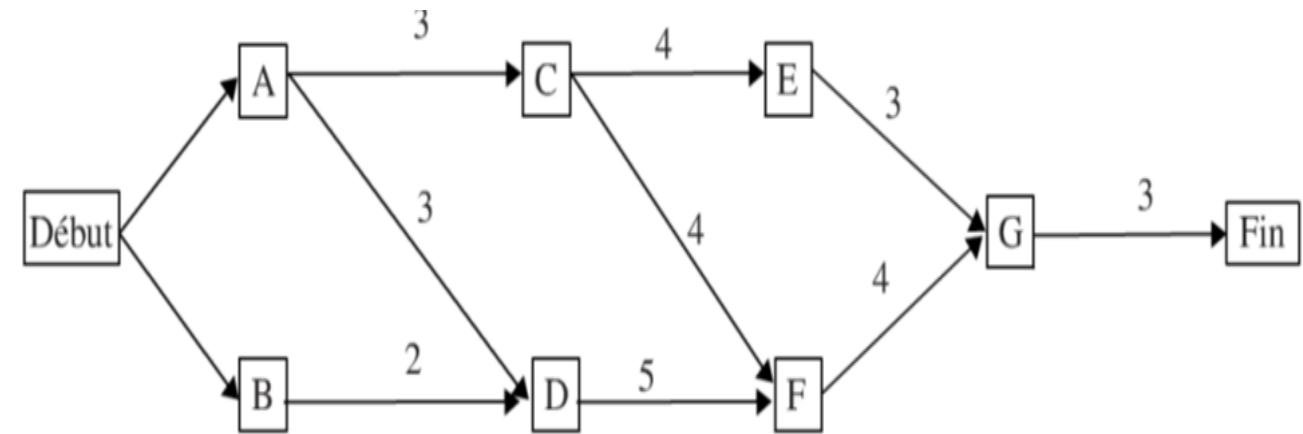
- Nous noterons $T(x_j)$ la **date au plus tard** d'une tâche x_j . $T(x_j)$ est le plus petit des nombres $T(x_k) - d(x_j)$ où x_k est une des tâches qui suit immédiatement la tâche x_j .
- **Exemple (suite):** Calculons les dates au plus tard des tâches du projet :

- On doit commencer par la fin puisqu'à chaque fois, on doit considérer les successeurs.



□ Ordonnancement d'un graphe - date au plus tard

- On a bien sur $T(\text{Fin}) = t(\text{Fin}) = 15$
 - $T(G) = T(\text{Fin}) - d(G) = 15 - 3 = 12$
 - $T(F) = T(G) - d(F) = 12 - 4 = 8$
 - $T(E) = T(G) - d(E) = 12 - 3 = 9$
 - $T(D) = T(F) - d(D) = 8 - 5 = 3$

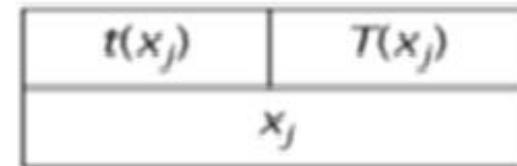


- Le sommet C a deux successeurs : E et F. On calcule $T(E) - d(C) = 9 - 4 = 5$ et $T(F) - d(C) = 8 - 4 = 4$. Le plus petit de ces deux nombres est 4, donc $T(C) = 4$.
- $T(B) = T(D) - d(B) = 3 - 2 = 1$
- Le sommet A a deux successeurs : C et D. On calcule :

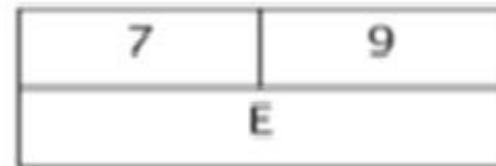
$$T(C) - d(A) = 4 - 3 = 1 \text{ et } T(D) - d(A) = 3 + 3 = 0.$$
 Donc $T(A) = 0$.

□ Ordonnancement d'un graphe : date au plus tôt-date au plus tard

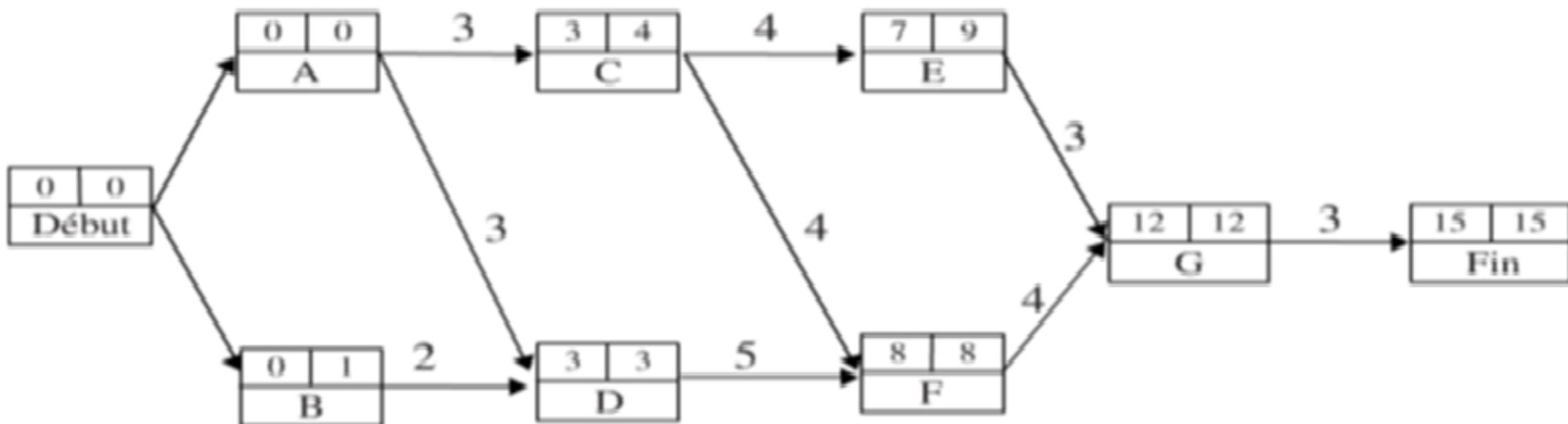
- Pour chaque tâche, on va rajouter les dates au plus tôt et au plus tard en présentant chaque tâche par un rectangle :



Par exemple, pour la tâche E :

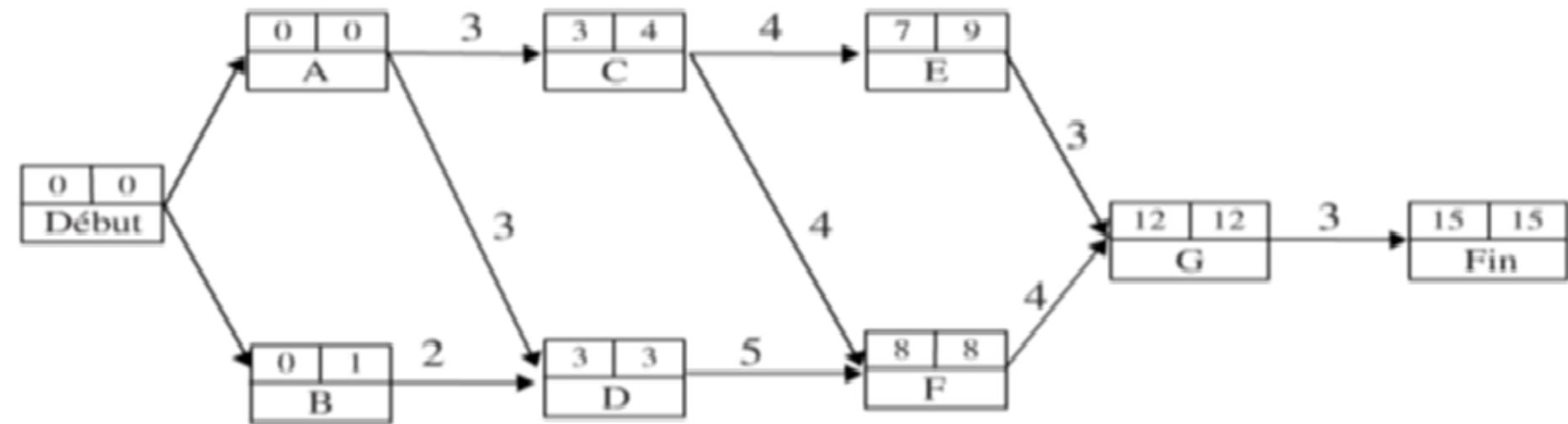


- Le graphe MPM devient :



□ Ordonnancement d'un graphe : date au plus tôt-date au plus tard

- Examinons la tâche C: on a $t(C) = 3$ et $T(C) = 4$. Au plus tôt, on peut commencer à 3 jours et au plus tard à 4 jours. Pour cette tâche, on dispose donc d'une « liberté » d'un jour. Ce n'est ainsi pour d'autres tâches (comme D par exemple) pour lesquelles les dates au plus tôt et au plus tard sont les mêmes.



- Ces tâches sont qualifiés de tâches critiques et l'enchaînement A-D-F-G-Fin, qui ne contient que des tâches critiques, est qualifié de chemin critique.

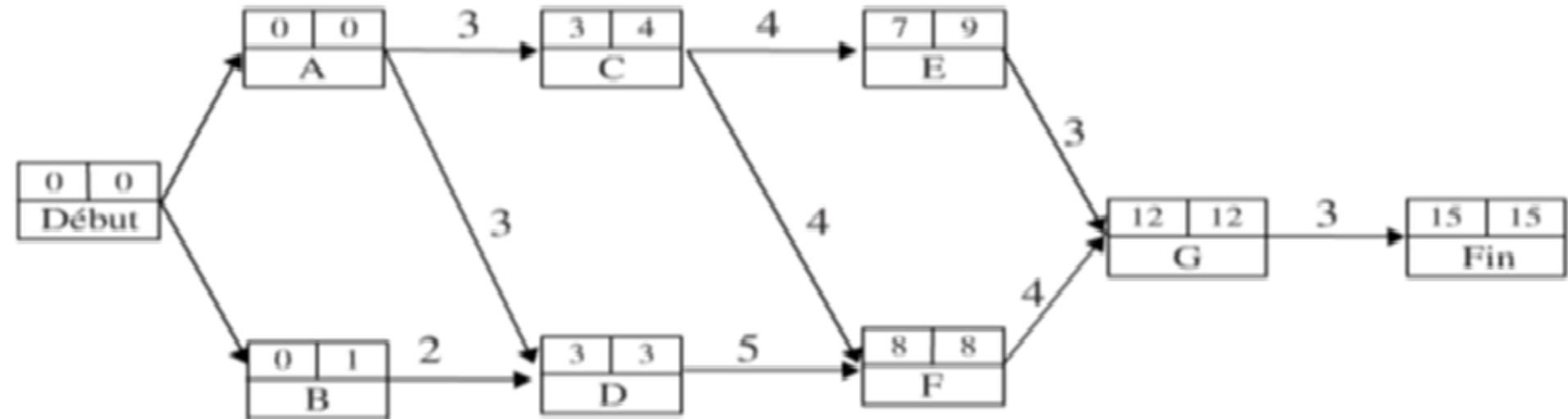
□ Ordonnancement d'un graphe

▪ Définitions :

- Une **tâche critique** est une tâche dont les dates au plus tôt et au plus tard sont égales. Donc x_j est critique si et seulement si $t(x_j) = T(x_j)$.
- Un **chemin critique** est un chemin reliant le début à la fin et qui n'est constitué que des tâches critiques.
- La marge totale d'une tâche c'est le retard maximum que l'on peut accepter sur la date de début de la tâche sans que cela ne retarde pas la date de fin du projet. La marge totale d'une tâche x_j se note $MT(x_j)$ et elle vaut $MT(x_j) = T(x_j) - t(x_j)$.
- **La marge totale d'une tâche critique est nulle.**

□ Ordonnancement d'un graphe

- Exemple (suite)



- La tâche C n'est pas critique. On peut constater qu'on peut retarder le début de C d'un jour sans retarder la date de fin du projet (et c'est le maximum). En effet, si C commence à la date 4, cela n'empêcherait pas F de commencer à la date 8, E pourrait commencer à la date 8 et n'empêcherait pas G de commencer à la date 12.
- Ce retard maximum s'appelle la marge totale de la tâche.

❑ Exercice 14

La mise en service d'un nouvel équipement routier demande la réalisation d'un certain nombre de tâches. Le tableau représente ces différentes tâches avec leurs relations d'antériorité.

Tâches	A	B	C	D	E	F	G
Durées	6	3	6	2	4	3	1
Tâches antérieures	Aucune	Aucune	Aucune	B	B	A, D	C, E, F

- A) déterminer le niveau de chacune tâches.
- B) Construire le graphe d'ordonnancement du projet et calculer les dates au plus tôt et au plus tard de chaque tâche.
- C) Déterminer le chemin critique. Quelle est la durée minimale de réalisation du projet?
- D) Calculer la marge totale de la tâche E. quelle est sa signification?

❑ Exercice 15

La réalisation d'un projet nécessite plusieurs tâches successives dont les durées en jours sont données dans le tableau suivant, ainsi que les tâches devant être réalisées antérieurement.

Tâches	A	B	C	D	E	F	G	H	I	J
Durées	4	2	2	1	2	5	3	3	3	4
Tâches antérieures	aucune	aucune	A	A	A, B	C	D, E	E, G	H	F,I

- A) Déterminer le niveau de chacune des tâches.
- B) Construire le graphe d'ordonnancement du projet et calculer les dates au plus tôt et au plus tard de chaque tâche.
- C) Déterminer le chemin critique. Quelle est la durée minimale de réalisation du projet?
- D) En réalité, la tâche C a nécessité une durée de 5 jours. Est-ce que cela a eu une incidence sur la durée de réalisation du projet?

❑ Exercice 16

Un réalisateur des films est confronté au problème de planning de son prochain film selon les tâches suivantes :

1. Déterminer les **dates au plus tôt** et les **dates au plus tard** de chaque tâche.
2. Déterminer le temps minimum de réalisation de l'ensemble
3. Proposer un chemin critique.
4. Préciser les marges totales de chaque tâche.

Tâche	description	durée (j)	antériorités
A	écriture du scénario	30	-
B	choix et recrutement des comédiens	12	15 jours après le début de A
C	choix du lieu de tournage	8	20 jours après le début de A
D	découpage technique	4	A et C doivent être terminées
E	préparation des décors	7	C et D doivent être terminées
F	tournage des extérieurs	10	A, B, C et D doivent être terminées
G	tournage des intérieurs	12	D, E et F doivent être terminées
H	synchronisation	3	F et G doivent être terminées
I	montage	14	H doit être terminée
J	accompagnement sonore	7	ne peut commencer que 3 jours après le début de I et après la fin de H
K	mixage	6	I et J doivent être terminées
L	tirage de la copie zéro	1	ne peut commencer que 2 jours après la fin de K

❑ Manipulation d'ordonnancement d'un graphe avec Gantt Project (TP)

- **Objectif**: le but de ce TP est de présenter une manipulation d'un graphe d'ordonnancement de manière synthétique à l'aide du logiciel **GanttProject**
 - Télécharger et installer **GanttProject**.

Votre tâche est modéliser un projet de création d'un site web « ensaf.ac.ma ». Il est composé de 3 parties:

- Créer un projet avec le nom (Nom étudiant), puis sélectionner les jours fériés et les week-ends.
- Créer les étapes du projet;
- Créer les 3 ressources (Chef de projet, Développeur et Graphite)
- Afficher le diagramme de Gantt des taches;
- Afficher le diagramme de Gantt des ressources

Manipulation d'ordonnancement d'un graphe avec Gantt Project (TP)

Tâches	Désignation	Durée	Tâches antérieures
A	Maquette du site	7J	
B	Ressources images sons et textes.	14J	A
C	Création Logo	7J	A
D	Création du fichier index.html	7J	A
E	Création du fichier styles.css	7J	D
F	Mise à jour du fichier index.html	7J	B, E
G	Création de l'en-tête	14J	F, C
H	Création du pied de page	14J	F
I	Création du menu	14J	F
J	Création des fichiers include	7J	G, H, I
K	Resources images et textes	42J	A
L	Création du fichier galerie.php	14J	J, K
M	Création du fichier visites.php	14J	J, K

Parcours de graphes

□ Parcours de graphes

- Beaucoup de problèmes sur les graphes nécessitent que l'on parcourt l'ensemble des sommets et des arcs/arêtes du graphe. Nous allons étudier dans ce chapitre les deux principales stratégies d'exploration :
 - **le parcours en largeur** consiste à explorer les sommets du graphe niveau par niveau, à partir d'un sommet donné ;
 - **le parcours en profondeur** consiste, à partir d'un sommet donné, à suivre un chemin le plus loin possible, puis à faire des retours en arrière pour reprendre tous les chemins ignorés précédemment.

□ Parcours de graphes

- Dans les deux cas, l'algorithme procède par coloriage des sommets :
 - Initialement, **tous les sommets sont blancs**. On dira qu'un sommet blanc **n'a pas encore été découvert**.
 - Lorsqu'un sommet est "**découvert**" (autrement dit, quand on arrive pour la première fois sur ce sommet), il est colorié en **gris**. Le sommet reste gris tant qu'il reste des successeurs de ce sommet qui sont blancs (autrement dit, qui n'ont pas encore été découverts).
 - Un sommet est **colorié en noir** lorsque tous ses successeurs sont **gris ou noirs** (autrement dit, lorsqu'ils ont tous été découverts).

□ Parcours de graphes

- De façon pratique, on va utiliser une **liste “d’attente au coloriage en noir”** dans laquelle on va stocker tous **les sommets gris** :
 - Un sommet **est mis** dans la liste d’attente dès qu’il est **colorié en gris**.
 - Un sommet gris dans la liste d’attente **peut faire rentrer dans la liste ses successeurs qui sont encore blancs** (en les coloriant en gris).
 - Quand tous les successeurs d’un sommet gris de la liste d’attente sont soit gris soit noirs, il est colorié en noir et **il sort de la liste d’attente**.
- La différence fondamentale entre **le parcours en largeur** et **le parcours en profondeur** provient de la façon de **gérer cette liste d’attente au coloriage en noir** :
 - Le parcours en largeur utilise une **file d’attente**.
 - Le parcours en profondeur utilise **une pile**.

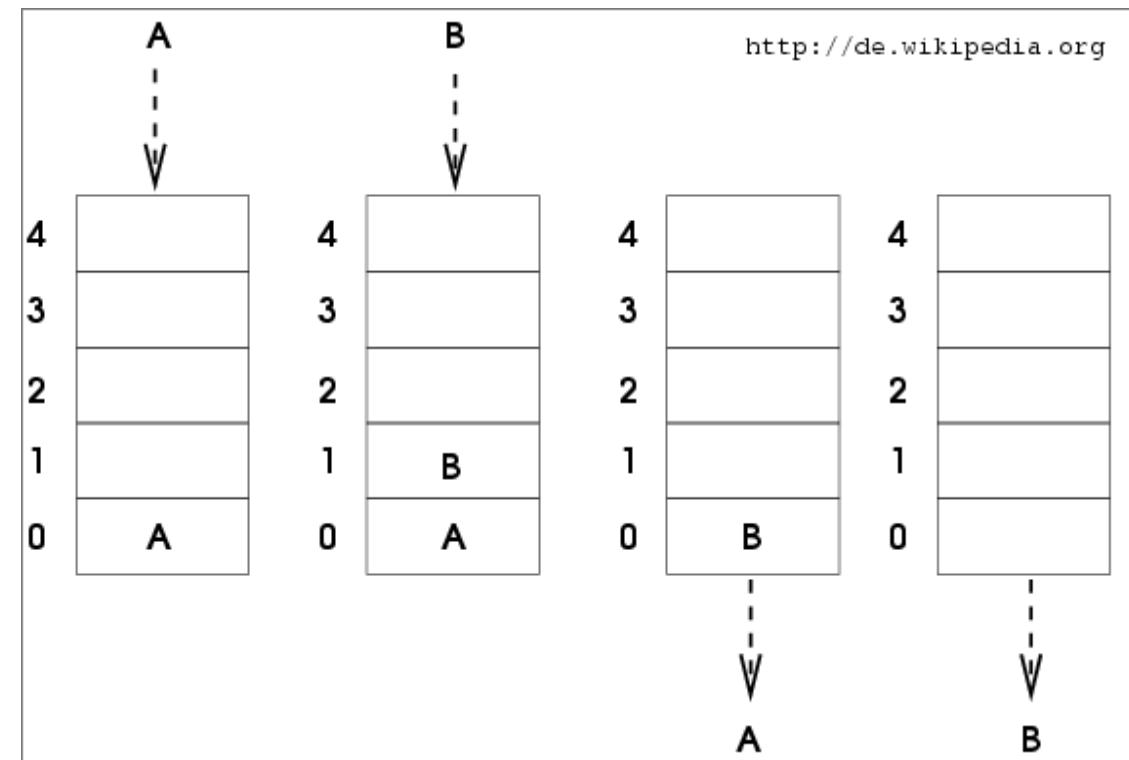
□ File (queue)

La structure de file est celle d'une file d'attente à un guichet :

- Les nouvelles personnes qui arrivent se rangent à la fin de la file d'attente.
- La personne servie est celle qui est arrivée en premier dans la file.

Structure FIFO (first in, first out).

- On ajoute les éléments « en queue » de la file
- On retire les éléments « en tête » de la file



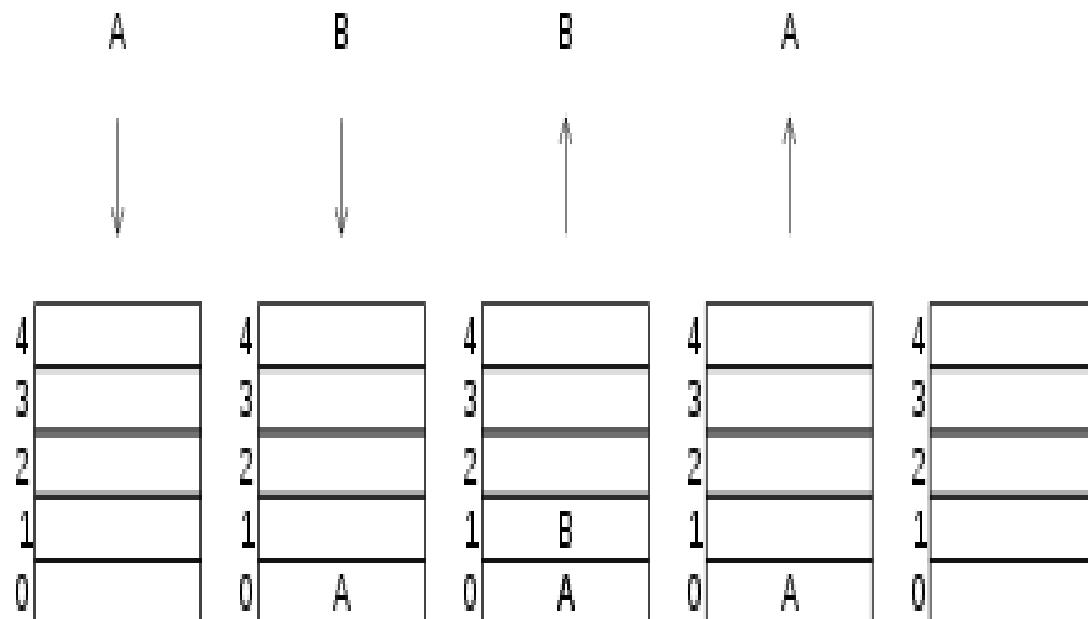
□ Pile (Stack)

La structure de pile est celle d'une pile d'assiettes :

- Pour ranger les assiettes, on les empile les unes sur les autres.
- Lorsqu'on veut utiliser une assiette, c'est l'assiette qui a été empilée en dernier qui est utilisée.

Structure LIFO (last in, first out)

- On ajoute les éléments « par le haut » de la pile
- On retire les éléments « par le haut » de la pile



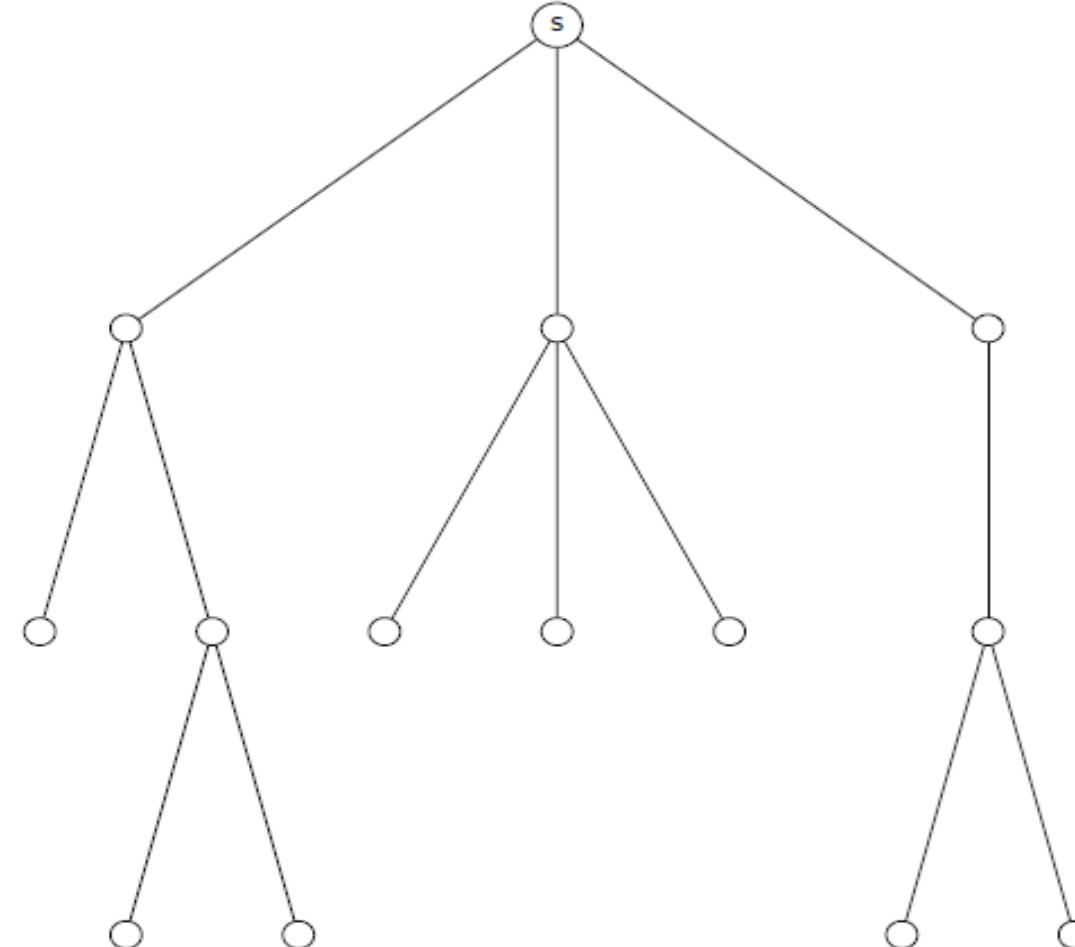
□ Parcours en largeur

■ Principe de l'algorithme

- Un parcours en largeur explore le graphe à partir **d'un sommet donné** (sommet de départ ou sommet source).
- L'algorithme permet de calculer **les distances de tous les sommets accessibles depuis le sommet source**.
- L'algorithme simule **la transmission d'un message** à partir d'un sommet source, en utilisant l'idée suivante :
 - Tout sommet qui reçoit le message, le transférera à tous ses voisins qui ne l'auront pas encore reçu.

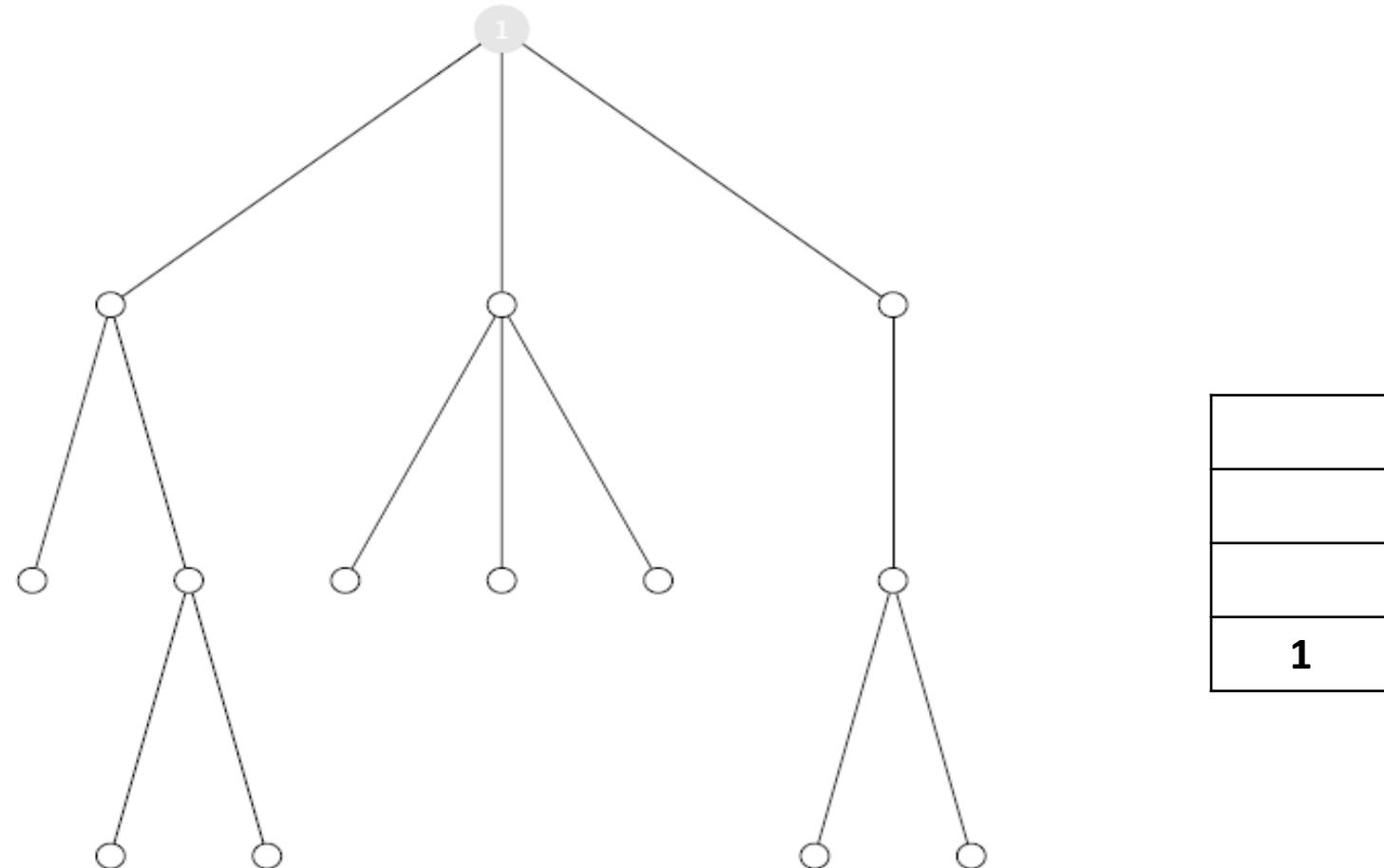
□ Parcours en largeur d'un arbre

- **Exemple** : Parcourir en largeur le graphe ci-dessous à partir du sommet s :



□ Parcours en largeur d'un arbre

- Exemple

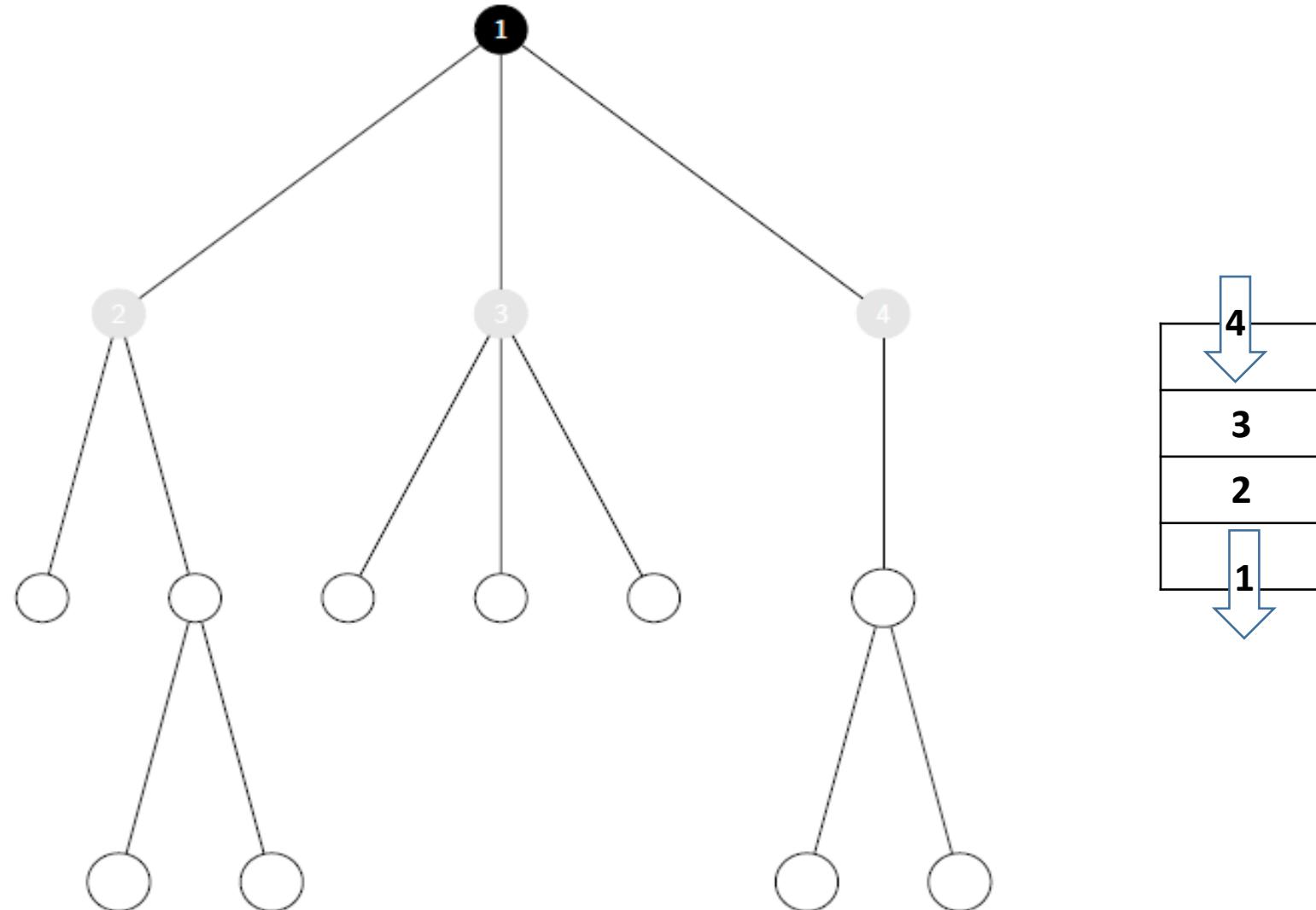


Enfiler : passage en gris. Défiler : passage en noir.

L'ordre pour enfiler les voisins (ni gris, ni noirs) dépend de l'implantation.

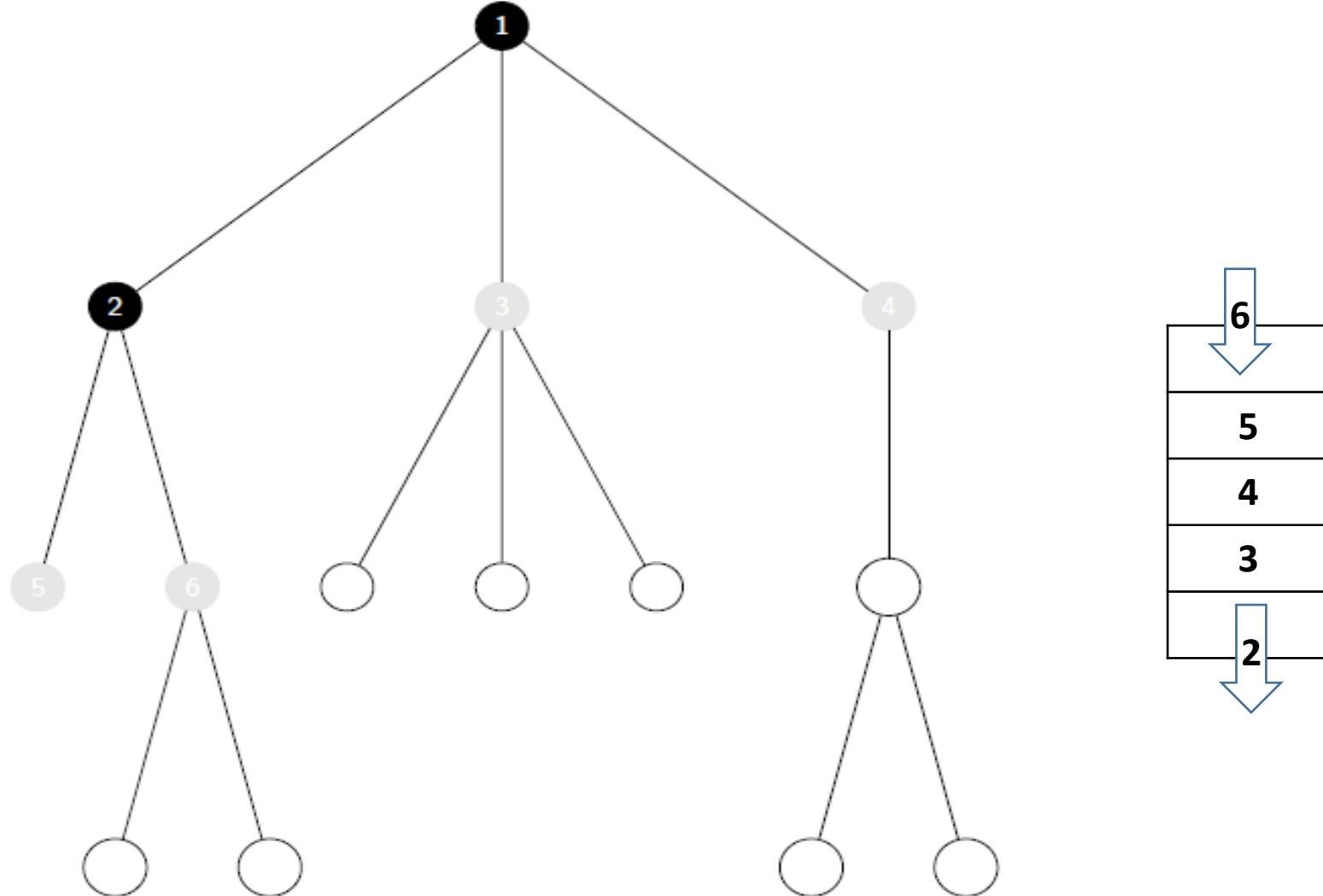
□ Parcours en largeur d'un arbre

- Exemple



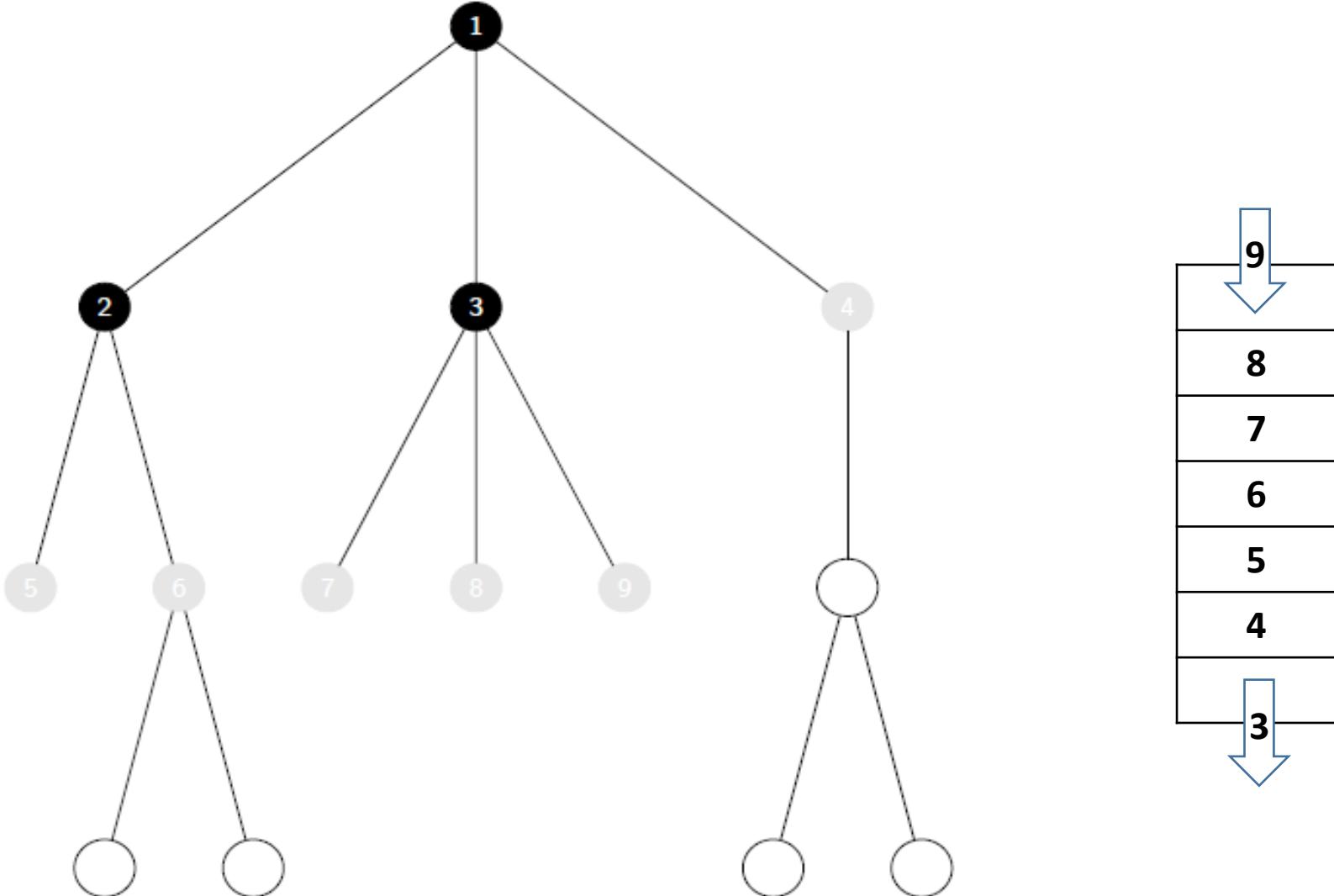
□ Parcours en largeur d'un arbre

- Exemple



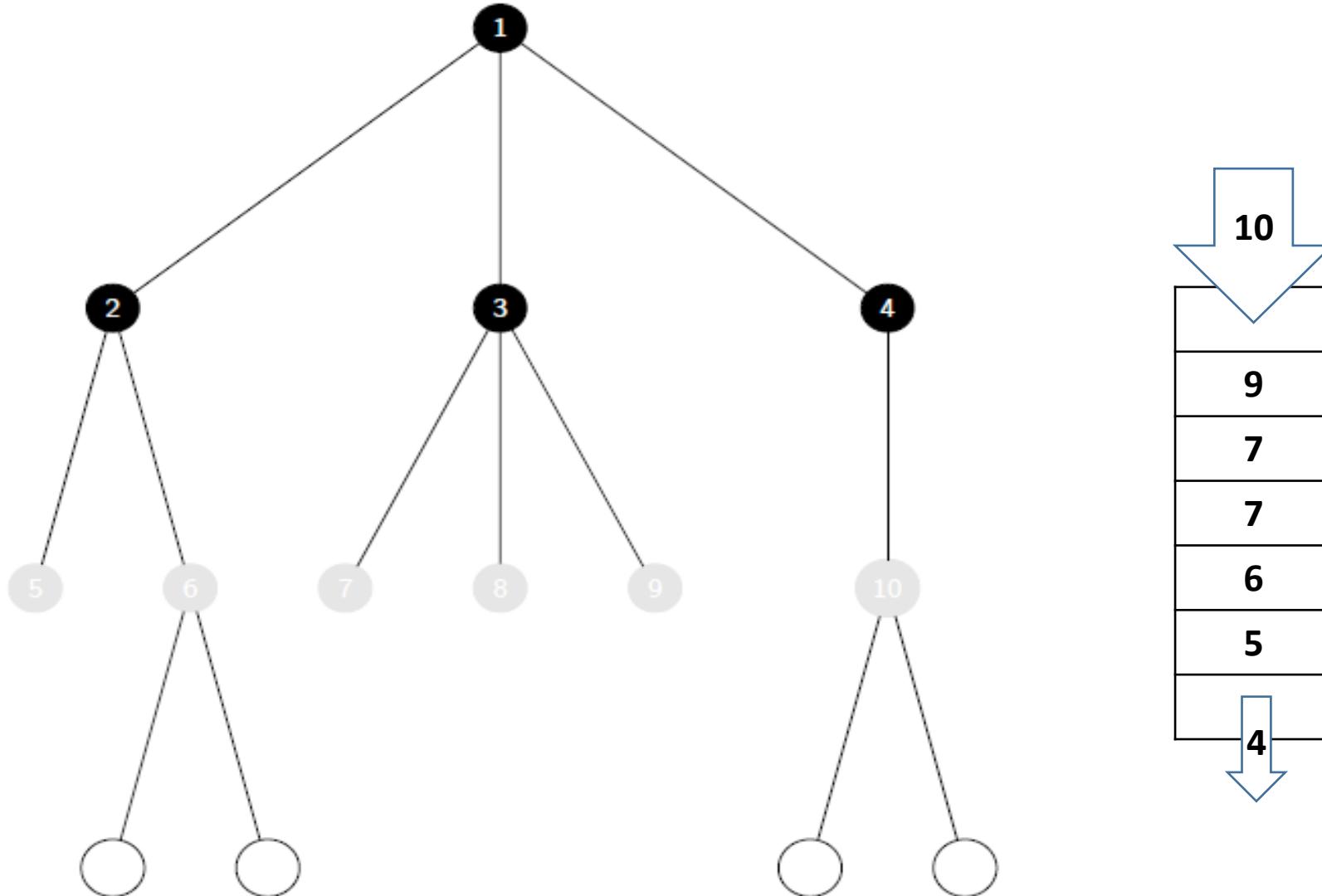
□ Parcours en largeur d'un arbre

- Exemple



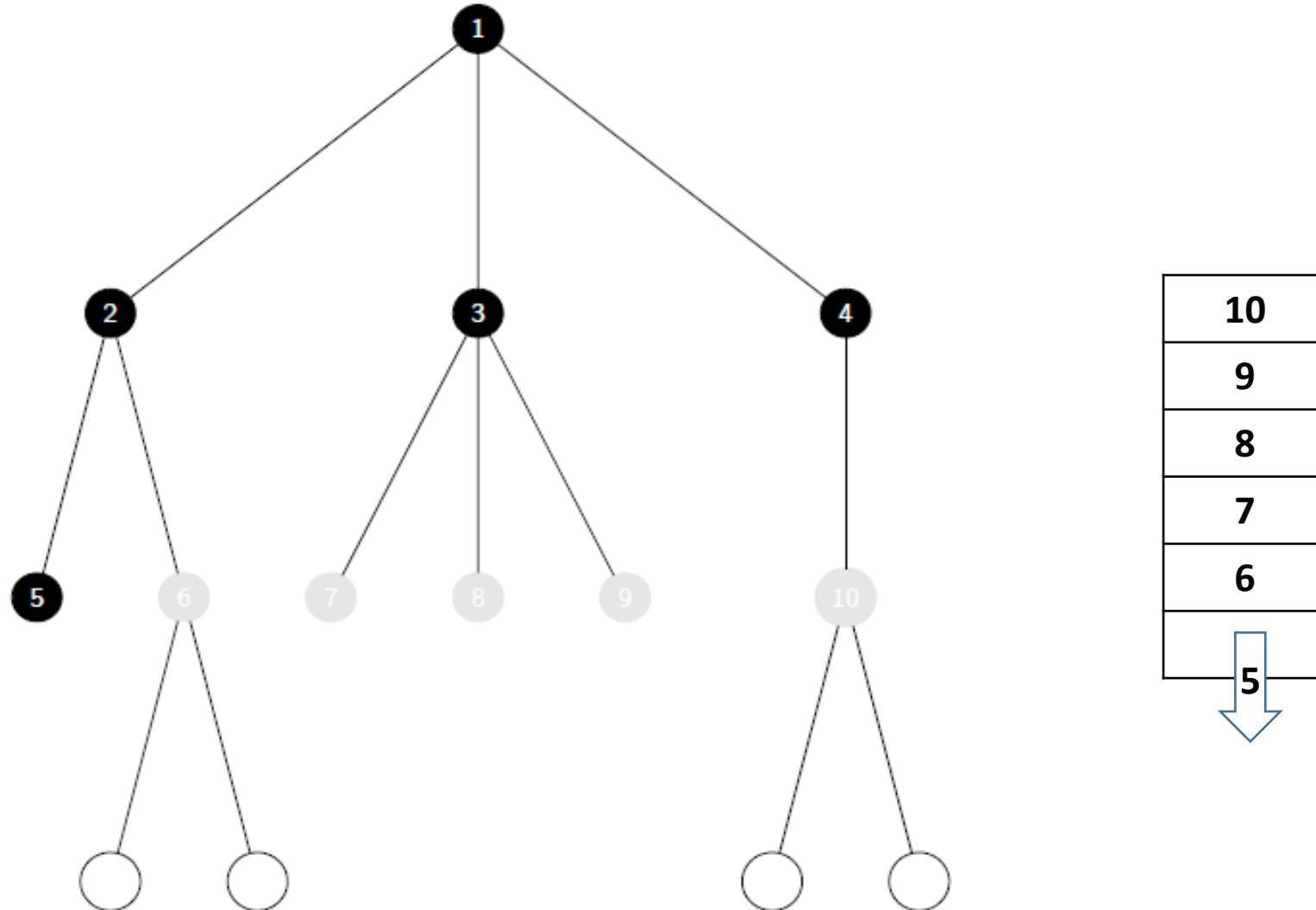
Parcours en largeur d'un arbre

- Exemple



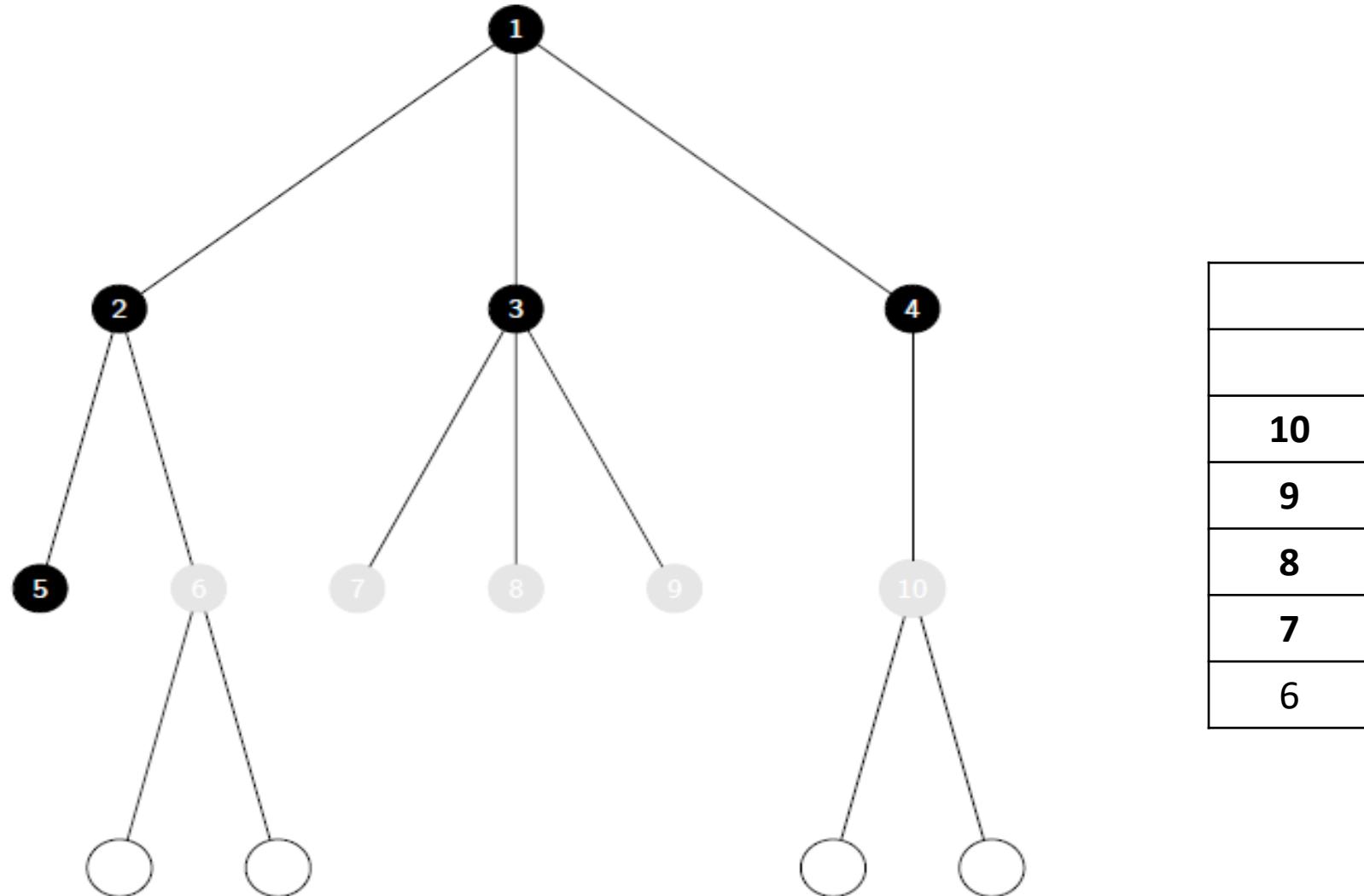
□ Parcours en largeur d'un arbre

- Exemple



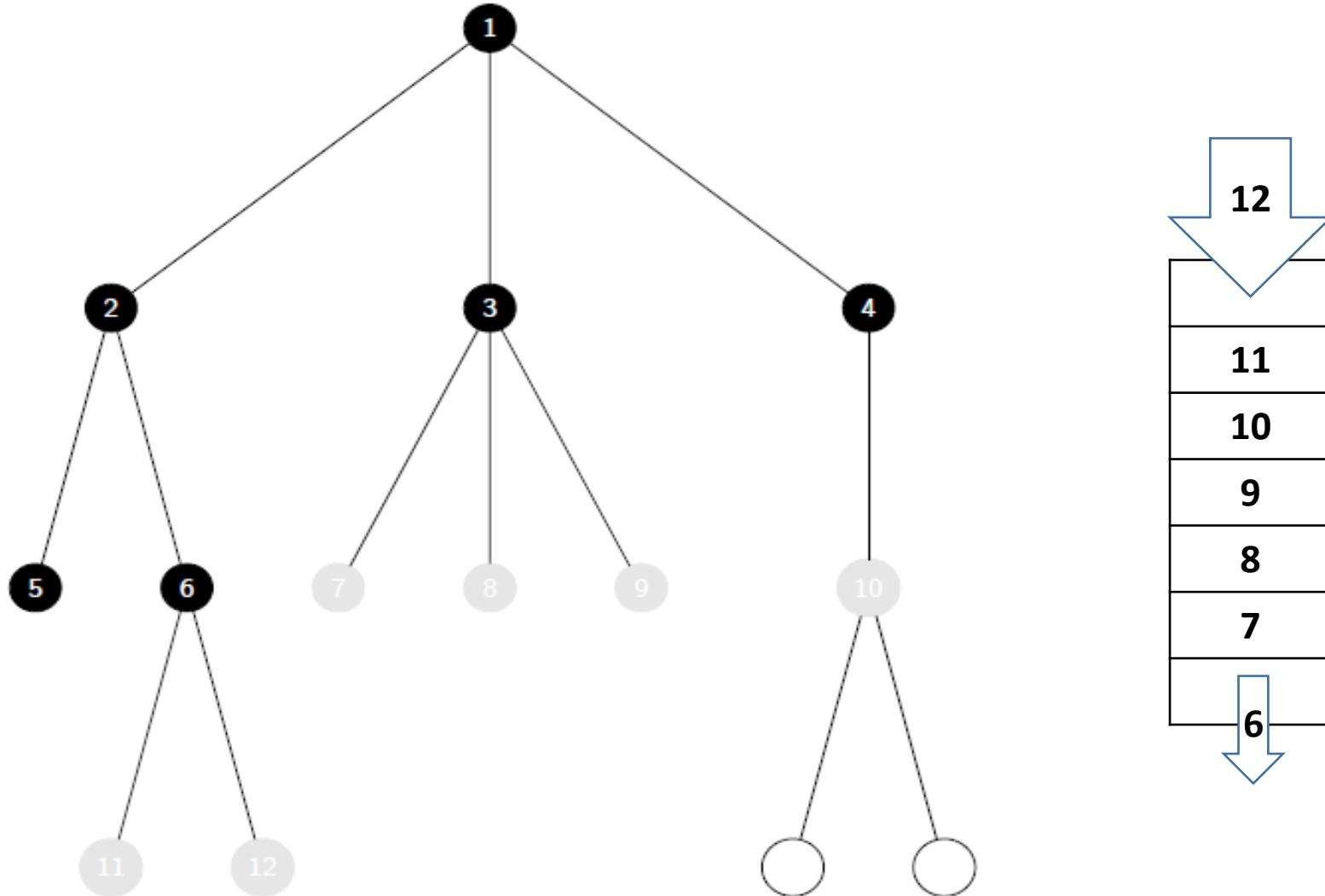
□ Parcours en largeur d'un arbre

- Exemple



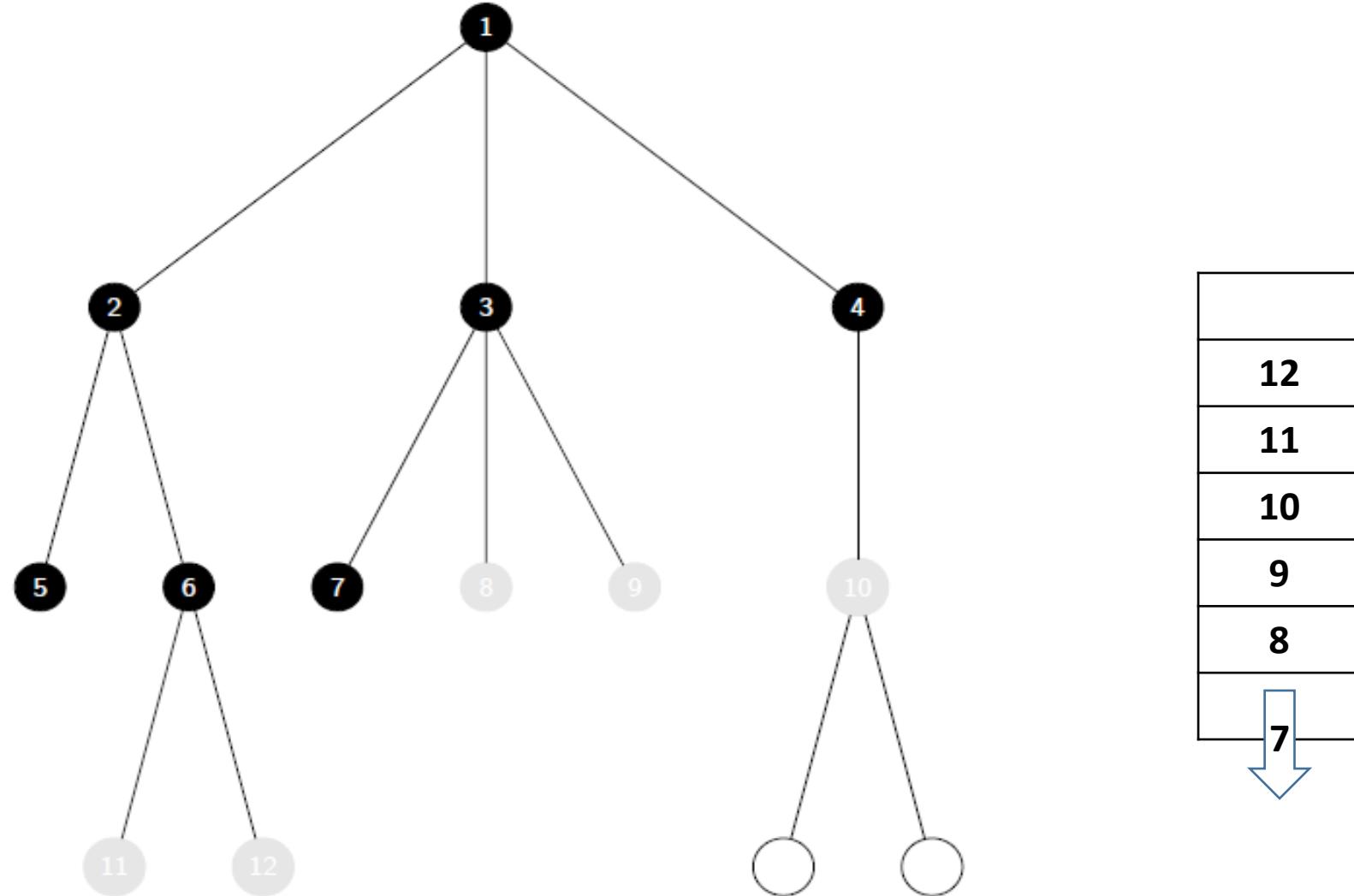
□ Parcours en largeur d'un arbre

- Exemple



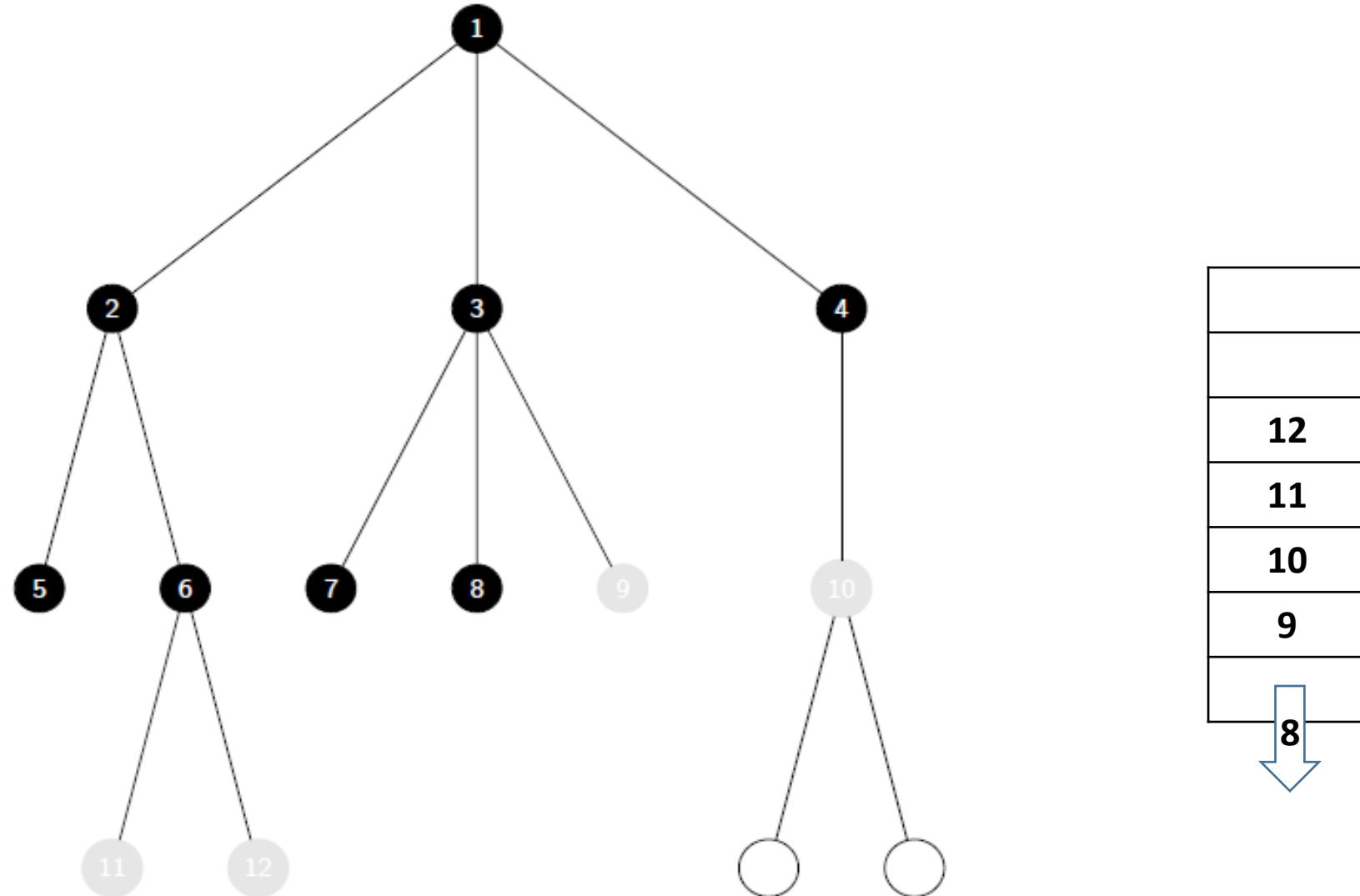
□ Parcours en largeur d'un arbre

- Exemple



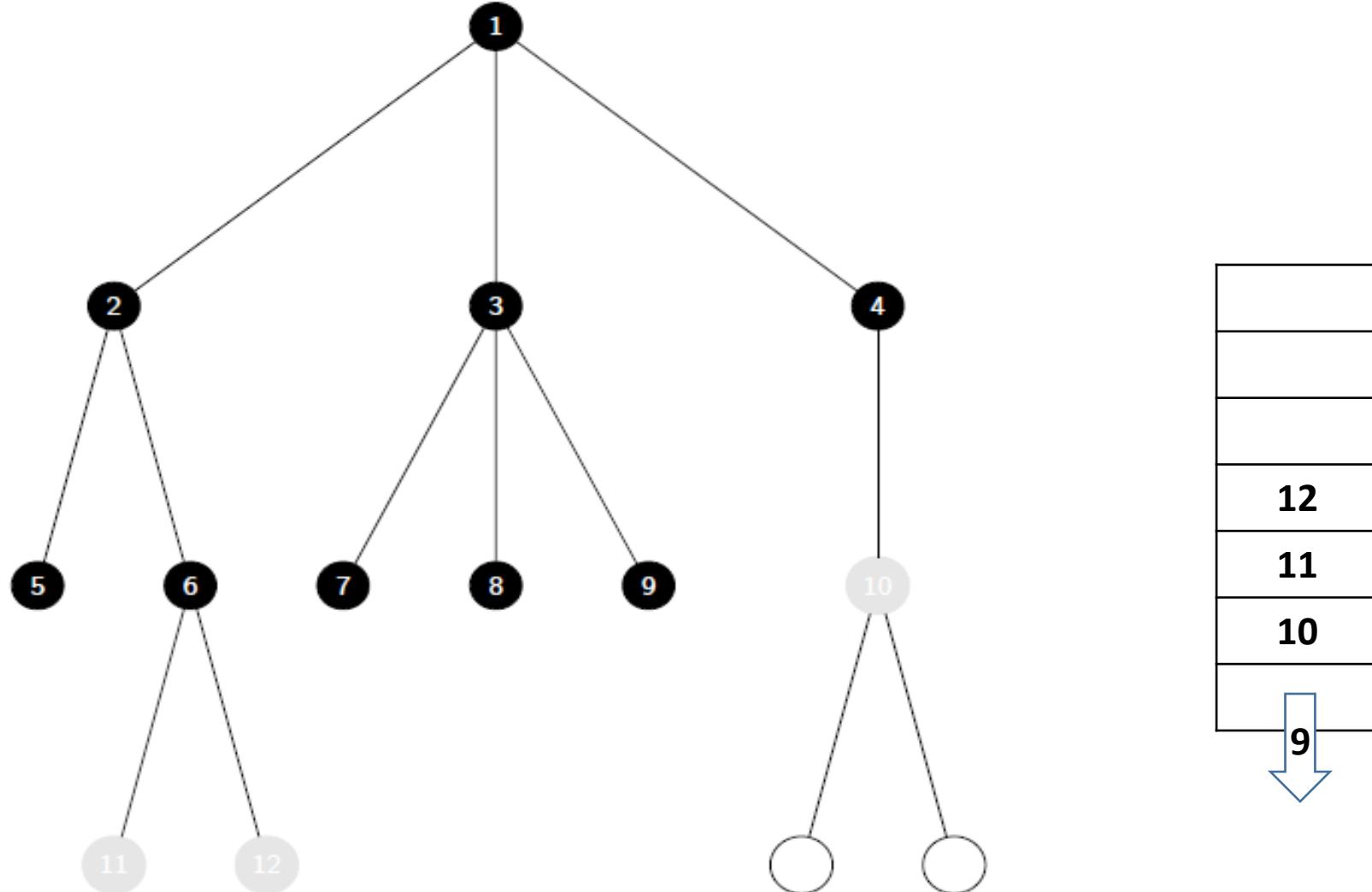
□ Parcours en largeur d'un arbre

- Exemple



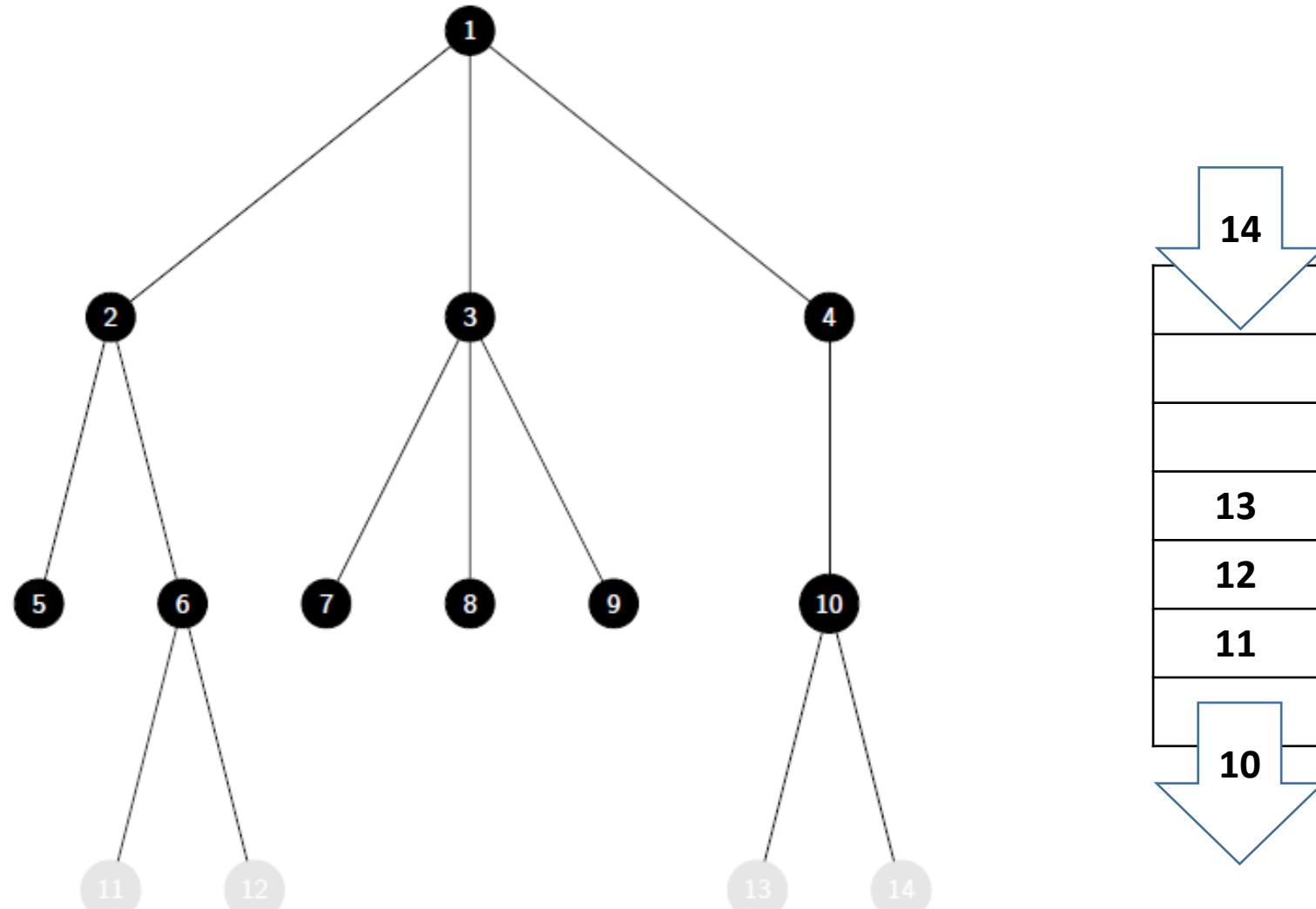
□ Parcours en largeur d'un arbre

- Exemple



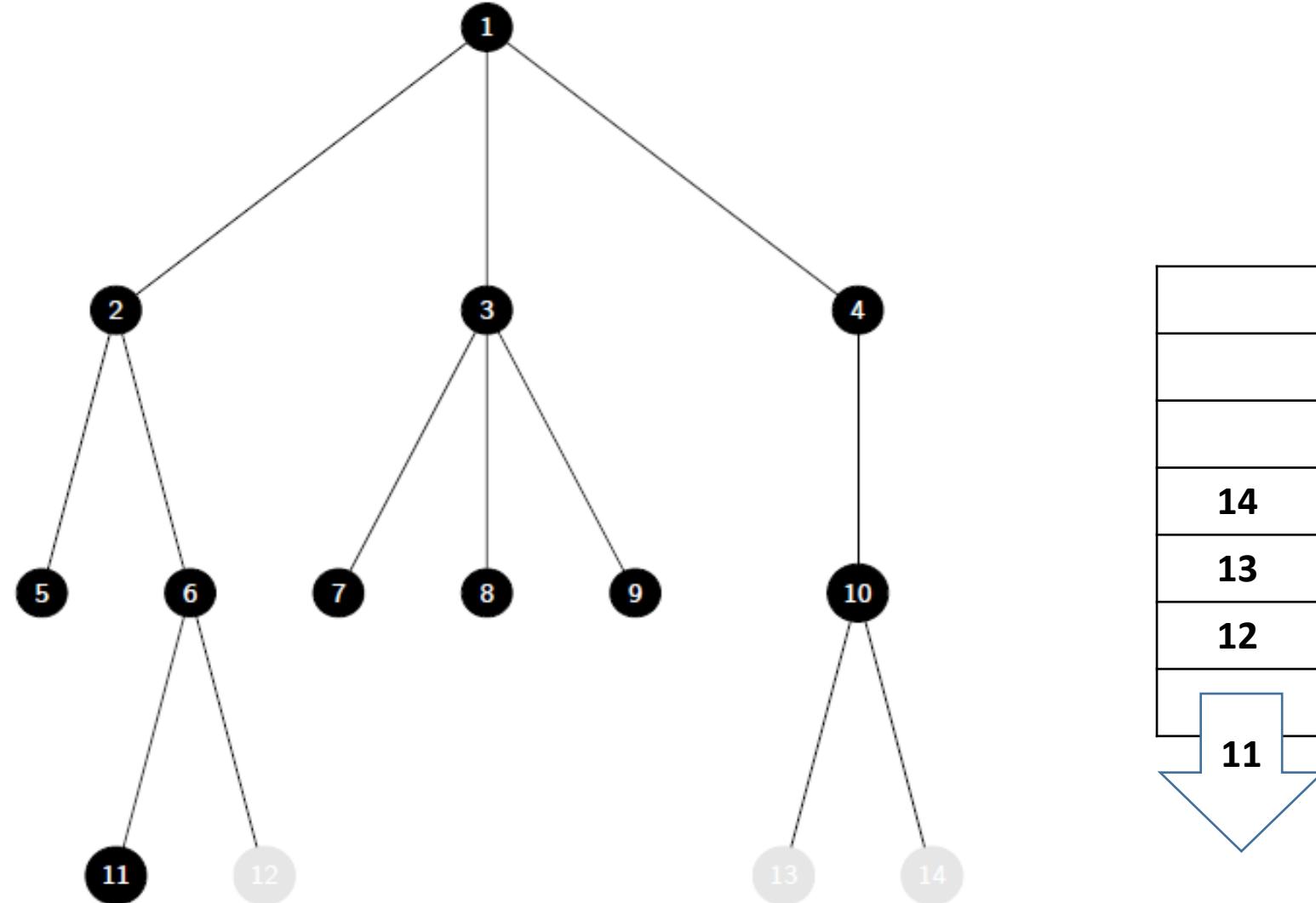
□ Parcours en largeur d'un arbre

- Exemple



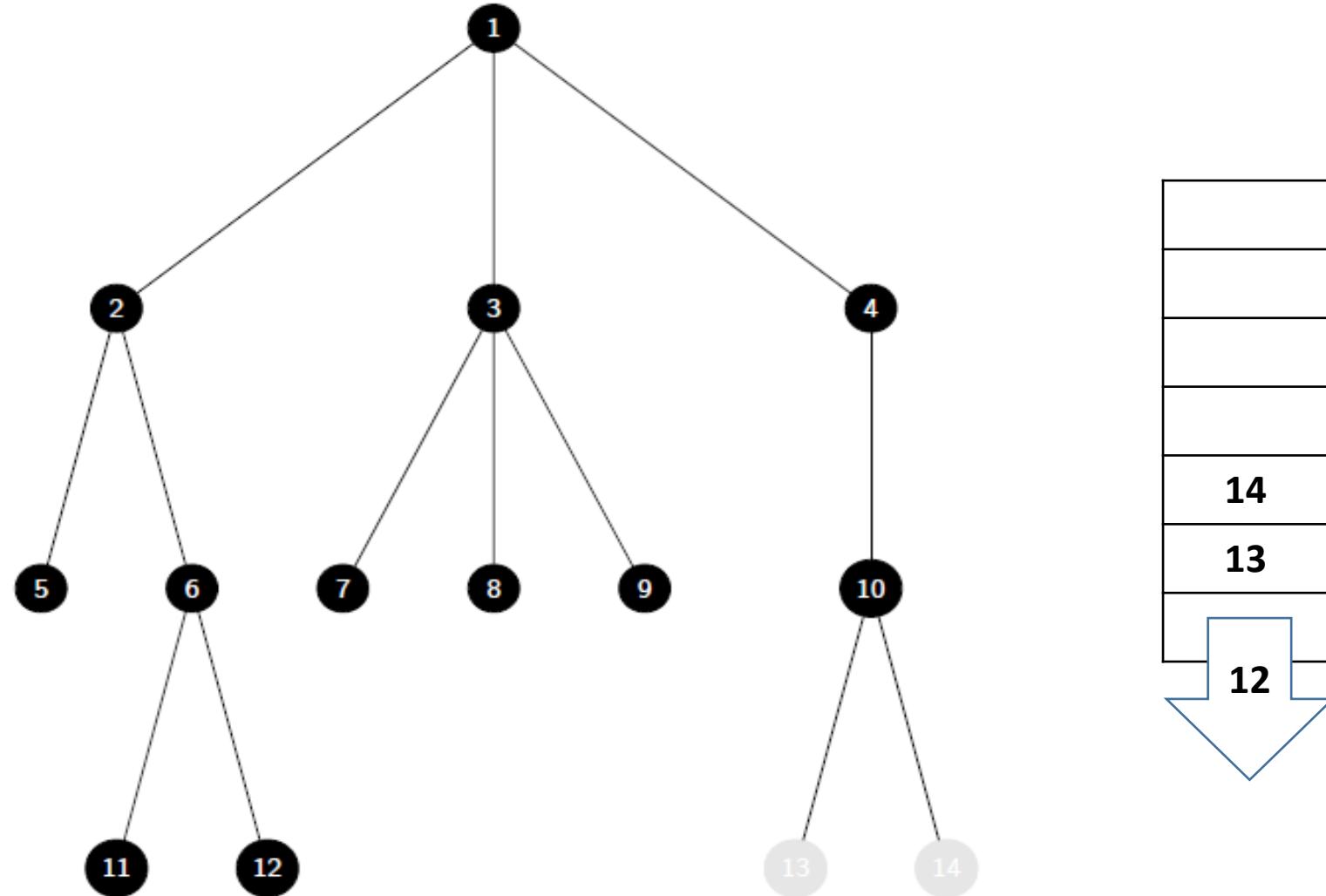
□ Parcours en largeur d'un arbre

- Exemple



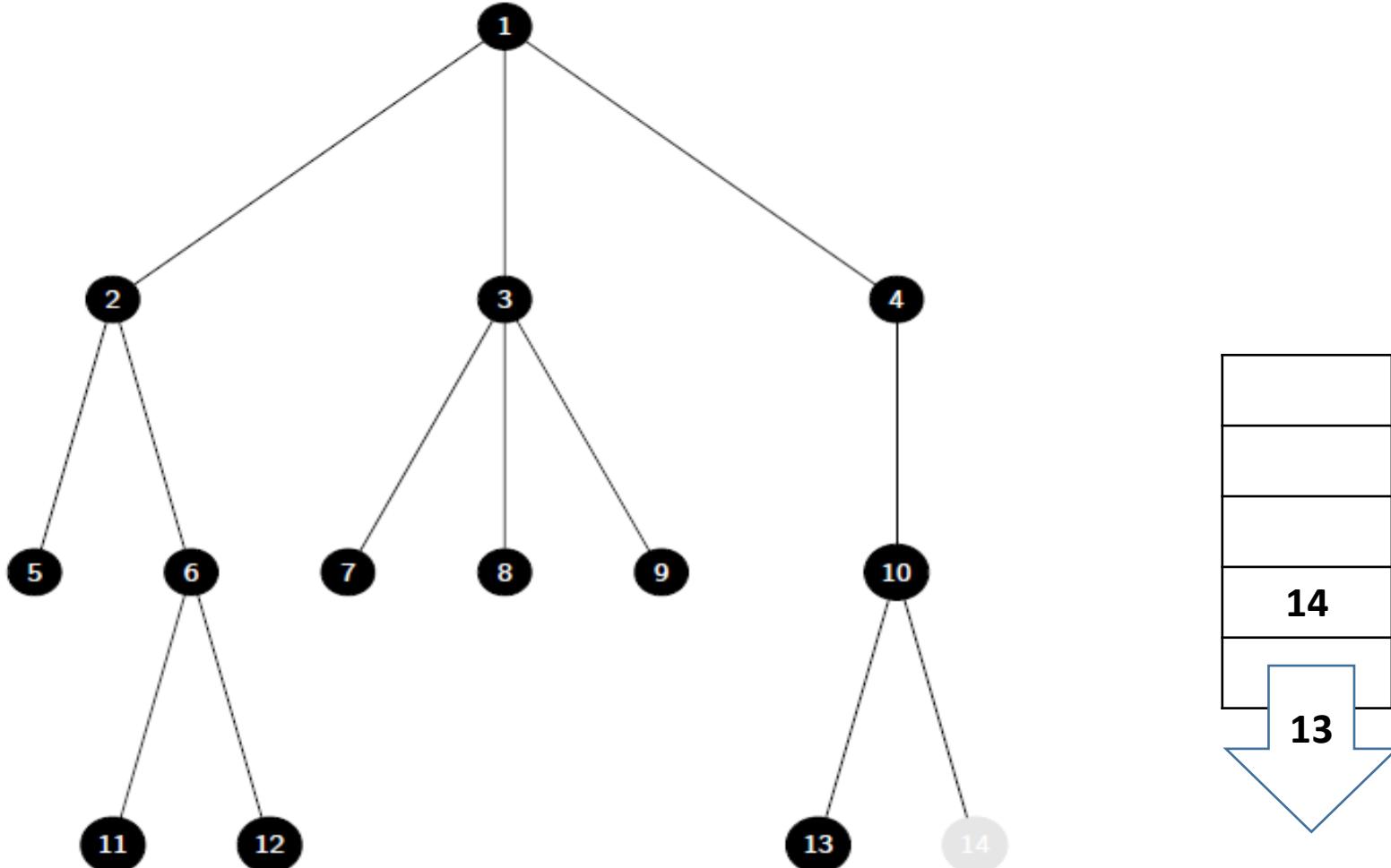
□ Parcours en largeur d'un arbre

- Exemple



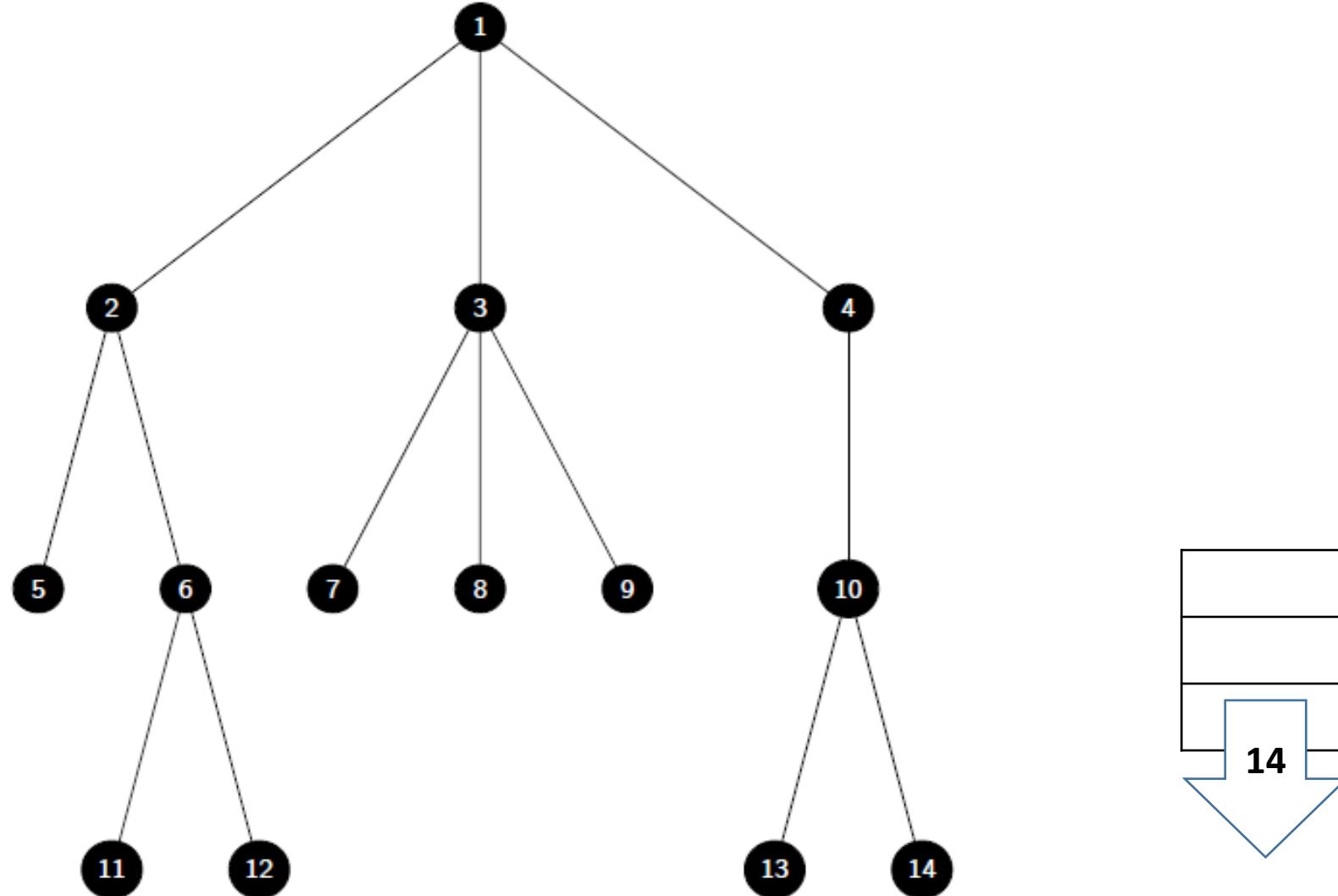
□ Parcours en largeur d'un arbre

- Exemple



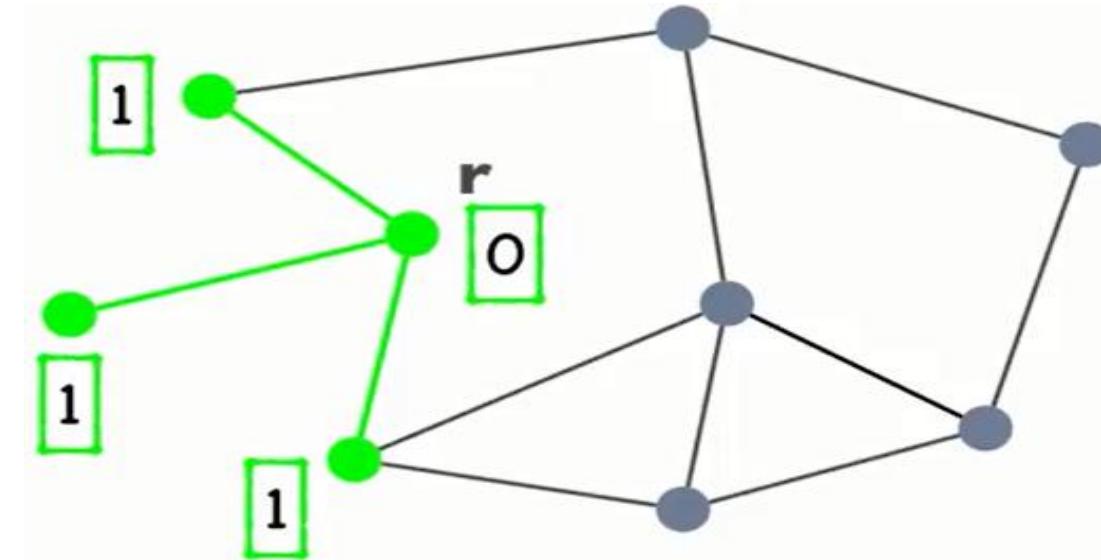
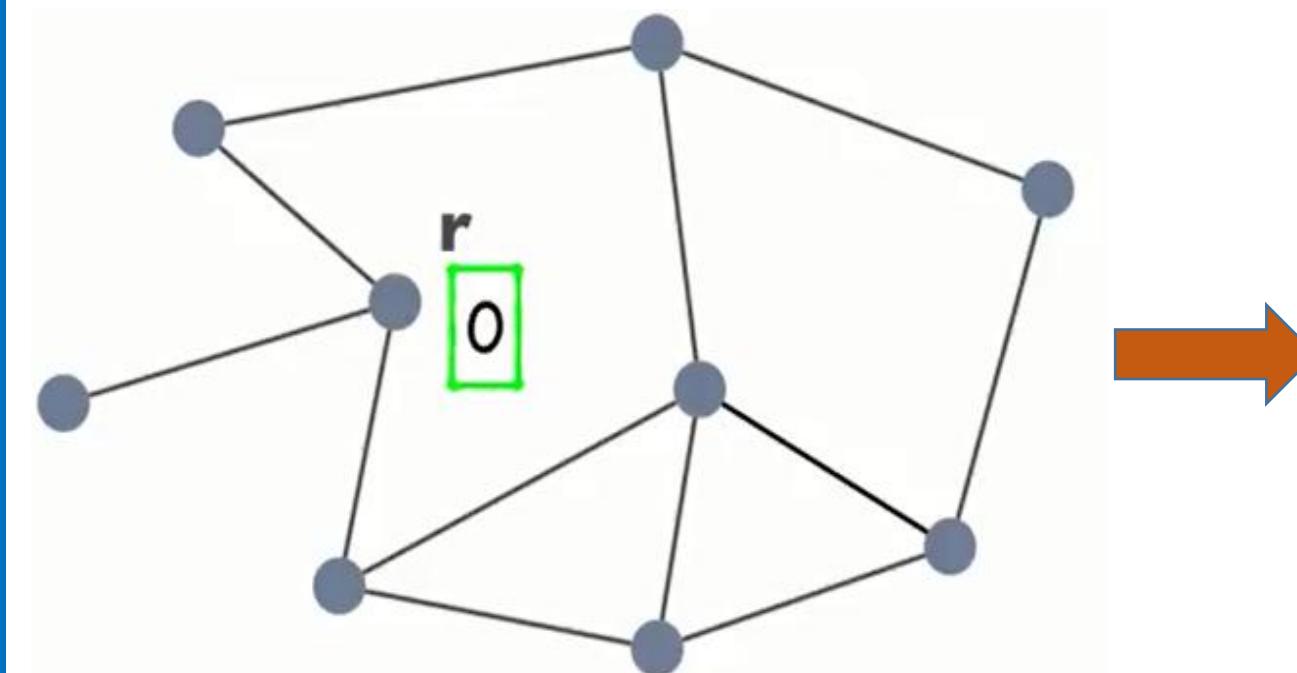
□ Parcours en largeur d'un arbre

- Exemple



□ Parcours en largeur d'un graphe

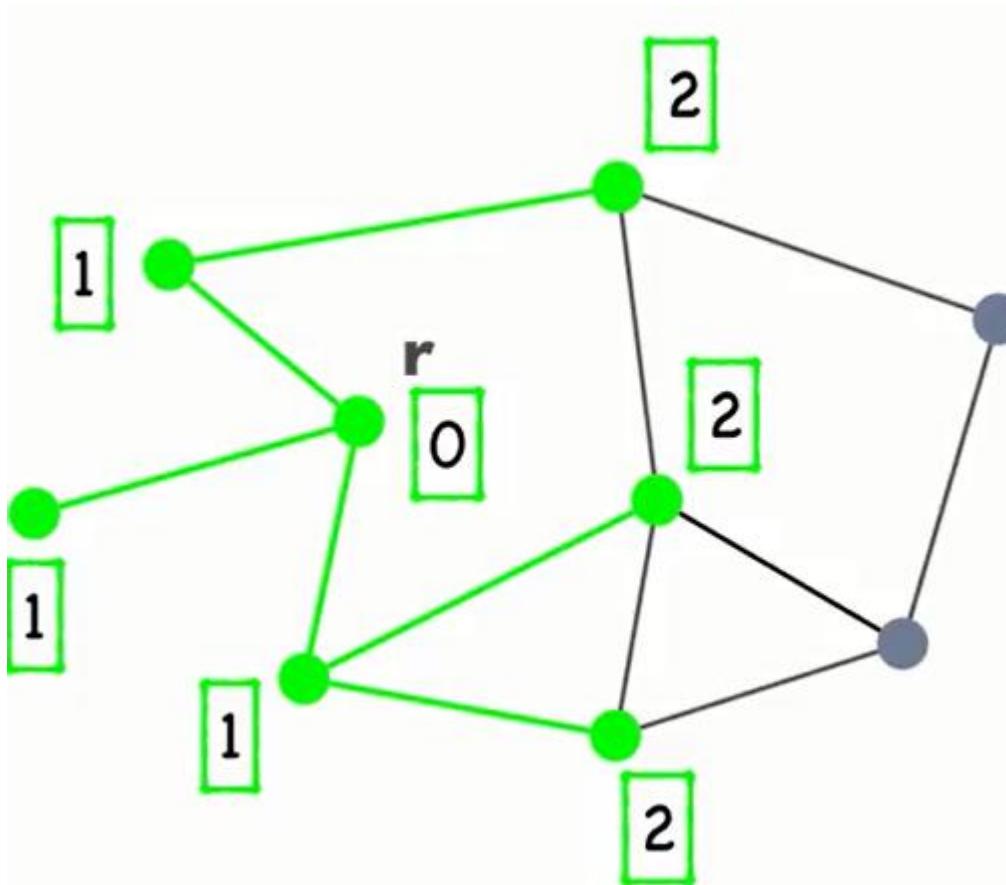
- **Exemple** Parcourir en largeur le graphe ci-dessous à partir du sommet r :



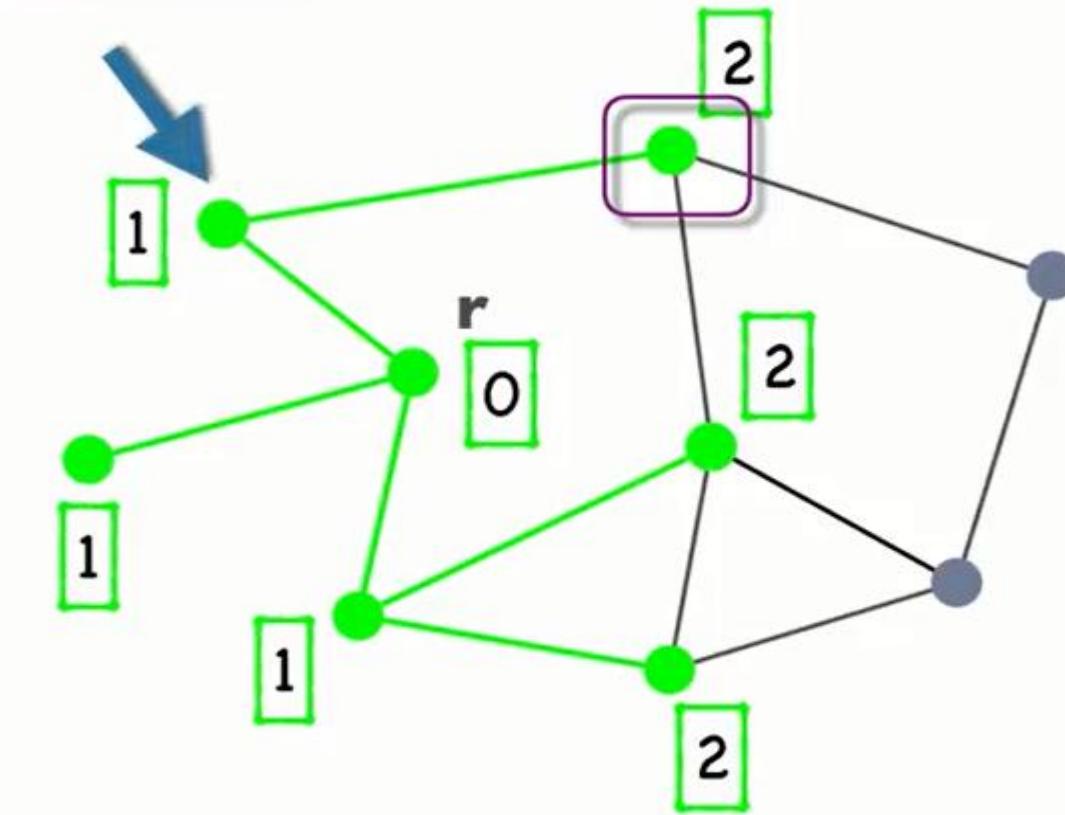
Mémorisation des arêtes
Mémorisation des sommets atteints

□ Parcours en largeur d'un graphe

▪ Exemple



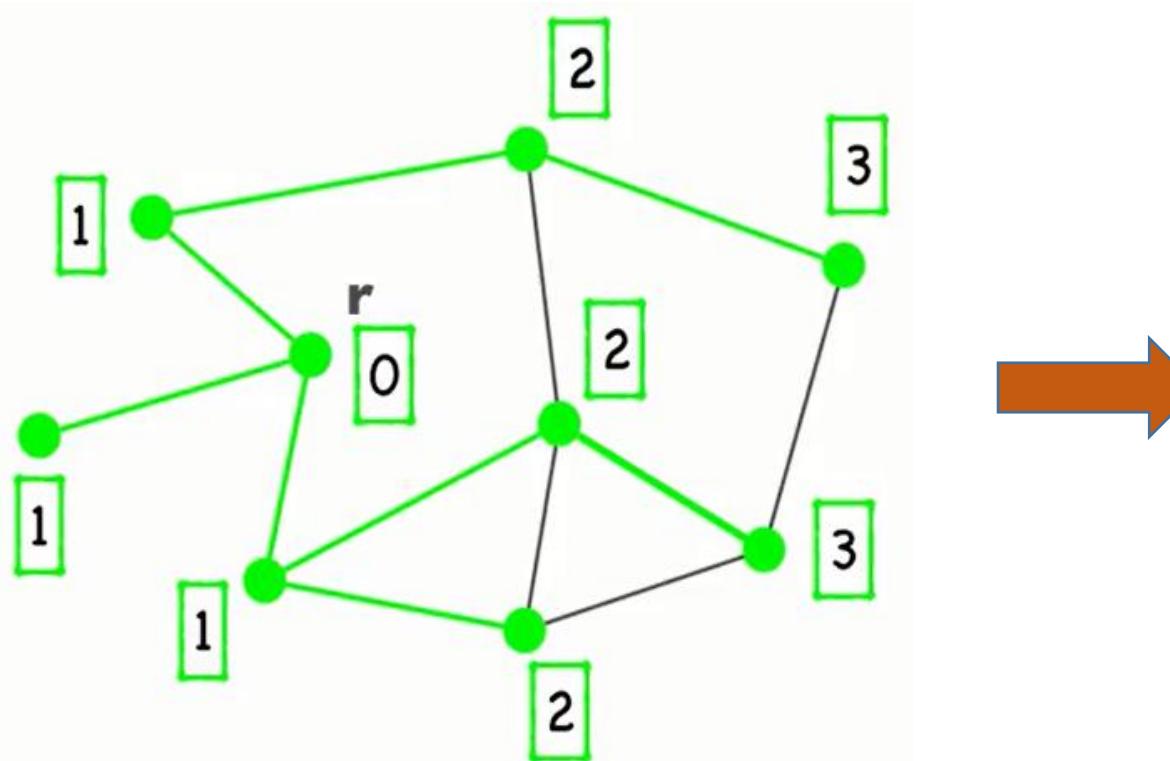
Déjà exploré



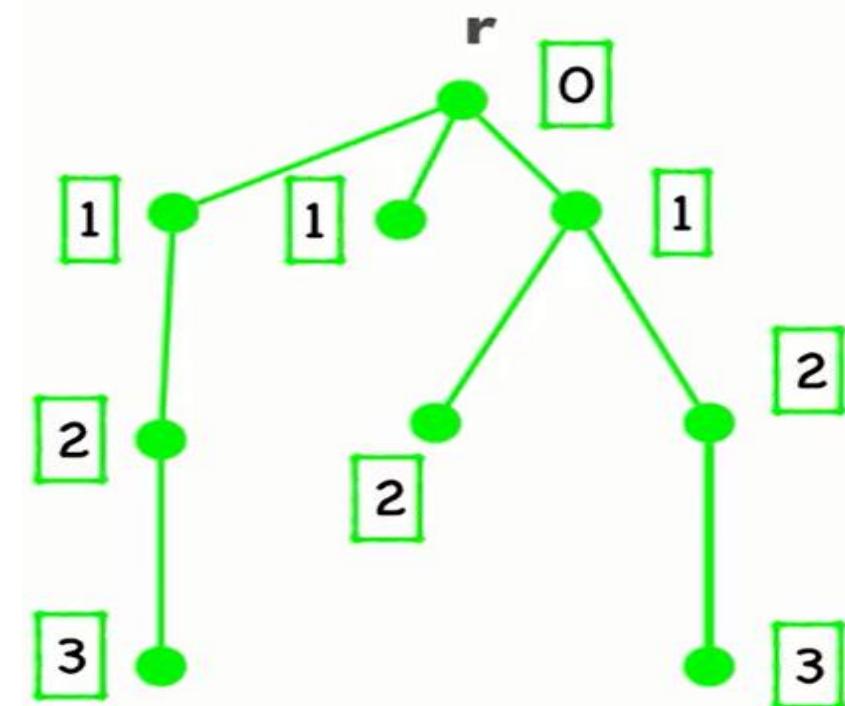
Examiner les voisins

□ Parcours en largeur d'un graphe

▪ Exemple



Fin de l'algorithme

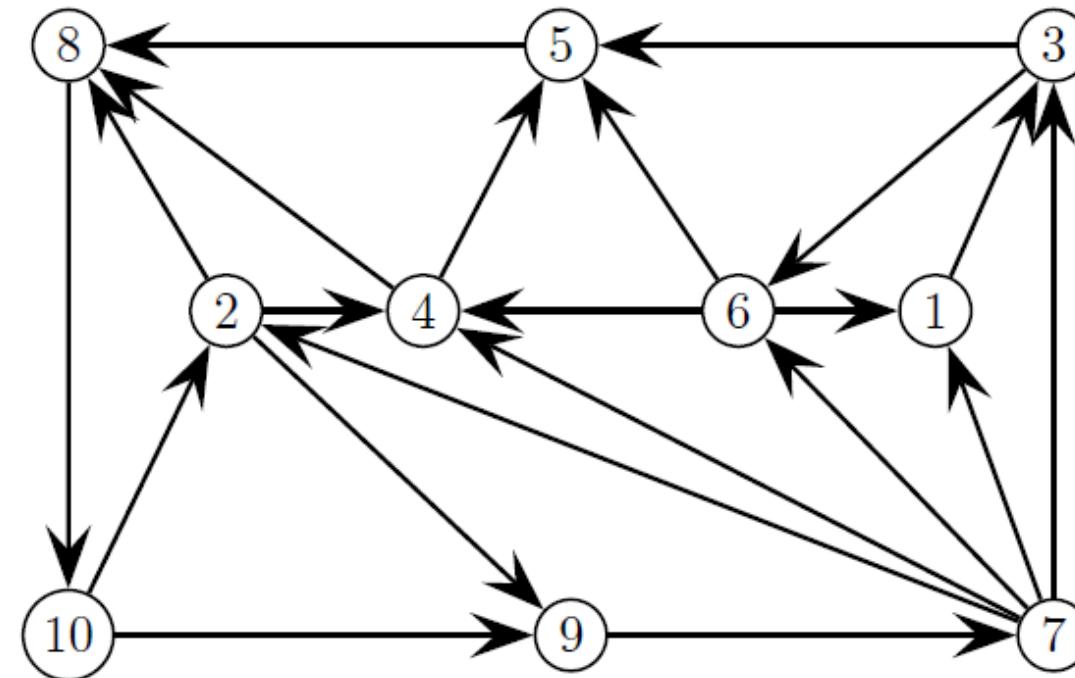


On obtient un arbre

□ Parcours en largeur

▪ Exercice 17

Parcours du graphe G en largeur, par ordre croissant des sommets, depuis le sommet 8. Vous donnerez à chaque fois l'ordre d'entrée des sommets dans la file



□ Parcours en largeur

▪ Exercice 18

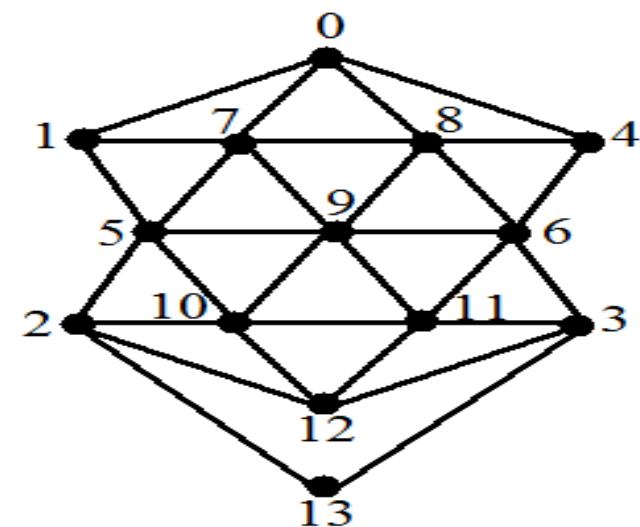
Pour ce graphe non orienté à 14 sommets, les voisins de chaque sommet sont supposés écrits dans l'ordre **croissant** de leurs numéros. Ainsi

0 a pour voisins 1, 4, 7, 8 ;

1 a pour voisins 0, 5, 7 ;

2 a pour voisins 5, 10, 12, 13 ;

etc.



Toujours en partant du sommet 0, faire une exploration en largeur du graphe , en utilisant l'ordre de voisins tel qu'il a été défini. On aura intérêt à utiliser l'évolution d'une file, afin de dessiner l'arbre final de l'exploration.

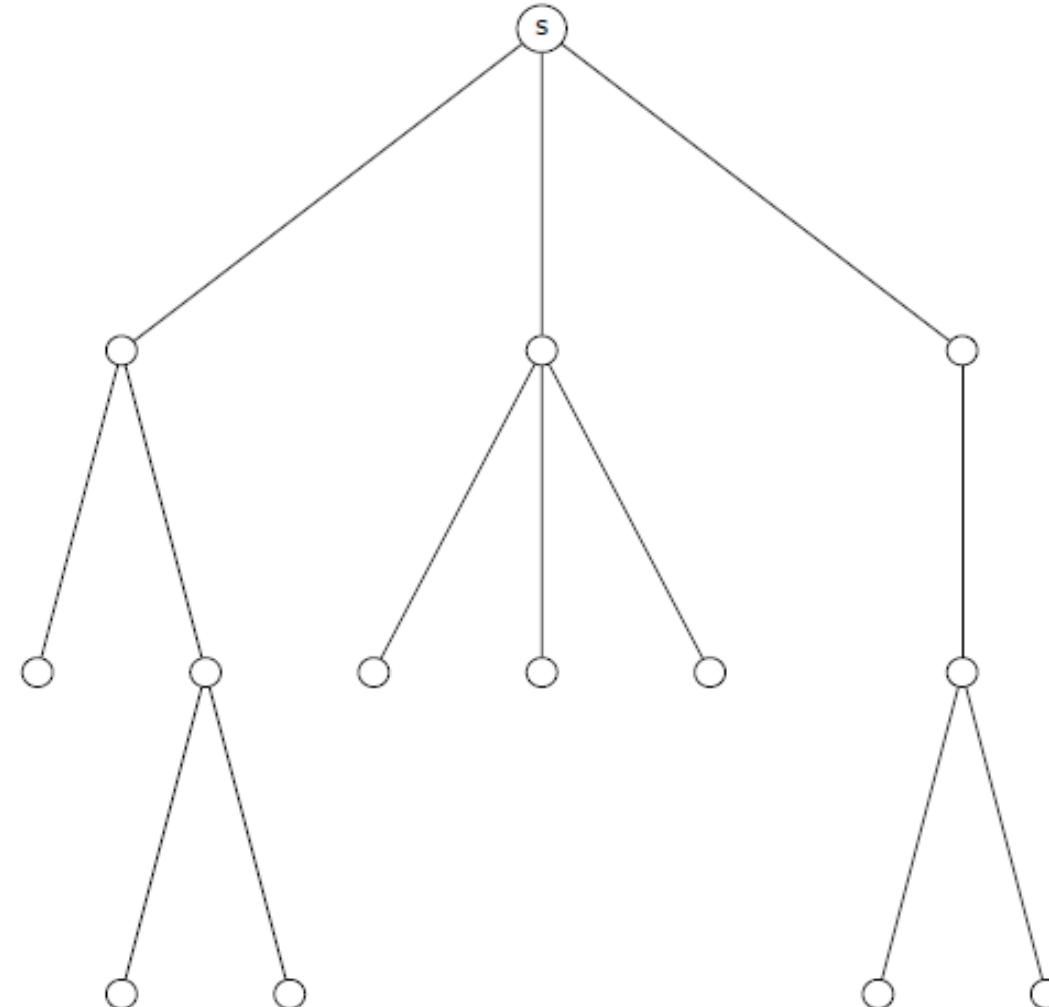
□ Parcours en profondeur

▪ Principe de l'algorithme

- Le parcours en profondeur consiste, à partir d'un sommet donné, **à suivre un chemin le plus loin possible, puis à faire des retours en arrière pour reprendre tous les chemins ignorés précédemment.**
- Dans ce parcours , on utilise **une pile**. On empile le sommet de départ.
- Si le sommet de la pile **présente des voisins qui ne sont pas dans la pile**, ni déjà passés dans la pile :
 - alors **on sélectionne l'un de ces voisins et on l'empile** (en le marquant de son numéro de découverte),
 - sinon **on dépile** (c'est à dire on supprime l'élément du sommet de la pile).
- On recommence au point 2 (tant que la pile n'est pas vide).

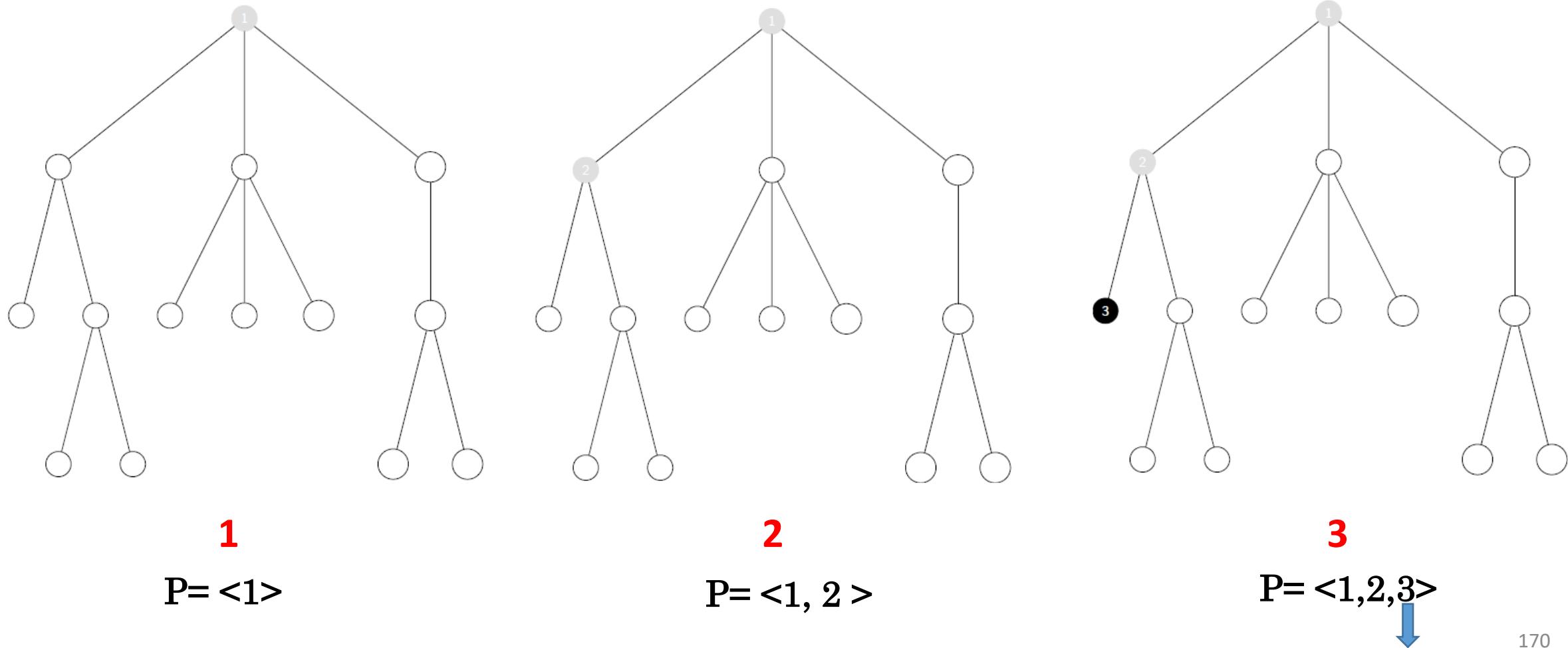
□ Parcours en profondeur d'un arbre

- Exemple : Parcourir en profondeur le graphe ci-dessous à partir du sommet s :



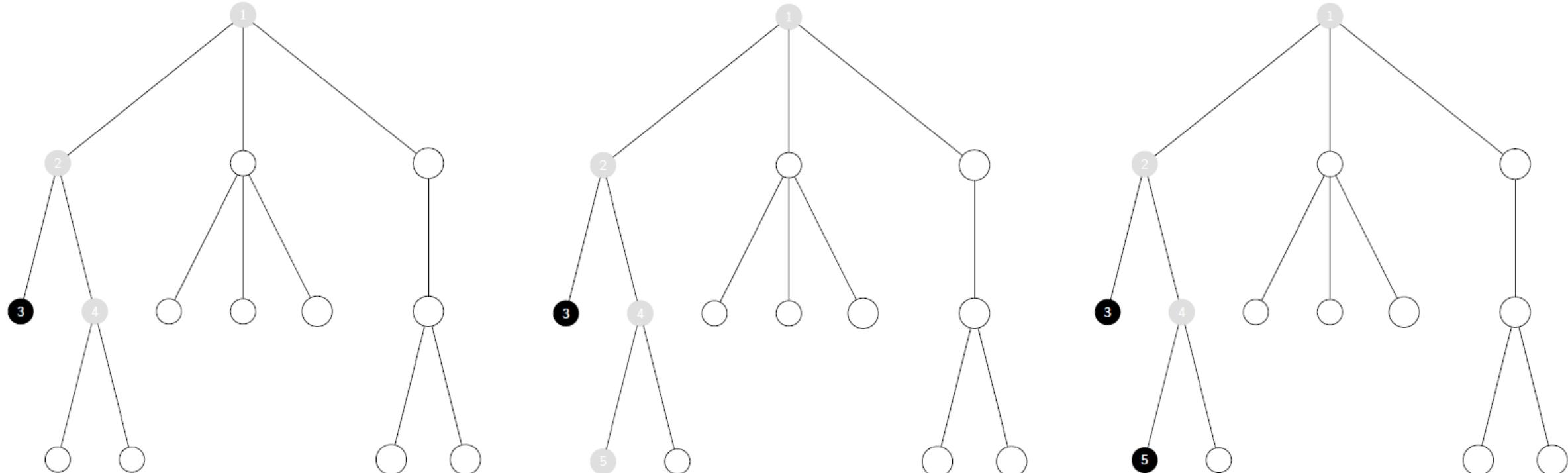
Parcours en profondeur d'un arbre

▪ Exemple



Parcours en profondeur d'un arbre

▪ Exemple



4

 $P = \langle 1, 2, 4 \rangle$

5

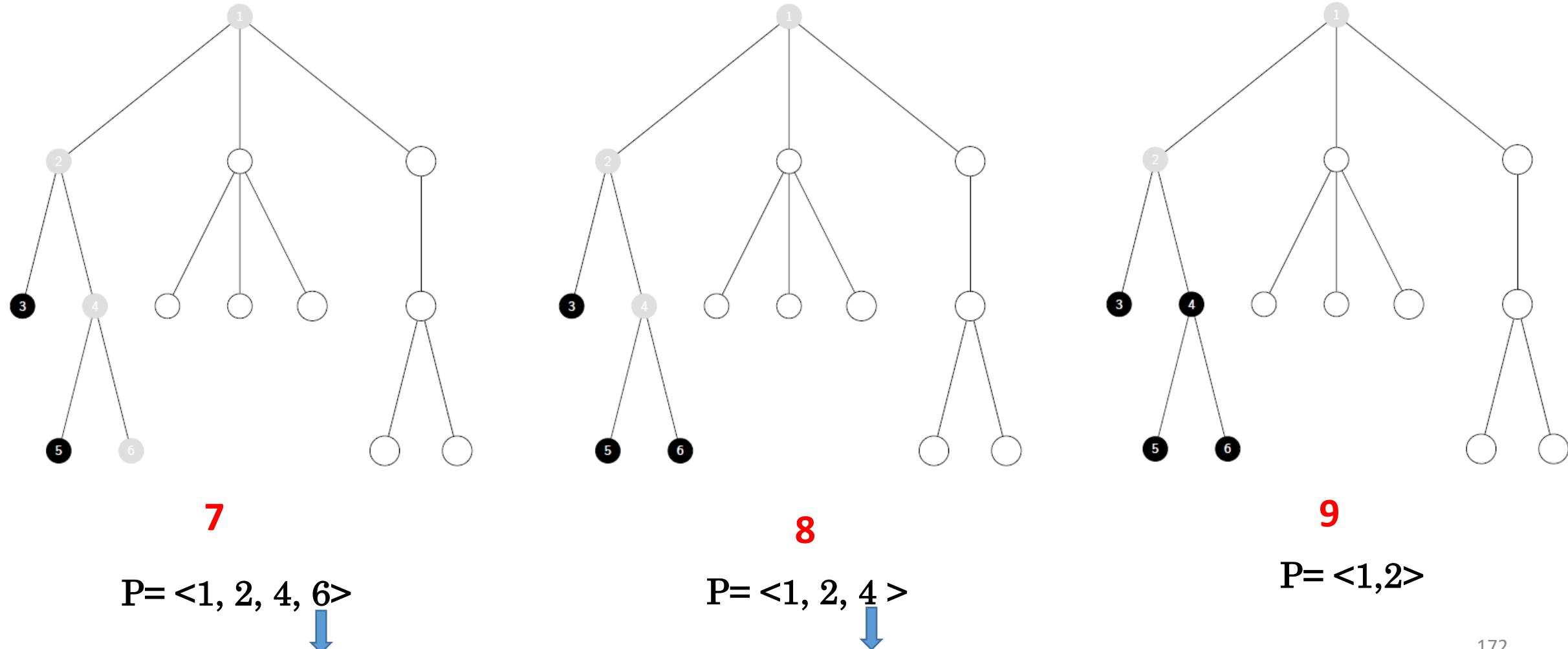
 $P = \langle 1, 2, 4, 5 \rangle$

6

 $P = \langle 1, 2, 4 \rangle$ 

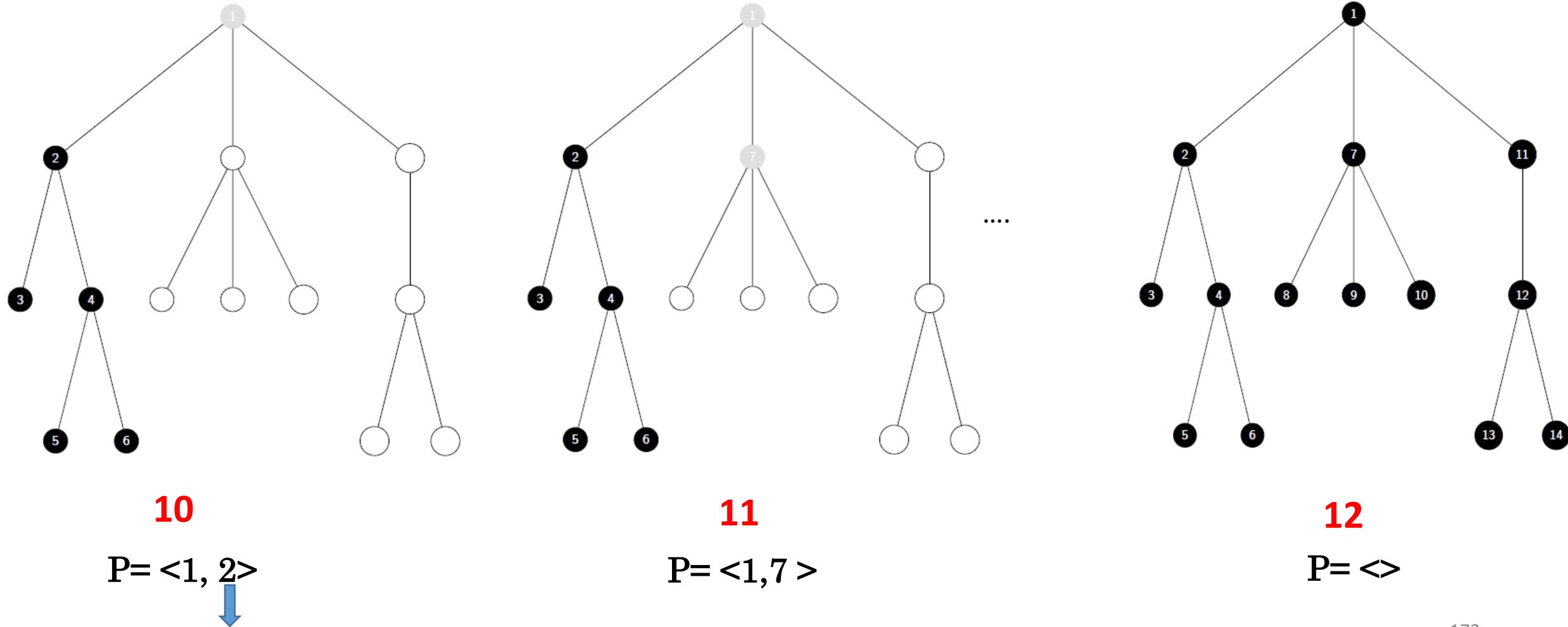
Parcours en profondeur d'un arbre

▪ Exemple



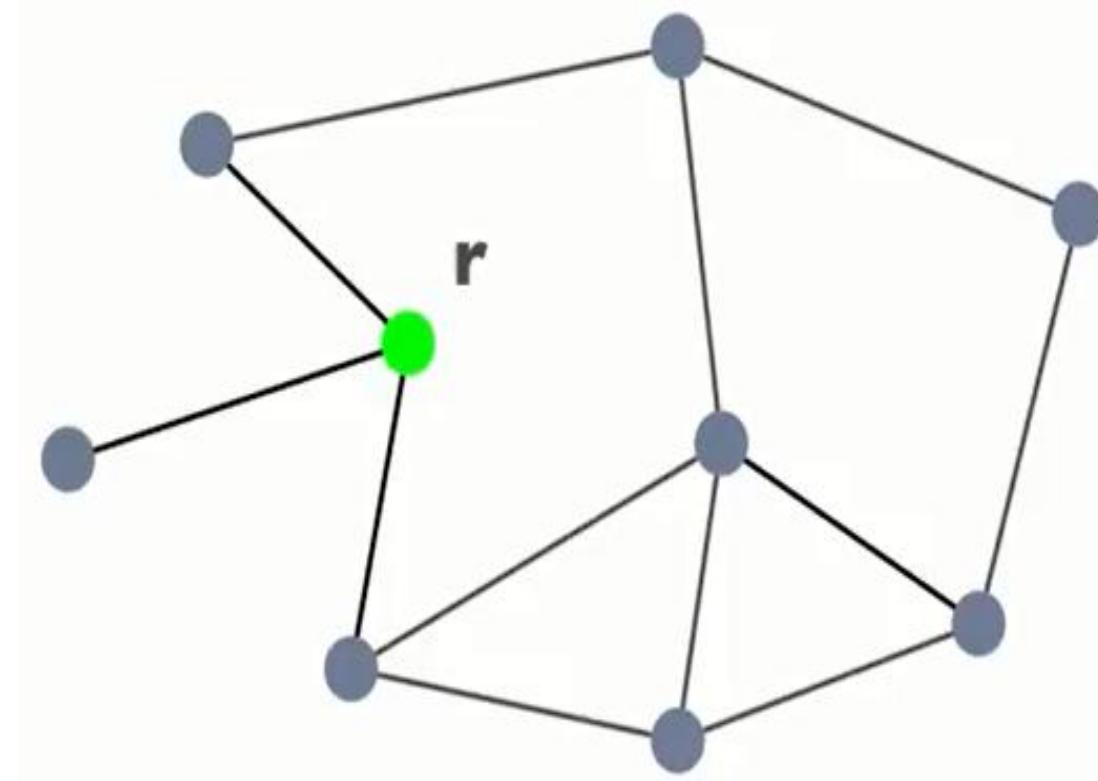
Parcours en profondeur d'un arbre

■ Exemple



□ Parcours en profondeur d'un graphe

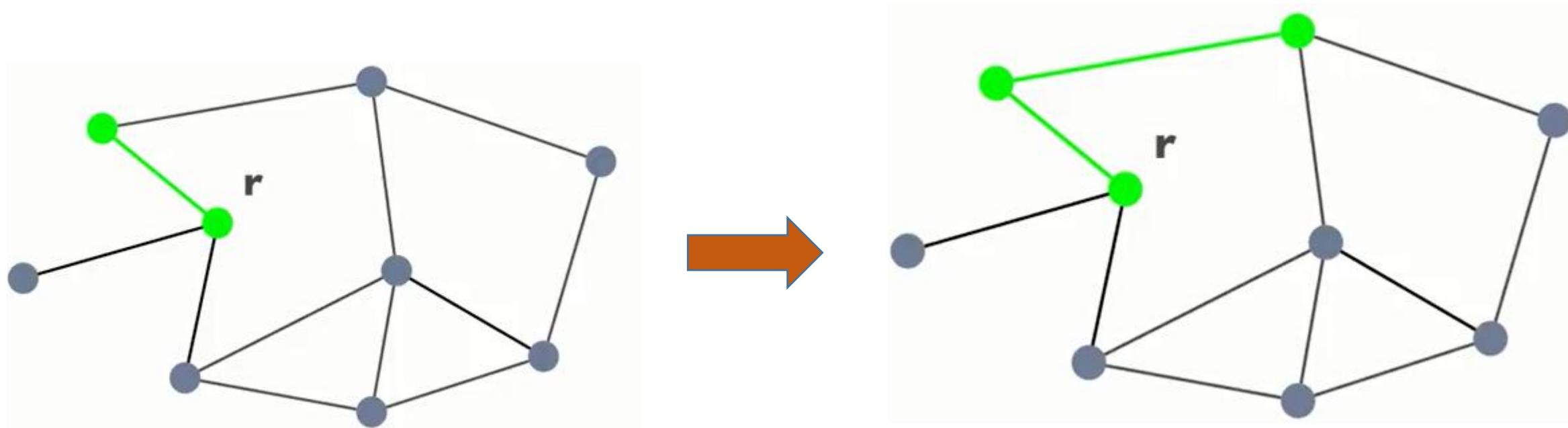
- Exemple : Parcourir en profondeur le graphe ci-dessous à partir du sommet r :



Mémorisation des sommets visités

Parcours en profondeur d'un arbre

▪ Exemple

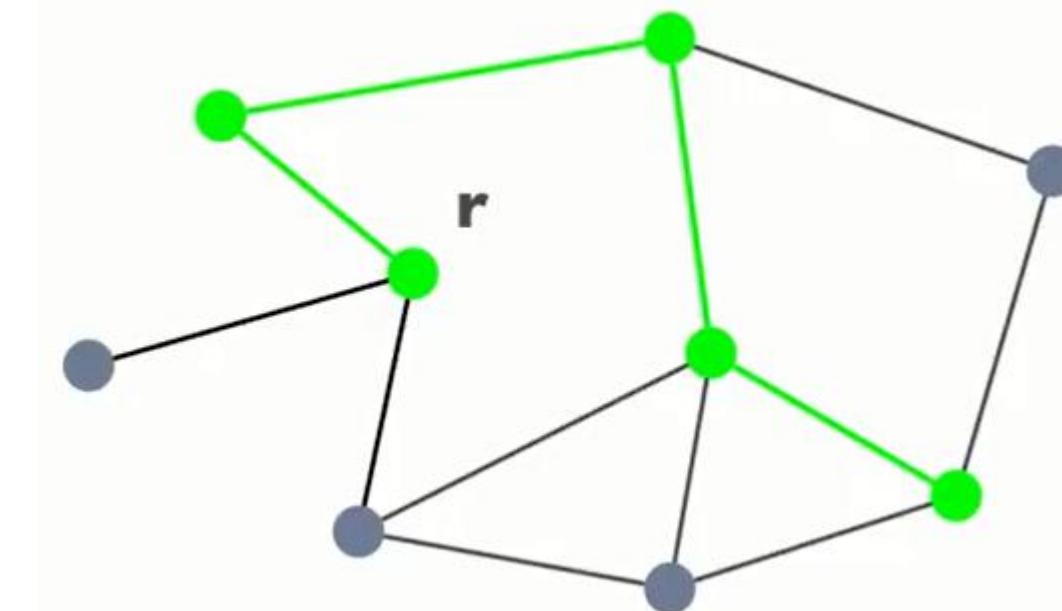
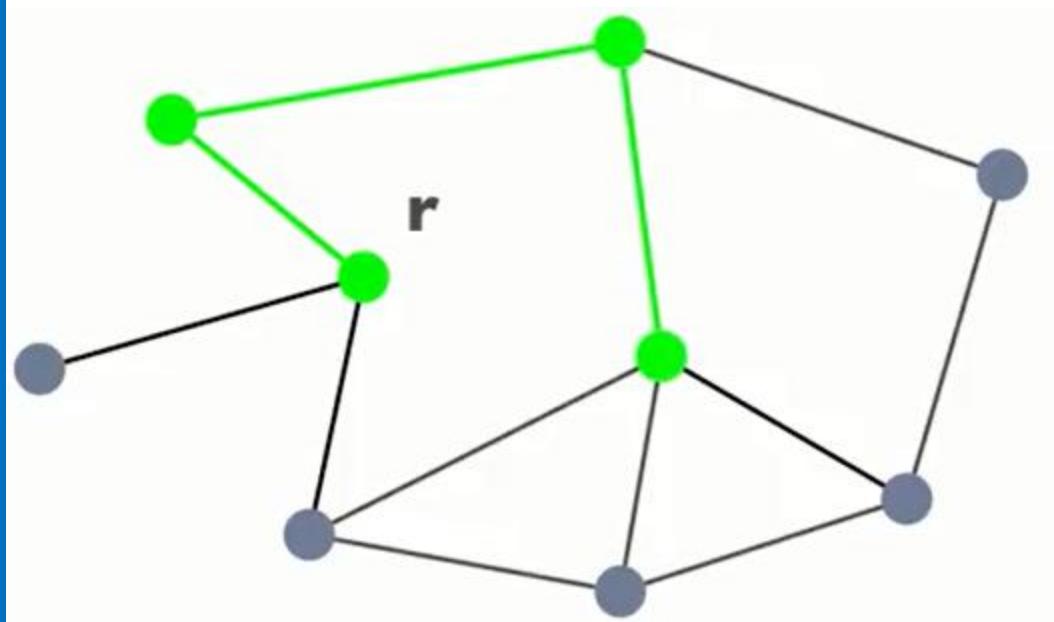


Mémorisation des sommets visités

Mémorisation des arêtes traversées

Parcours en profondeur d'un graphe

■ Exemple

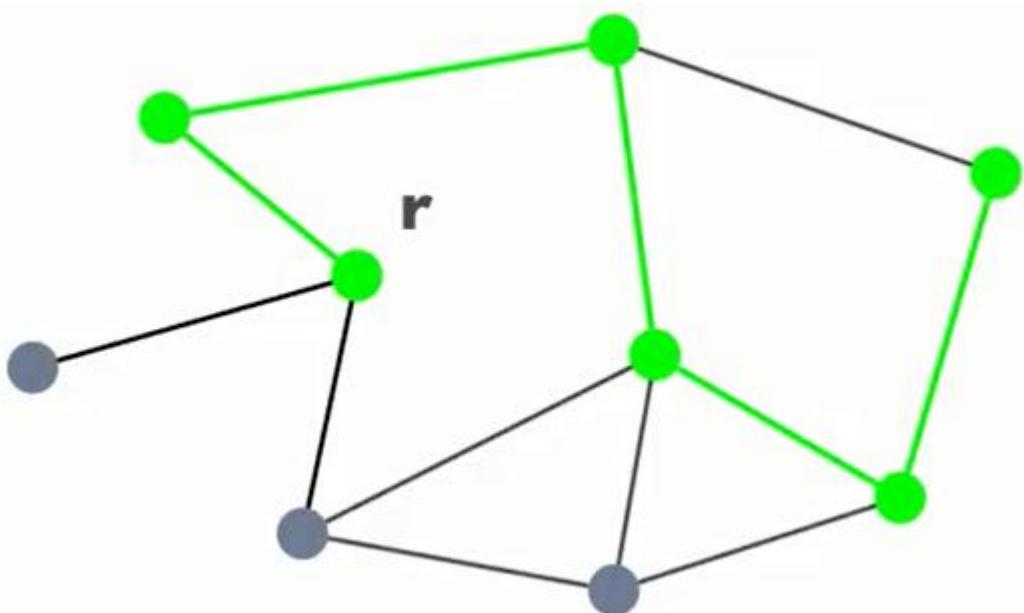


Mémorisation des sommets visités

Mémorisation des arêtes traversées

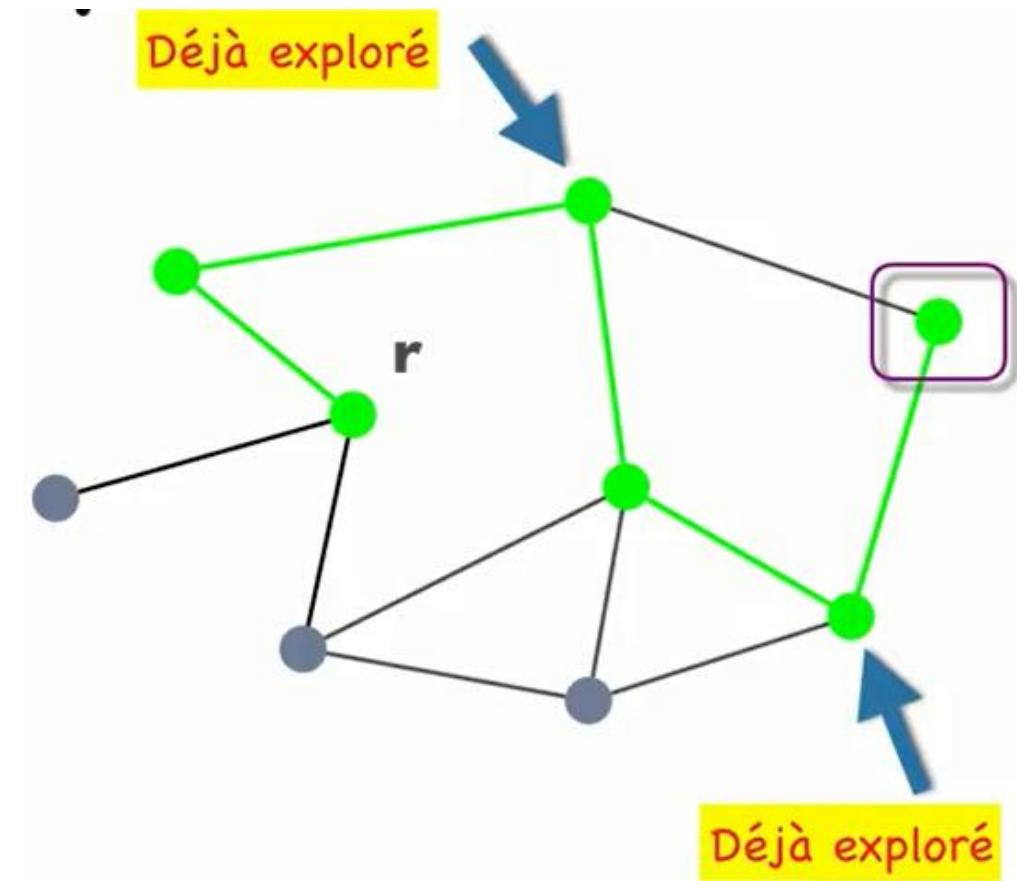
□ Parcours en profondeur d'un graphe

■ Exemple



Mémorisation des sommets visités

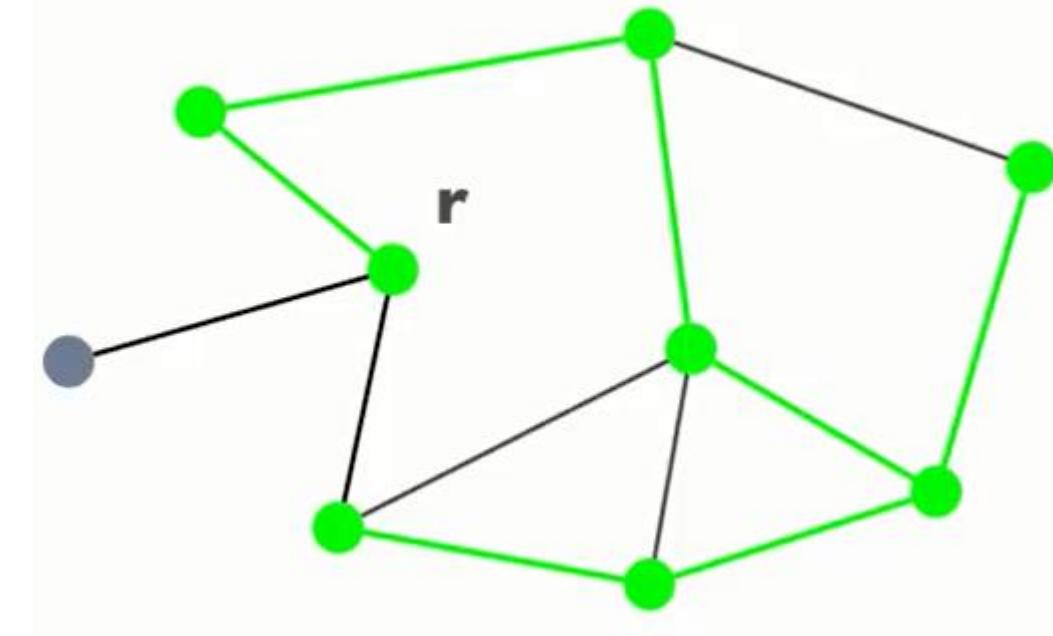
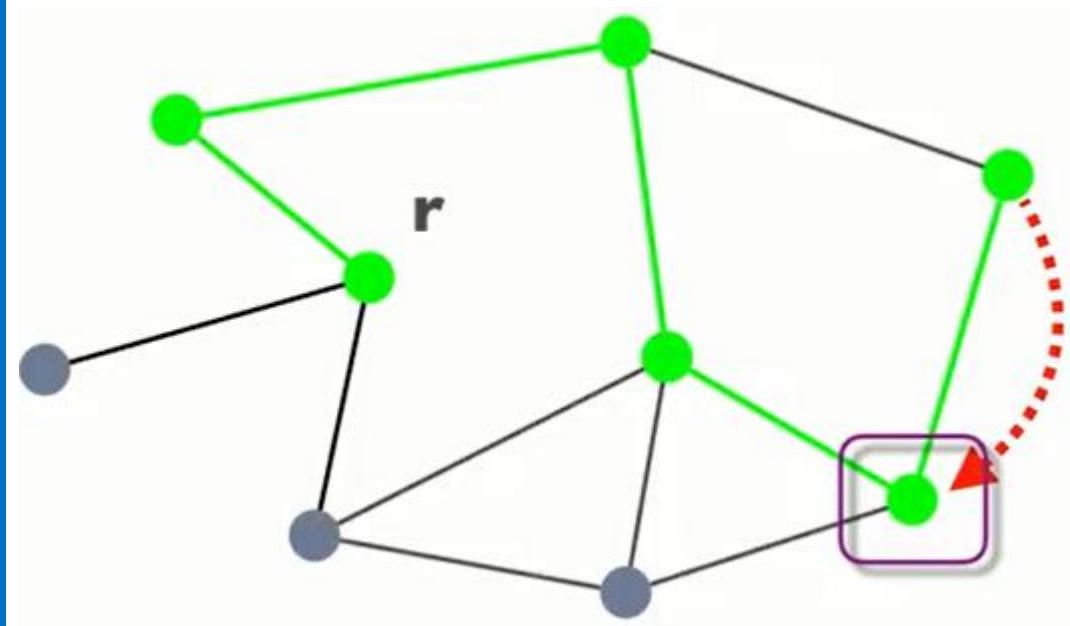
Mémorisation des arêtes traversées



Impossible d'aller plus en profondeur

Parcours en profondeur d'un graphe

■ Exemple



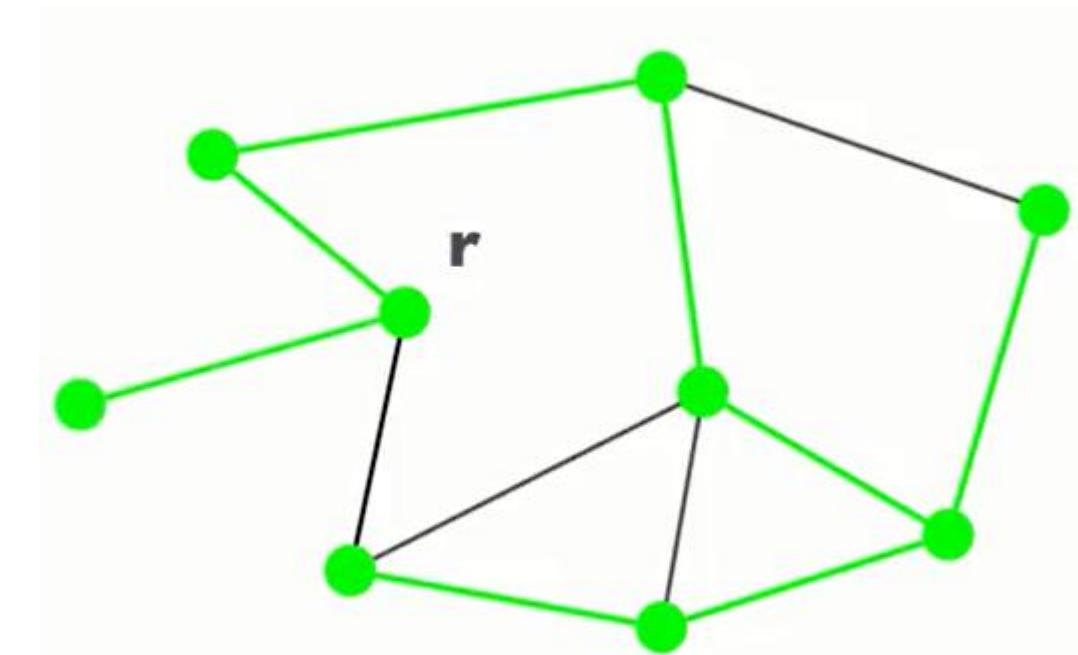
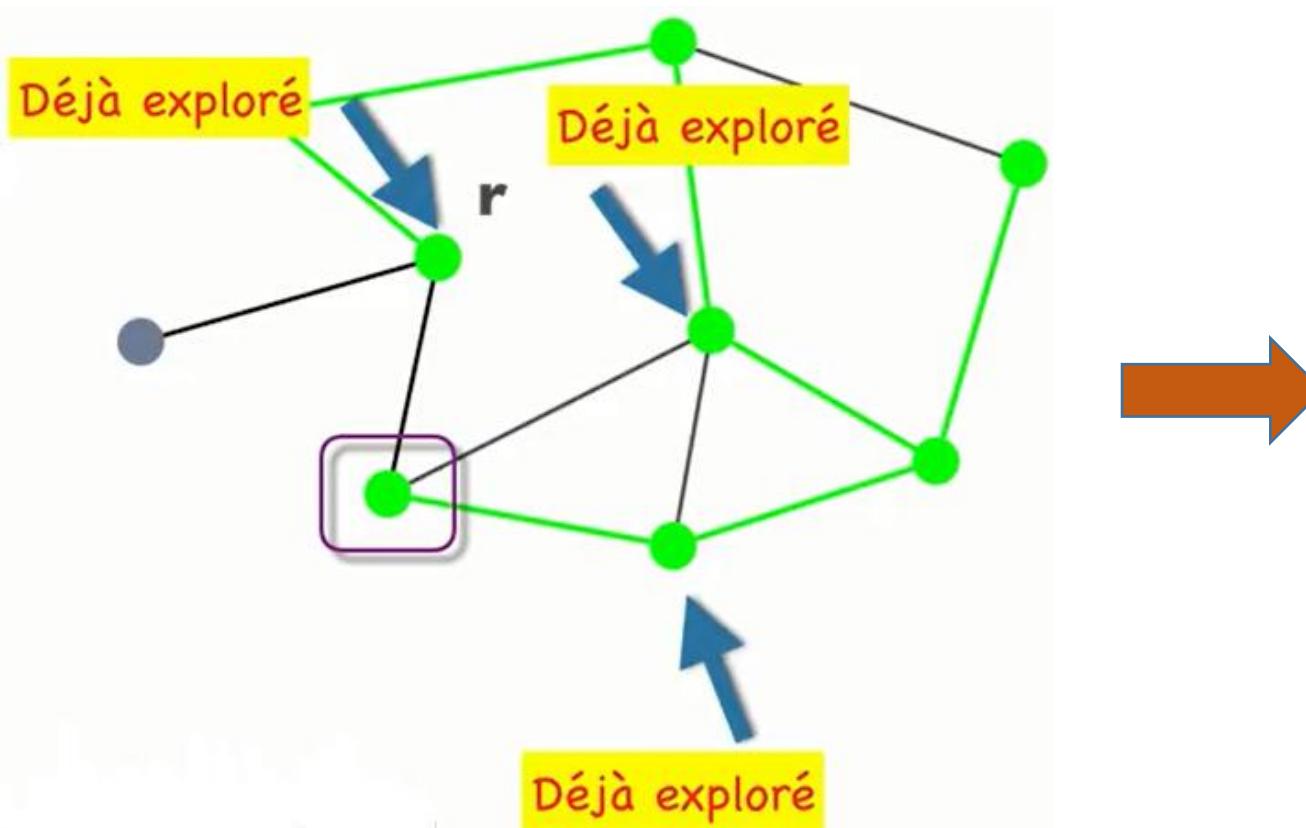
Coincé ? Revenir sur ses pas

Essayer, à nouveau, d'aller en profondeur

Actualizado
2023-09-11

Parcours en profondeur d'un graphe

■ Exemple

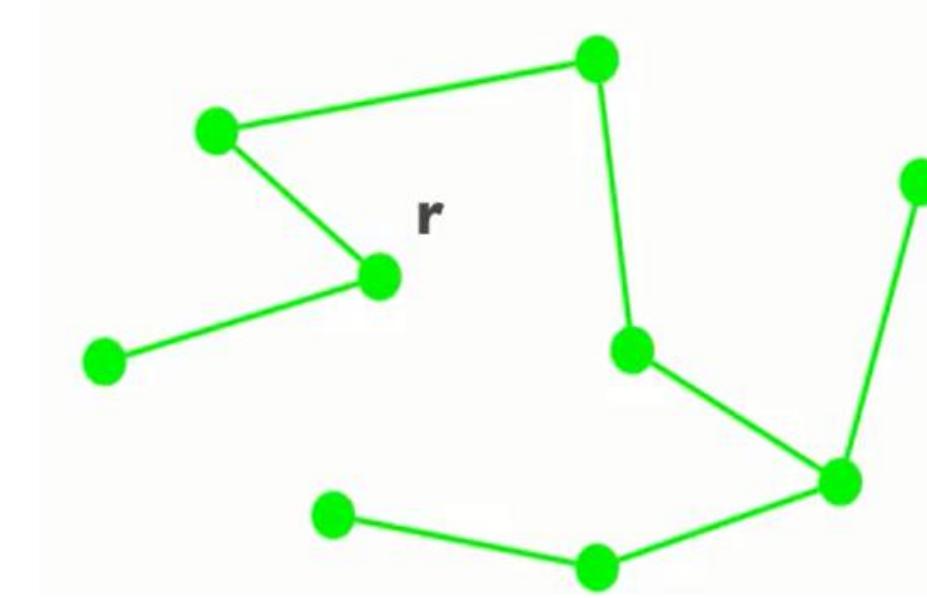
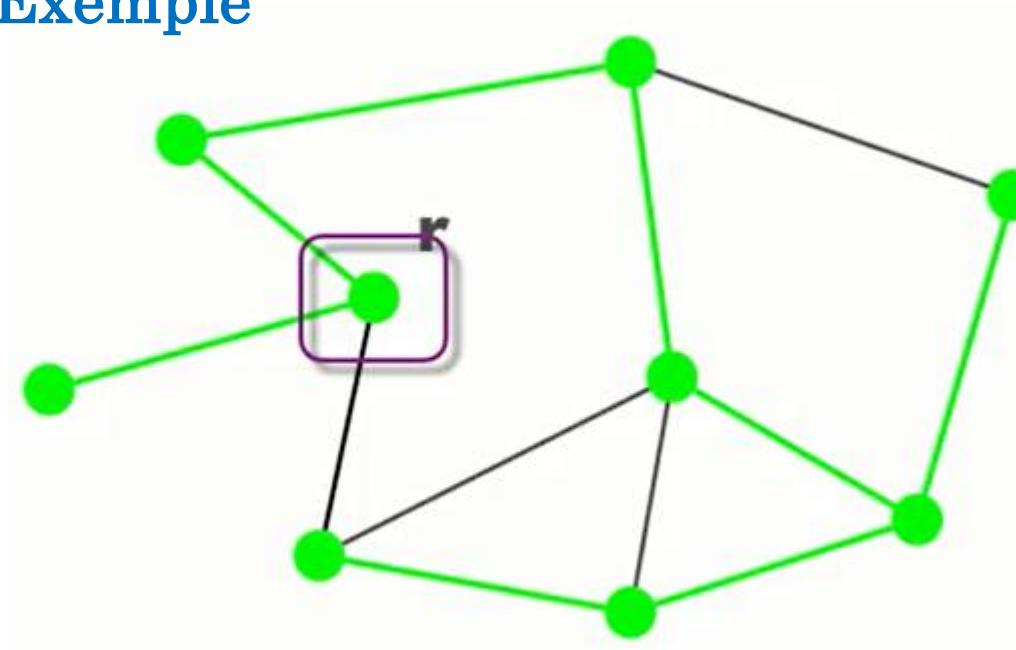


Essayer, à nouveau, d'aller en profondeur

Coincé ? Revenir sur ses pas

Parcours en profondeur d'un graphe

■ Exemple



Coincé au sommet de départ ?

FIN

Fin de l'algorithme

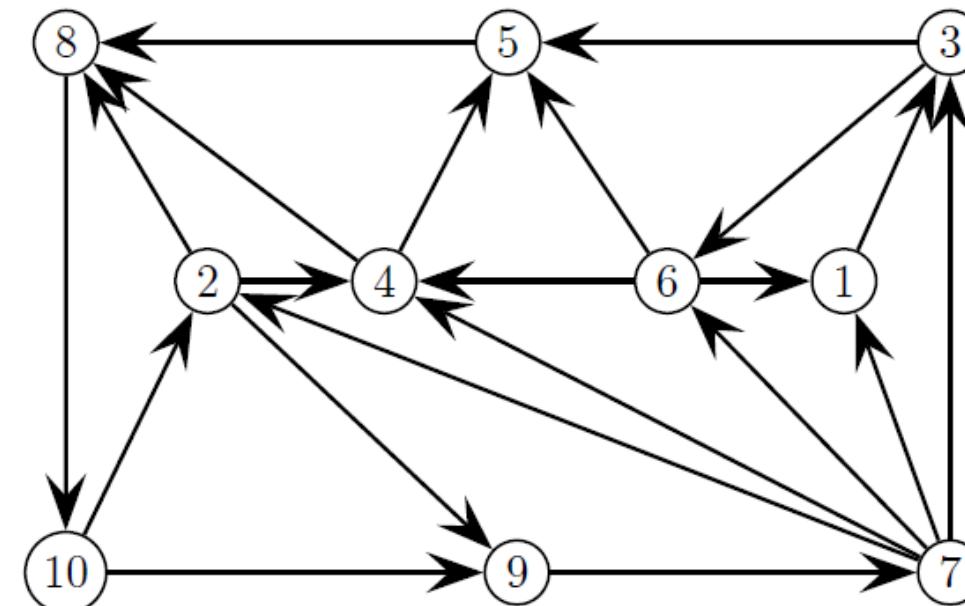
Résultat

Parcours en profondeur d'un arbre

▪ Exercice 19

Parcours du graphe G en profondeur, par ordre décroissant des sommets, depuis le sommet 8.

Vous donnerez à chaque fois l'ordre d'entrée des sommets dans la pile:



□ Parcours en profondeur d'un arbre

■ Exercice 20

Pour ce graphe non orienté à 14 sommets, les voisins de chaque sommet sont supposés écrits dans l'ordre croissant de leurs numéros. Ainsi

0 a pour voisins 1, 4, 7, 8 ;

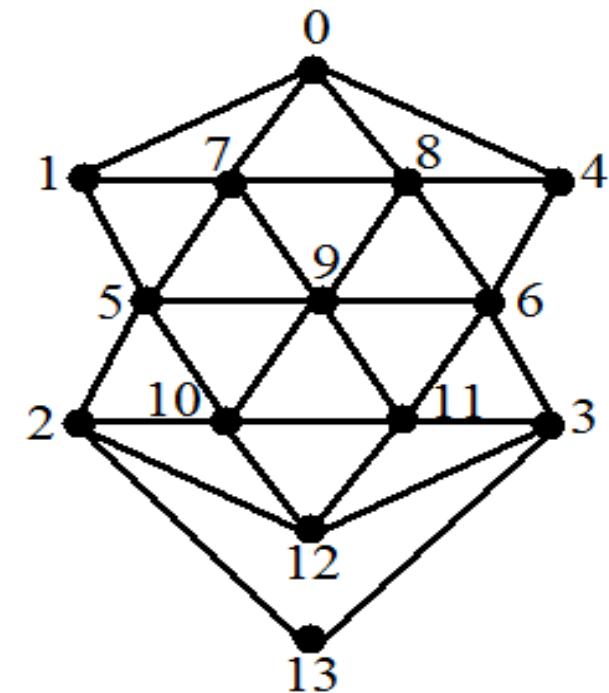
1 a pour voisins 0, 5, 7 ;

2 a pour voisins 5, 10, 12, 13 ;

etc.

En partant du sommet 0, faire une exploration en profondeur de ce graphe, en utilisant l'ordre de voisins tel qu'il a été défini.

Dessiner l'arbre obtenu.



Problèmes d'optimisation pour des graphes pondérés

□ Problèmes d'optimisation pour des graphes pondérés

- Arbre de recouvrement minimal (Spanning Tree):

Définition: Soit $G=(V,E)$ un graphe non orienté à valuations quelconques.

Un **arbre couvrant de poids minimal** de G est un **arbre couvrant** de G dont la valuation est la plus petite parmi celles de tous les arbres couvrants de G .

Là encore, dès qu'un graphe sera connexe, un tel arbre existera. En effet le nombre d'arbres couvrants est nécessairement fini, il suffit donc de prendre celui qui a la valuation minimale

Théorème

Soit $G=(V,E)$ un graphe non orienté à valuations quelconques.

Si G est connexe alors G possède un arbre couvrant de poids minimal.

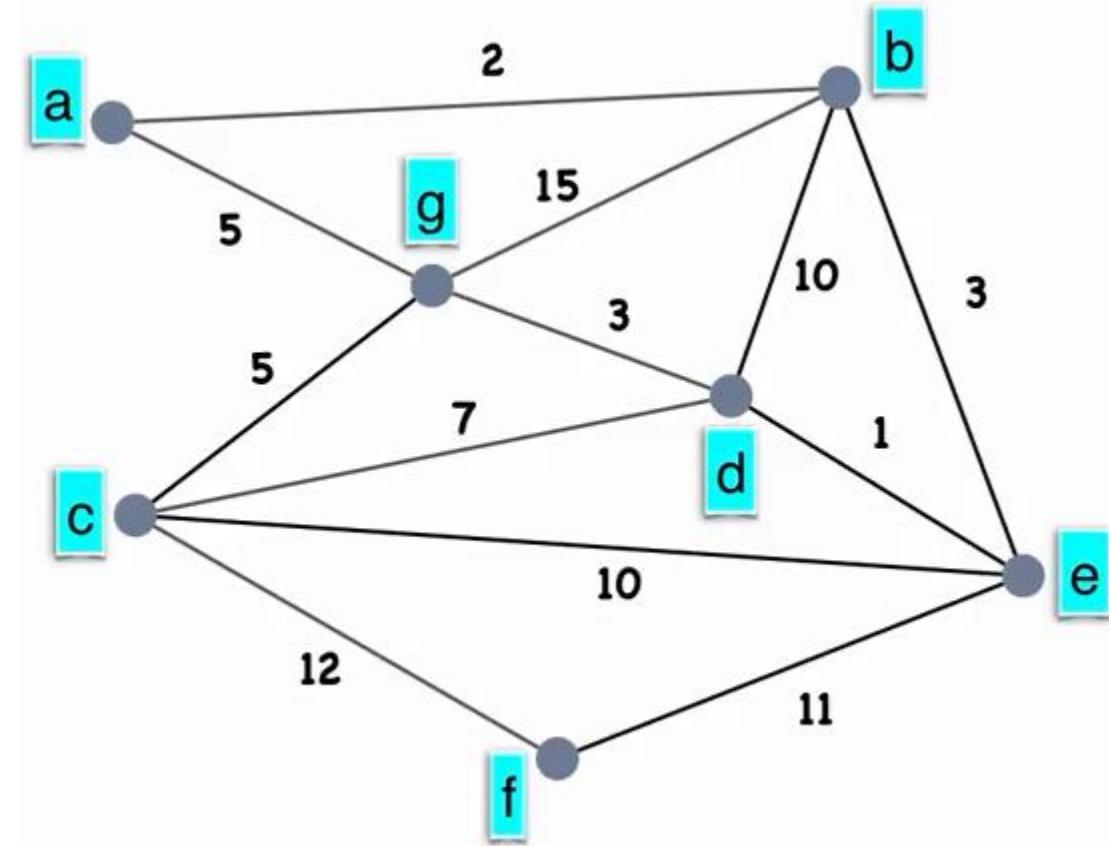
□ Problèmes d'optimisation pour des graphes pondérés

- **Recherche d'un arbre de recouvrement minimal**
- Le problème de l'**arbre couvrant minimal** est un des plus vieux problèmes en théorie des graphes. La problématique se pose comme suit : étant donné un **graphe avec un nombre de sommets et un nombre d'arêtes ayant des poids de valeurs dans l'ensemble des entiers relatifs**, l'**arbre couvrant minimal** consiste à trouver l'**ensemble des arêtes permettant de rejoindre tous les sommets sans former de cycle, et ce, avec un coût minimal**.
- Ce problème trouve des applications pratiques variées : il est directement applicable pour l'optimisation et la conception de divers types de réseaux (électrique, internet, etc.)
- On souhaite bien sûr que tous les nœuds du réseau puissent communiquer entre eux, il nous faut donc rechercher un **sous-graphe couvrant connexe**. Pour avoir à établir un minimum de liaisons nous rechercherons même un **arbre de poids minimal**.

□ Problèmes d'optimisation pour des graphes pondérés

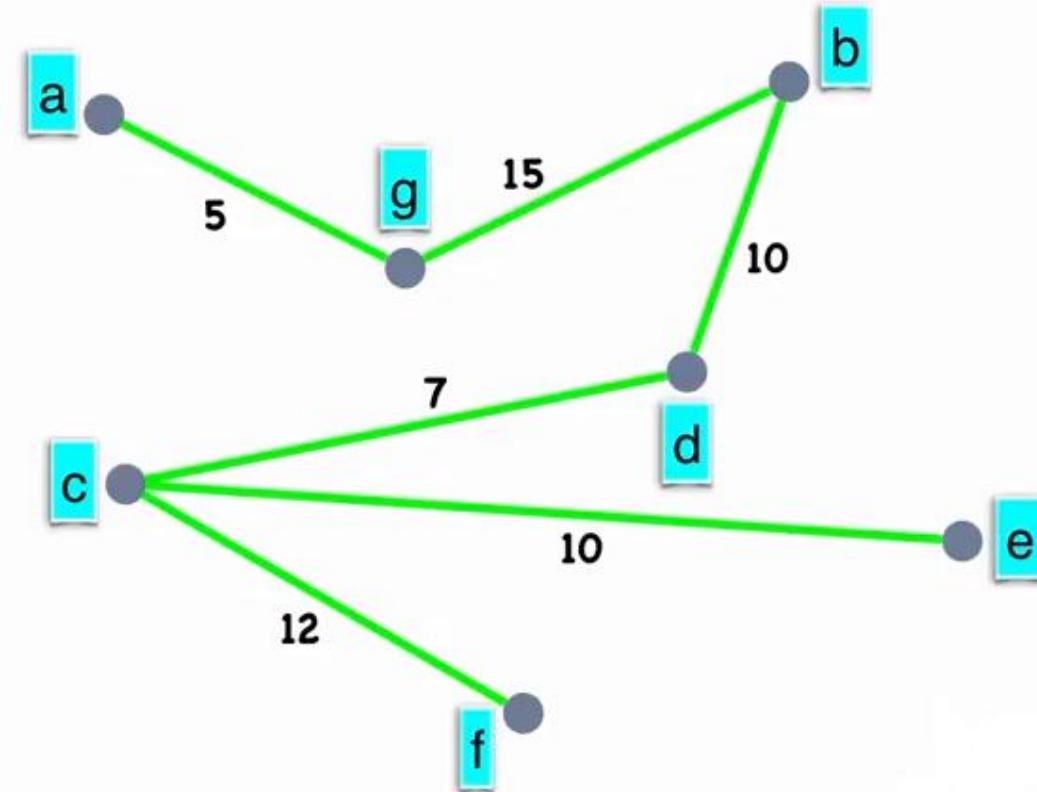
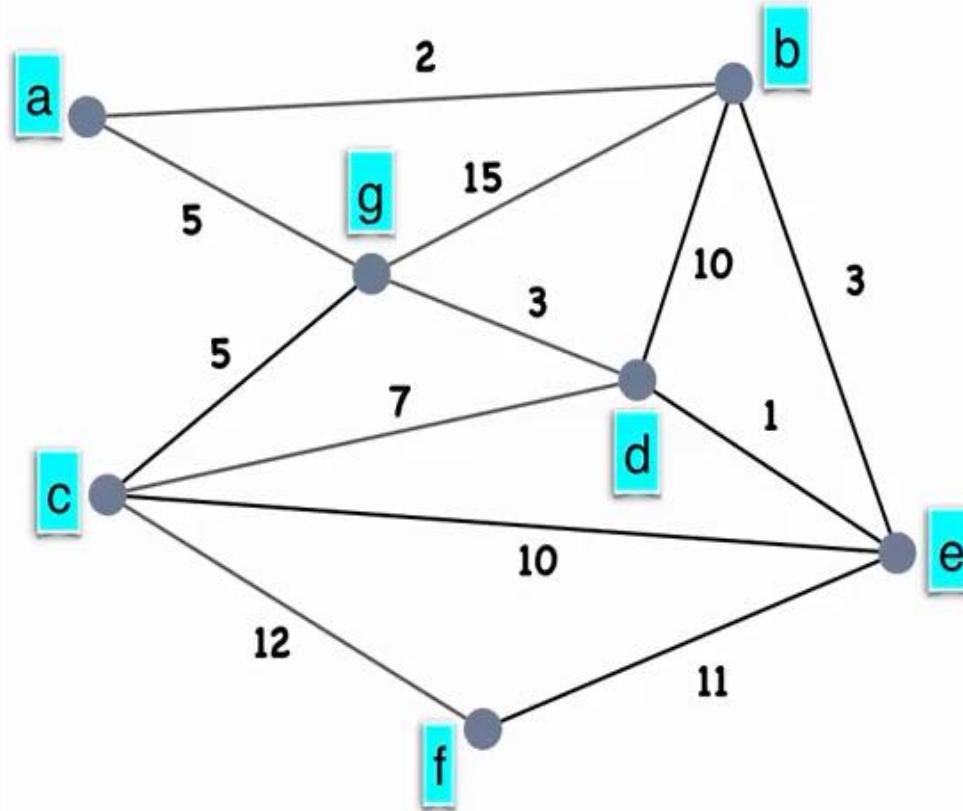
▪ Exemple

- Pour le graphe suivant, on cherche :
 - Un arbre (Graphe connexe sans cycle)
 - Contenant tous les sommets
 - De poids minimum



Problèmes d'optimisation pour des graphes pondérés

▪ Exemple



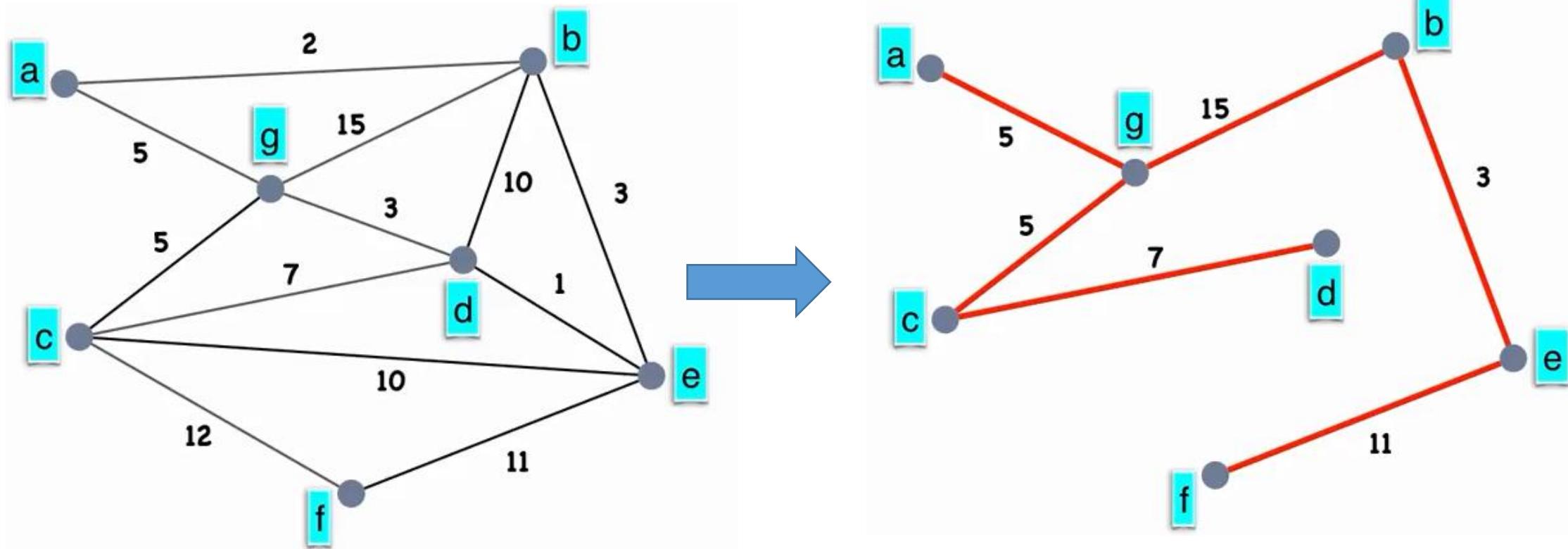
Pour l'arbre suivant :

$$\text{Poids (T1)} = 5 + 15 + 10 + 7 + 10 + 12 = 59$$

Ce poids est-il minimal ?

□ Problèmes d'optimisation pour des graphes pondérés

▪ Exemple



Poids (**T2**) = $5+5+15+3+7+11= 46 <$ poids(**T1**)=59

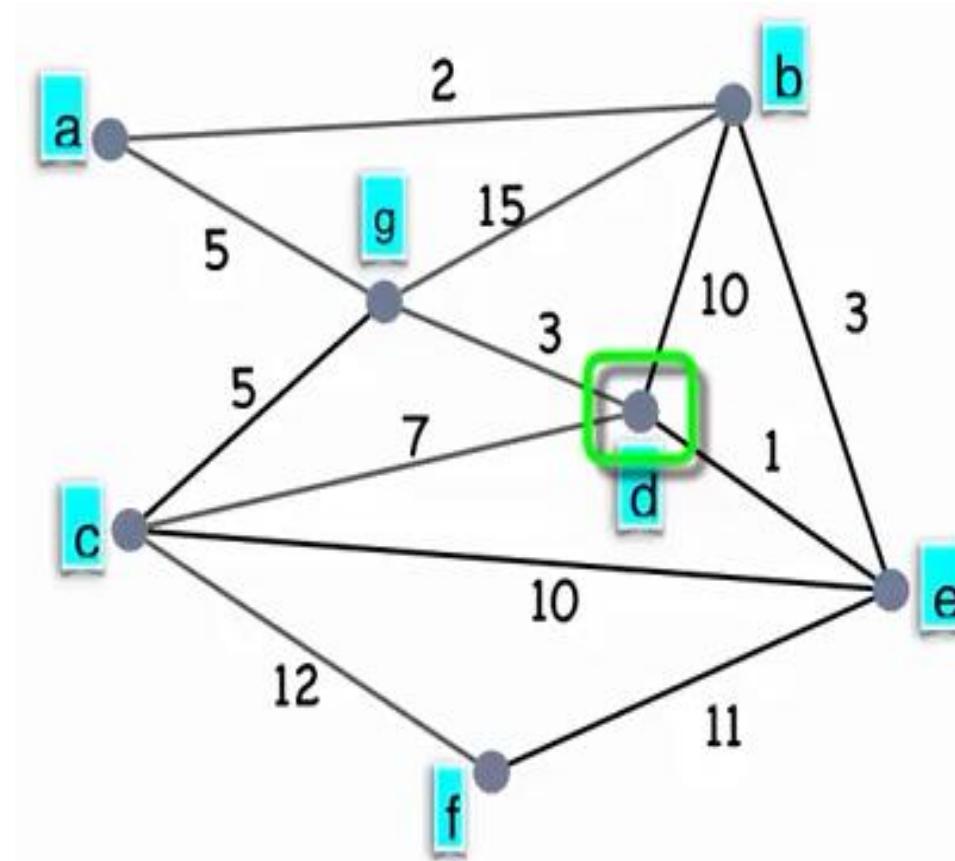
- Pour trouver l'arbre de recouvrement minimal, on dispose de 2 algorithmes: Prim et Kruskal.

□ Arbre couvrant de poids minimal - Algorithme Prim

- L'algorithme de Prim est un **algorithme de recherche d'un arbre couvrant de poids minimal dans un graphe non orienté connexe à valuations quelconques.**
- Son principe comme suit:
 - 1) On part d'un sommet s
 - 2) On ajoute l'arête de plus faible poids qui
 - (a) part d'un sommet visité
 - (b) et qui ne crée pas de cycle
 - 3) On itère (2) tant qu'il reste des arêtes

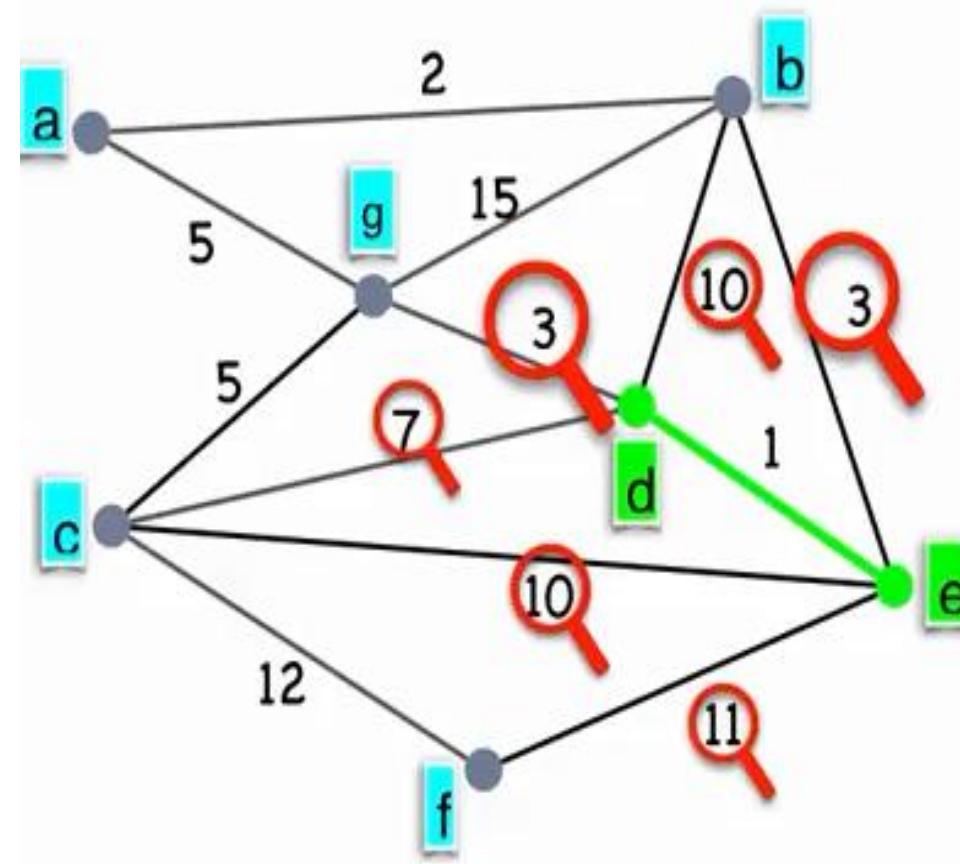
□ Arbre couvrant de poids minimal -Algorithme Prim

- Les données du problème: graphe non orienté- connexe-pondéré
- Itération 1: choix d'un sommet quelconque



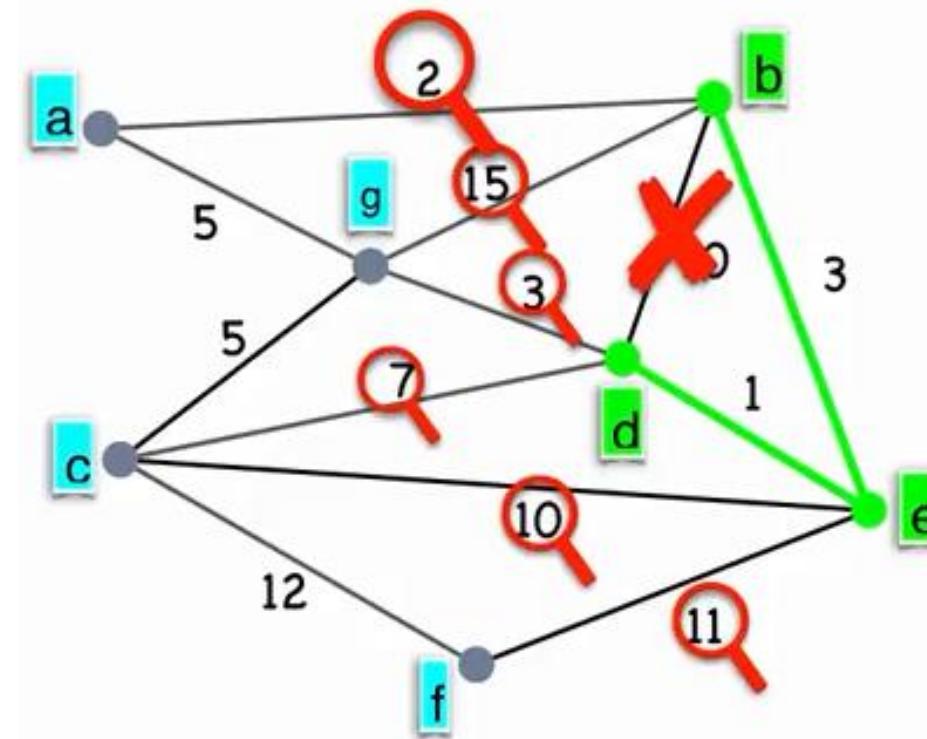
□ Arbre couvrant de poids minimal -Algorithme Prim

- Itération 2: l'Addition d'une arête de plus faible valeur et qui ne crée aucun cycle



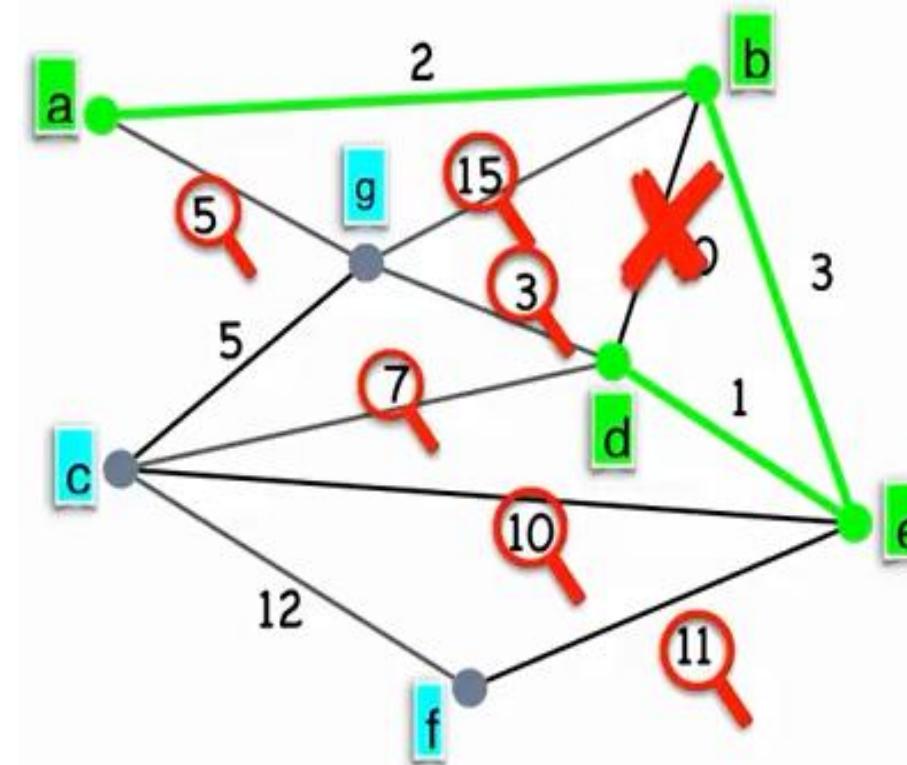
□ Arbre couvrant de poids minimal -Algorithme Prim

- Itération 2: l'Addition d'une arête de plus faible valeur et qui ne crée aucun cycle



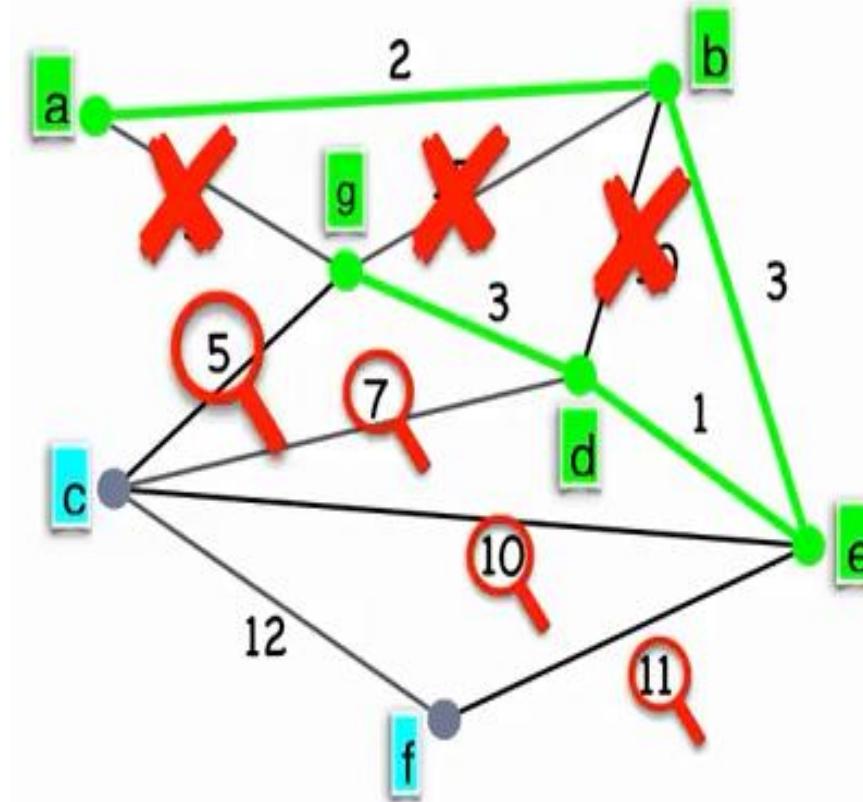
□ Arbre couvrant de poids minimal -Algorithme Prim

- Itération 2: l'Addition d'une arête de plus faible valeur et qui ne crée aucun cycle



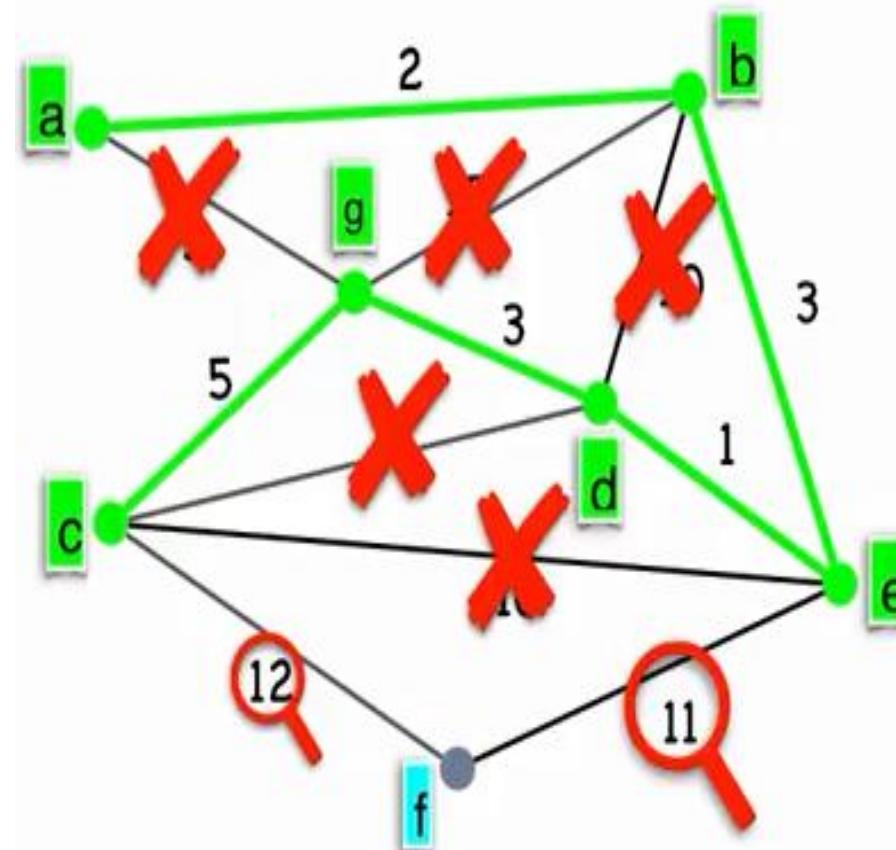
□ Arbre couvrant de poids minimal -Algorithme Prim

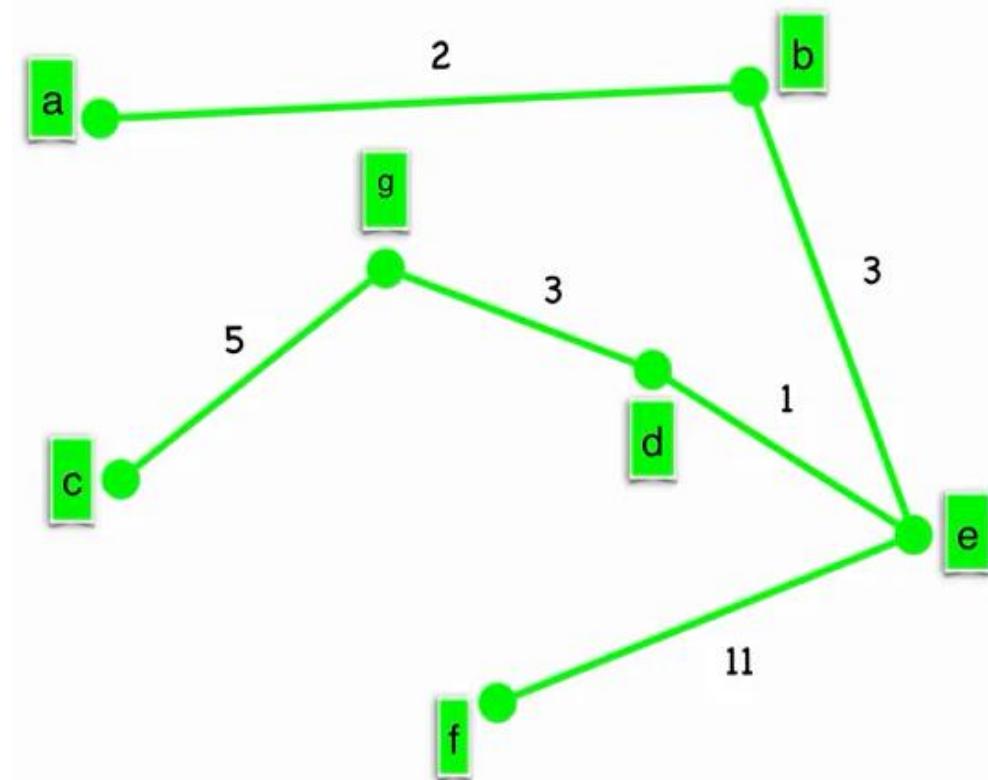
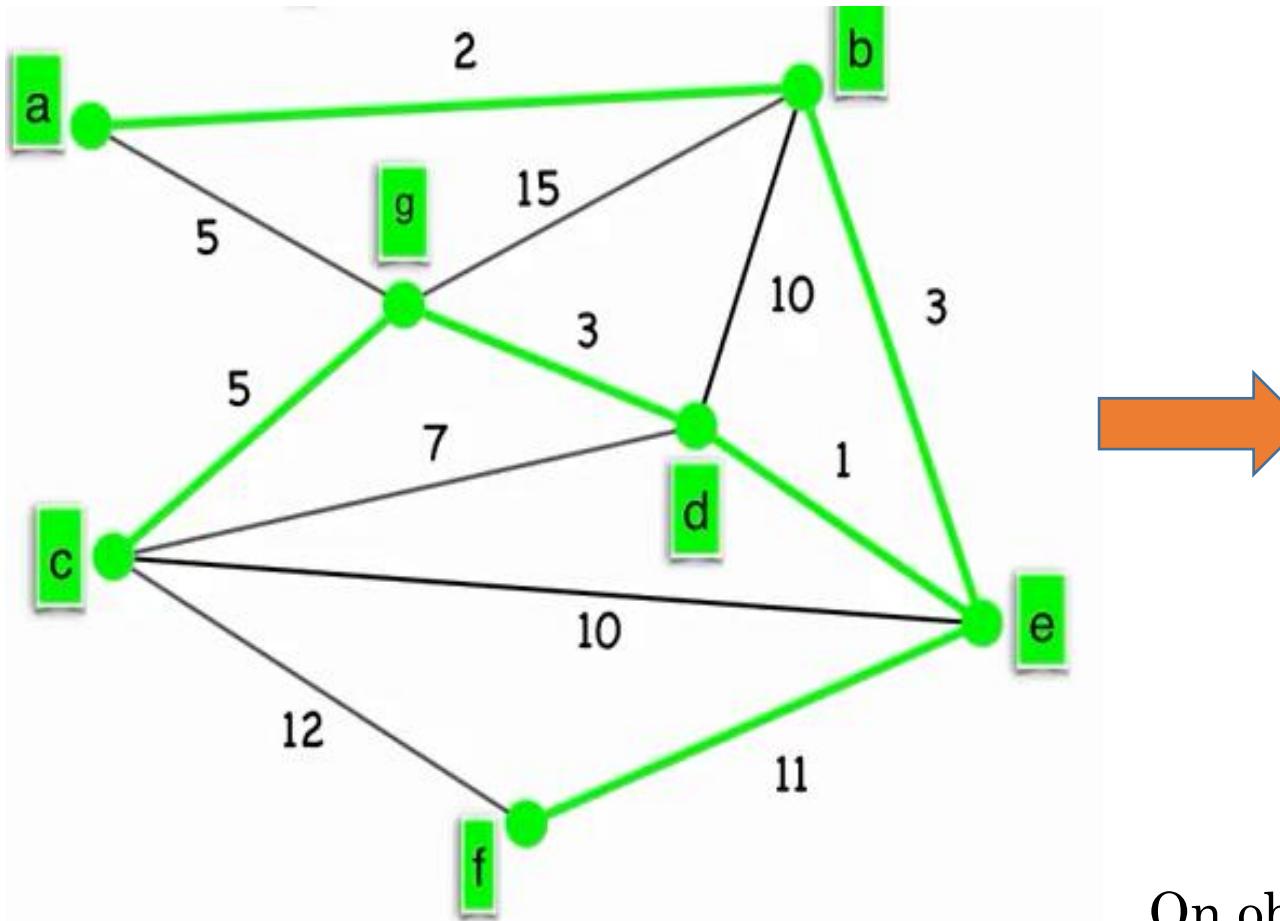
- Itération 2: l'Addition d'une arête de plus faible valeur et qui ne crée aucun cycle



□ Arbre couvrant de poids minimal -Algorithme Prim

- Itération 2: l'Addition d'une arête de plus faible valeur et qui ne crée aucun cycle



Arbre couvrant de poids minimal -Algorithme Prim

On obtient un arbre couvrant de poids minimal

$$\text{Poids (T)} = 2+3+11+1+3+5= 25$$

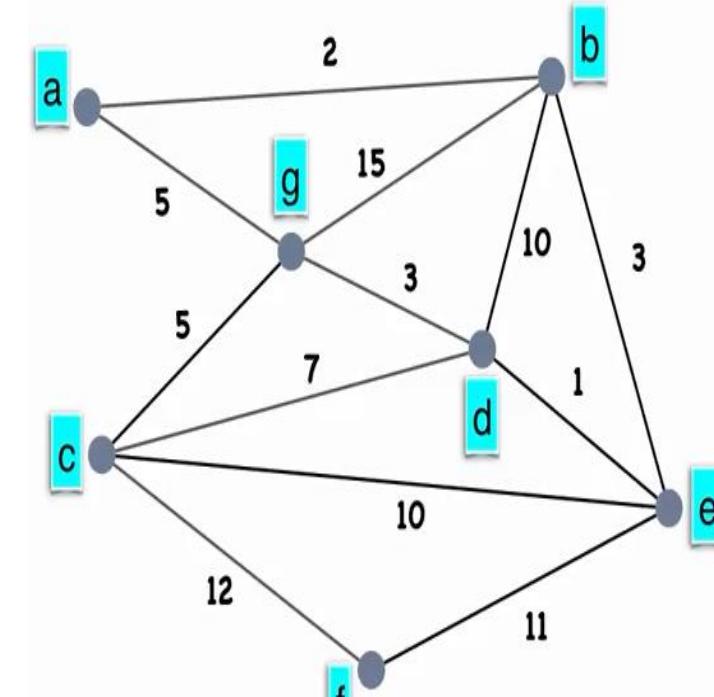
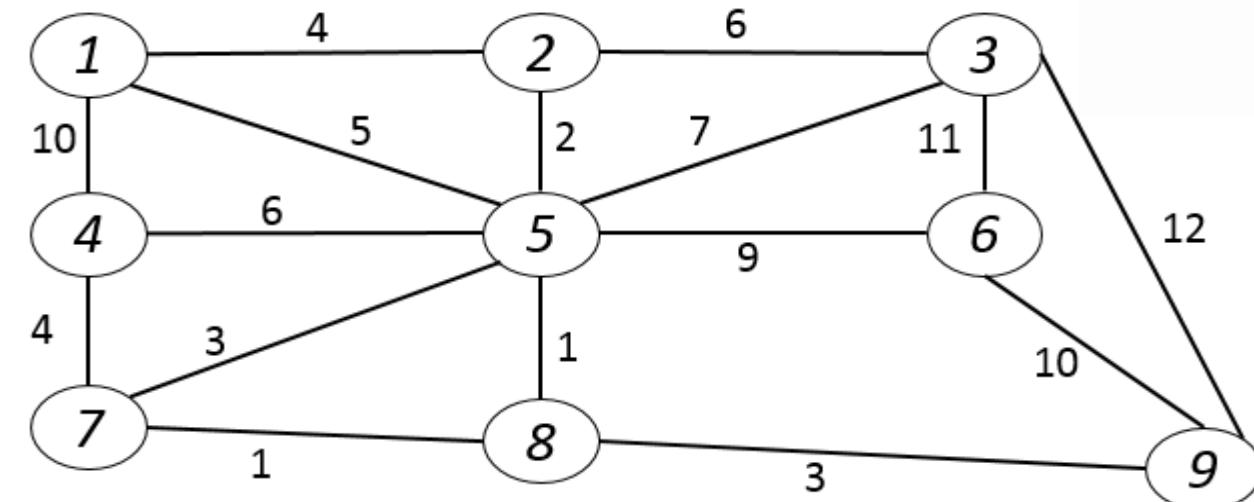
□ Arbre couvrant de poids minimal - Algorithme Prim

▪ Exercice 21

- Sur le graphe précédent, partez d'un autre sommet pour vérifier la stabilité du poids minimal

▪ Exercice 22

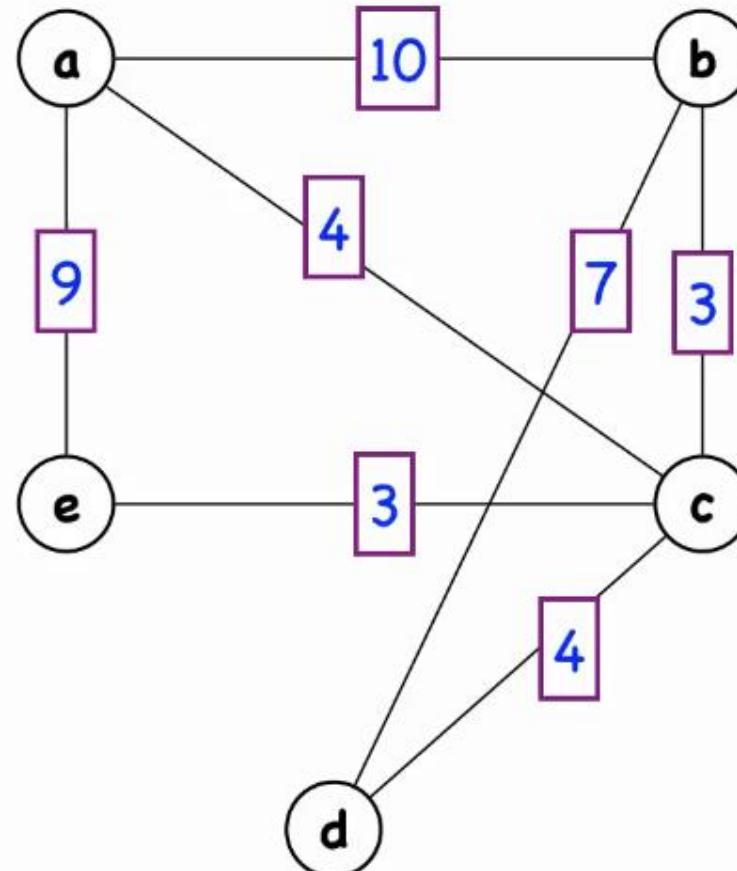
- Trouver l'arbre de recouvrement minimal en utilisant l'algorithme de Prim



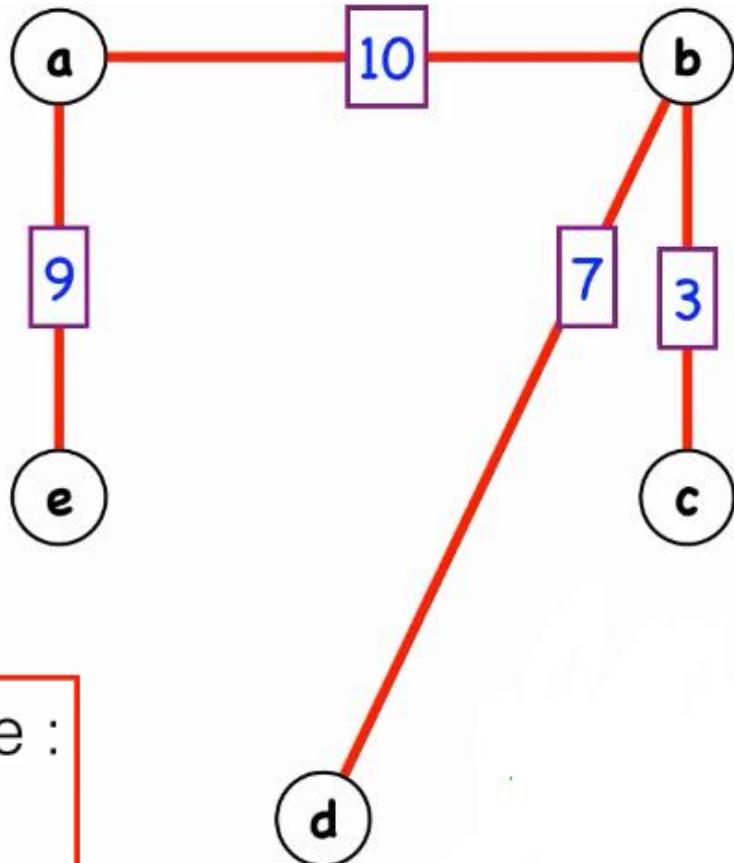
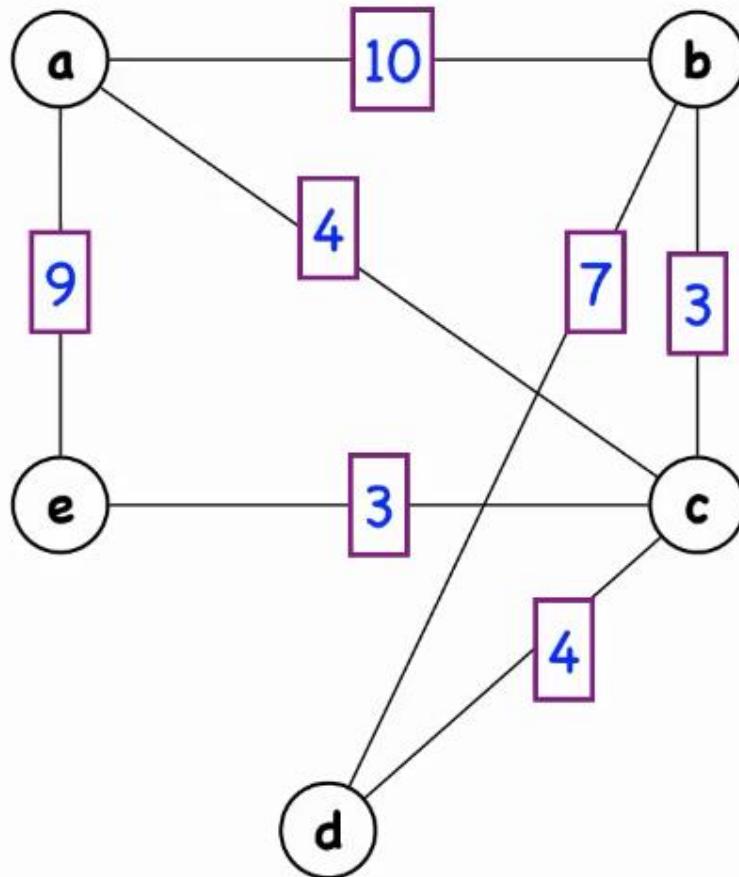
□ Arbre couvrant de poids minimal - Algorithme Kruskal

Les données du problème

- ▶ Un graphe G connexe et pondéré
- ▶ Poids de l'arête



On cherche un arbre couvrant de poids min.

Arbre couvrant de poids minimal - Algorithme Kruskal

Poids de cet arbre :
 $9+10+3+7 = 29$

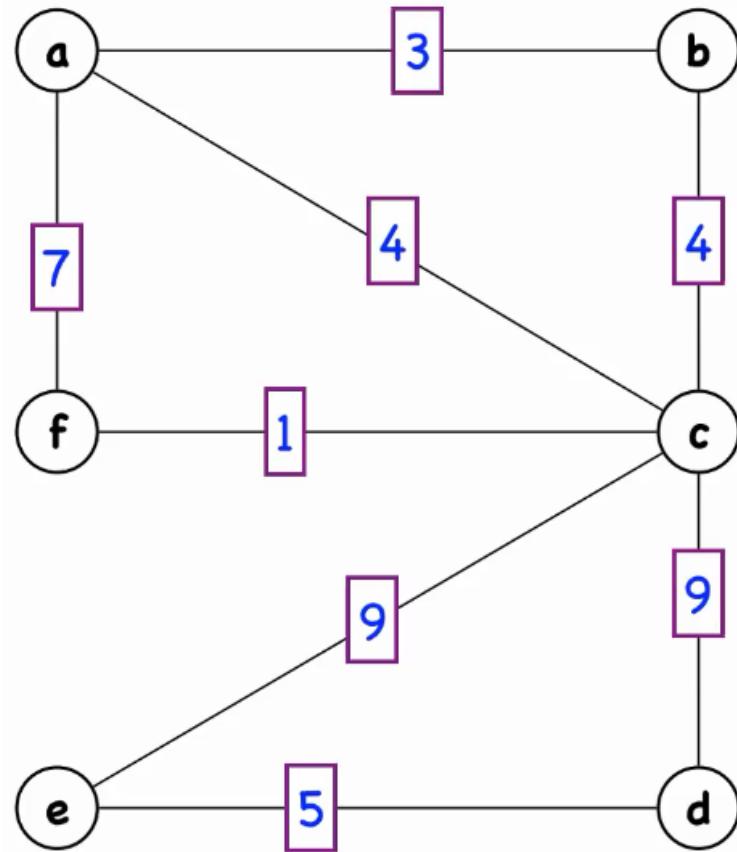
Ce poids est-il minimal ?

□ Arbre couvrant de poids minimal - Algorithme Kruskal

- Algorithme de Kruskal consiste à balayer les arêtes triées dans l'ordre (croissant ou décroissant selon le cas) et à choisir l'arête si les sommets ne sont pas déjà connectés
- Son principe est comme suit:
 - On dresse la liste des arrête d'un graphe, triés par poids croissant
 - On parcourt ensuite la liste triée :
 - on sélectionne l'arête de poids le plus faible; si elle ne crée pas de cycle, on l'ajoute
 - si la liste ne contient plus d'élément, on a terminé et l'algorithme s'arrête

□ Arbre couvrant de poids minimal - Algorithme Kruskal

- Comment trouver un tel arbre de poids minimal



► Étape 1: Trier les arêtes par poids croissant

f,c : 1

a,b : 3

b,c : 4

a,c : 4

e,d : 5

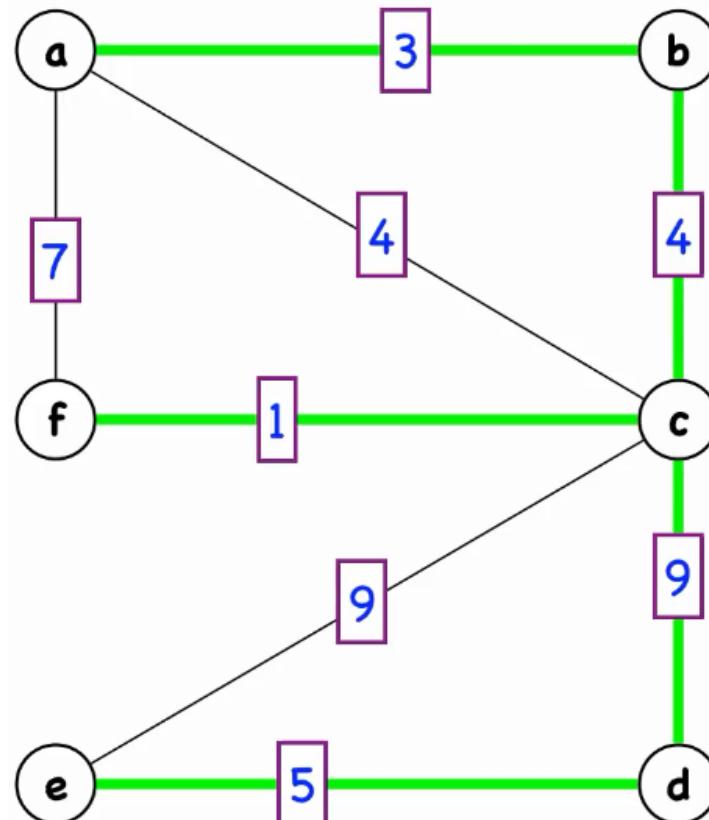
a,f : 7

c,d : 9

c,e : 9

□ Arbre couvrant de poids minimal - Algorithme Kruskal

- Comment trouver un tel arbre de poids minimal



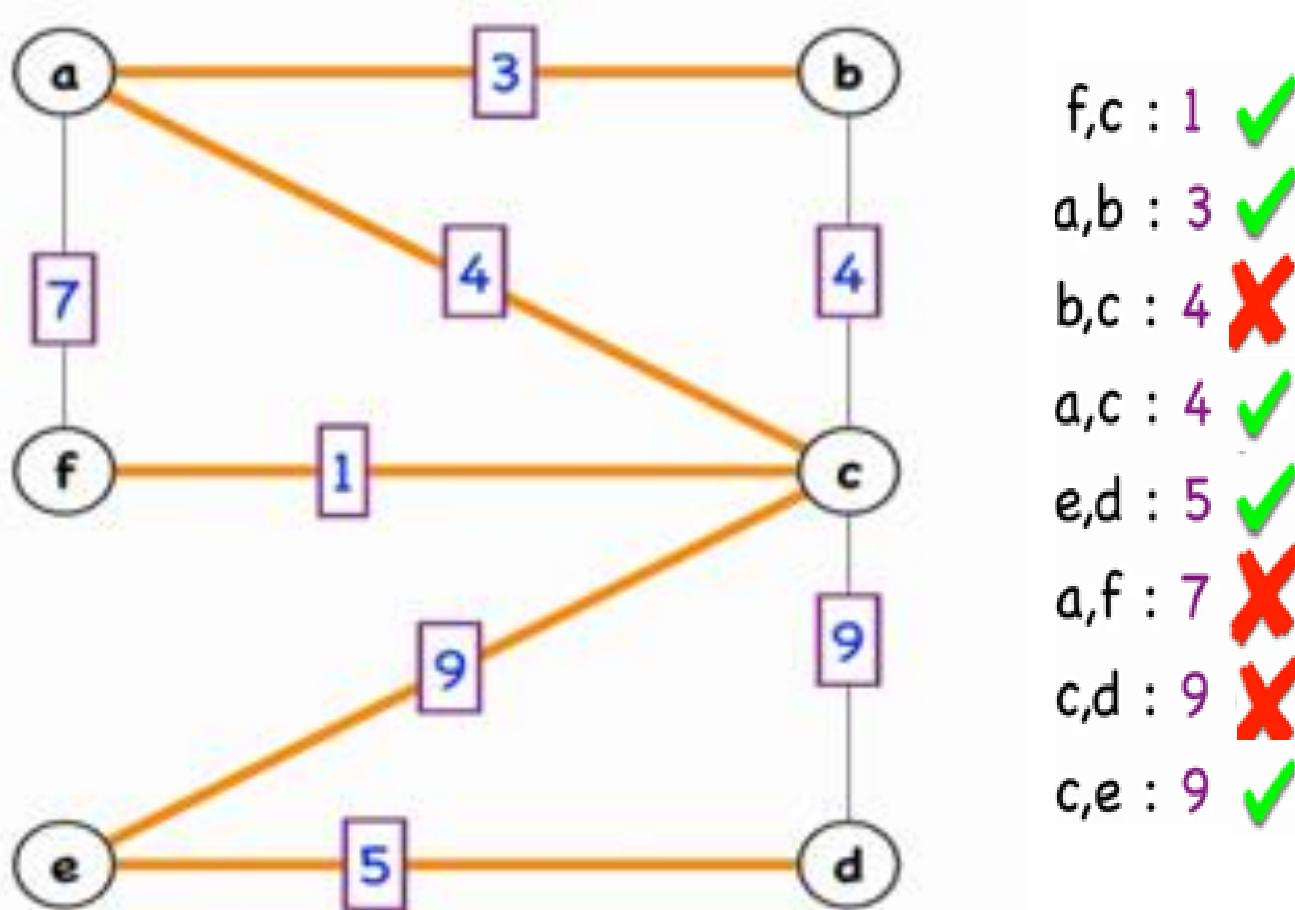
- Étape 1: Trier les arêtes par poids croissant

- f,c : 1 ✓
 - a,b : 3 ✓
 - b,c : 4 ✓
 - a,c : 4 ✗
 - e,d : 5 ✓
 - a,f : 7 ✗
 - c,d : 9 ✓
 - c,e : 9 ✗

Poids de cet arbre :
 $3+4+9+1+5= 22$

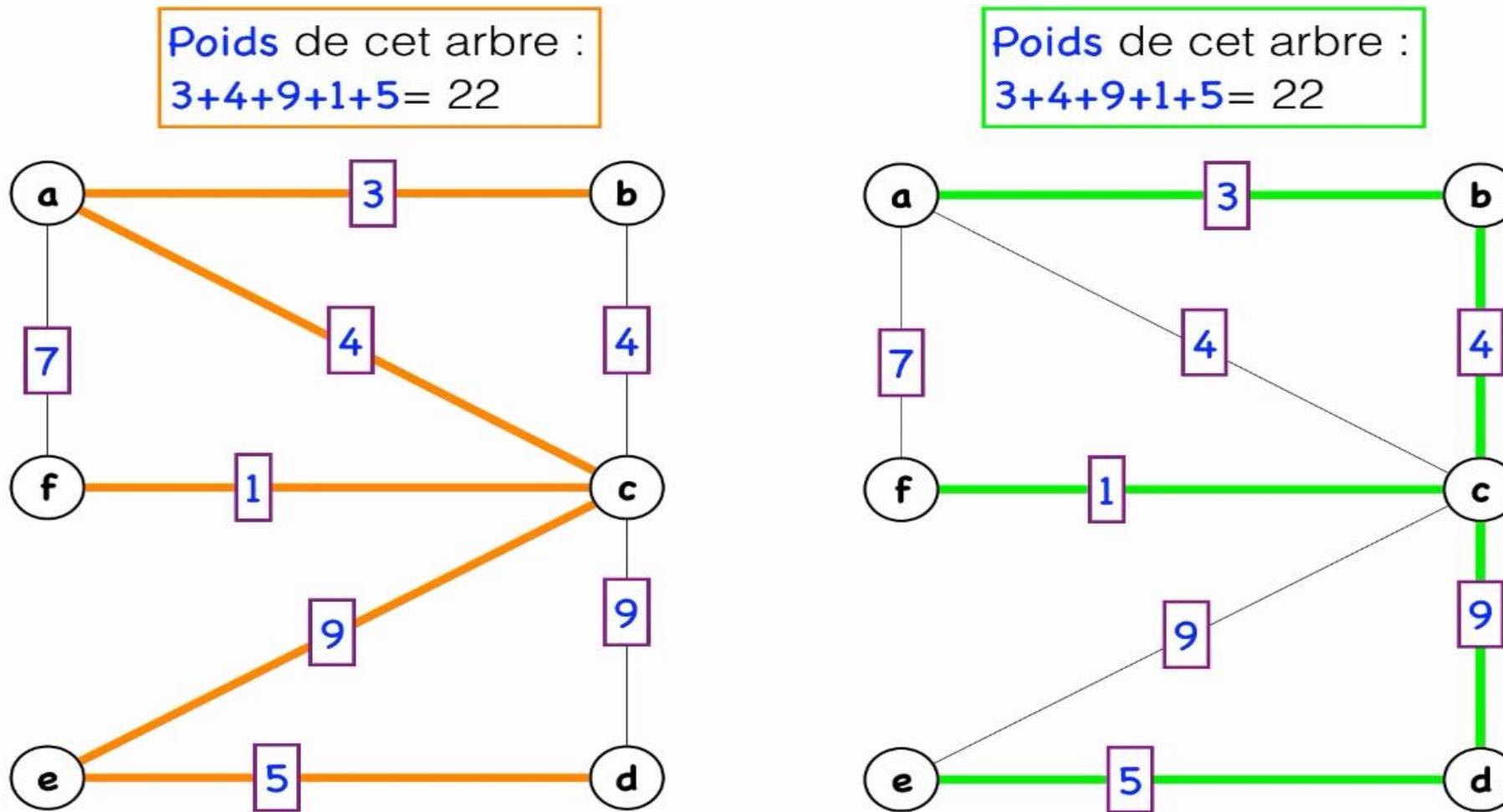
□ Arbre couvrant de poids minimal - Algorithme Kruskal

- Comment trouver un tel arbre de poids minimal



f,c : 1 ✓
a,b : 3 ✓
b,c : 4 ✗
a,c : 4 ✓
e,d : 5 ✓
a,f : 7 ✗
c,d : 9 ✗
c,e : 9 ✓

Poids de cet arbre :
 $3+4+9+1+5= 22$

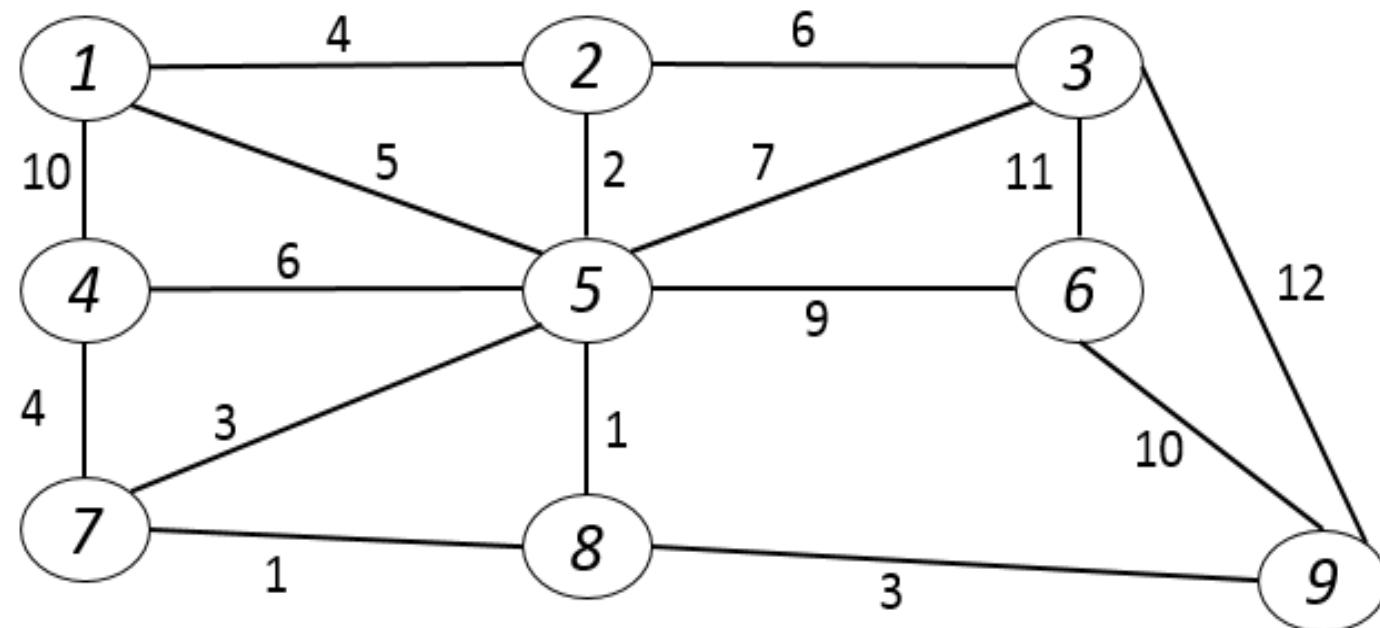
Arbre couvrant de poids minimal - Algorithme Kruskal

L'algorithme de Kruskal est stable

□ Arbre couvrant de poids minimal - Algorithme Kruskal

▪ Exercice 23

- Trouver l'Arbre couvrant de poids minimal en utilisant l'algorithme de Kruskal



□ Arbre couvrant de poids minimal

▪ Exercice 24

- Un Opérateur Télécom désire installer un réseau de transmissions de données entre son agence centrale située dans le quartier administratif à El Jadida et sept de ses succursales.
- Quelles sont les lignes à construire pour minimiser le coût du réseau ?

	Centrale	Essalam	Medina	Essaada	Manar	Facultés	Jawhara	Elmatar
Centrale	---							
Essalam	10	---						
Medina	5	10	---					
Essaada	15	14	30	---				
Manar	14	20	20	23	---			
Facultés	12	15	34	3	45	---		
Jawhara	20	29	16	4	18	10	---	
Elmatar	9	10	17	19	20	7	8	---

□ Arbre couvrant de poids minimal

- **Prim** construit l'arbre de recouvrement minimal en commençant par un sommet quelconque du graphe, alors que **Kruskal** construit l'arbre de recouvrement minimal à partir des arêtes.
- Le choix d'un de ces 2 algorithmes découle le plus souvent du choix de la représentation du graphe.

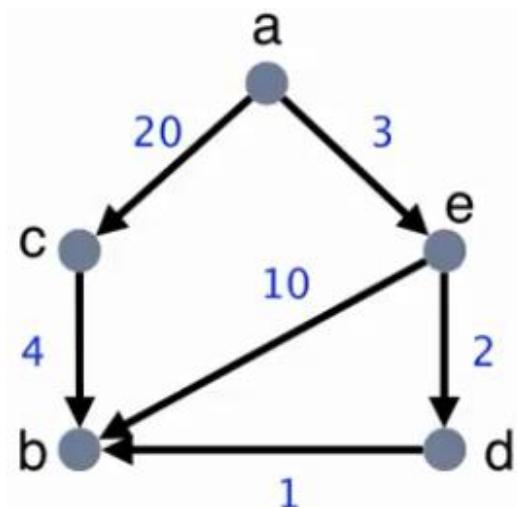
Problème du plus court chemin

□ Le problème du plus court chemin (PCC)

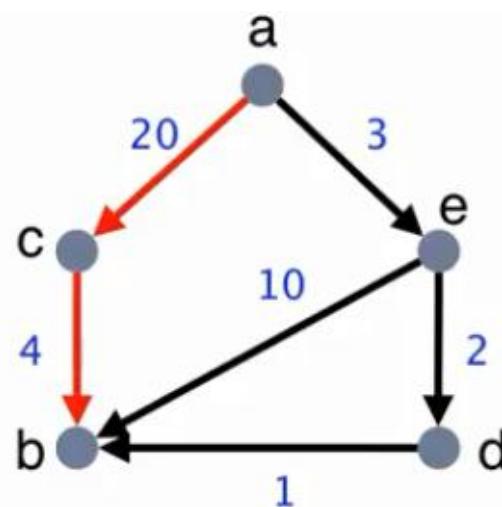
- Beaucoup de problèmes peuvent être modélisés en utilisant des graphes pondérés. Les problèmes de cheminement dans les graphes, en particulier la recherche du plus court chemin, comptent parmi les problèmes les plus anciens de la théorie des graphes et les plus importants par leurs applications : coût de transport, temps de parcours, problème de trafic, . . .
- Les algorithmes de recherche de plus court chemin seront différents selon les caractéristiques du graphe.
- On va étudier 2 algorithmes qui permettent de résoudre des problèmes de recherche de plus courts chemins à origine unique :
 - l'algorithme de **Dijkstra** résout ce problème lorsque tous les coûts sont **positifs ou nuls**,
 - l'algorithme de **Ford-Bellman** résout ce problème lorsque les coûts sont **positifs, nuls ou négatifs**,

□ Le problème du plus court chemin (PCC)

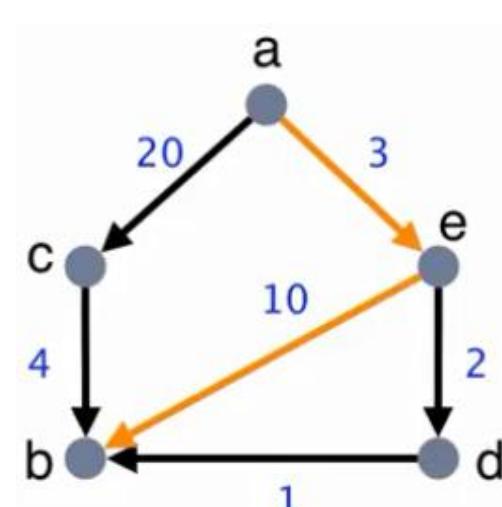
- Le problème du plus court chemin (PCC) compte parmi les problèmes le plus classiques de la théorie des graphes et les plus importants dans leurs applications (les problèmes d'optimisation de réseaux routiers ou télécommunications, les problèmes de tournées,...).



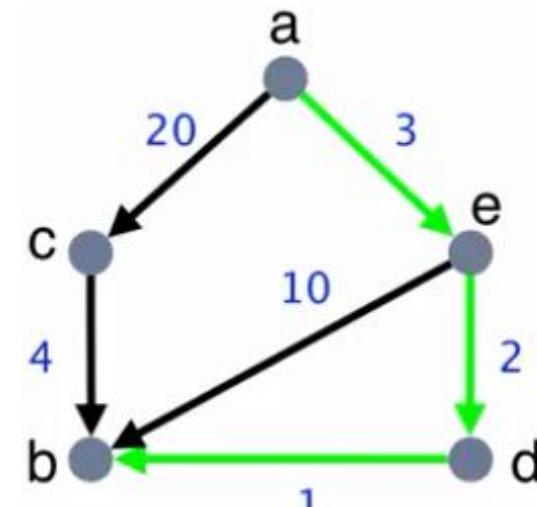
Le graphe donné



Solution 1



Solution 2



Solution 3

□ Algorithmes de résolution du (PCC)

- Selon les propriétés du graphe traité (orienté / non orienté, avec / sans circuit ou longueurs positives / quelconques) et selon le problème considéré (recherche du plus court chemin d'un sommet vers tous les autres, ou entre tous les couples de sommets), il existe de nombreux algorithmes permettant l'obtention d'une solution.

Algorithmes	Type du PCC	Propriétés du graphe	
		Type de graphe	Longueur
Dijkstra	D'un sommet à tous les autres sommets	Graphe orienté (et non orienté)	Longueur positives
Bellman-Ford		Graphe orienté sans circuit (sommet d'origine doit être sans prédecesseur)	Longueur quelconque (nombre réel)
Floyd	Entre tous les couples de sommets	Graphe orienté sans circuit absorbant	

□ Algorithme Dijkstra

- L'**algorithme de Dijkstra** sert à résoudre le problème du plus court chemin.
- Il permet, par exemple, de déterminer un plus court chemin pour se rendre d'une ville à une autre connaissant le réseau routier d'une région.
- Il calcule aussi des plus courts chemins à partir d'une source dans un graphe orienté ou non, et pondéré par des réels positifs.
- On peut aussi l'utiliser pour calculer un plus court chemin entre **un sommet de départ et un sommet d'arrivée**.
- Dijkstra peut être utilisé si le graphe (orienté ou non) **contient un cycle**.
- Dijkstra ne prend pas en considération les arcs qui ont **une pondération négative**.

□ Algorithme Dijkstra

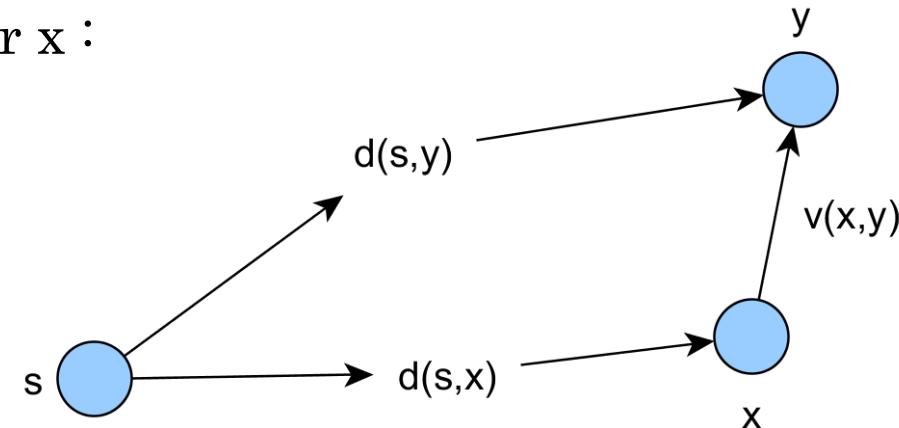
■ Principe :

- Si **n** est l'ordre du graphe, après une phase d'initialisation, cet algorithme procède **en n-1 itérations**, une par sommet différent de **s**.
- L'**initialisation** va consister à attribuer à chaque sommet successeur de **s** une distance provisoire égale **à la valuation** entre **s** et lui-même. Les autres sommets n'étant pas reliés directement à **s**, on initialise leur distance avec **la valeur $+\infty$** afin de ne pas interférer avec notre future recherche.
- Une fois cette étape terminée, on va commencer le **traitement** et les **itérations**.
- Au début de chaque itération, **on choisit un sommet x** parmi ceux qui n'ont pas encore été **traités**, dont la distance à **s** est **minimale**.

□ Algorithme Dijkstra

▪ Principe :

- Pour chacun des successeurs y de x , on compare ensuite l'évaluation actuelle de la distance $d(s,y)$ avec la distance $d(s,x)+v(x,y)$. On se demande donc si pour arriver à y il est plus judicieux ou non de passer par x :



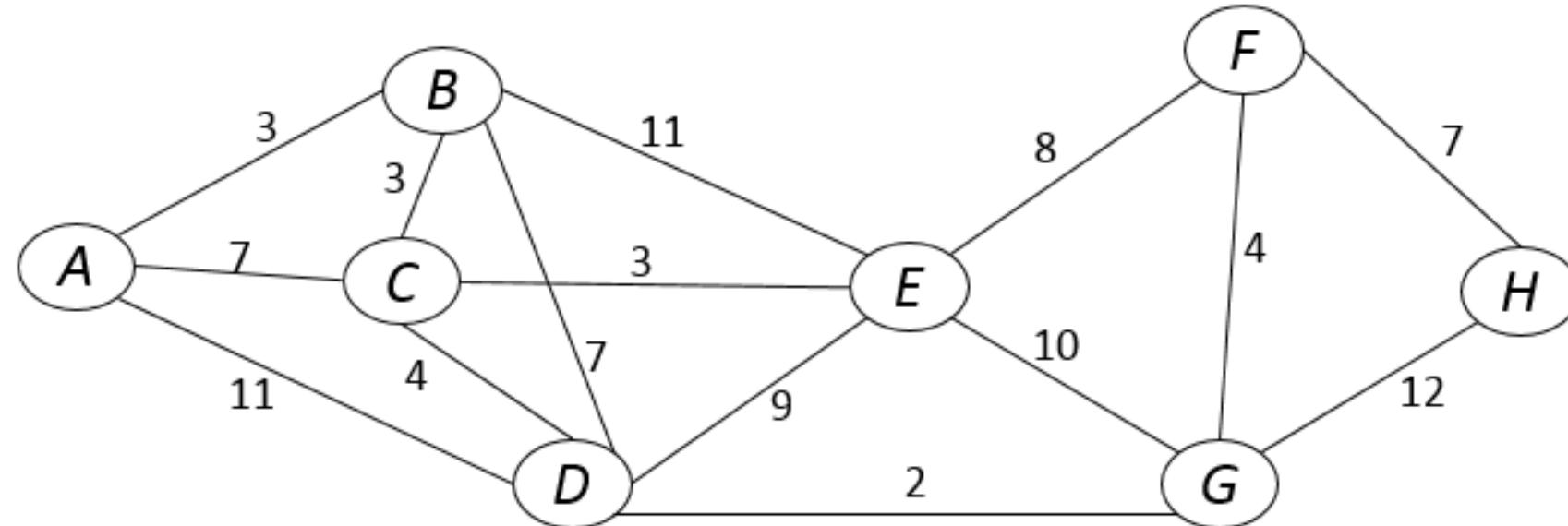
- On met alors éventuellement à jour la distance $d(s,y)$.
- Quand tous les successeurs du sommet x ont été examinés, on rajoute x à la liste des sommets traités, et l'on repart sur une nouvelle itération, avec le choix d'un nouveau sommet.

□ Algorithme Dijkstra

■ Exemple de l'application de Dijkstra

➤ But: Un conducteur veut se rendre de A à H en minimisant le temps du trajet.

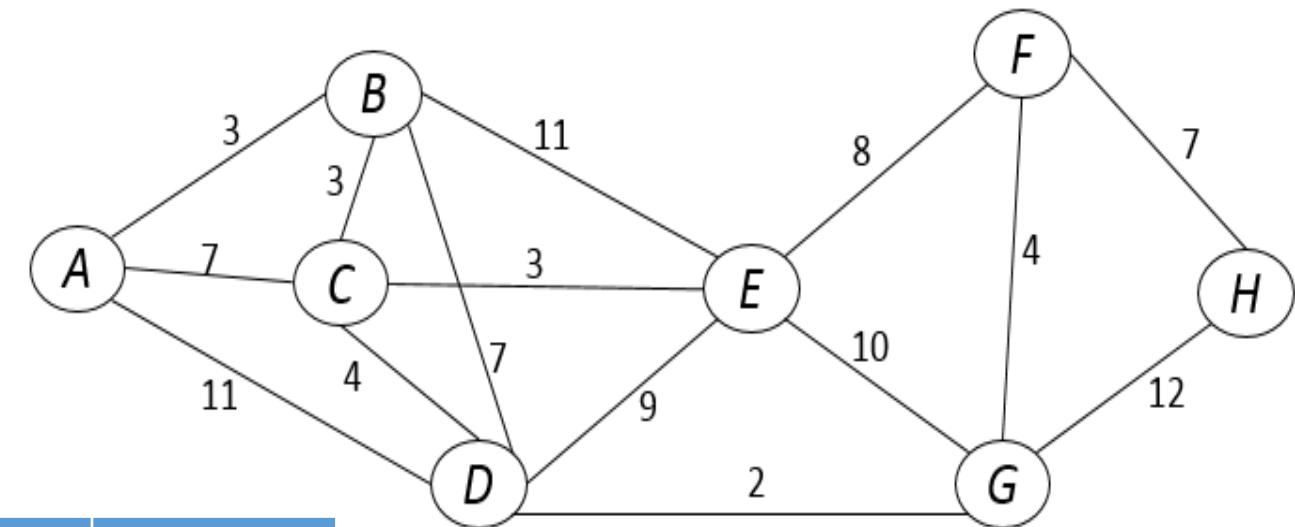
Déterminer le chemin et préciser le temps minimal pour ce parcours.



Problème du plus court chemin

□ Algorithme Dijkstra

- Processus de l'application de Dijkstra



A	B	C	D	E	F	G	H	Sommet
0	3(A)	7(A)	11(A)	∞	∞	∞	∞	B(3)
		6(B)	10(B)	14(B)	∞	∞	∞	C(6)
			10(C)	9(C)	∞	∞	∞	E(9)
			10(C)		17(E)	19(E)	∞	D(10)
					17(E)	12(D)	∞	G(12)
					16(G)		24(G)	F(16)
							23(F)	H(23)

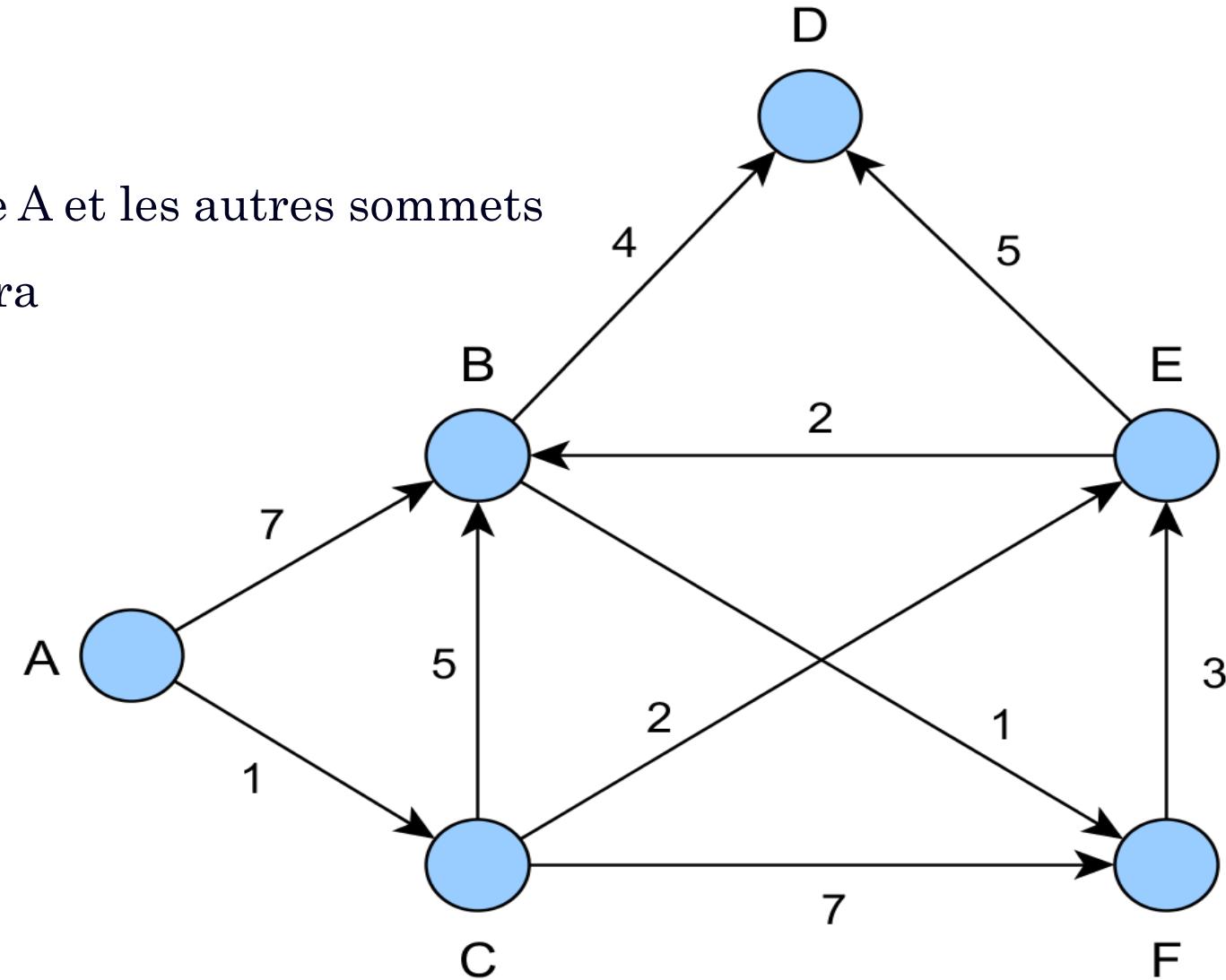
Le temps minimal pour aller de
A à H: 23 min

Le chemin: A B C D G F H

Algorithme Dijkstra

■ Exercice 25

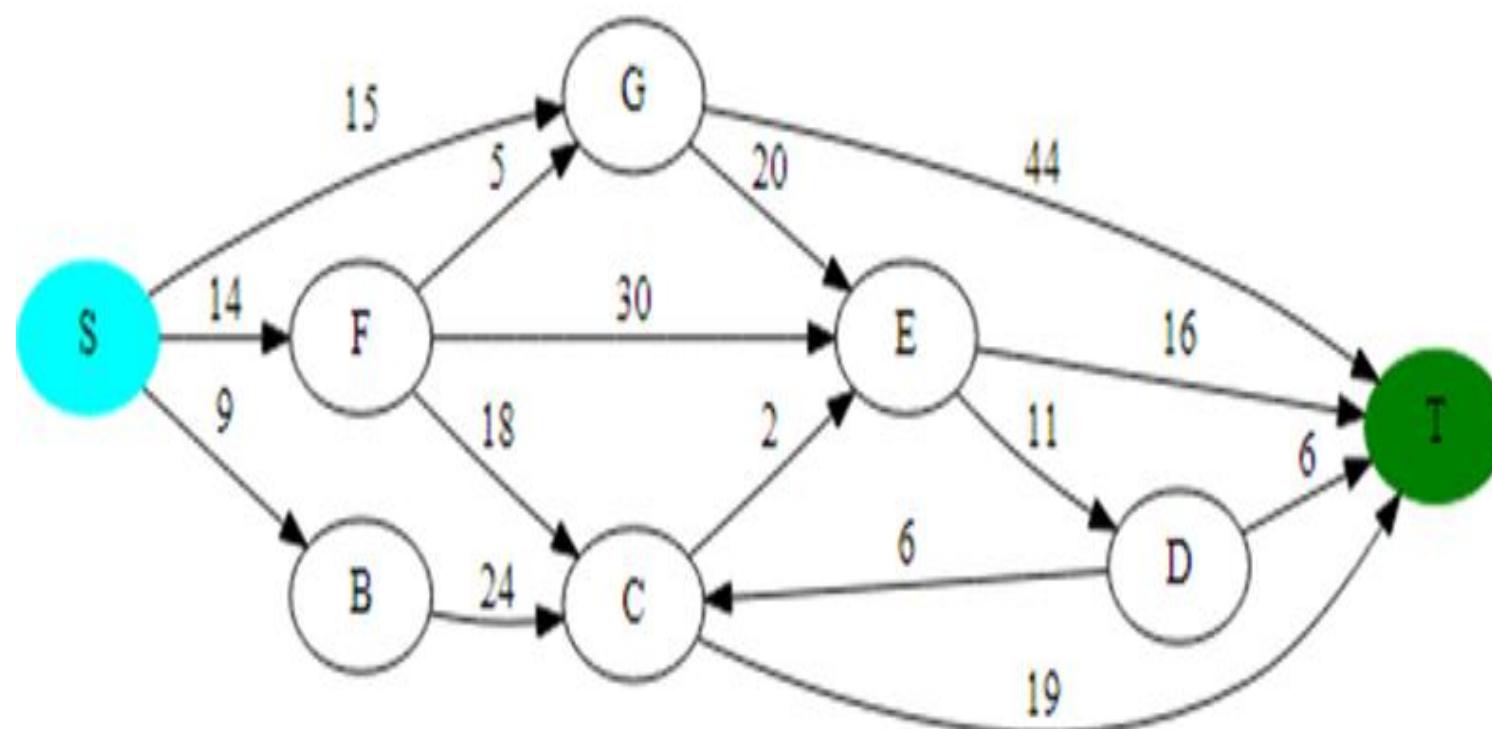
- Trouver le plus court chemin entre A et les autres sommets en utilisant l'algorithme de Dijkstra



Algorithme Dijkstra

- Exercice 26

Appliquez l'algorithme de Dijkstra sur le graphe suivant, entre le nœud bleu et le nœud vert :



□ Algorithme Dijkstra

▪ Exercice 27 facultatif

La matrice qui suit donne en heures les durées des vols entre certains aéroport a₁, a₂, ..., a₆ :

$$\begin{pmatrix} 0 & 3 & \infty & 5 & \infty & \infty \\ 3 & 0 & 5 & 2 & 4 & \infty \\ \infty & 4 & 0 & \infty & 4 & 3 \\ 6 & 2 & \infty & 0 & 4 & 4 \\ \infty & 4 & 4 & 5 & 0 & 2 \\ \infty & \infty & 5 & 4 & 3 & 0 \end{pmatrix}$$

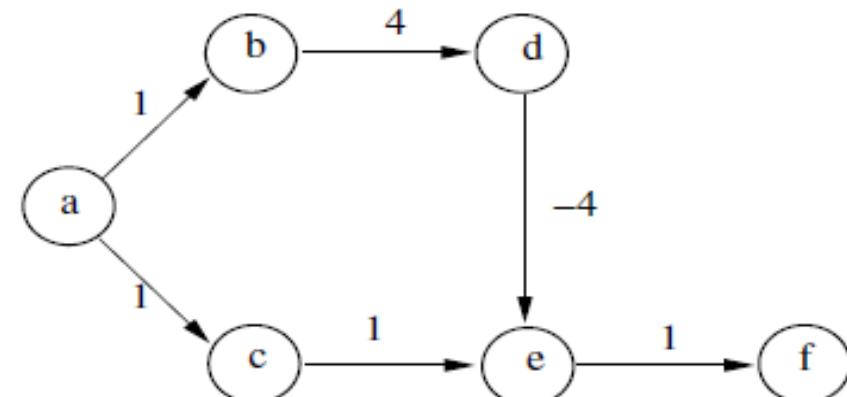
Le terme (i,j) de cette matrice est égal à ∞ lorsque le vol au départ de l'aéroport a_i à destination de a_j n'existe pas.

- En ne tenant pas compte de la durée des escales**, quel est l'itinéraire le plus rapide de a₁ à a₆.
- S'il y a une escale obligatoire respectivement 2, 3, 1, 1, 4, 5 heures aux aéroports a₁, ..., a₆, quel est alors l'itinéraire le plus rapide de a₁ à a₆ ?

□ Algorithme Bellman-Ford

- L'algorithme de Dijkstra ne marche pas toujours quand le graphe contient des arcs dont les coûts sont négatifs.

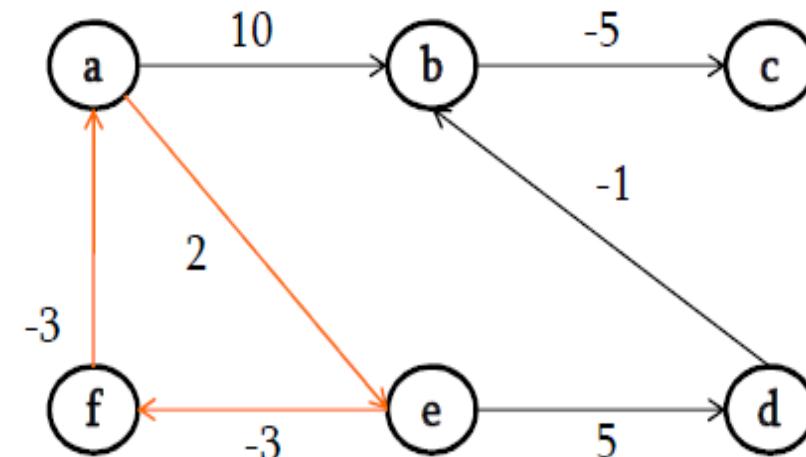
Considérons par exemple le graphe suivant :



- Pour aller de « **a** à **f** », l'algorithme de Dijkstra va trouver le chemin $\langle a; c; e; f \rangle$, de poids 3, alors que le chemin $\langle a; b; d; e; f \rangle$ est de poids 2.
- L'algorithme de Bellman-Ford permet de trouver les plus courts chemins à origine unique dans le cas où le **graphe contient des arcs dont le coût est négatif**, sous réserve que le graphe ne contient pas de **circuit absorbant** (dans ce cas, l'algorithme de Bellman-Ford va détecter l'existence de circuits absorbants).

□ Algorithme Bellman-Ford

- **Circuit absorbant:** Dans les graphes pondérés, le poids d'un circuit est la somme des poids des arcs qu'il contient. Si ce poids est négatif, on parle de **circuit absorbant**.
 - Un circuit négatif est appelé circuit absorbant
- **Condition Nécessaire:** Le problème du plus court chemin a une solution si et seulement s'il n'existe pas dans le graphe de circuit absorbant pouvant être atteint à partir de l'origine (s).
- Pour savoir s'il existe un circuit de cout négatif, il suffit de regarder s'il est possible d'aller de i à j pour un cout négatif.



□ Algorithme Bellman-Ford

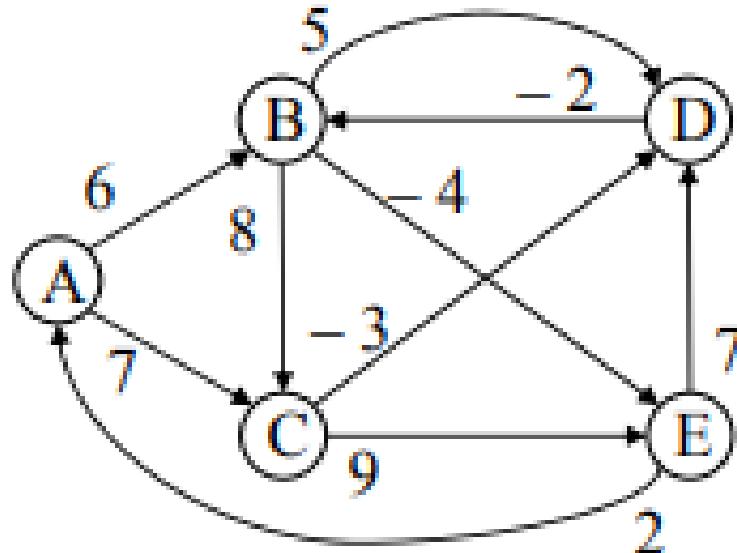
■ Principe:

Les notations sont les mêmes que précédemment. Au départ, la distance est initialisée à 0 pour l'origine et à ∞ pour les autres sommets.

```
pour k = 1 à n - 1 faire
{
    pour chaque arc (u, v) faire
        si (D[u] + L[u, v] < D[v]) alors
        {
            D[v] = D[u] + L[u, v];
            C[v] = u;
        }
    }
}
```

□ Algorithme Bellman-Ford- Exemple illustratif

► Exemple:



Exemple: $d(B) = \infty > d(A) + L(A,B) = 0 + 6$

$$d(B) = d(A) + L(A,B) = 0 + 6$$

Ordre d'examen des arcs et valuations :

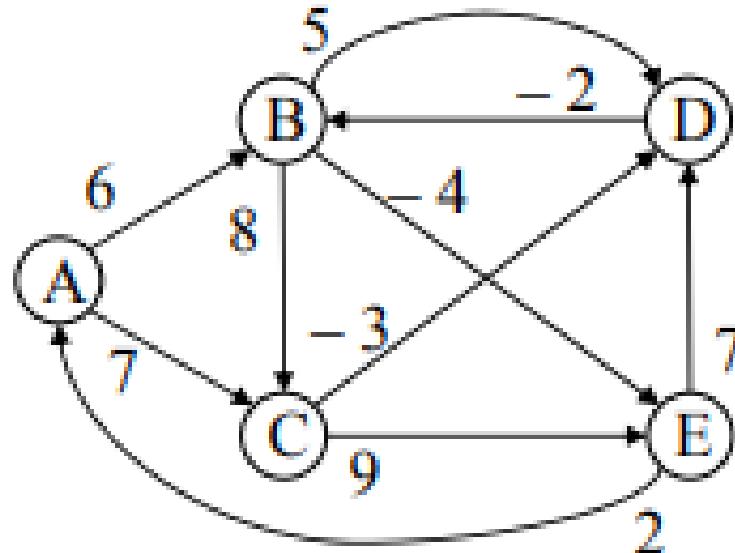
$$\begin{array}{llll} (A, B) = 6 & (A, C) = 7 & (B, C) = 8 & (B, D) = 5 \\ (B, E) = -4 & (C, D) = -3 & (C, E) = 9 & (D, B) = -2 \\ (E, A) = 2 & (E, D) = 7 \end{array}$$

It	A	B	C	D	E	Sommet
0	0	-	-	-	-	
1	0	6 ; A	7 ; A	-	-	
2	0	6 ; A	7 ; A	4 ; C	2 ; B	
3	0	2 ; D	7 ; A	4 ; C	2 ; B	
4	0	2 ; D	7 ; A	4 ; C	-2 ; B	

□ Algorithme Bellman-Ford- Exemple illustratif

- Pour vérifier qu'un graphe contient un circuit absorbant, on se base sur la remarque suivante:
si un graphe ne contient pas de circuits absorbants, l'algorithme de Bellman-Ford se termine en au plus $n-1$ itérations.
- Ainsi, dans un graphe quelconque si l'algorithme effectue encore une mise à jour des distances au bout de $n-1$ itérations, c'est que ce graphe contient **nécessairement un circuit absorbant**.
- Dans notre exemple, plus rien ne change dans les itérations suivantes (donc inutile de continuer...). Donc, **pas de circuit absorbant et les longueurs des plus courts chemins obtenues**.
- L'algorithme de Bellman-Ford permet de déterminer le PCC entre le sommet choisi A et tous les autres sommets. En effet, les PCC dans l'exemple précédent entre A et les autres sommets sont :

□ Algorithme Bellman-Ford- Exemple illustratif



It	A	B	C	D	E	Sommet
0	0	-	-	-	-	
1	0	6 ; A	7 ; A	-	-	A
2	0	6 ; A	7 ; A	4 ; C	2 ; B	C
3	0	2 ; D	7 ; A	4 ; C	2 ; B	D
4	0	2 ; D	7 ; A	4 ; C	-2 ; B	B

Le PCC entre :

- A-E : A-C-D-B-E
- A-D : A-C-D
- A-C : A-C
- A-B : A-C-D-B

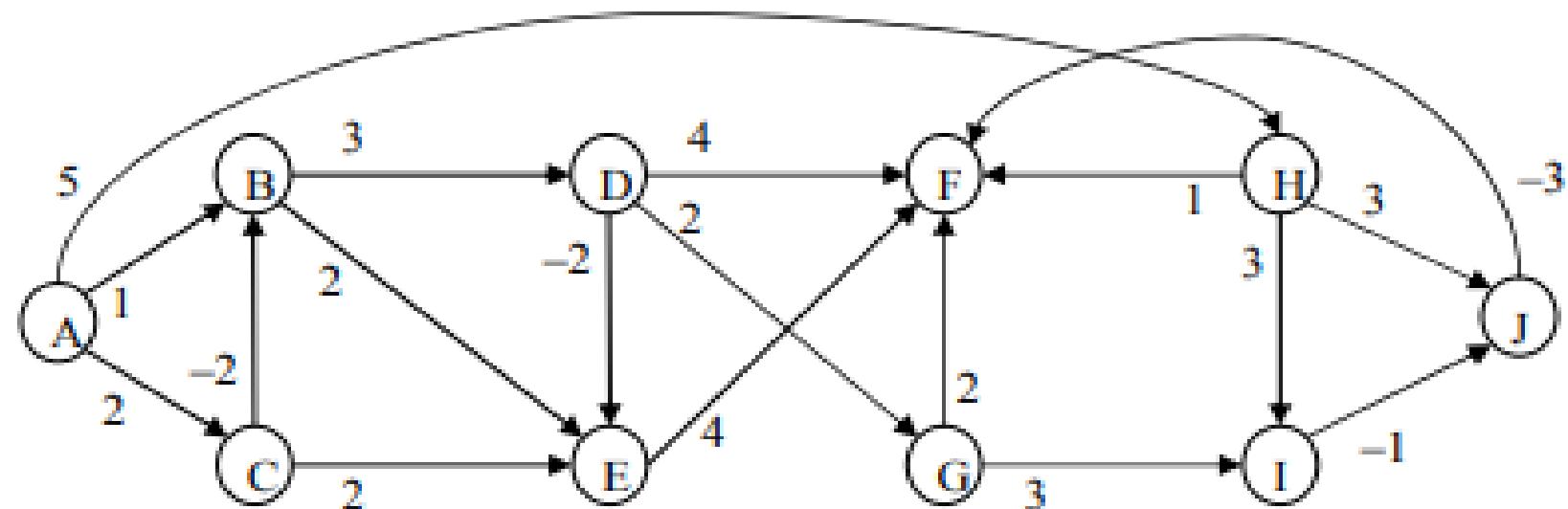
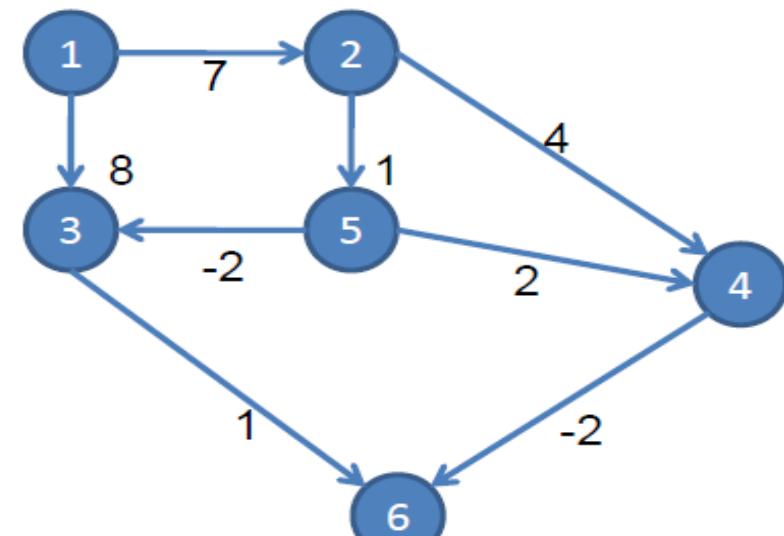
Algorithme Bellman-Ford

Exercice 28

- Trouver le plus court chemin entre le sommet 1 et tous les autres sommets

Exercice 29

- Trouver le plus court chemin de A à F



□ Algorithme de Floyd

- Cet algorithme permet **de calculer le PCC entre tous les couples de sommets** dans un graphe orienté **sans circuit absorbant** de longueur quelconque

- Numéroter les sommets de 1 à n ($|X| = n$)
- Soit la matrice $A = \{a_{ij}\}$ de taille $n \times n$ définie initialement comme suit:
- Soit π la matrice qui contient dans chaque cellule (i,j) ,

le prédécesseur de j dans le PCC entre i et j :

$$a_{ij} = \begin{cases} 0 & \text{si } i = j \\ l_{ij} & \text{si } (i,j) \in U \\ +\infty & \text{sinon} \end{cases}$$

$$\pi_{ij}^{(0)} = \begin{cases} Nil & \text{si } i = j \text{ ou } a_{ij} = \infty \\ i & \text{sinon} \end{cases}$$

- Cet algorithme permet de calculer le PCC de la façon suivante:
 - A la première itération (0), on cherche le PCC entre chaque couple (i, j) passant éventuellement par le sommet 1 ;
 - A l'itération k on cherche le PCC entre chaque couple (i, j) passant par des sommets d'indice inférieur ou égal à k ($\{1, \dots, n\}$).

□ Algorithme de Floyd

- Cet algorithme est valable quelles que soient les valuations des arcs, L'algorithme est constitué de N itérations principales ; pour chaque itération k , on calcule les plus courts chemins entre toute paire de sommets avec des sommets intermédiaires appartenant uniquement à l'ensemble $\{1,2,\dots,k\}$.
- A l'initialisation, on calcule le plus court chemin entre toute paire de sommets n'ayant pas de sommets intermédiaires, donc il suffit de prendre la longueur des arcs qui existent et mettre un poids infini si l'arc n'existe pas. Par la suite, si on note $A_{ij}^{(k)}$ la valeur du plus court chemin de i à j dont les seuls sommets intermédiaires sont dans l'ensemble $\{1,2,\dots,k\}$, alors on a l'égalité suivante :

$$A_{ij}^{(k)} = \min\left(A_{ij}^{(k-1)}, A_{ik}^{(k-1)} + A_{kj}^{(k-1)}\right) \text{ pour } 1 \leq k \leq n$$

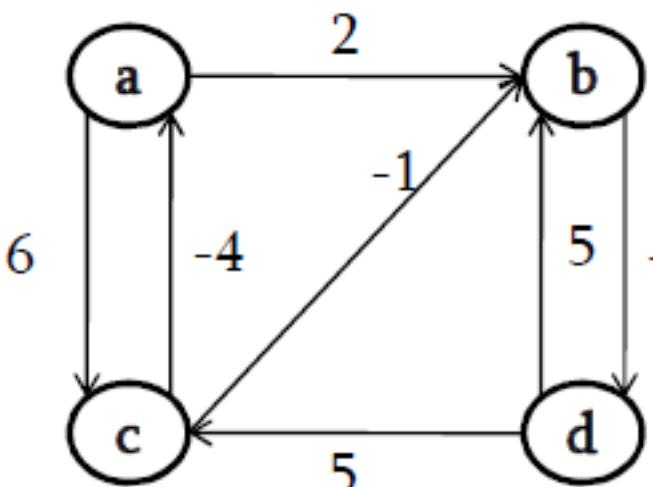
Pour la matrice pi on a :

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{si } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ \pi_{kj}^{(k-1)} & \text{si } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \end{cases}$$

Problème du plus court chemin

□ Algorithme de Floyd

- **Exemple:** Trouver PCC entre tous les couples des sommets



$$A^0 = \begin{pmatrix} 0 & 2 & 6 & \infty \\ \infty & 0 & \infty & -2 \\ -4 & -1 & 0 & \infty \\ \infty & 5 & 5 & 0 \end{pmatrix}$$

$$\pi^{(0)} = \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 3 & \text{Nil} & \text{Nil} \\ \text{Nil} & 4 & 4 & \text{Nil} \end{pmatrix}$$

$$A^1 = \begin{pmatrix} 0 & 2 & 6 & \infty \\ \infty & 0 & \infty & -2 \\ -4 & \mathbf{-2} & 0 & \infty \\ \infty & 5 & 5 & 0 \end{pmatrix}$$

$$\pi^{(1)} = \begin{pmatrix} \text{Nil} & 1 & 1 & \text{Nil} \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & \mathbf{1} & \text{Nil} & \text{Nil} \\ \text{Nil} & 4 & 4 & \text{Nil} \end{pmatrix}$$

$$A_{32} = \min(A_{32}, A_{31} + A_{12}) = -4 + 2 = -2$$

$$\pi_{32}^{(1)} = \pi_{12}^{(0)}$$

$$A^2 = \begin{pmatrix} 0 & 2 & 6 & 0 \\ \infty & 0 & \infty & -2 \\ -4 & -2 & 0 & \mathbf{-4} \\ \infty & 5 & 5 & 0 \end{pmatrix}$$

$$\pi^{(2)} = \begin{pmatrix} \text{Nil} & 1 & 1 & 2 \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & \mathbf{1} & \text{Nil} & 2 \\ \text{Nil} & 4 & 4 & \text{Nil} \end{pmatrix}$$

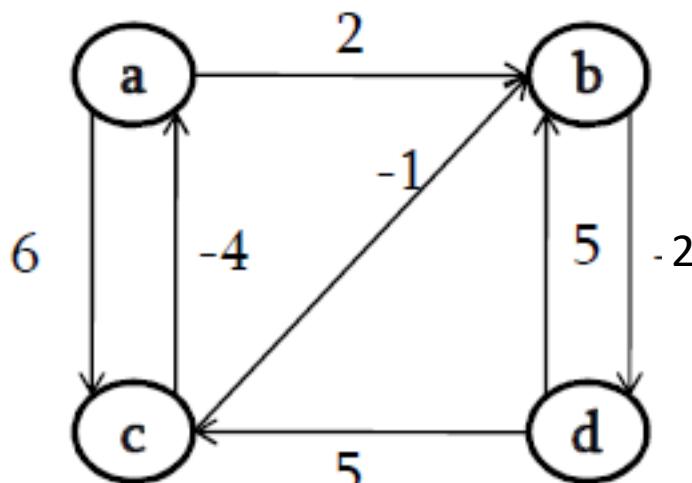
$$A_{14} = \min(A_{14}, A_{11} + A_{14}, A_{12} + A_{24})$$

$$\pi_{14}^{(2)} = \pi_{24}^{(1)} \quad \pi_{34}^{(2)} = \pi_{24}^{(1)}$$

$$A_{34} = \min(A_{34}, A_{31} + A_{14}, A_{32} + A_{24})$$

□ Algorithme de Floyd

- Exemple:



$$A^3 = \begin{pmatrix} 0 & 2 & 6 & 0 \\ \infty & 0 & \infty & -2 \\ -4 & -2 & 0 & -4 \\ 1 & 3 & 5 & 0 \end{pmatrix}$$

$$\pi^{(3)} = \begin{pmatrix} \text{Nil} & 1 & 1 & 2 \\ \text{Nil} & \text{Nil} & \text{Nil} & 2 \\ 3 & 1 & \text{Nil} & 2 \\ 3 & 1 & 4 & \text{Nil} \end{pmatrix}$$

$$A_{41} = \min(A_{41}, A_{41} + A_{11}, A_{42} + A_{21}, A_{43} + A_{31})$$

$$A_{42} = \min(A_{42}, A_{41} + A_{12}, A_{42} + A_{22}, A_{43} + A_{32})$$

$$A^4 = \begin{pmatrix} 0 & 2 & 5 & 0 \\ -1 & 0 & 3 & -2 \\ -4 & -2 & 0 & -4 \\ 1 & 3 & 5 & 0 \end{pmatrix}$$

$$\pi^{(4)} = \begin{pmatrix} \text{Nil} & 1 & 4 & 2 \\ 3 & \text{Nil} & 4 & 2 \\ 3 & 1 & \text{Nil} & 2 \\ 3 & 1 & 4 & \text{Nil} \end{pmatrix}$$

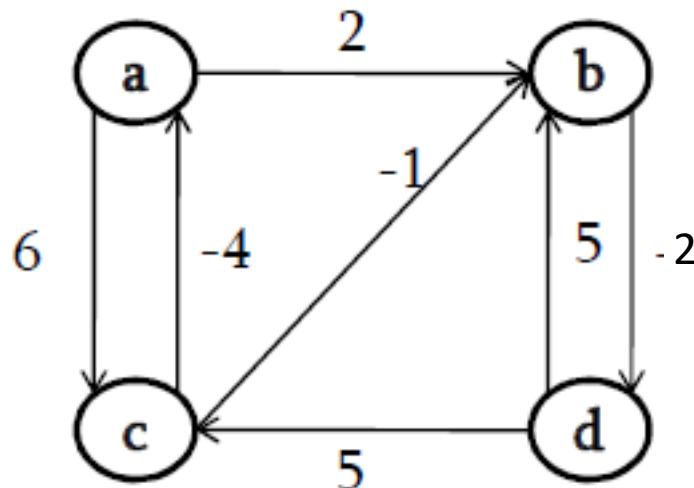
$$\pi_{21}^{(4)} = \pi_{41}^{(3)}$$

$$\pi_{13}^{(4)} = \pi_{43}^{(3)}$$

$$\pi_{23}^{(4)} = \pi_{43}^{(3)}$$

Algorithme de Floyd

- **Exemple:** Trouver PCC entre tous les couples des sommets



$$A^4 = \begin{pmatrix} 0 & 2 & 5 & 0 \\ -1 & 0 & 3 & -2 \\ -4 & -2 & 0 & -4 \\ 1 & 3 & 5 & 0 \end{pmatrix}$$

$$\pi^{(4)} = \begin{pmatrix} Nil & 1 & 4 & 2 \\ 3 & Nil & 4 & 2 \\ 3 & 1 & Nil & 2 \\ 3 & 1 & 4 & Nil \end{pmatrix}$$

Le PCC entre tous les sommets A et D est de langueur 0 : A-B-D

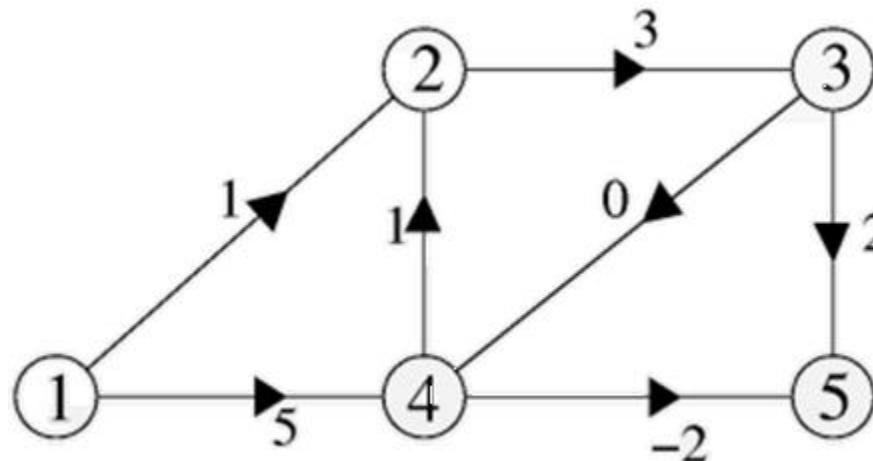
Le PCC entre tous les sommets A et C est de langueur 5 : A-B-D-C

Le PCC entre tous les sommets C et D est de langueur -4 : C-A-B-D

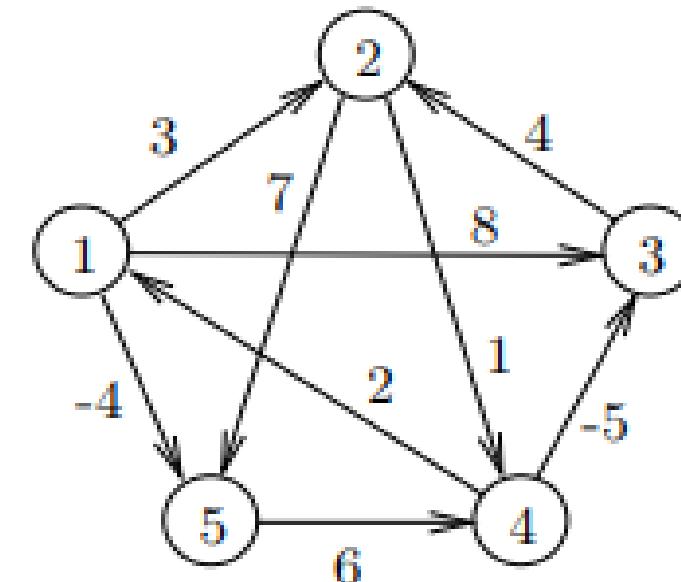
Le PCC entre tous les sommets B et C est de langueur 3 : B-D-C

Exercice 30

Appliquer l'algorithme de Floyd sur les graphes suivants:



$$A^{(5)} = \begin{pmatrix} 0 & 1 & 4 & 4 & 2 \\ \infty & 0 & 3 & 3 & 1 \\ \infty & 1 & 0 & 0 & -2 \\ \infty & 1 & 4 & 0 & -2 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$



Flots dans les réseaux

❑ Réseau de flot

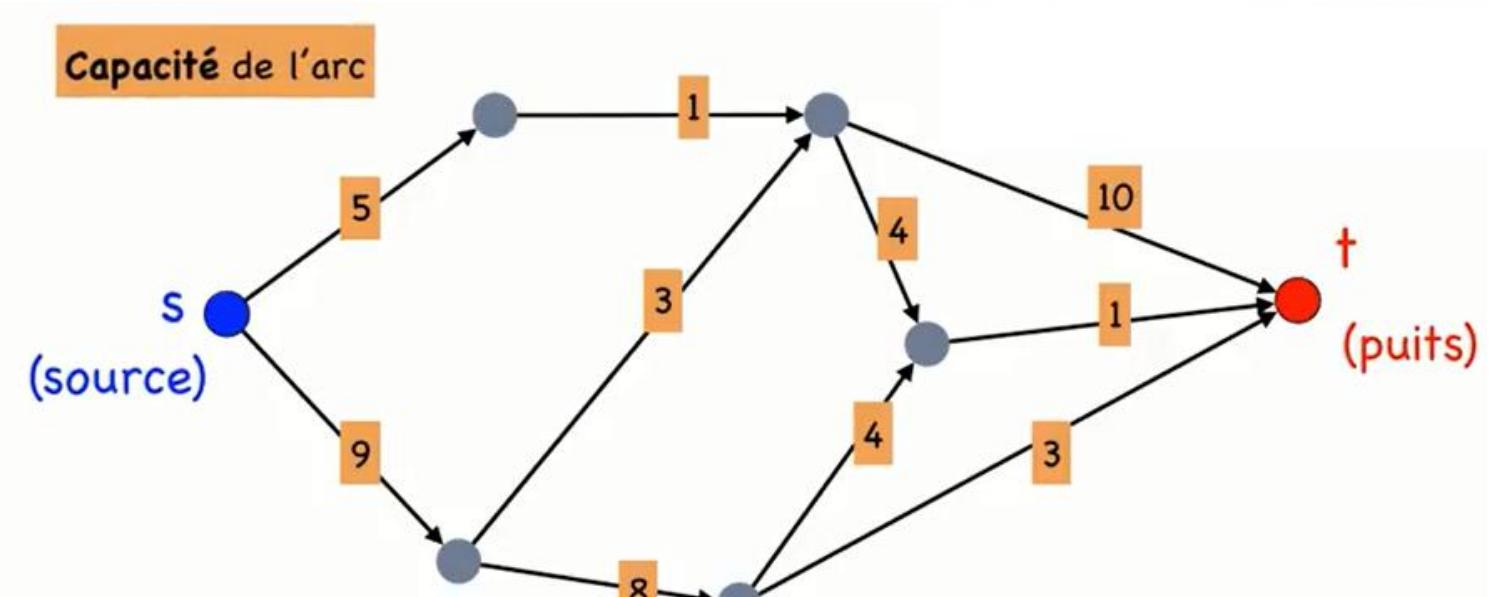
- Le problème de flot constitue un domaine d'application très important de la théorie des graphes.
- Il consiste à organiser, les mouvements de certaines quantités d'un bien dans un réseau.
- Le problème de flot maximum a été d'abord formulé en 1954 par TE Harris comme un modèle simplifié de la circulation ferroviaire soviétique.

Applications :

- Logistique : transport de marchandises : train, camion, bateau,...
- Distribution d'eau (canalisations)
- Transport de pétrole : réseau de pipelines
- Energie : réseau de transport et de distribution d'électricité
- Information : réseau téléphonique, réseau d'entreprises, internet.

□ Définition : réseau de transport

Un réseau de transport note $R = (G = (V, \vec{E}), s, t, c)$ est formé de :
 $G = (V, E)$ un graphe orienté
 $s \in V$ appelé **sommet source**
 $t \in V$ appelé **sommet destination ou puits**
 $c : \vec{E} \rightarrow N, Q+$ fonction capacité (à chaque arc $(i ; j) \in \vec{E}$ est associée une capacite $c(i ; j) \geq 0$).



□ Définition : Flot Réalisable

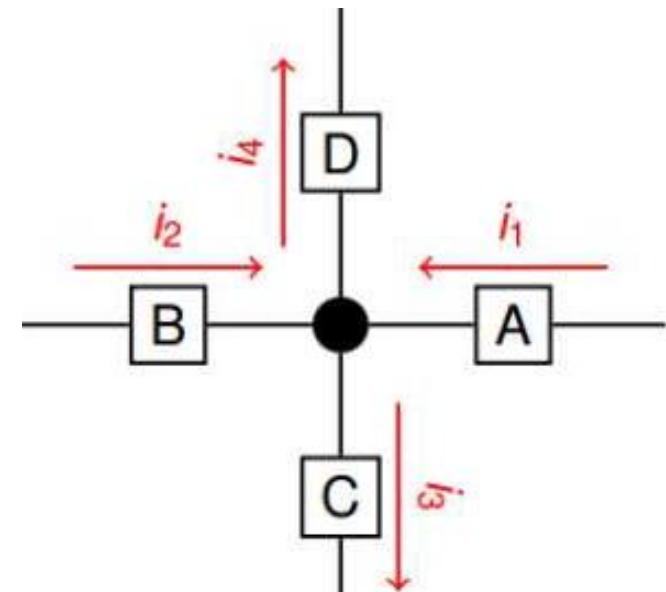
Soit $R = (G = (V, \vec{E}), s, t, c)$ un réseau. Un flot f dans R est une application $f : \vec{E} \rightarrow N, Q^+$.

Un flot f est réalisable dans R si :

- Contrainte de capacité $0 \leq f(i,j) \leq c(i,j) \quad \forall (i,j) \in \vec{E}$, on dit **flot compatible**
- Contraintes de conservation de flot (Loi de Kirschoff)

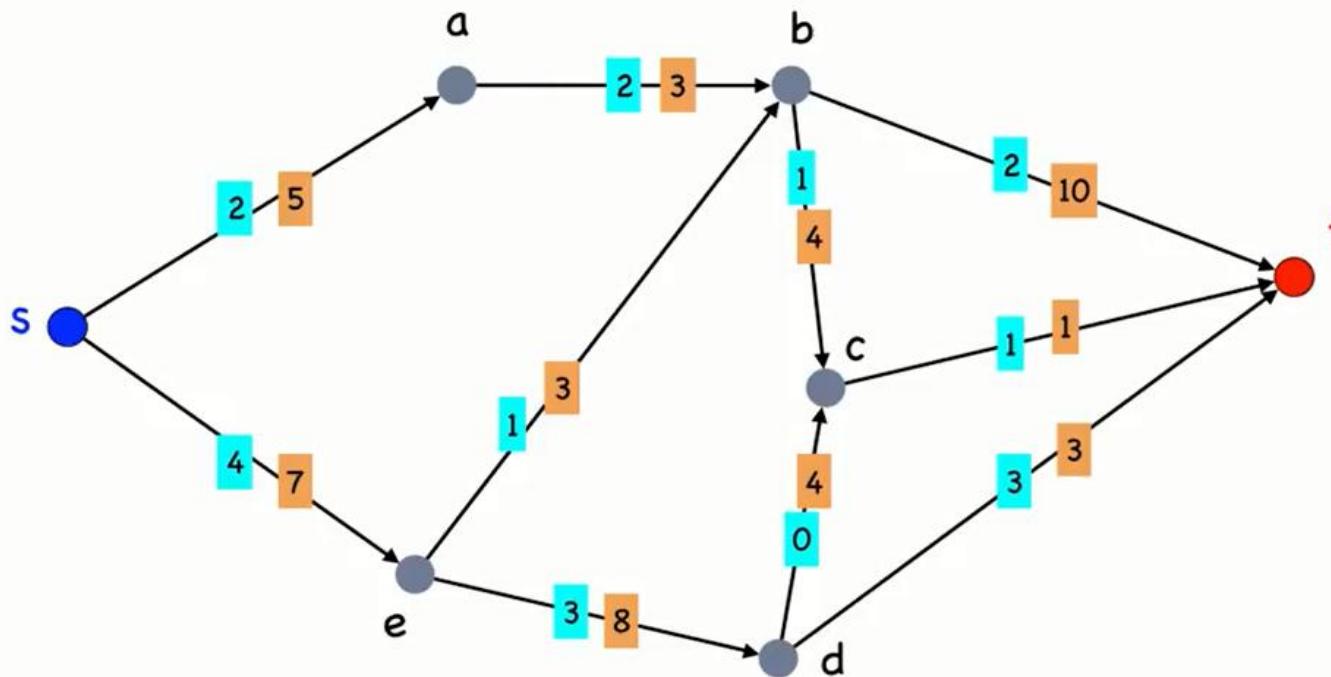
$$\sum_{i \mid (i,j) \in \vec{E}} f(i,j) - \sum_{k \mid (j,k) \in \vec{E}} f(j,k) = 0 \quad \forall j \in V \setminus \{s, t\}$$

(quantité qui entre dans j = quantité qui sort de j)



□ Définition : Flot Réalisable

- Exemple:



Contraintes de capacité: ?

Contraintes de conservation de flot: ?

Le flot est réalisable: ??

□ Définition : Valeur d'un Flot

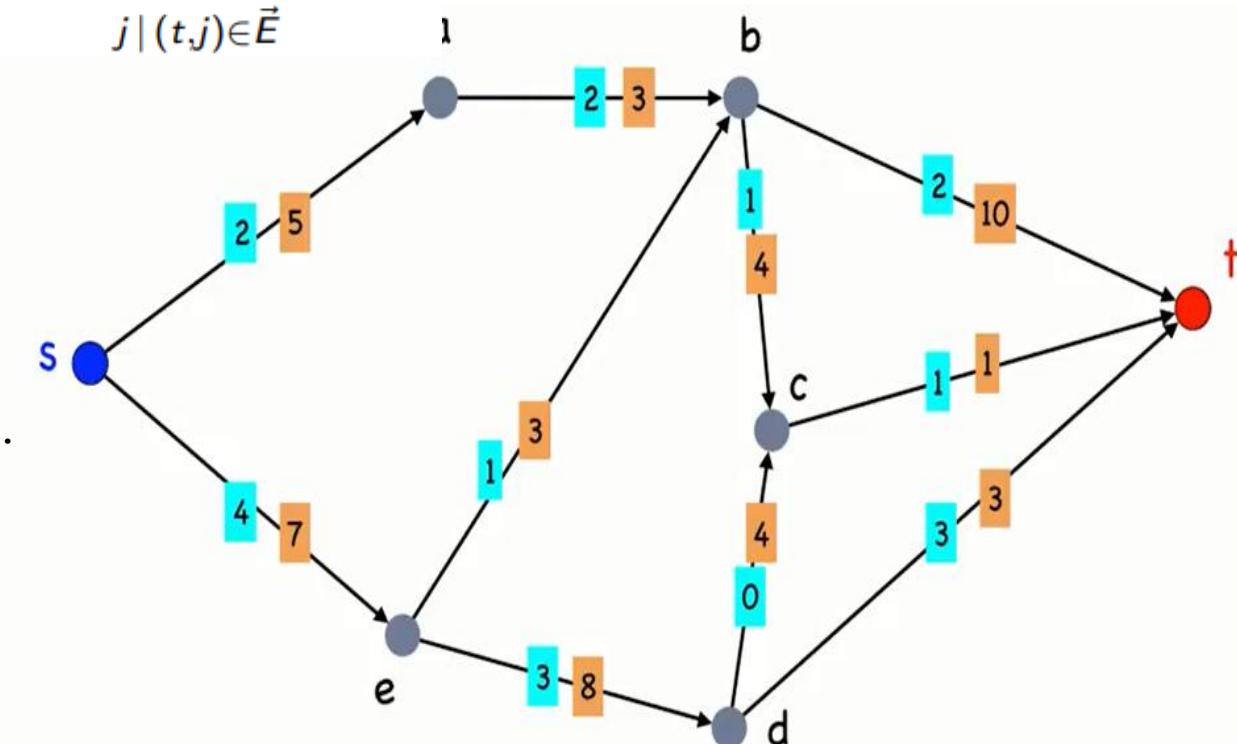
Soit $R = (G = (V, \vec{E}), s, t, c)$ un réseau

La valeur d'un flot réalisable F entre s et t est la quantité de flot envoyée de s à t . On la note F et

$$F = \sum_{i | (s,i) \in \vec{E}} f(s, i) - \sum_{j | (j,s) \in \vec{E}} f(j, s) = \sum_{i | (i,t) \in \vec{E}} f(i, t) - \sum_{j | (t,j) \in \vec{E}} f(t, j)$$

▪ Exemple :

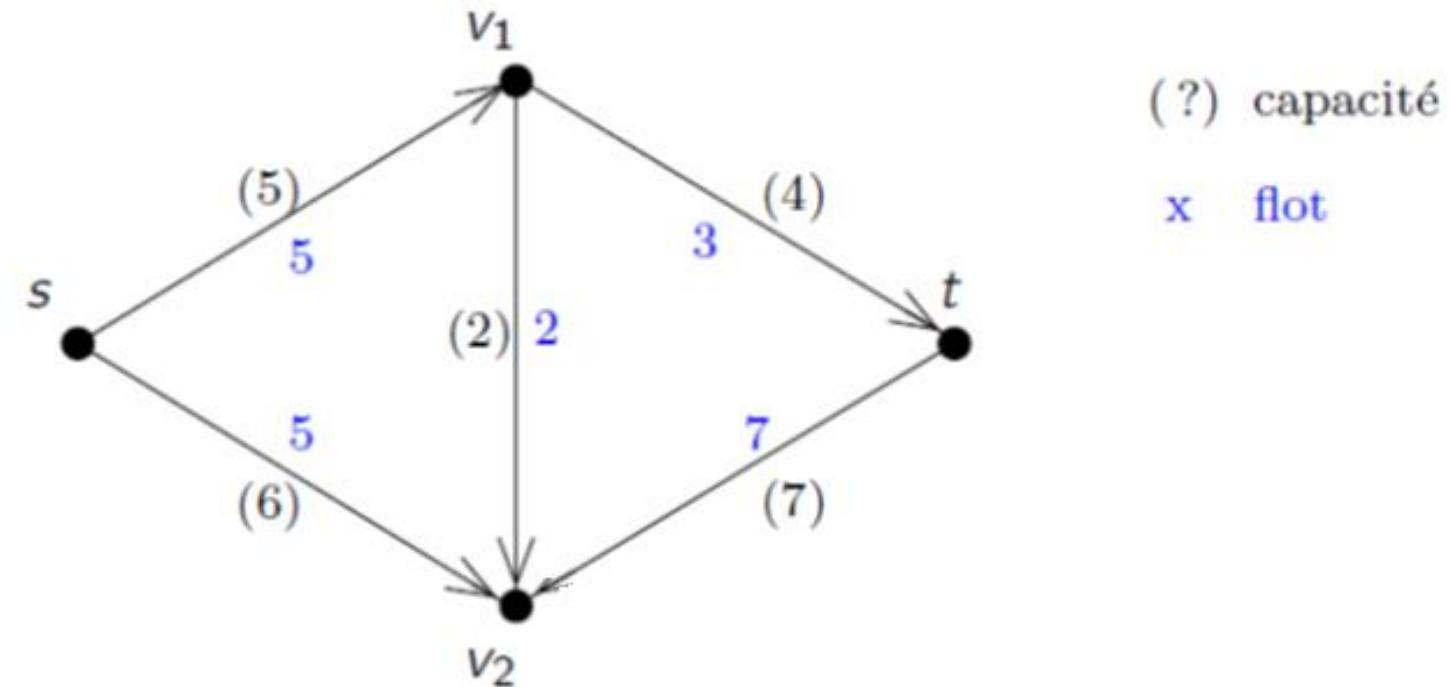
Ici, la valeur du flot est de 6. ($F = 2 + 4 = 2 + 1 + 3 = 6$).



□ Définition : Arc saturé

Un arc (i, j) est dit **saturé** pour un flot f si $f(i, j) = c(i, j)$.

- Exemple



Ici, les arcs $(s; v_1)$; $(v_1; v_2)$ et $(v_2; t)$ sont saturés.

Les arcs $(s; v_2)$ et $(v_1; t)$ ne le sont pas.

□ Définition : flot complet

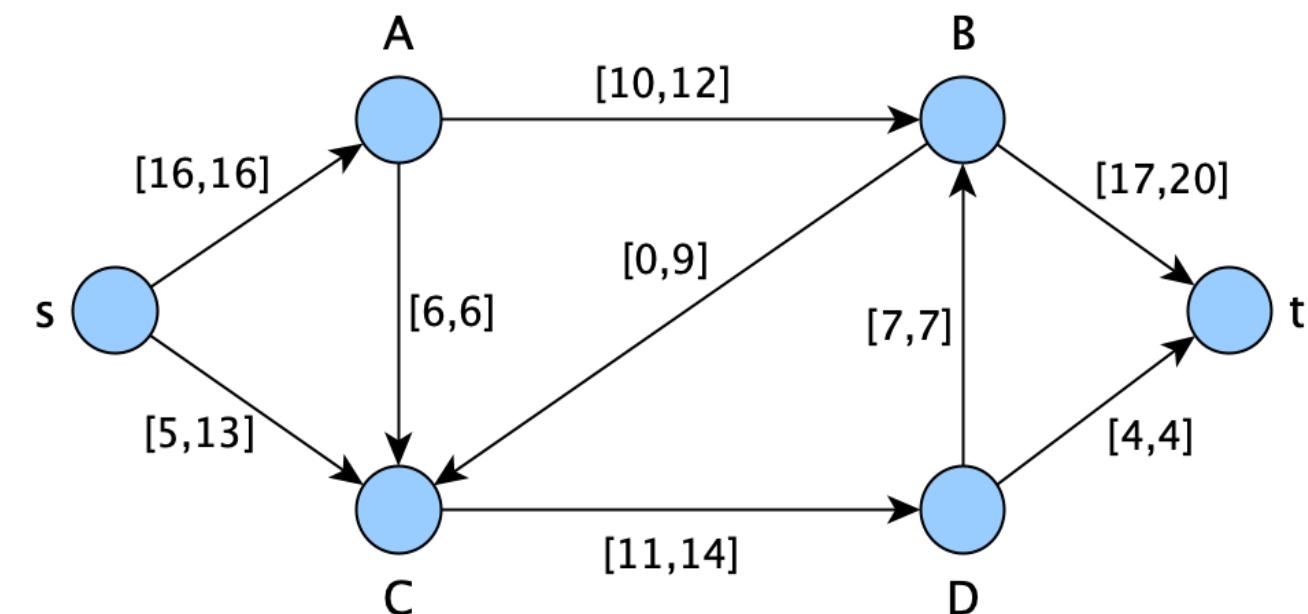
- **Définition**

Soit $G = (V, E)$ un réseau de transport.

On dit qu'un flot f circulant dans G est **complet** s'il est compatible et si chaque chemin de la source vers le puits possède un arc **saturé**.

- **Exemple: Un flot complet**

Voici un réseau de transport et
un **flot complet** y circulant

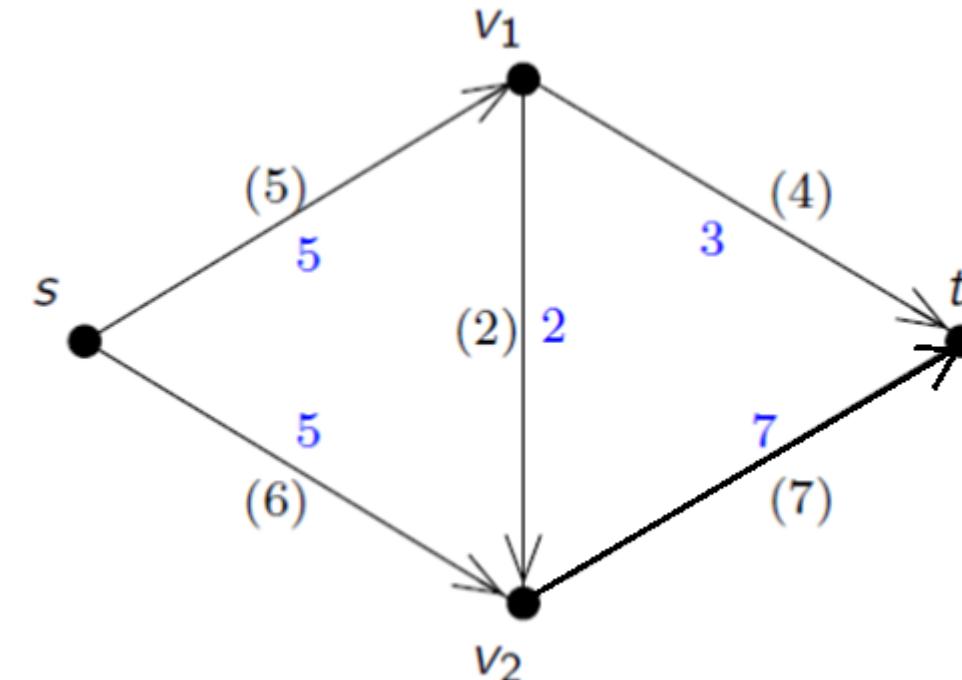


□ Le problème du flot maximum

▪ Problème :

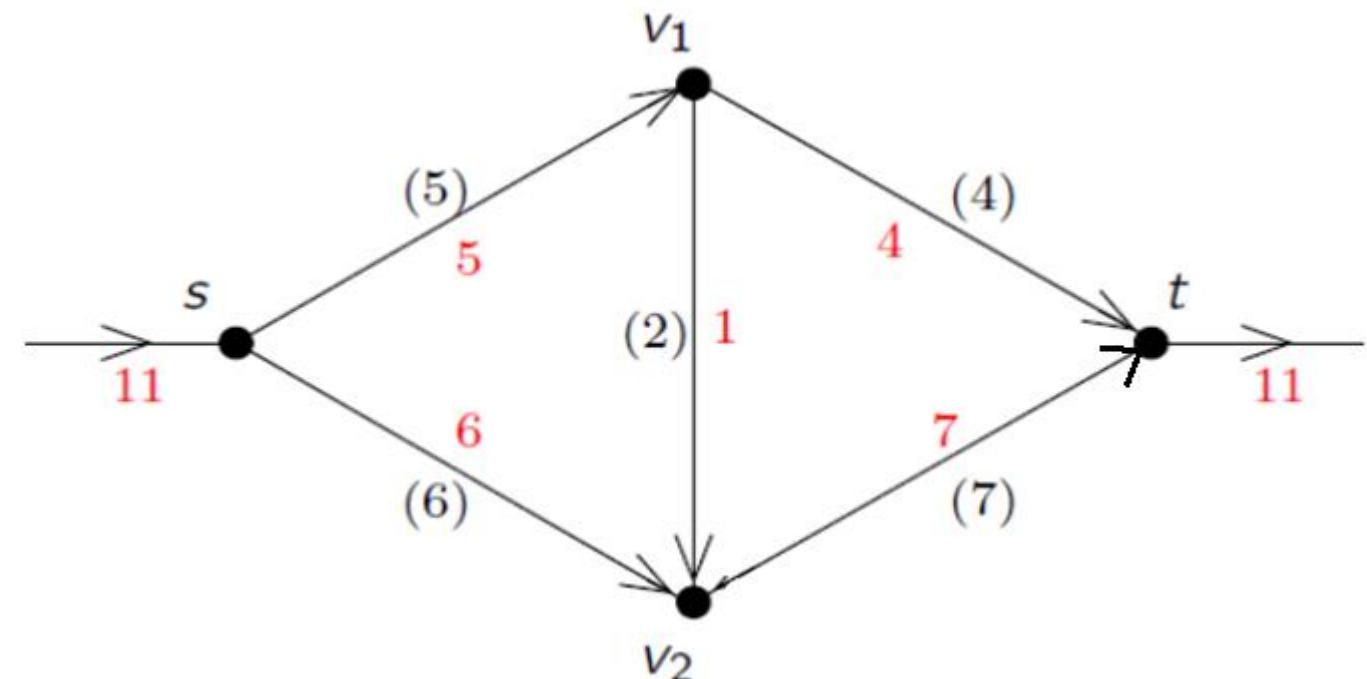
Soit un réseau $R = (G = (V, E), s, t, c)$.

Le problème du flot maximum consiste à déterminer un **flot réalisable** entre s et t qui soit de valeur maximum.



Le problème du flot maximum: Exemple■ Exemple :

On remarque que le flot donné dans le réseau précédent n'est pas maximum. En effet, on peut trouver un flot de valeur 11.



Ce nouveau flot est maximum.

En effet, on remarque qu'au mieux il peut rentrer 11 unités de flot dans t à cause des capacités 4 et 7 sur les arcs entrants.

□ Le problème du flot maximum: *Solution*

- En 1956, deux mathématiciens américains Lester Randolph Ford Jr et Delbert Ray Fulkerson ont proposé un algorithme, qui permet d'écouler un graphe depuis le sommet entrée S vers le sommet sortie P tout en déterminant le flot maximum .
- L'algorithme de **Ford - Fulkerson** est le premier à avoir été spécialement conçu pour la résolution du problème de flot maximum.

□ Le problème du flot maximum: *Solution*

- Avant de présenter l'algorithme en lui-même nous allons introduire la notion de **graphe d'écart**.
 - Il s'agit d'un graphe associé à un flot compatible d'un réseau de transport.
 - Il possède les mêmes sommets que le réseau mais ses arcs vont traduire les **augmentations possibles** du flot afin de le transformer en un flot maximal.
- **Définition**

Soit $G=(V,E)$ un réseau de transport dont on note c les capacités des arcs. On considère un flot compatible f circulant dans G .

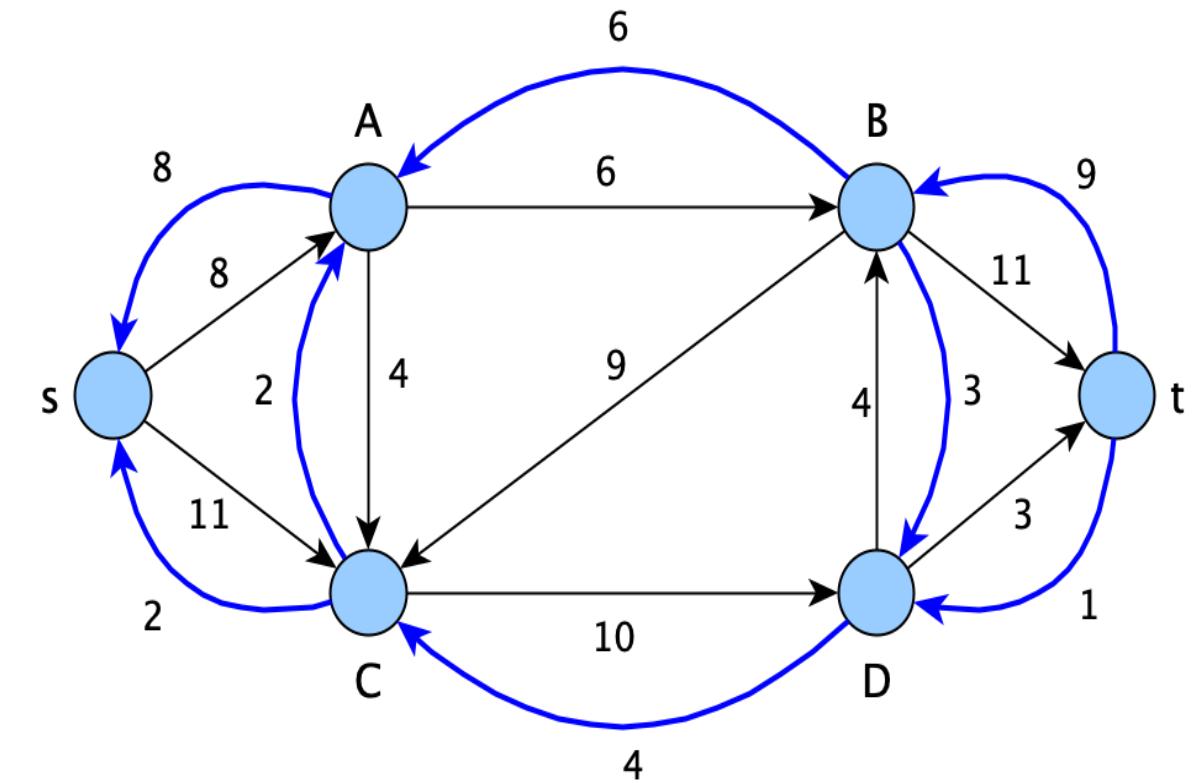
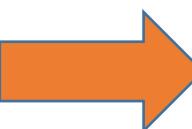
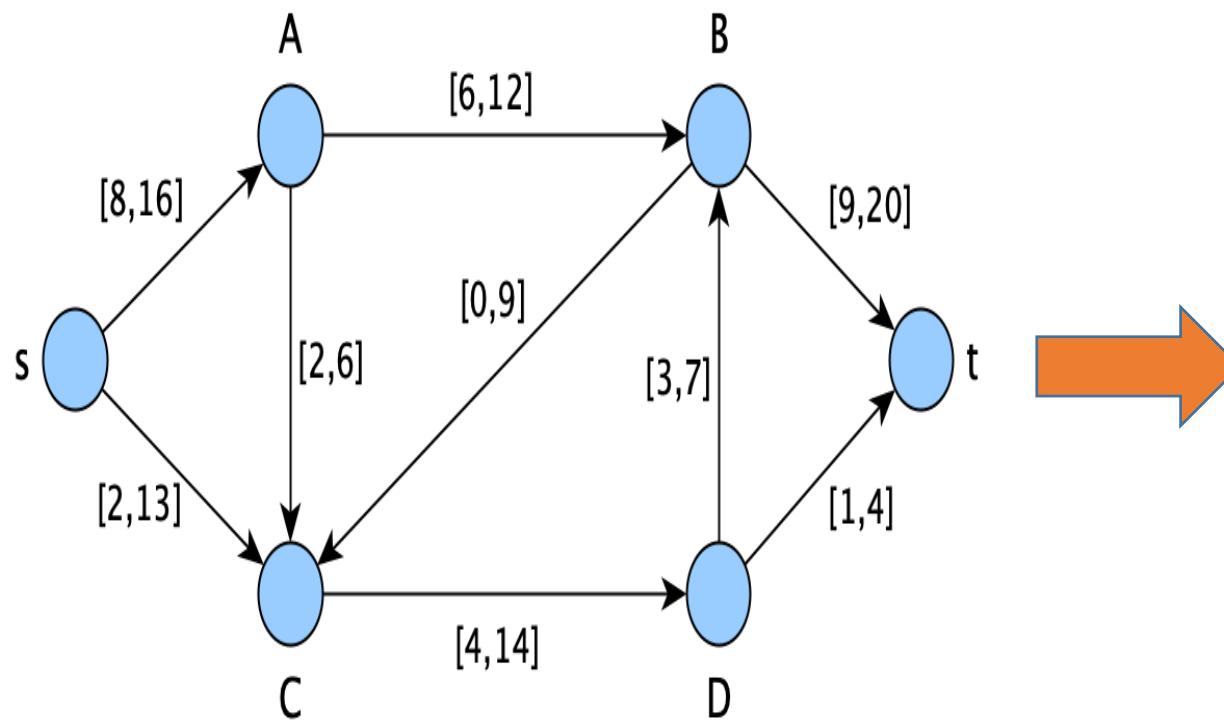
Le **graphe d'écart** de ce flot f dans G est un graphe possédant les mêmes sommets que G , et dont les arcs vont dépendre de ceux de G ainsi que de la valeur du flot y circulant. Plus précisément **pour un arc** (u,v) de E , on va construire un ou deux arcs dans le graphe d'écart selon la règle :

- Si $f(u,v) < c(u,v)$ on définit un arc (u,v) de valuation $c(u,v)-f(u,v)$.
- Si $f(u,v) > 0$ on définit un arc (v,u) de valuation $f(u,v)$.

□ Le problème du flot maximum: *Solution*

- Exemple.

Voici un réseau de transport et un flot compatible y circulant :



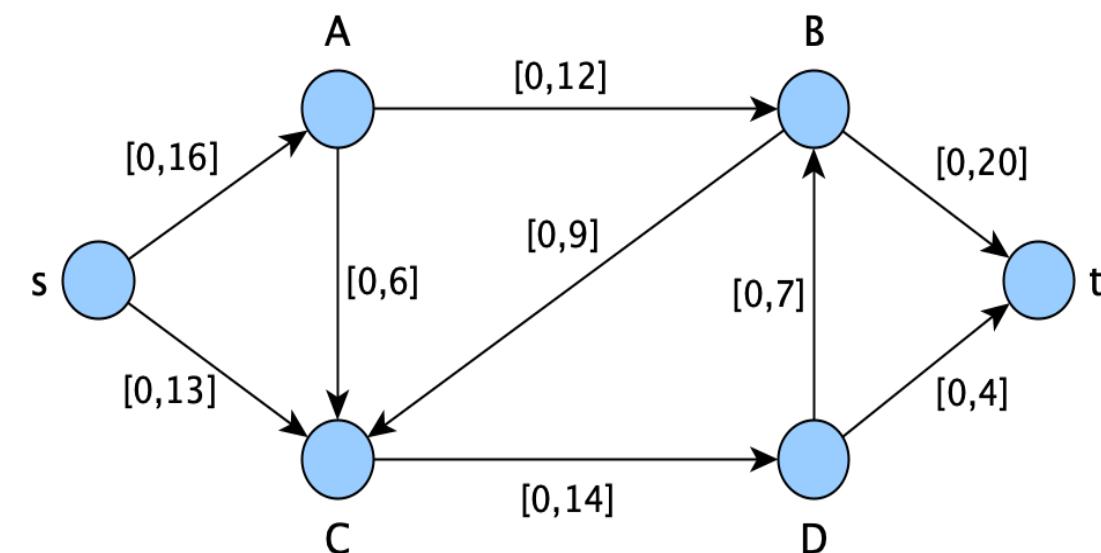
□ Le problème du flot maximum: *Solution*

- L'algorithme de Ford et Fulkerson permet donc de déterminer un flot maximal dans un réseau de transport.
 1. Partir d'un flot compatible, par exemple le flot nul.
 2. Construire le graphe d'écart correspondant.
 3. Rechercher un chemin allant de la source vers le puits dans le graphe d'écart.
 4. Il y a alors deux possibilités :
 - a. S'il existe un tel chemin, augmenter le flot circulant sur ce chemin dans le réseau du minimum des valuations des arcs du chemin dans le graphe d'écart. Recommencer alors à l'étape 2.
 - b. Sinon, l'algorithme est terminé et le flot courant est maximal.

□ Le problème du flot maximum: *Solution*

- Cet algorithme est tout à fait naturel, le graphe d'écart indiquant les possibilités restantes d'amélioration du flot. C'est donc lui qui permet d'affirmer si le flot courant est maximal ou non, et c'est lui qui le cas échéant nous permet de l'améliorer.
- **Exemple : Application de l'algorithme de Ford et Fulkerson**

Considérons le réseau de transport G dont la représentation sagittale est la suivante :



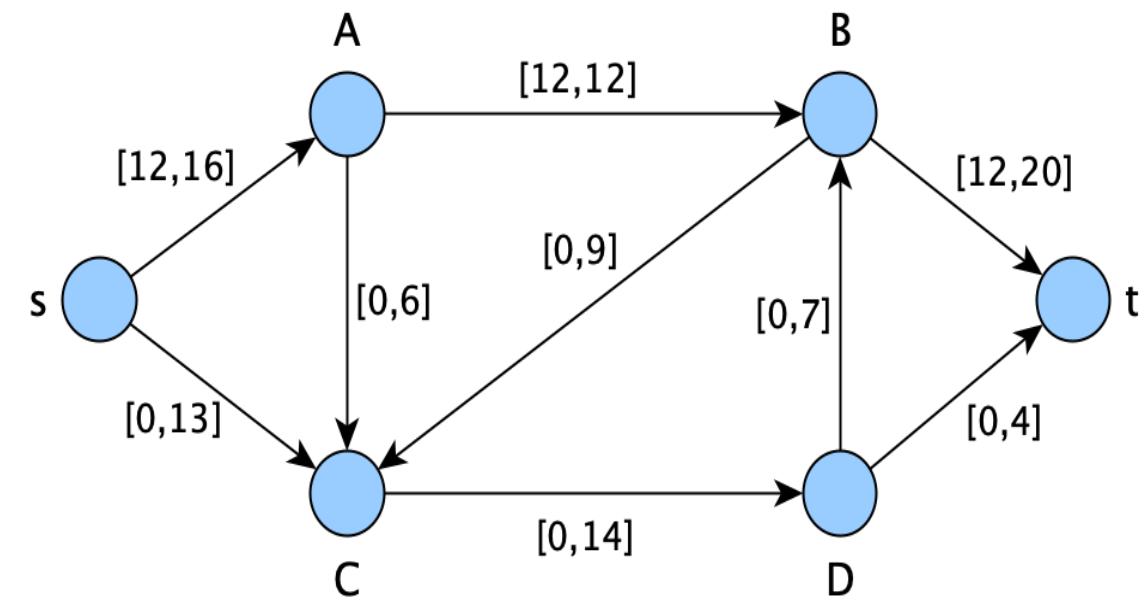
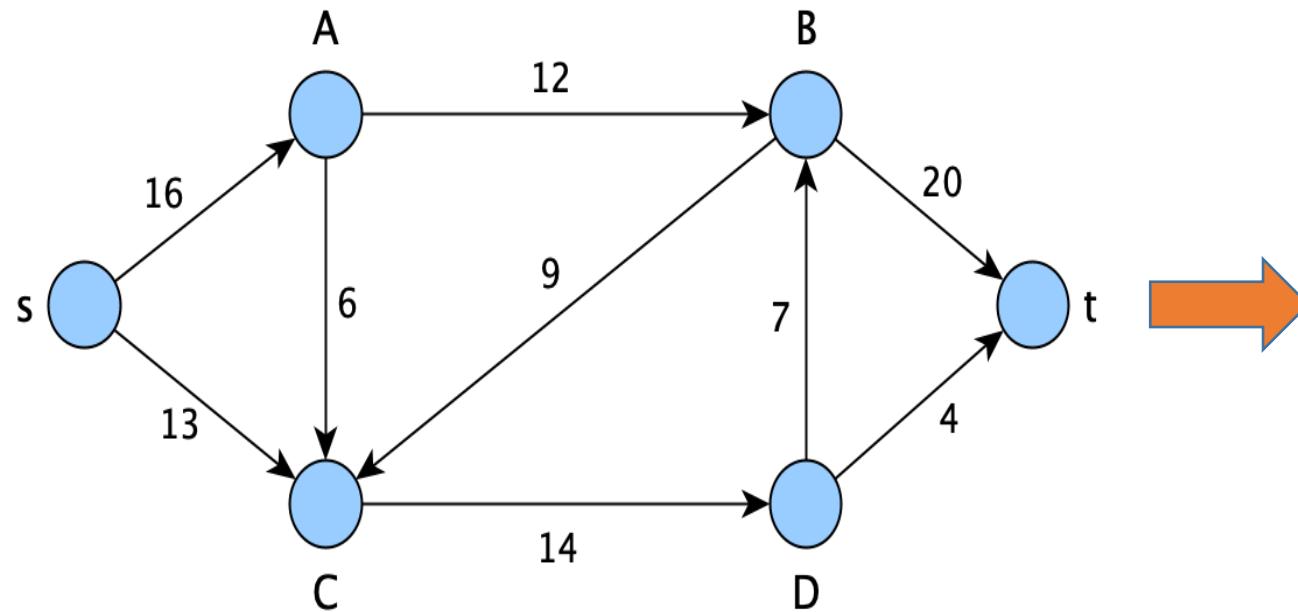
□ Le problème du flot maximum: *Solution*

- Exemple . Application de l'algorithme de Ford et Fulkerson

Nous avons pour l'instant un flot nul, dont le graphe d'écart correspond est bien sûr :



Considérons le chemin (s, A, B, t). La valuation minimale de ses arcs est égale à 12, on va donc augmenter le flot de cette valeur.



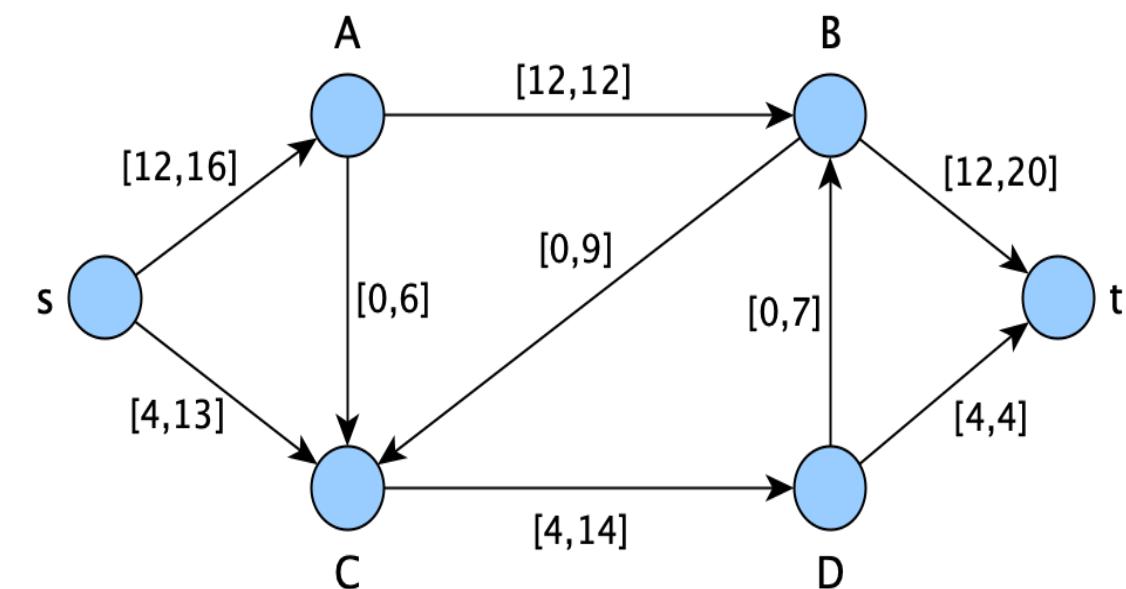
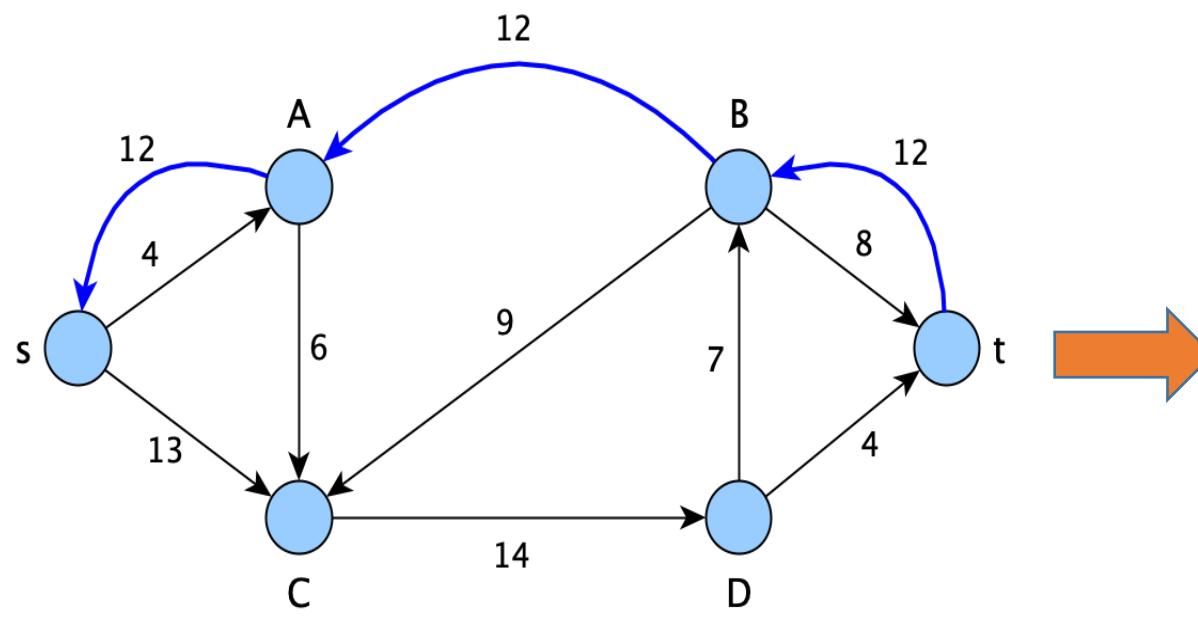
□ Le problème du flot maximum: *Solution*

- Exemple . Application de l'algorithme de Ford et Fulkerson

Mettons à jour le graphe d'écart :



Considérons maintenant le chemin (s, C, D, t). La valuation minimale de ses arcs est égale à 4, on va donc augmenter le flot de cette valeur.



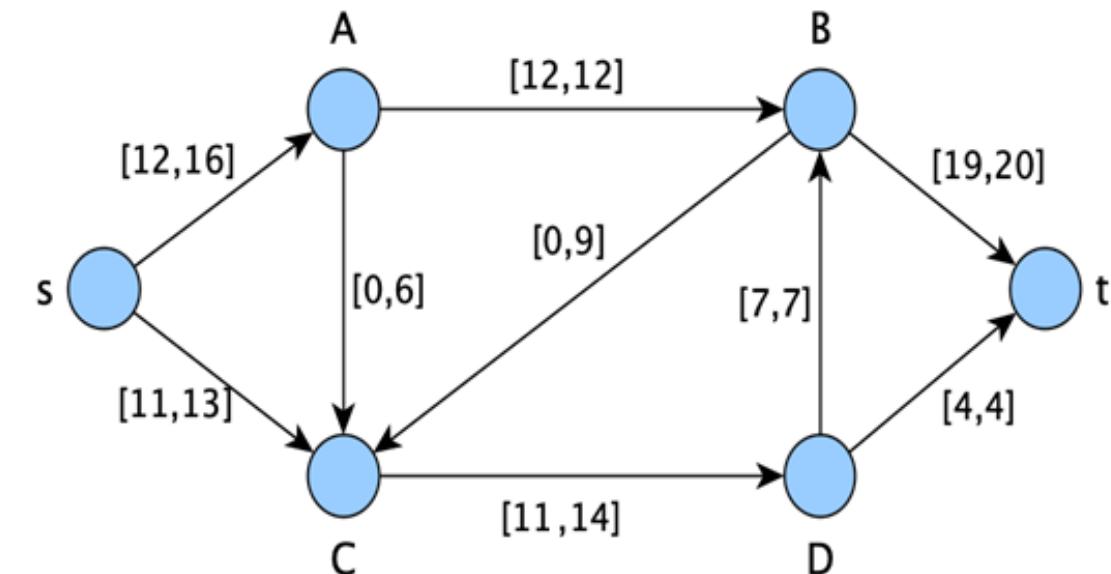
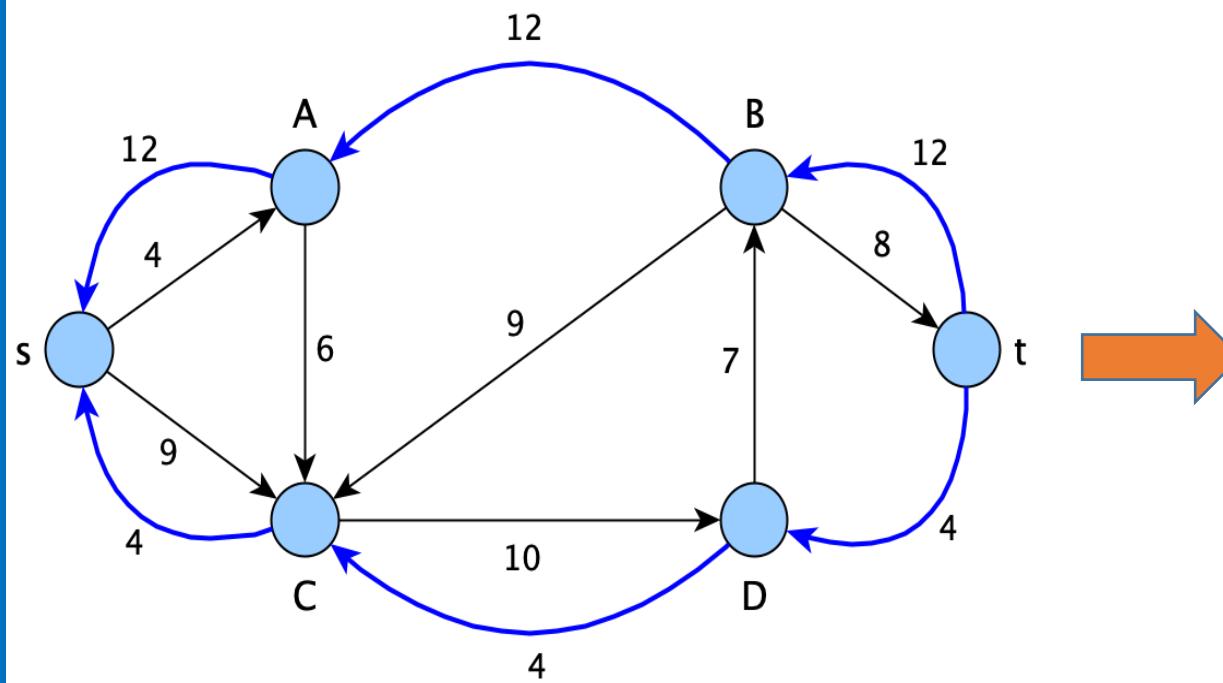
□ Le problème du flot maximum: *Solution*

- Exemple . Application de l'algorithme de Ford et Fulkerson

Mettons à jour le graphe d'écart :



Considérons maintenant le chemin (s, C, D, B, t). La valuation minimale de ses arcs est égale à 7, on va donc augmenter le flot de cette valeur



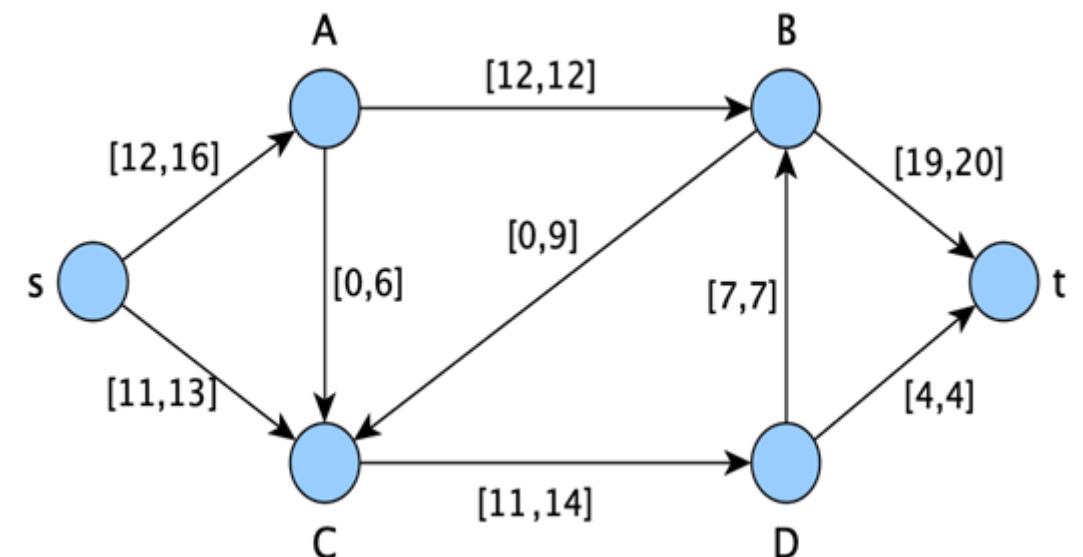
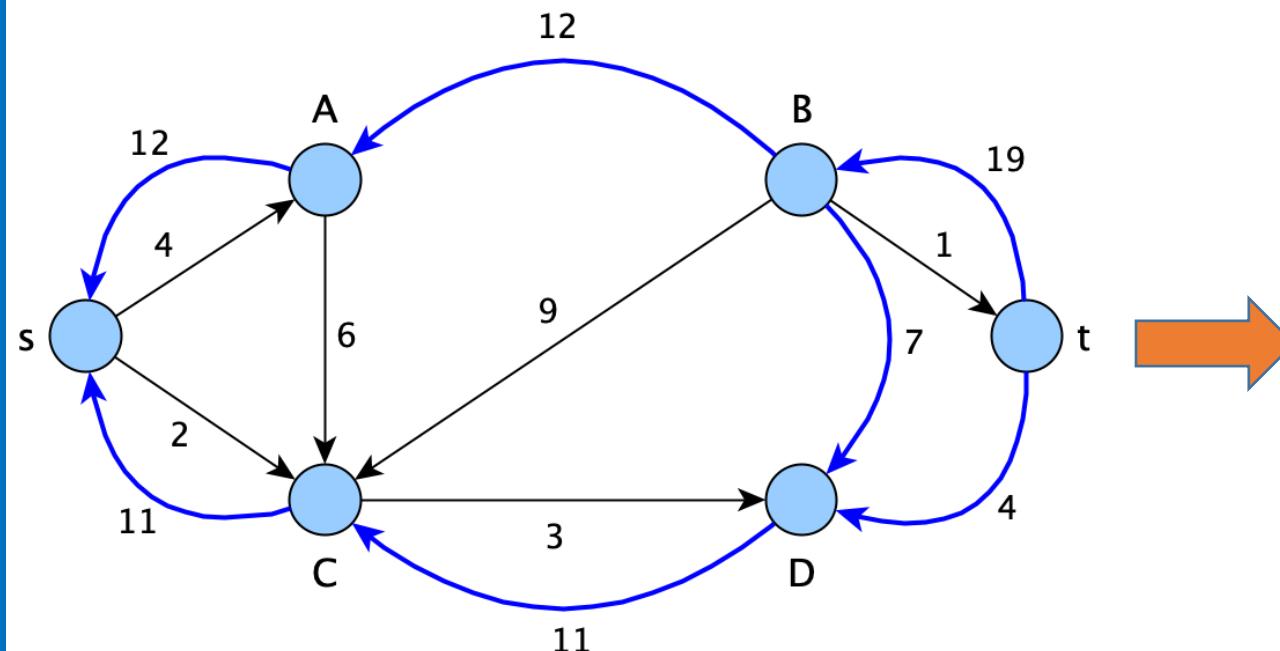
□ Le problème du flot maximum: *Solution*

- Exemple . Application de l'algorithme de Ford et Fulkerson

Mettons à jour le graphe d'écart :



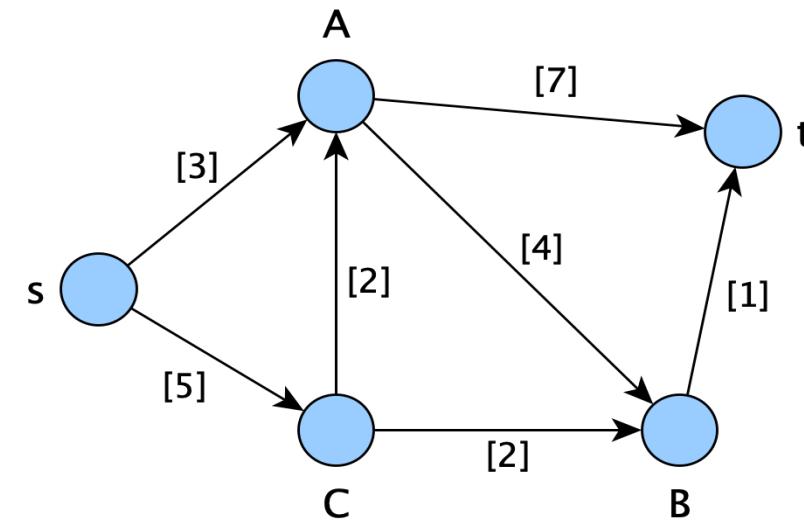
Il n'y a plus de chemins allant de la source vers le puits dans le graphe d'écart, le flot courant est donc maximal :



La valeur du flot maximal est ainsi de 23

Exercice 31

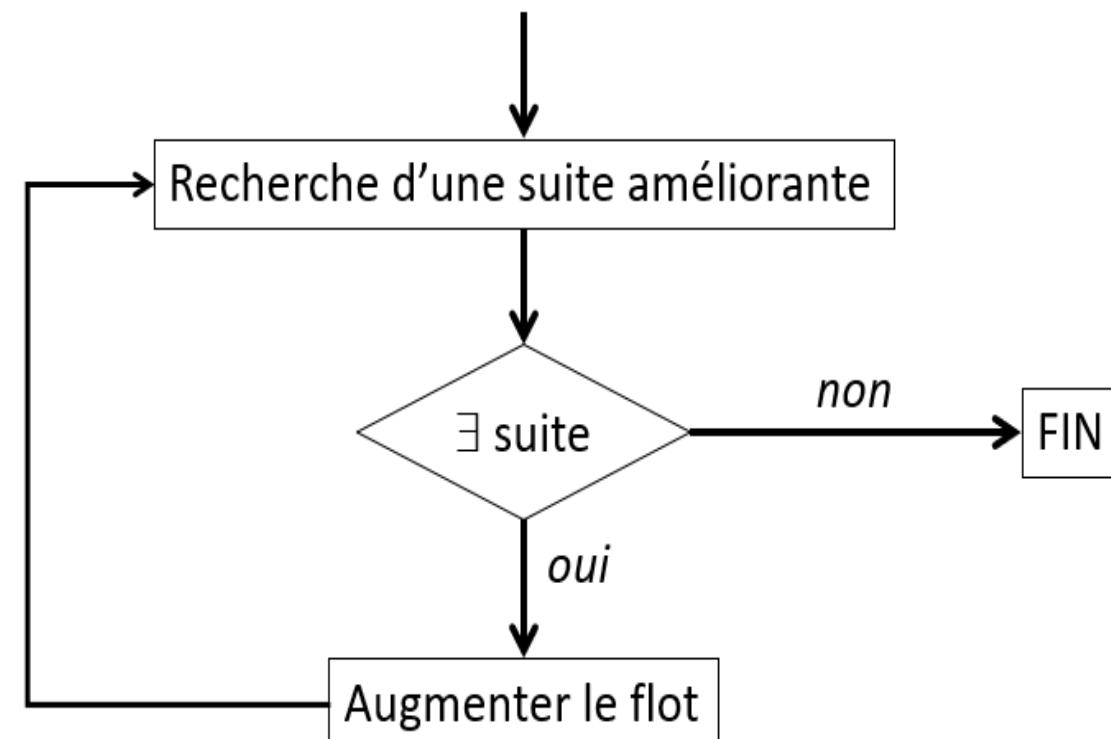
Considérons le réseau de transport donc la représentation sagittale est la suivante :



Appliquer l'algorithme de Ford et Fulkerson à ce graphe.

□ Le problème du flot maximum: *Solution*

L'algorithme de Ford-Fulkerson permet aussi de calculer un flot à valeur maximale sur un graphe de flot en se basant sur l'augmentation de la valeur de flot au moyen de suites améliorantes. On peut le schématiser de la façon suivante :



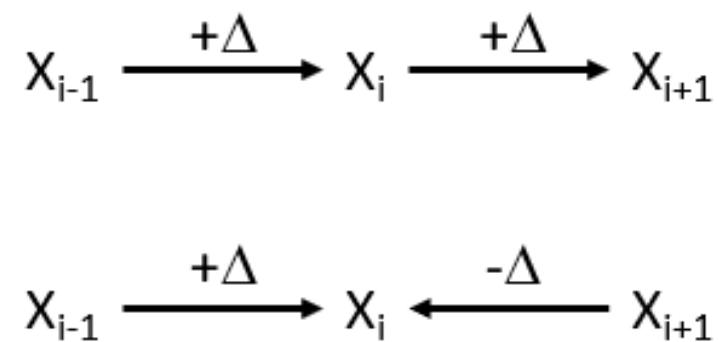
□ Le problème du flot maximum: *Solution*

- Chaine Améliorante

Dans un problème de flot conservatif, une **chaîne améliorante** est une chaîne, qui commence par un arc qui a pour origine **la source s**, qui se termine par un arc qui a pour extrémité le puits **t**, et telle que pour tout (x, y) arête de la chaîne:

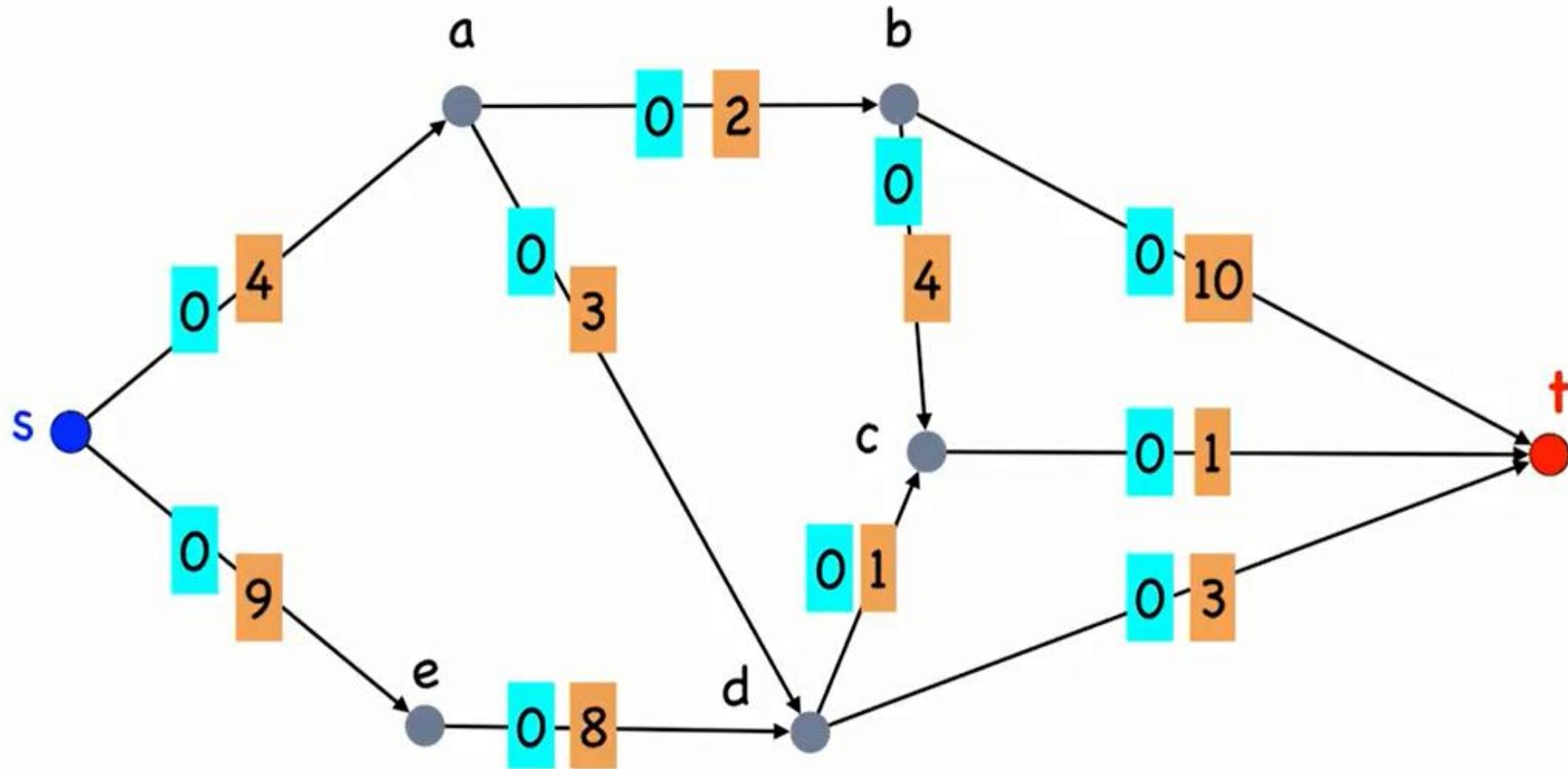
- si l'arc correspondant va de x à y alors $F(x,y) < C(x,y)$, c'est un ***arc avant non saturé***,
- si l'arc correspondant va de y à x alors $F(y,x) > 0$, c'est un ***arc arrière à flux strictement positif***.

Selon le sens des deux arcs, la loi de Kirshoff détermine les ***augmentations*** ou ***diminutions*** possibles. Si l'on suppose une augmentation ou diminution du premier arc de la suite, l'action à effectuer sur le second se déduit immédiatement



Le problème du flot maximum: Algorithme de Ford -Fulkerson

Flot initial nul (valeur 0)

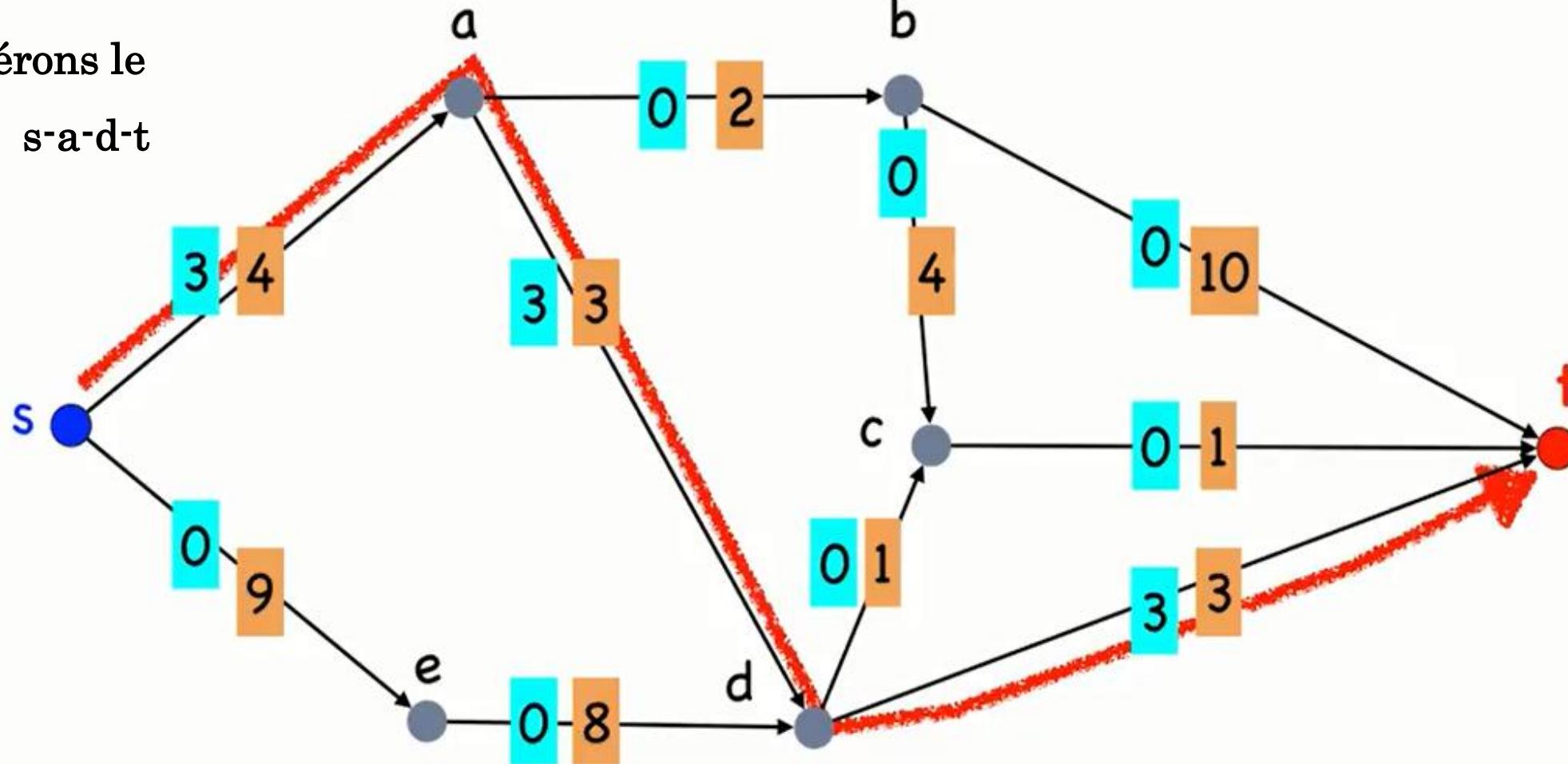


Le problème du flot maximum: Algorithme de Ford -Fulkerson

Flot initial nul (valeur 0)

Flot 1 (valeur 3)

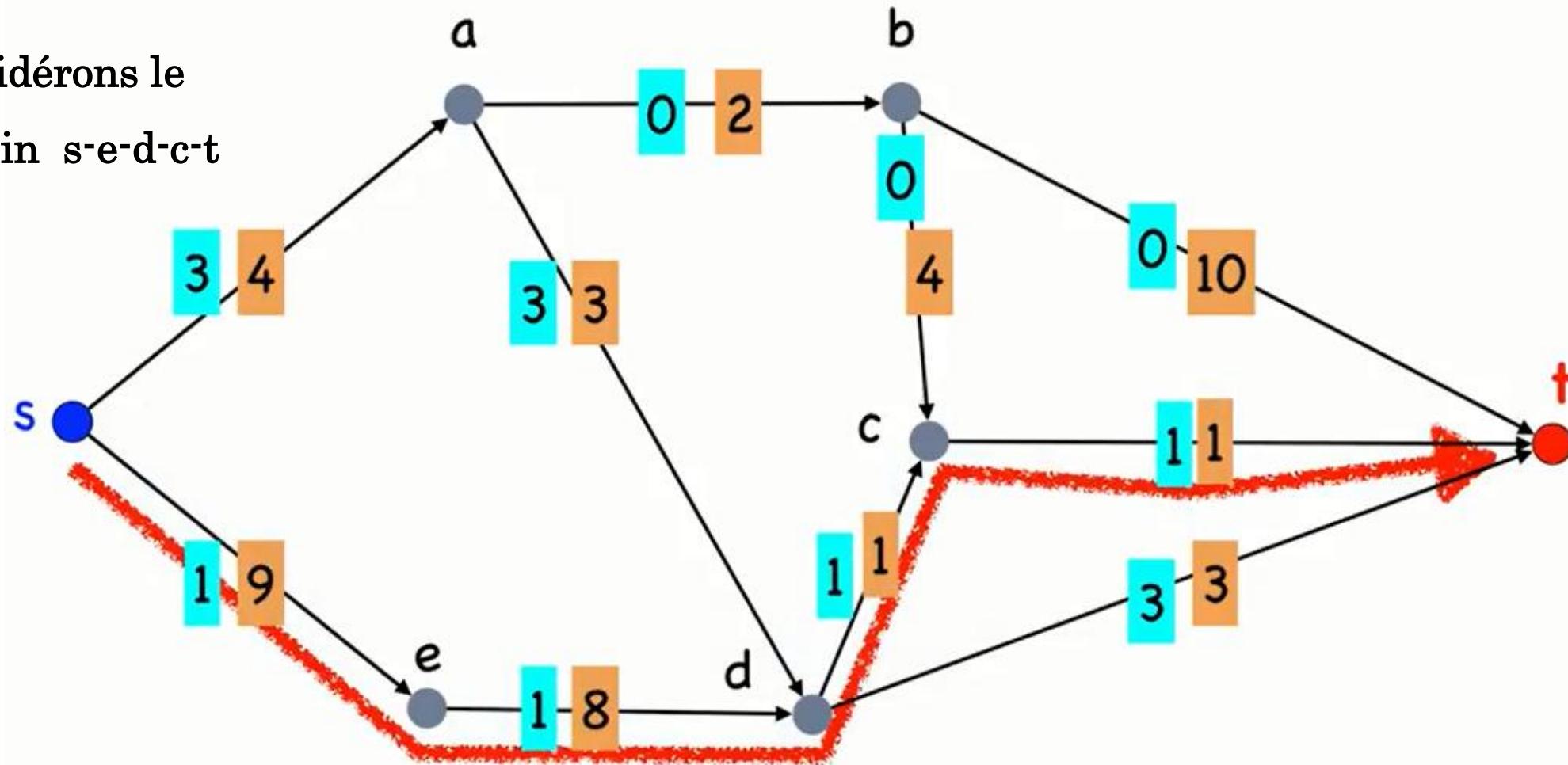
Considérons le
chemin s-a-d-t



Le problème du flot maximum: Algorithme de Ford -Fulkerson

Flot initial nul (valeur 0) \rightarrow Flot 1 (valeur 3) \rightarrow Flot 2 (valeur 4)

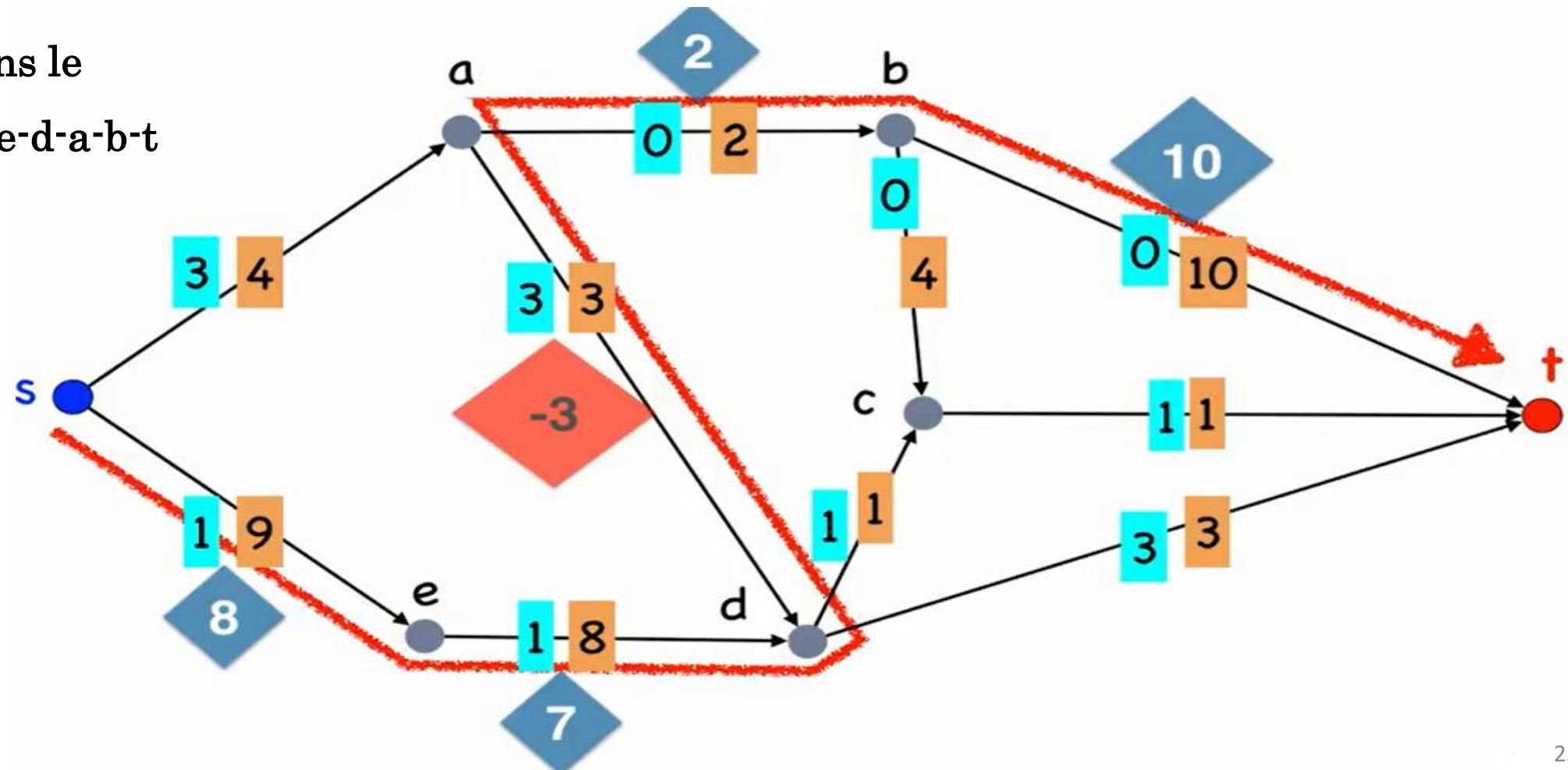
Considérons le
chemin s-e-d-c-t



Le problème du flot maximum: Algorithme de Ford -Fulkerson

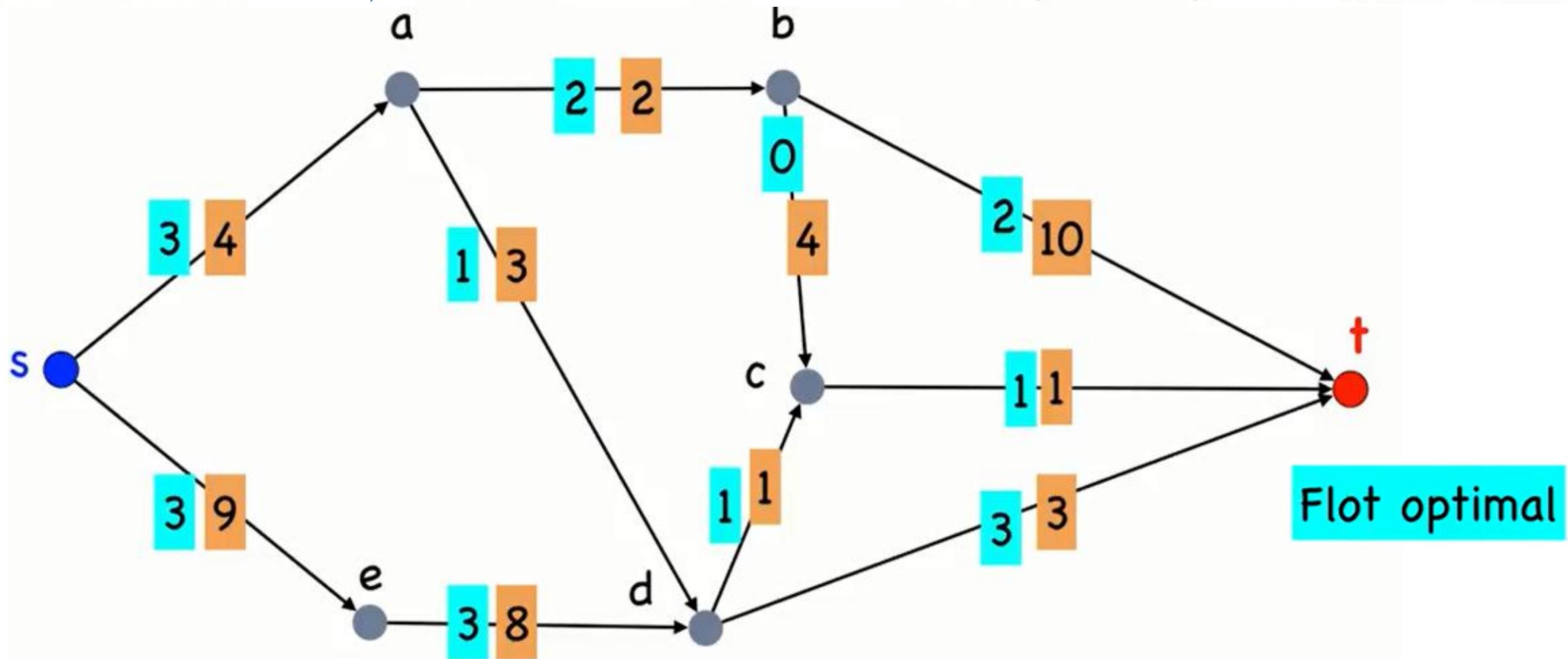
Flot initial nul (valeur 0) \rightarrow Flot 1 (valeur 3) \rightarrow Flot 2 (valeur 4)

Considérons le
chemin s-e-d-a-b-t



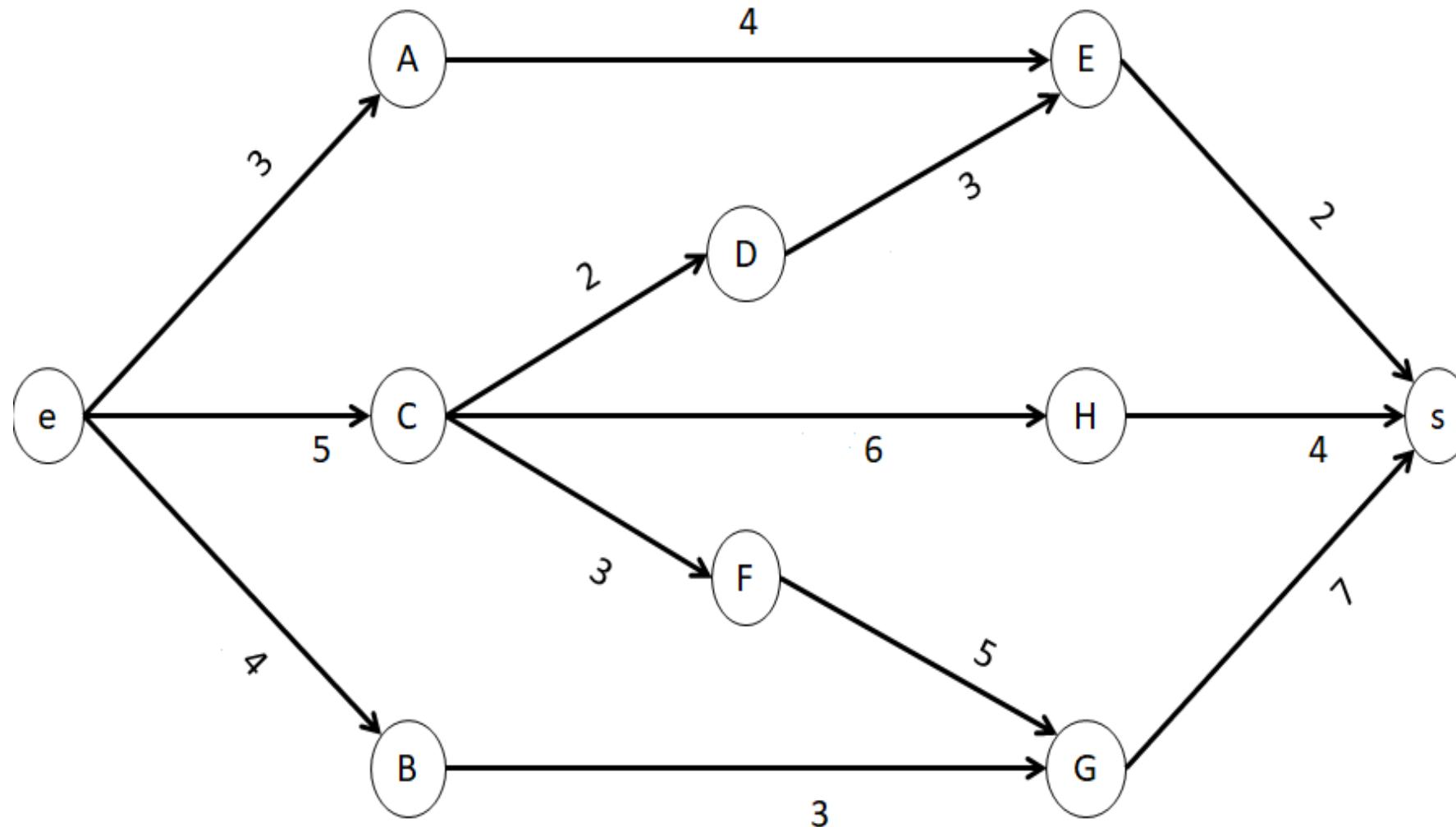
Le problème du flot maximum: Algorithme de Ford -Fulkerson

Flot initial nul (valeur 0) \rightarrow Flot 1 (valeur 3) \rightarrow Flot 2 (valeur 4) \rightarrow Flot 3 (valeur 6)



Exercice 32

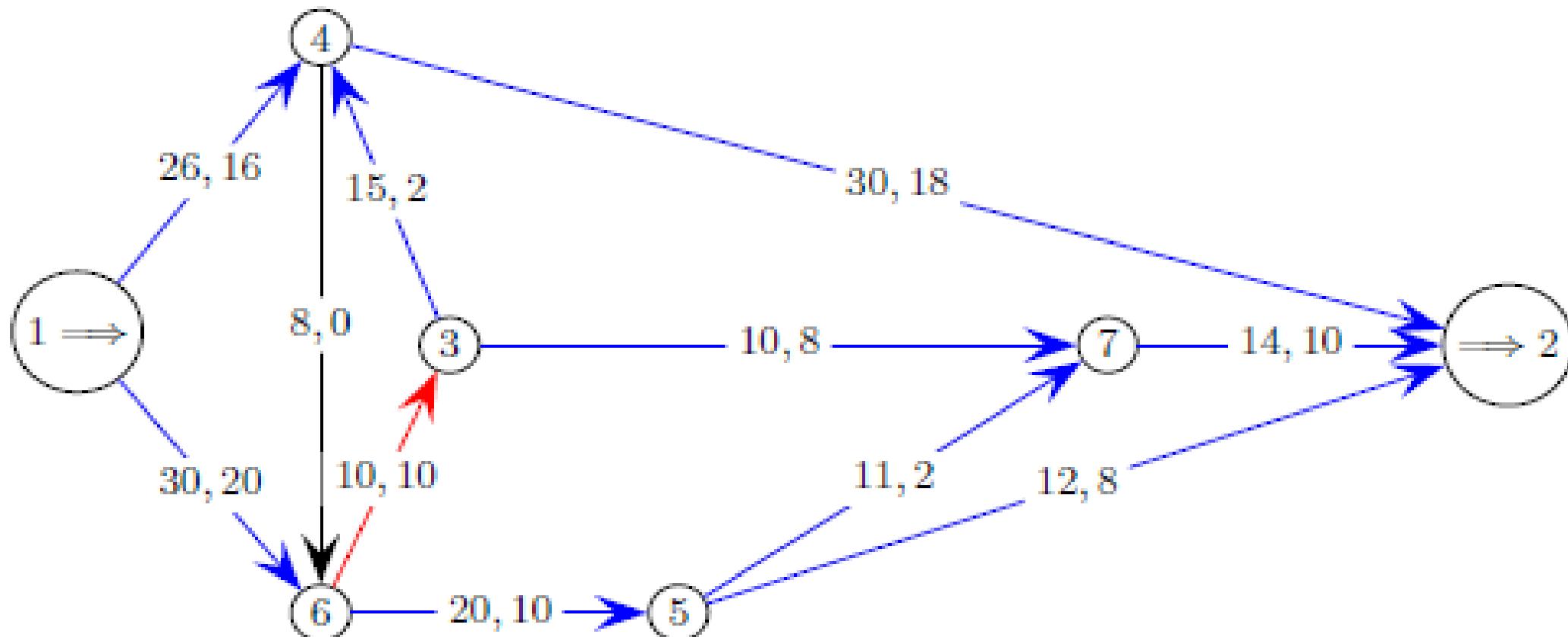
Calculer le flot maximum pour ce graphe en suivant l'algorithme de Ford-Flukerson



Exercice 33

Calculer le flot maximum pour ce graphe en suivant l'algorithme de Ford-Fulkerson

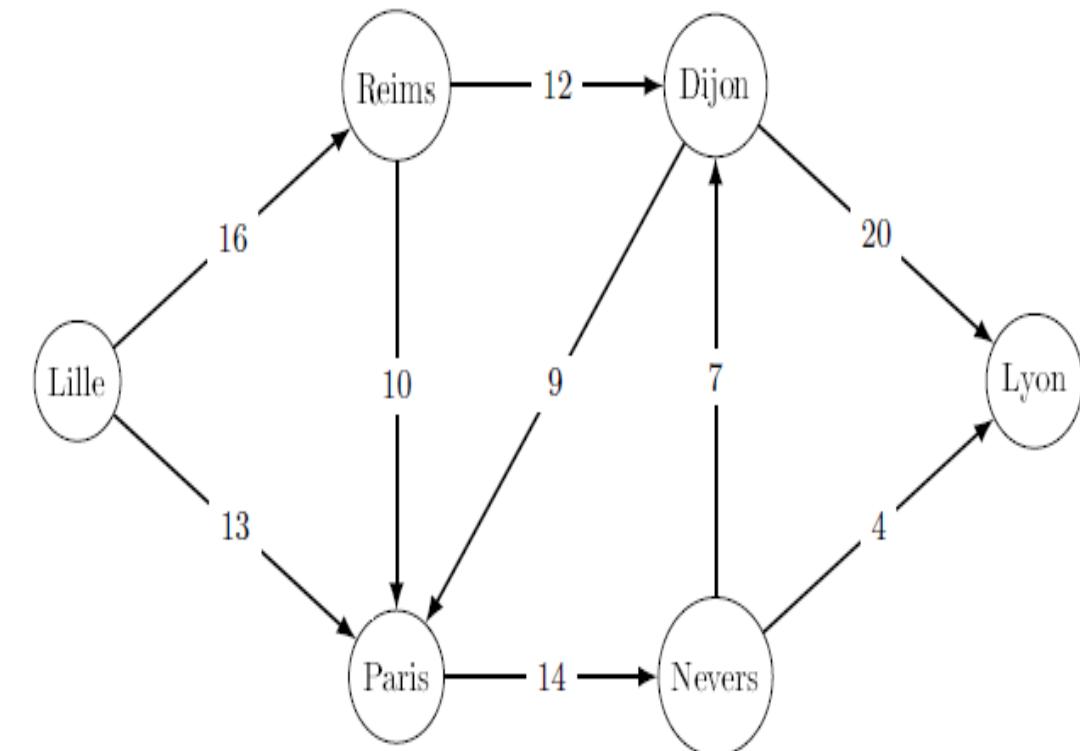
- 1) En supposant en premier lieu le flot de départ est null.
- 2) Prenons en considération les valeurs des flots affichées sur le graphe.



Exercice 34

L'usine « France Land », localisée à Lille, produit des voitures. Ces voitures sont acheminées en train jusqu'à Lyon, où elles sont stockées dans un entrepôt puis vendues. Les capacités des trains sont :

- sur la ligne Lille/Reims : 16 voitures par jour,
- sur la ligne Lille/Paris : 13 voitures par jour,
- sur la ligne Reims/Paris : 10 voitures par jour,
- sur la ligne Reims/Dijon : 12 voitures par jour,
- sur la ligne Paris/Nevers : 14 voitures par jour,
- sur la ligne Dijon/Paris : 9 voitures par jour,
- sur la ligne Nevers/Dijon : 7 voitures par jour,
- sur la ligne Nevers/Lyon : 4 voitures par jour,
- sur la ligne Dijon/Lyon : 20 voitures par jour.



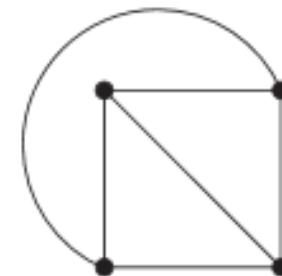
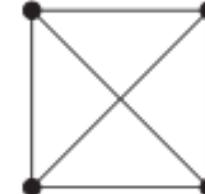
Ce réseau de transport sera modélisé par le graphe suivant :

Question: Calculer pour ce graphe son flot maximum

Fin

□ Graphe Planaire

Définition: Un graphe est planaire s'il peut être tracé dans un plan sans qu'aucune de ses arêtes en croise une autre. Un tel tracé est appelé une représentation planaire du graphe.



Définition: Soit G un graphe planaire. Une face F de G est une région maximale du plan délimité par un ensemble d'arêtes de G , et qui n'en contient aucune.

Le degré de F , noté $\deg(F)$, est le nombre d'arêtes de G qui bordent F .

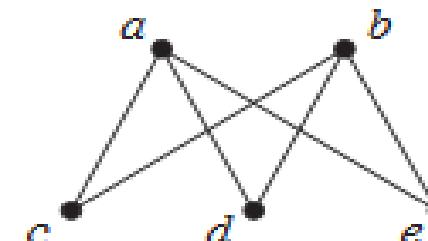
Soit G un graphe planaire et a le nombre d'arêtes de G . Alors

$$\sum_{\text{faces } F} \deg(F) = 2a$$

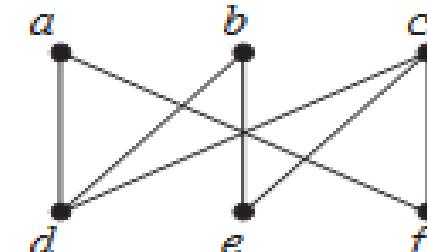
□ Graphe Planaire

▪ Exercice

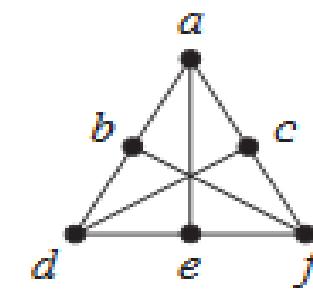
Déterminer si les graphes suivants sont des graphes planaires. Si oui, donner leur représentation planaire, déterminer le nombre de faces et le degré de chaque face ainsi que la somme des degrés de toutes les faces.



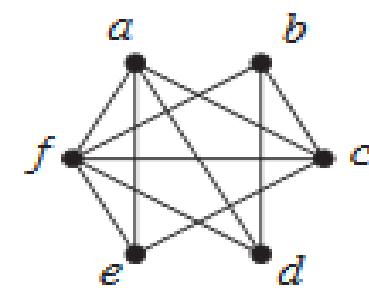
a)



b)



c)



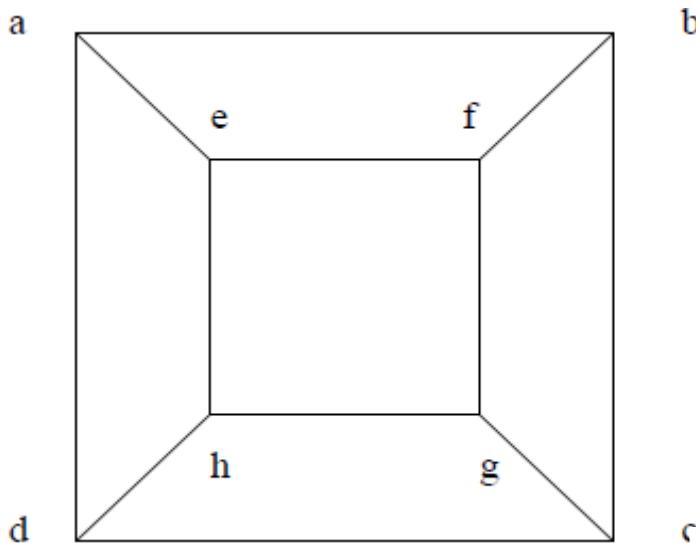
d)

□ Graphe Biparti

Un graphe est biparti si ses sommets peuvent être divisés en deux ensembles X et Y, de sorte que toutes les arêtes du graphe relient un sommet dans X à un sommet dans Y. Les arbres sont des exemples des graphes bipartis. Si G est biparti, il est habituellement noté par $G = (X, Y, E)$, où E est l'ensemble des arêtes

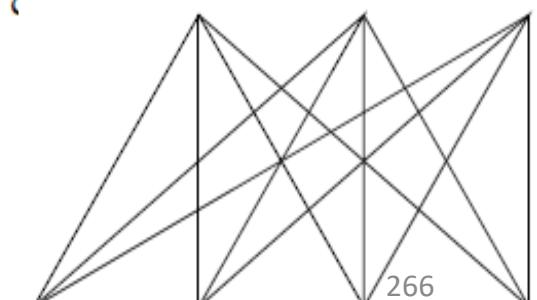
Exemple : Ce graphe est biparti.

les deux classes sont $\{a, c, f, h\}$ et $\{b, d, e, g\}$



G est biparti complet si G est biparti et si tout sommet d'une partie est adjacent à tout sommet de l'autre partie.

Le graphe biparti complet $K_{3,4}$



□ Algorithme Bellman-Ford- Exercice

- ▶ Trouver le plus long chemin entre le sommet α et tous les autres sommets
- ▶ Trouver le plus long chemin de A à F

