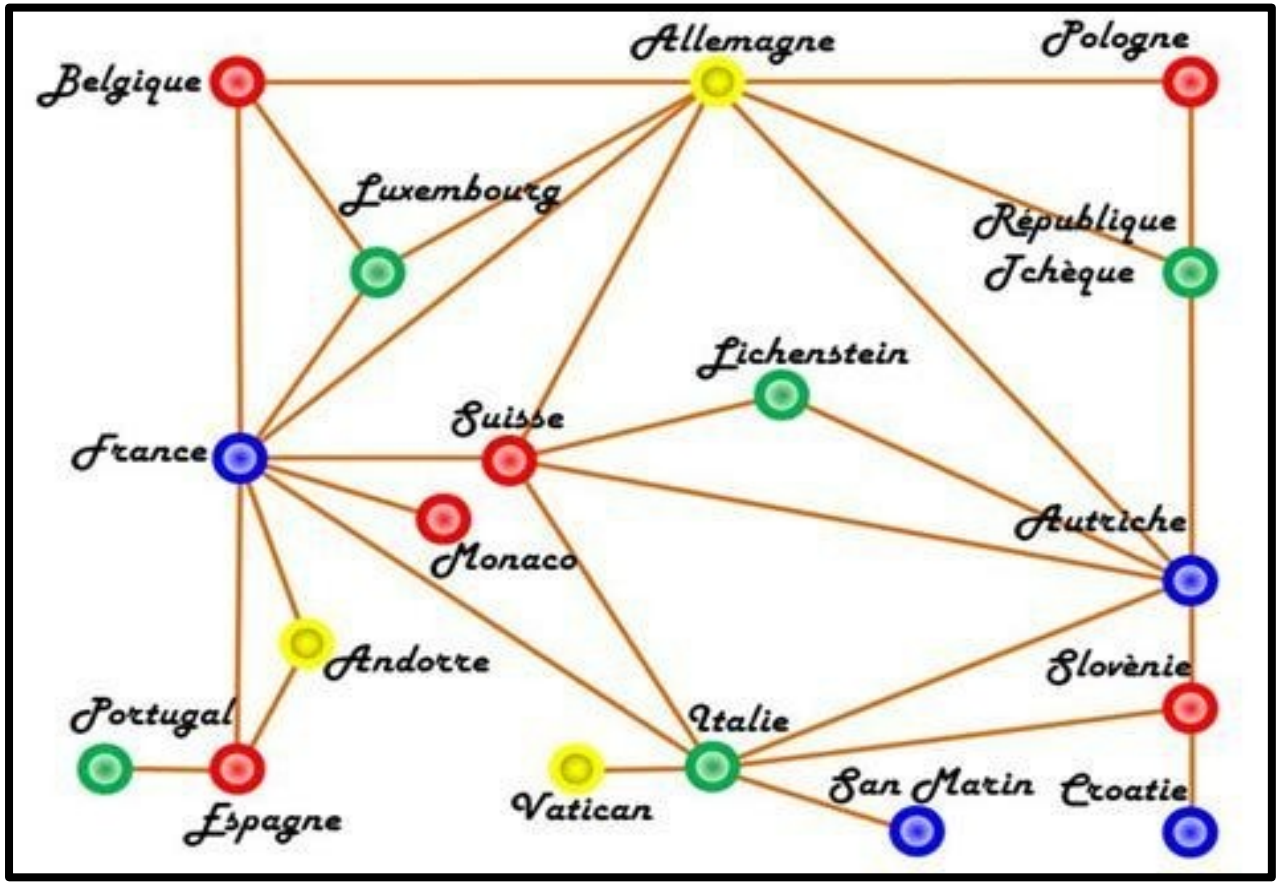
COMPTER les CHEMINS et PROBLÈMES de PARCOURS dans les GRAPHES

(14 novembre 2024)

K A F E M A T H



C J U P

(**C**haque **J**our **U**n **P**eu)

14/11/2024

… Il est bien plus beau de savoir quelque chose de tout que de savoir tout d'une chose ; cette universalité est la plus belle. (Pascal)

K A F E M A T H

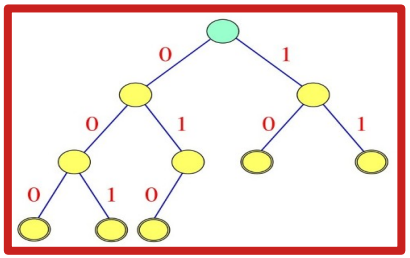
MAJ : 16/11/2024

COMPTER les CHEMINS et PROBLÈMES de PARCOURS dans les GRAPHES

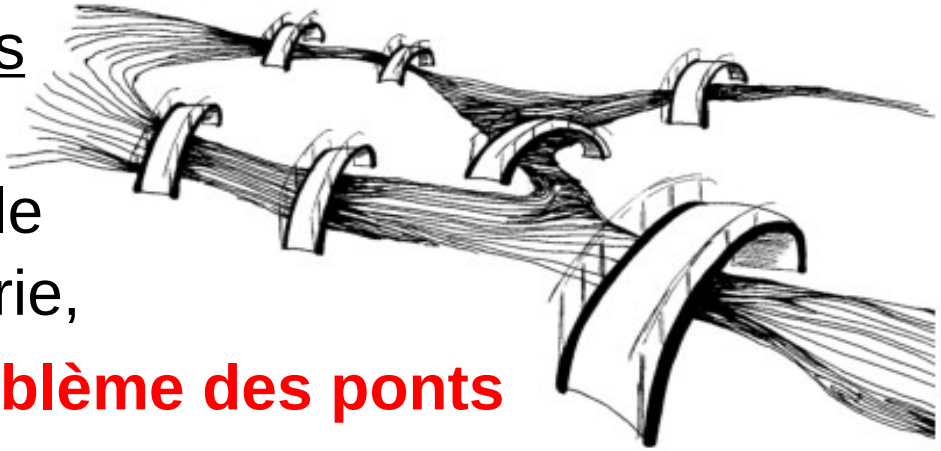
S O M M A I R E

1

Un GRAPHE, QUOI de PLUS SIMPLE ?



► La naissance de la théorie des graphes nous renvoie au siècle d'**Euler** (*) : c'est à lui qu'on doit le 1er résultat important de la théorie, solution du **problème des ponts de Königsberg**.



Leonhard EULER
(1707-1783)

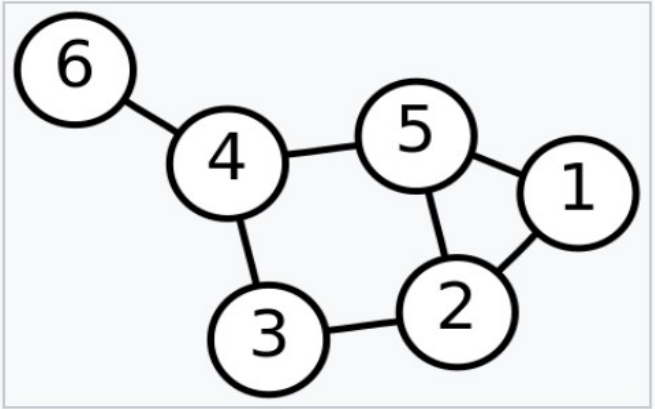
► L'énoncé du problème : Une ville est construite sur deux îles reliées au continent par **six** ponts, et entre elles par **un** pont. Trouver un chemin tel que, en partant d'un point quelconque, on puisse y revenir en ne passant qu'**une seule fois** sur chacun des ponts.

(*) Voir : https://fr.wikipedia.org/wiki/Leonhard_Euler et
https://fr.wikipedia.org/wiki/Probl%C3%A8me_des_sept_ponts_de_K%C3%B6nigsberg

1-1 QU'EST-CE QU'UN GRAPHE ?

Un graphe est un objet mathématique *bicéphale*, noté **G(S,A)** :

- **S** : ensemble de **sommets** ou **nœuds** ;
- **A** : ensemble d'**arêtes** ou **arcs** reliant ces sommets.



Remarque : on trouve aussi la notation **G(V,E)** qui renvoie aux mots anglais **vertex** et **edge**.

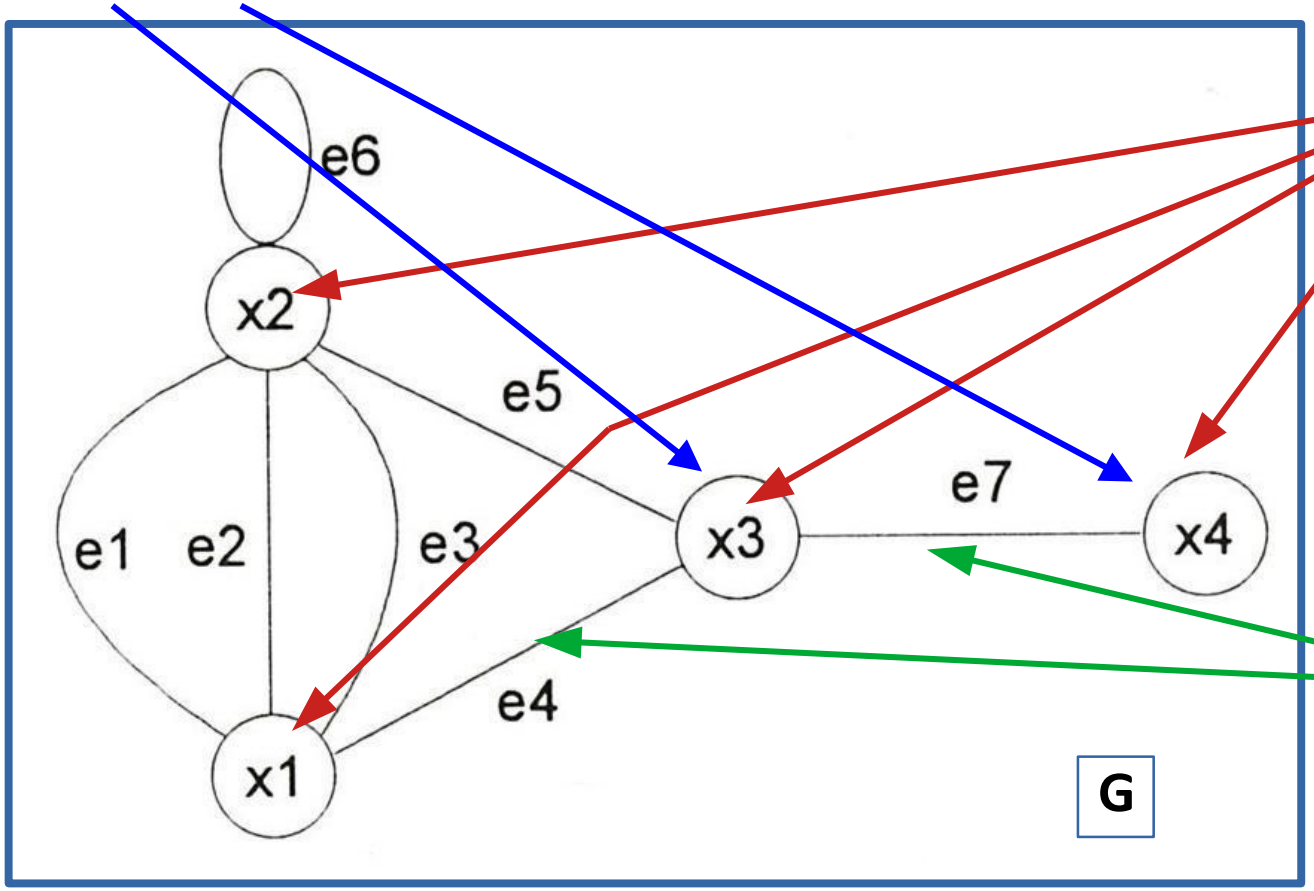
Un graphe est un *outil* qui permet de représenter les **relations** (connections) entre les *choses* …

Exemples : montrer comment des informations sont *liées*, comment des individus forment un *réseau social*, etc.

EXEMPLES de GRAPHES

Exemple 1

Dans le graphe **G**, les sommets x_3 et x_4 sont dits ***adjacents*** (ou ***voisins***).



sommets

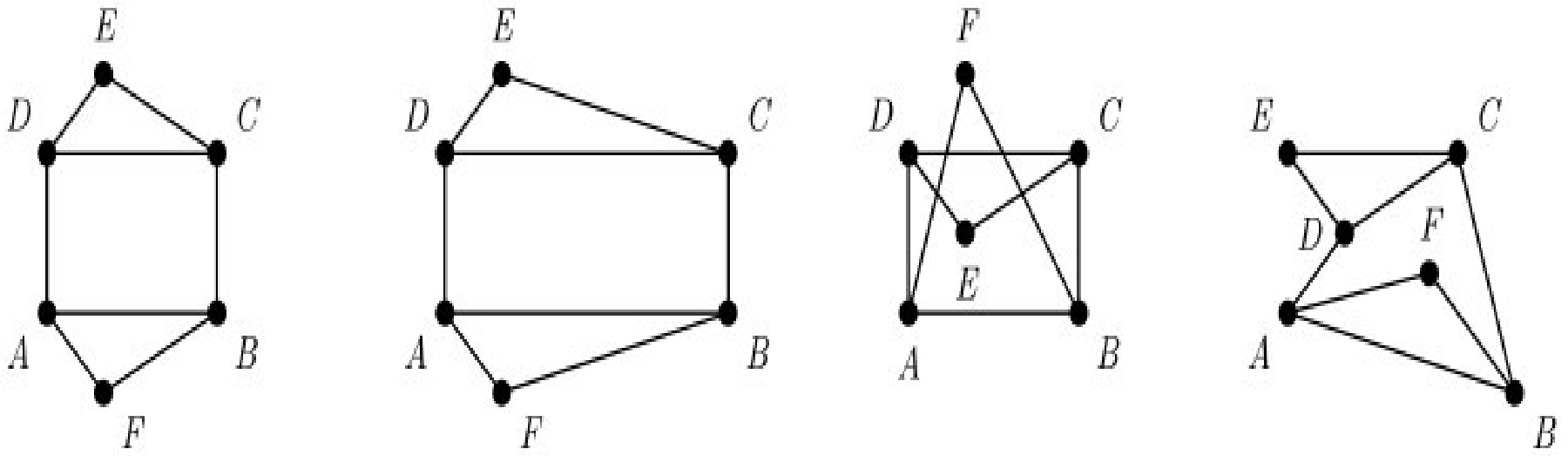
S=$\{x_1,x_2,x_3,x_4\}$

arêtes

A=$\{e_1,e_2,e_3,e_4,e_5,e_6,e_7\}$

Exemple 2

- Un *même* graphe peut être *dessiné* de plusieurs façons (graphes alors dits **isomorphes**) :



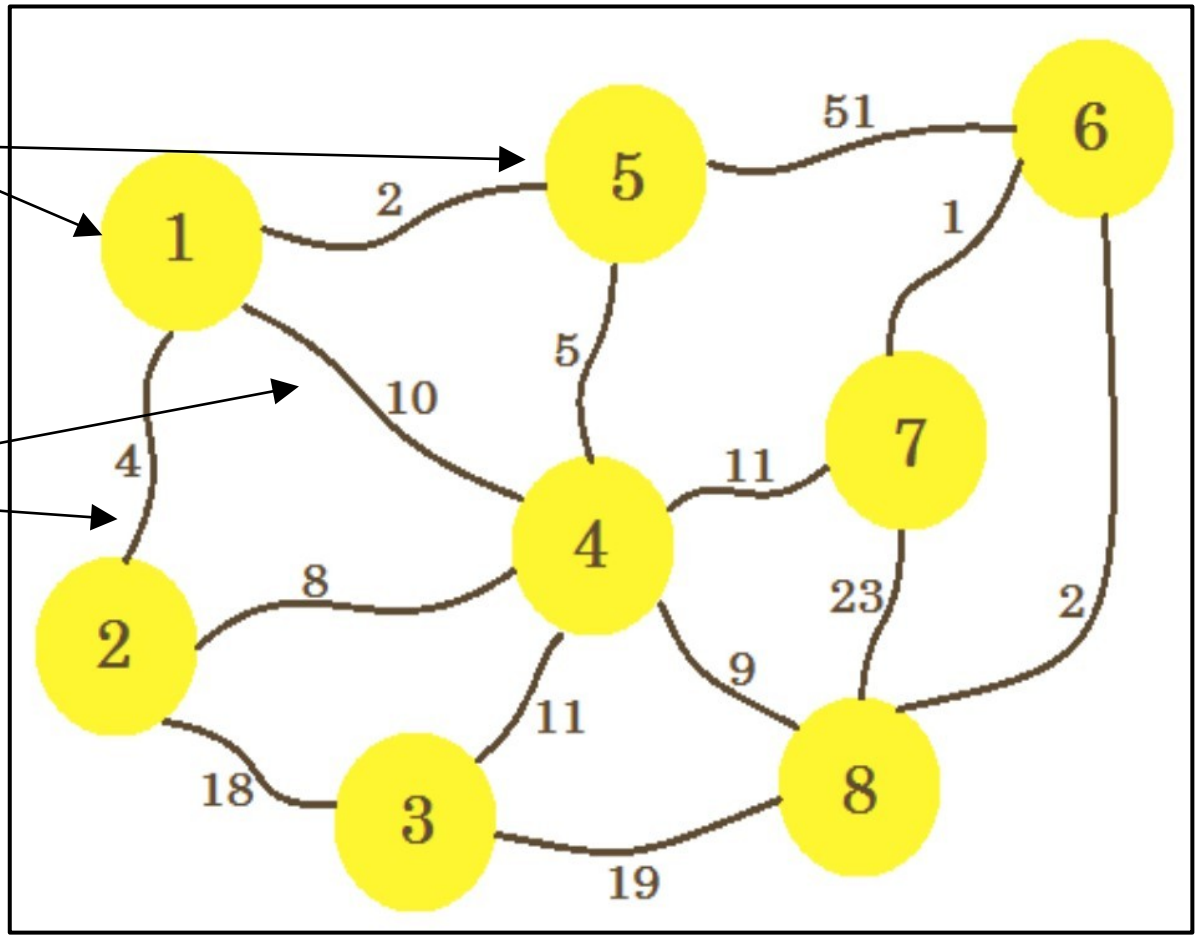
Un *même* graphe *représenté* de 4 façons

Exemple 3

Le graphe **G** ci-dessous est un graphe **valué** **non orienté** :

Les sommets sont **étiquetés**

Les arêtes sont **valuées**



G

Dans un graphe **valué**, chaque arête de **G** reçoit une « valeur » convenue, en général un *nombre* représentant une *mesure* dans l'espace métrique considéré.

1-3 Un GRAPHE PARTICULIER : l'ARBRE

Un graphe **A** est appelé **arbre** s'il est à la fois :

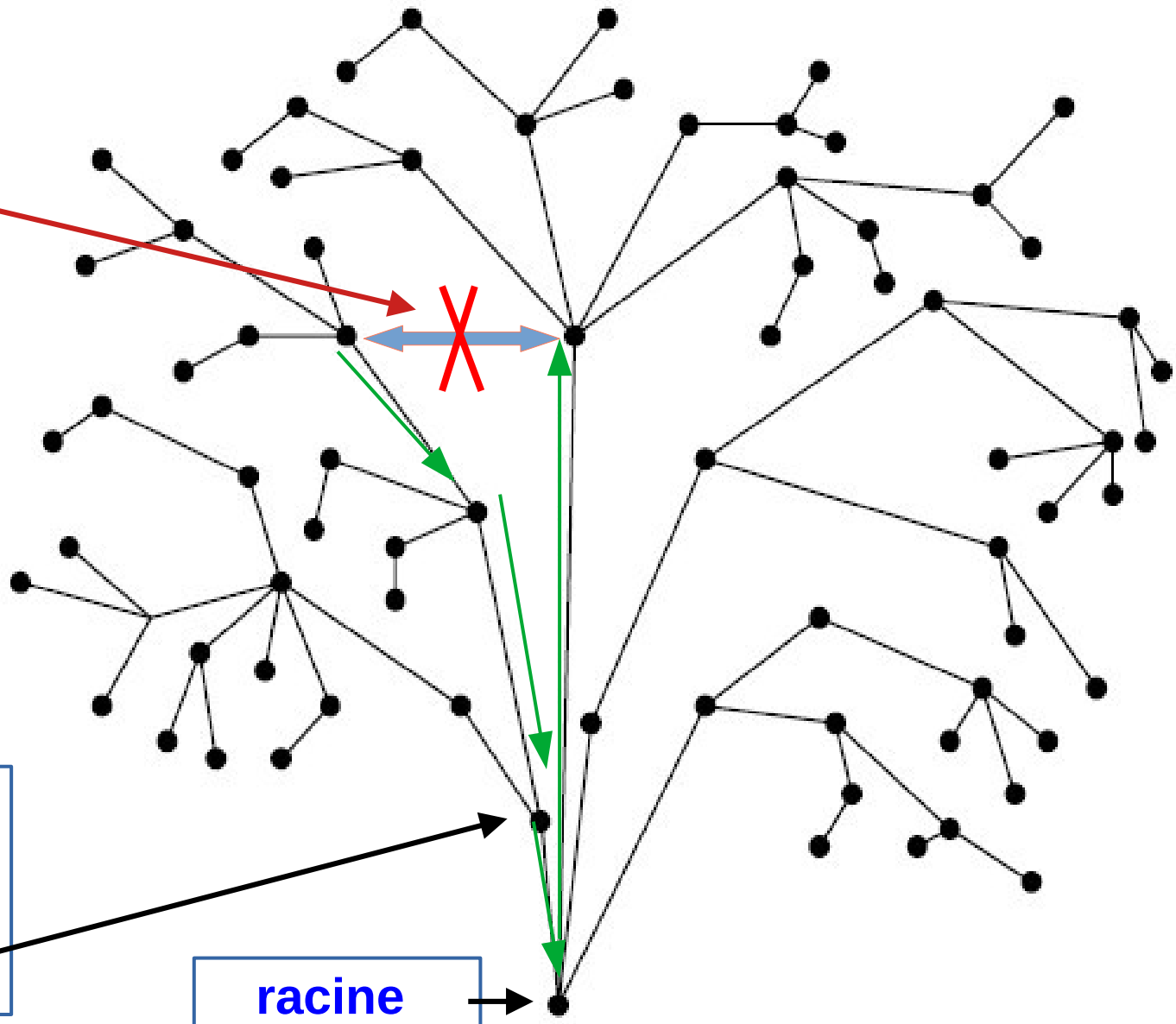
- **simple** : le graphe **A** ne contient ni *boucle*, ni *cycle*, ni *multi-arête* ;
- **connexe** : il y a *toujours un chemin* (une succession finie d'arêtes) reliant sans interruption toute paire de sommets de **A** ;

et

- **sans cycle** : il n'y a *jamais de chemin* reliant un sommet de **A** à lui-même.

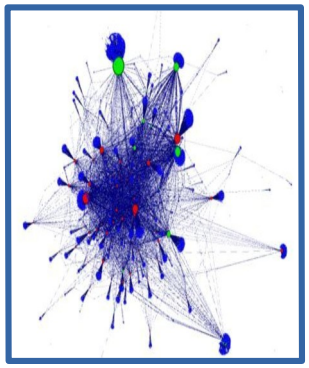
Les CARACTÉRISTIQUES d'un ARBRE

Dans un **arbre**, on ne peut pas sauter d'une ***branche*** à l'autre!
Il faut repasser par la ***racine*** !



Les sommets d'un **arbre**, souvent appelés **nœuds**.

racine



2 PROBLÈMES CLASSIQUES de PARCOURS et de DÉNOMBREMENT dans les GRAPHES

(Un petit aperçu !)

2-1 Le *PARCOURS d'*un GRAPHE

Qu'est-ce que Parcourir (Explorer) un Graphe ?

C'est **énumérer** l'ensemble des sommets accessibles par un **chemin** à partir d'un sommet **s** donné, les descendants de **s**, dans l'objectif de faire subir à ces sommets un certain traitement.

Types de Parcours : élémentaire, eulérien, hamiltonien, ...

► Comment définit-on le **parcours d'un graphe** ?

Le terme **parcours** désigne l'ensemble des *chemins*, des *circuits*, des *chaînes* et des *cycles* d'un graphe.

► Un parcours peut être :

- **élémentaire**, s'il n'emprunte qu'une seule fois ses *sommets*,
- **eulérien**, s'il passe une seule fois par chaque *arête* ou *arc*,
- **hamiltonien**, s'il passe une seule fois par chaque *sommet* (et donc au plus une fois par chaque *arête*).

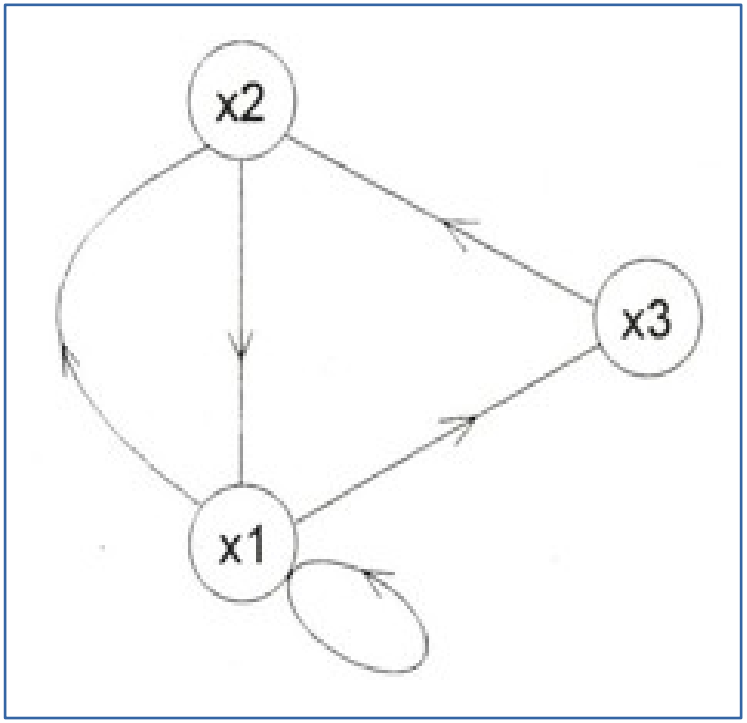
► Exemples de parcours **élémentaires**

Un parcours dans un graphe G est **élémentaire** s'il n'emprunte qu'**une seule fois** chacun des sommets qui sont sur ce parcours.

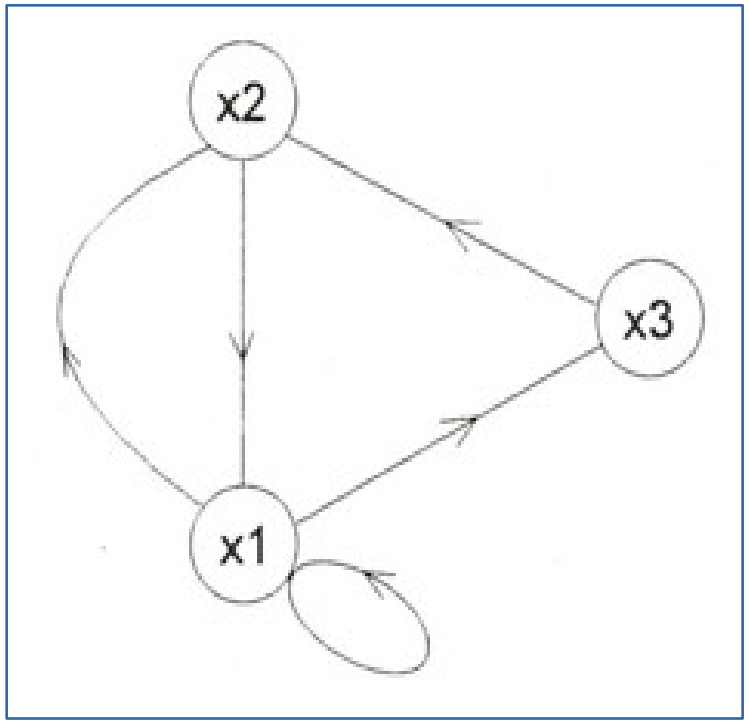
Dans le graphe *orienté* **G** ci-dessous :

1 - le parcours $\{x_2, x_1, x_1, x_3\}$ *n'est pas élémentaire*, car il passe deux fois par x_1 ;

2 - le parcours $\{x_1, x_1\}$ *est élémentaire* ; c'est un cas-limite, celui d'une boucle, qui ne passe qu'une fois par x_1 ;



3 - le parcours $\{x_1, x_3, x_2, x_1\}$ est un **circuit** élémentaire ;



4 - le parcours $\{x_3, x_1, x_2\}$ est une **chaîne** élémentaire du graphe **G'** non orienté, associé au graphe **G** ;

5 - le parcours $\{x_1, x_2, x_3, x_1\}$ est un **cycle** élémentaire du graphe **G'** non orienté associé à **G**.

► Exemples de parcours **eulérien**

Le parcours d'un graphe **G** est **eulérien** s'il passe **une seule fois** par **chaque *arête*** ou ***arc*** de **G**.

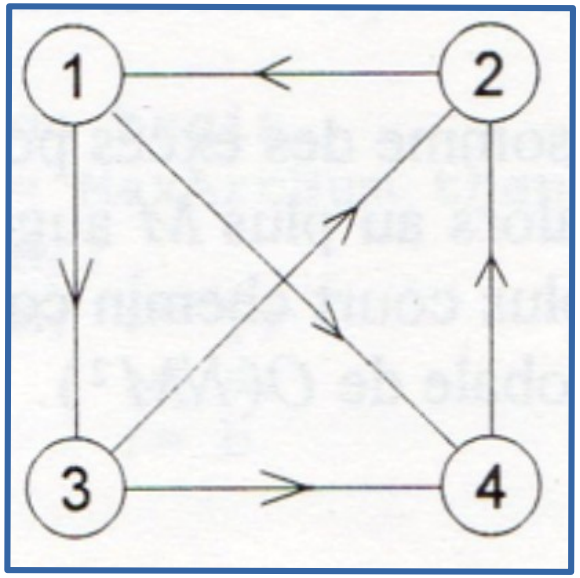
(le parcours peut passer plus d'une fois par chaque *sommet*)

C'est le mathématicien **Euler** qui est à l'origine de cette notion, élaborée à l'époque du problème des **ponts de Königsberg**.

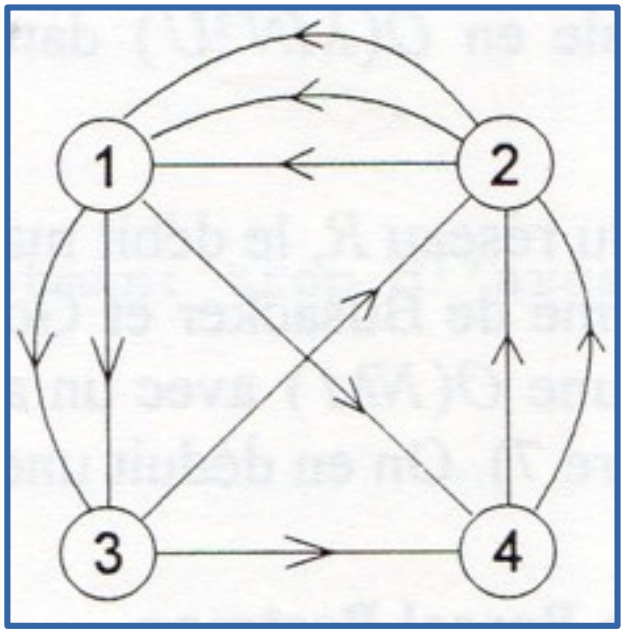
Les **graphes eulériens** renvoient notamment à deux problèmes célèbres :

- le problème du **postier chinois**,
- le problème du **voyageur de commerce**.

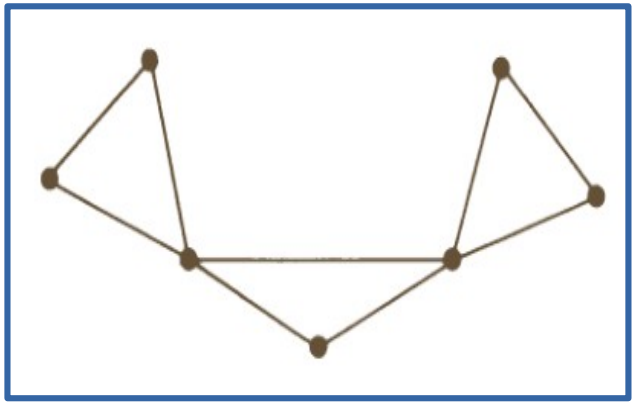
Les **graphes eulériens** se rencontrent dans des questions pratiques comme l'*optimisation* des tournées (distribution du courrier, ramassage scolaire, livraisons à un réseau de magasins, etc).



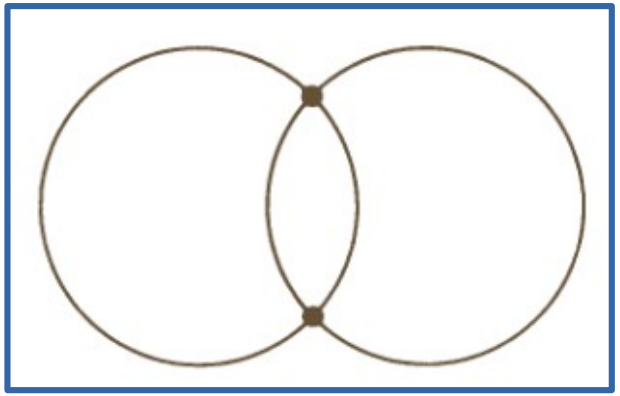
Graphe fortement connexe *non* eulérien



Graphe rendu eulérien par ajout des pseudo-chemins (4,2,1,3) et (2,1)



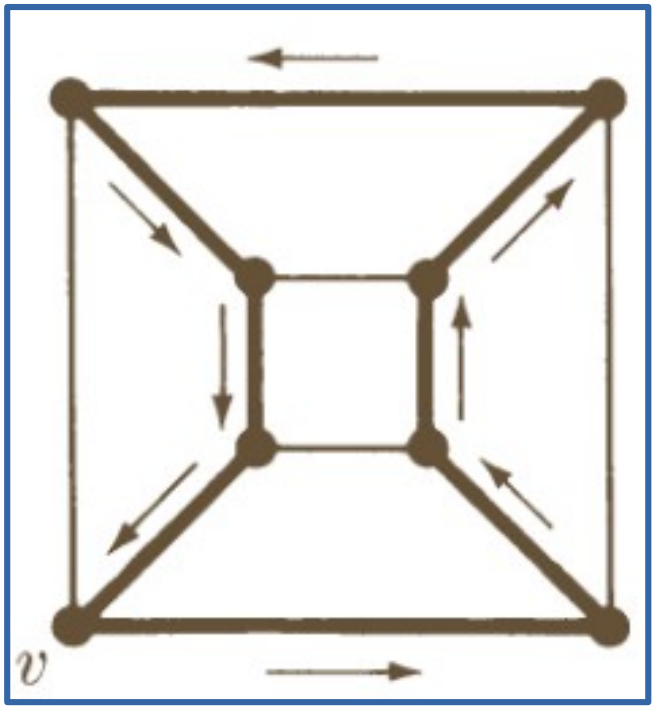
Deux graphes eulériens

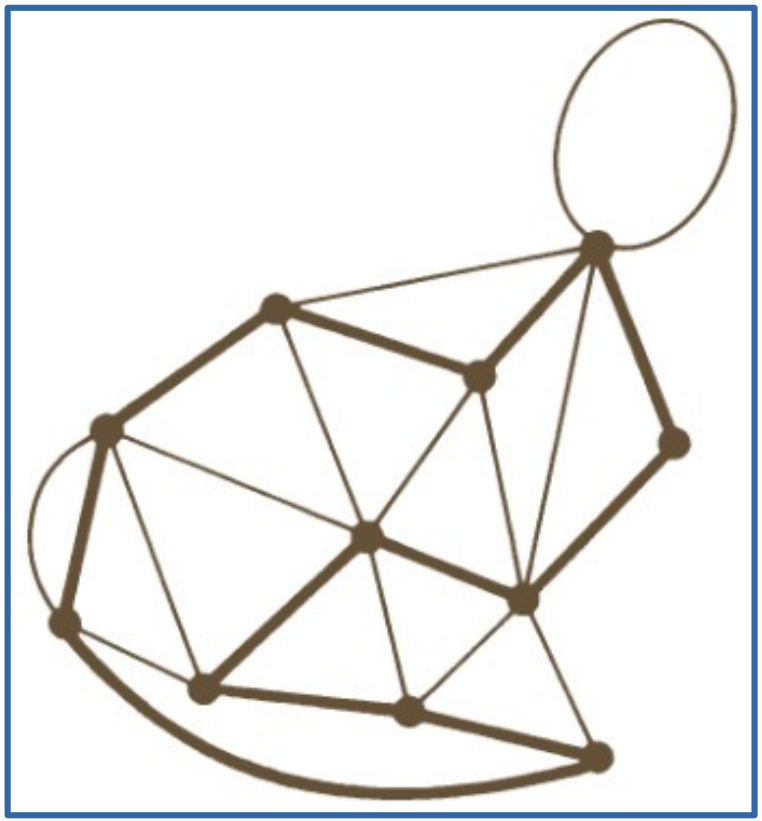


► Exemples de parcours **hamiltoniens**

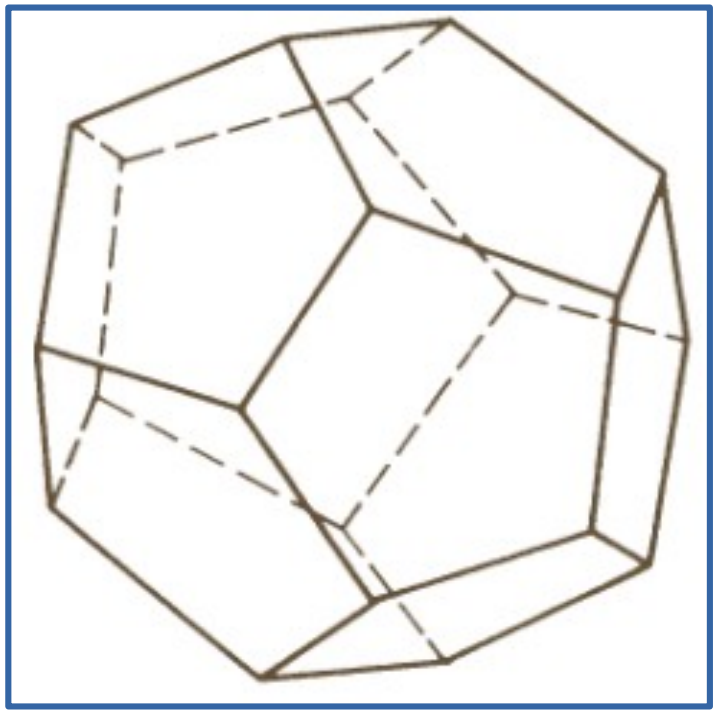
Le parcours d'un graphe **G** est **hamiltonien** s'il passe **une seule fois** par *chaque* ***sommet*** de **G**.

C'est le mathématicien **Hamilton** qui est à l'origine de cette notion.



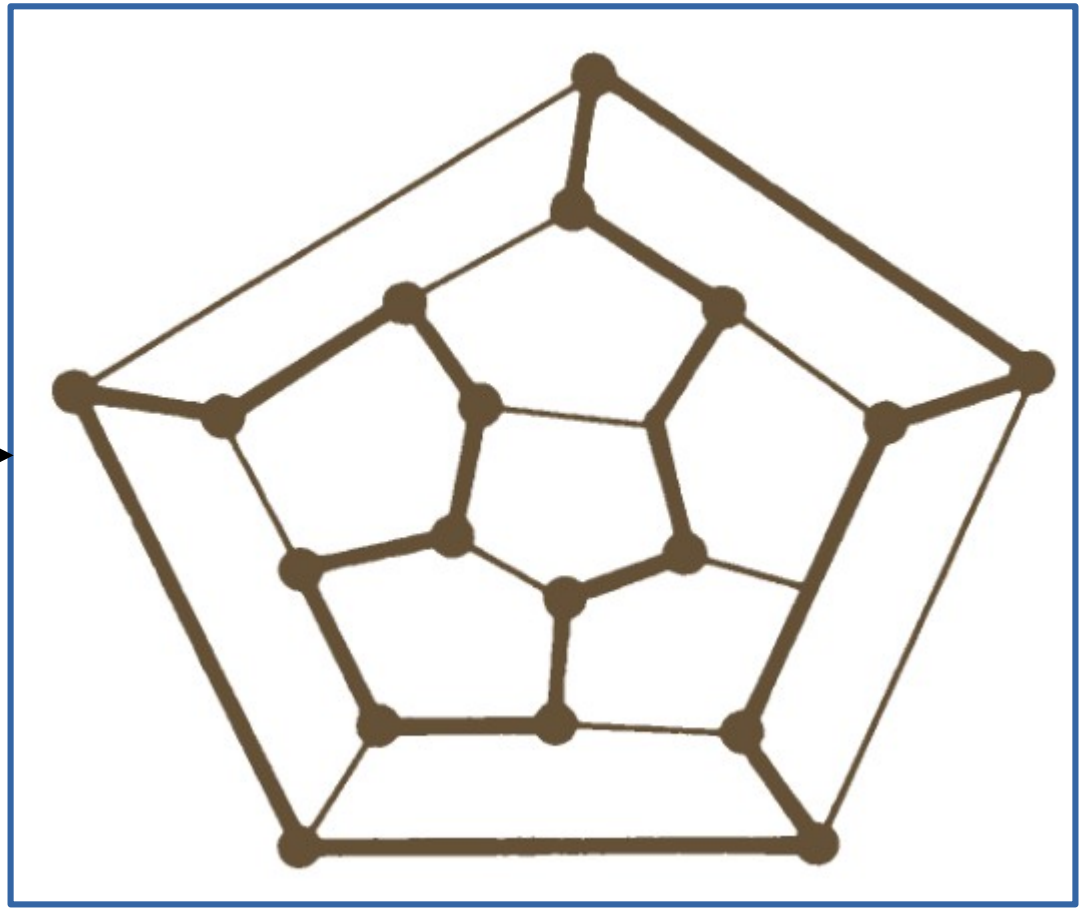


William Rowan Hamilton (1805-1865)



Un **dodécaèdre D** ...

… et le **circuit hamiltonien**
passant 1 seule fois
par chacun des *sommets*
de **D** (en traits gras)



2-2

Le DÉNOMBREMENT des ARBRES : CAYLEY et PRÜFER

- Dans le parcours d'un graphe, on utilise un *arbre* pour représenter les données, mais cette représentation n'est pas *optimale* !
- Afin d'évaluer la complexité de l'exploration des parcours possibles d'un graphe, on cherche *d'abord* à dénombrer les arbres à traiter.

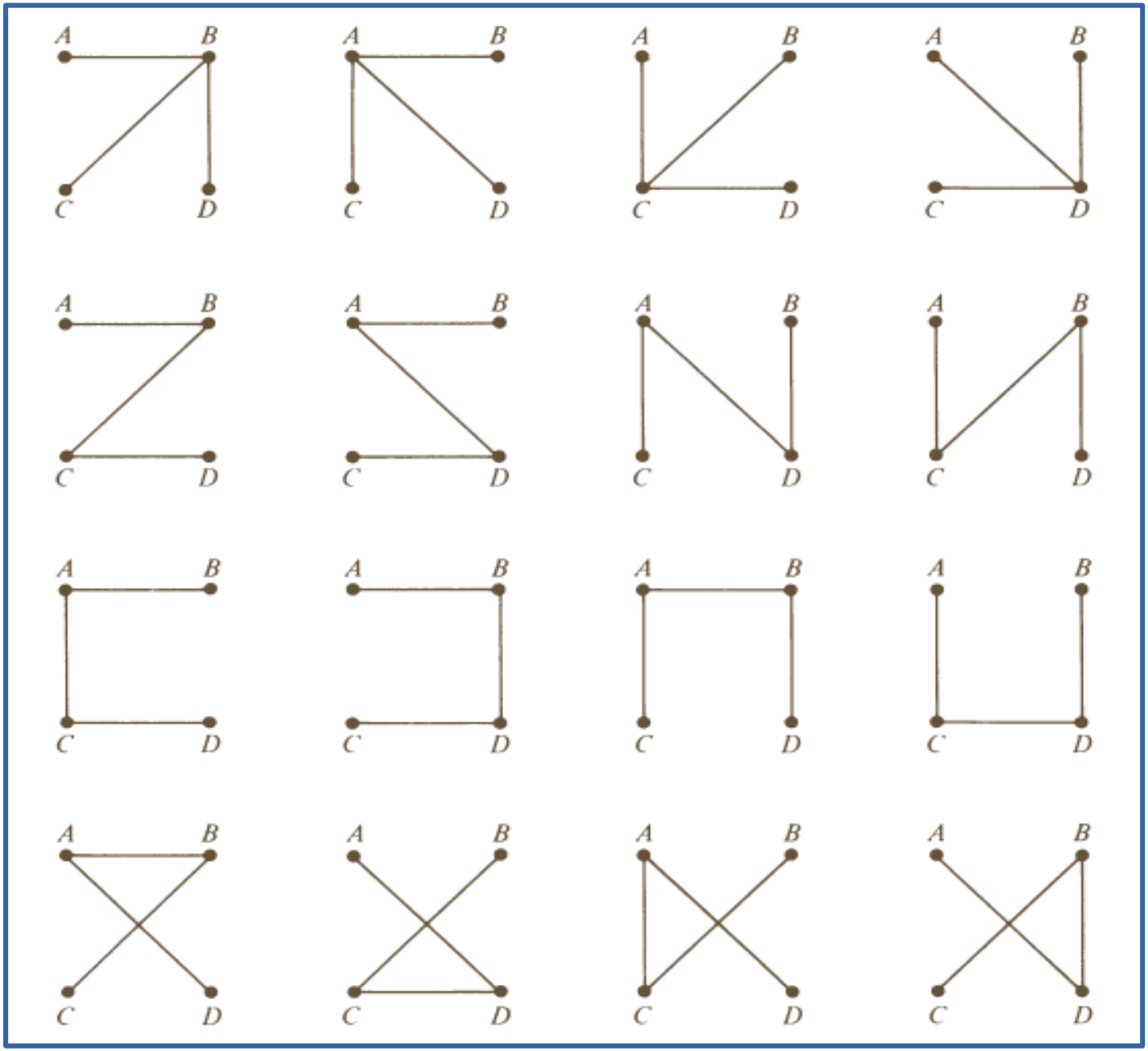
Le THÉORÈME de CAYLEY et le DÉNOMBREMENT des ARBRES *ÉTIQUETÉS*

- Calcul du Nombre d'**Arbres Etiquetés**

Le nombre **N** d'arbres *étiquetés* (non orientés) de **n** sommets qu'on peut construire est égal à n^{n-2} : c'est le **théorème de Cayley**.

Voir : https://fr.wikipedia.org/wiki/Formule_de_Cayley

► Les $n^{n-2} = 4^2 = 16$ arbres étiquetés de **n=4** sommets



DÉNOMBREMENT des ARBRES *ÉTIQUETÉS* : le CODE de PRÜFER

En **1918**, le mathématicien autrichien **Heinz Prüfer** proposa une *procédure* qui constituait une preuve *constructive* simple du résultat de Cayley ...

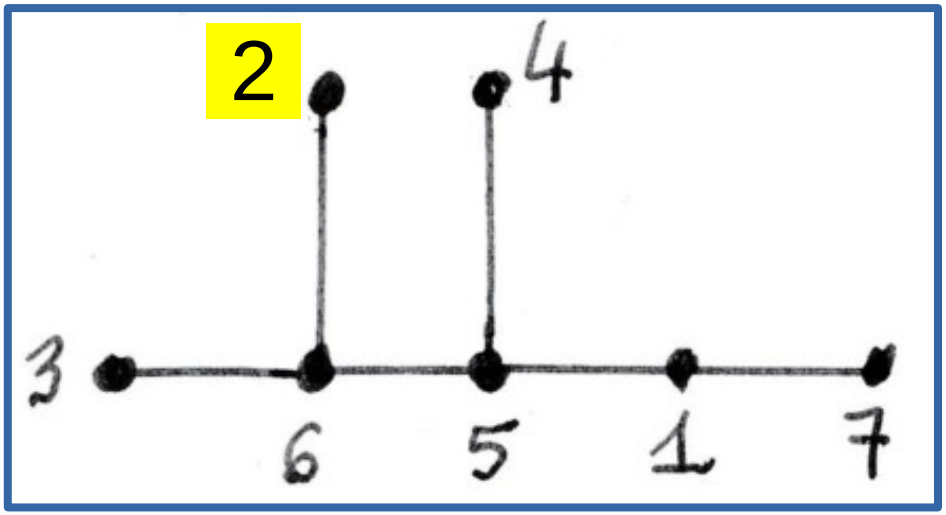
Heinz Prüfer
1896-1934

La **méthode** (ou : **code**) **de Prüfer** consiste à établir une **bijection** entre l'ensemble des arbres de **n** sommets et l'ensemble des chaînes de **n-2** entiers compris entre **1** et **n**.

► 1) Exemple de Construction d'une **suite de Prüfer** à partir d'un **Arbre Etiqueté**

On se donne l'arbre de **n** = 7 sommets :

$A_1 = \{1,2,3,4,5,6,7\}$



$A_1 = \{1,2,3,4,5,6,7\}$

Opération **1** : Chercher dans A_1 les sommets de degré **deg=1**.

Ce sont les sommets: 3, 2, 4 et 7.

Retenir celui qui a **la plus petite étiquette**. C'est le sommet 2

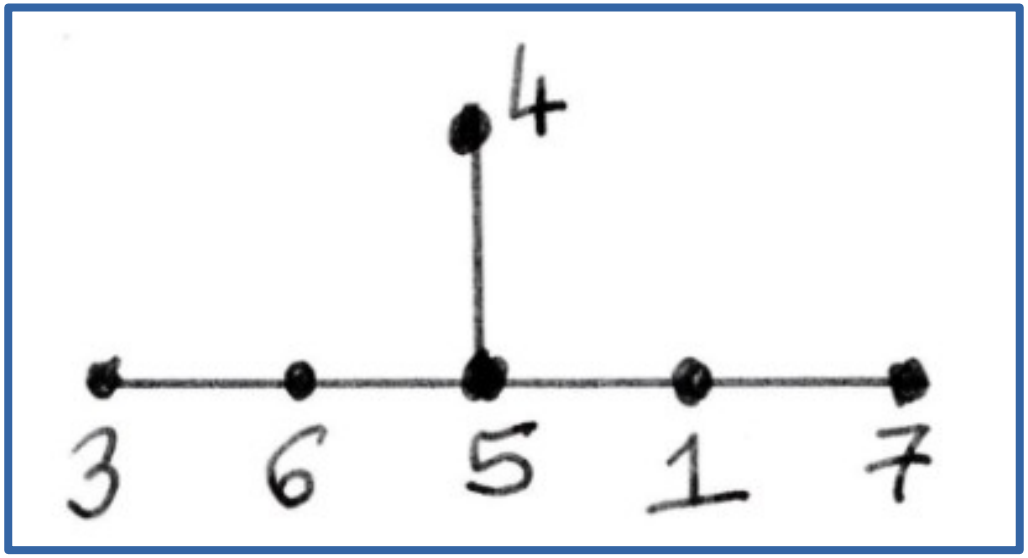
Opération **2** : Chercher dans A_1 le **sommet adjacent** au sommet **2**.

C'est le sommet **6**. C'est le 1er terme de la **suite de Prüfer** :

P = (**6**, ?, ?, ?, ?)

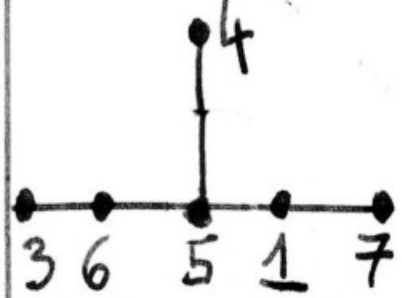
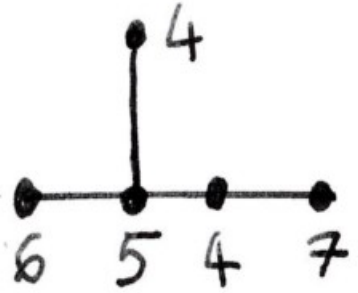
Opération **3** : On supprime dans l'arbre A_1 le **sommet 2** et la branche **(2,6)**. L'arbre A_1 devient :

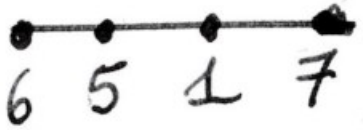
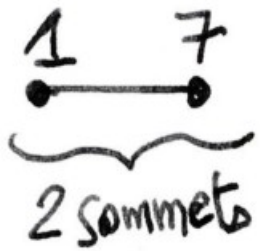
$A_2 = \{1,3,4,5,6,7\}$



Opération **4** : On réitère les opérations **1** à **3** jusqu'à obtenir un arbre A_{n-2} qui n'ait plus que **2** sommets ...

Dans notre exemple, on obtient :

K	Arbre A_K	Sommets de degré 1	Plus petite étiquette et sommet adjacent	Branche à supprimer	Suite de PRUFER
2		3,4,7	3 et 6	(3,6)	(6,6,?,?,?)
3		4,6,7	4 et 5	(4,5)	(6,6,5,?,?)

k	Arbre A_k	Sommets de degré 1	Plus petite Etiquette et sommet adjacent	Branche à supprimer	Suite de PRÜFER
4		6, 7	6 et 5	(6, 5)	(6, 6, 5, 5, ?)
5		5, 7	5 et 1	(5, 1)	(6, 6, 5, 5, 1)
					

Ici, la procédure est terminée, car l'arbre A_1 de départ n'a plus que **2** sommets !

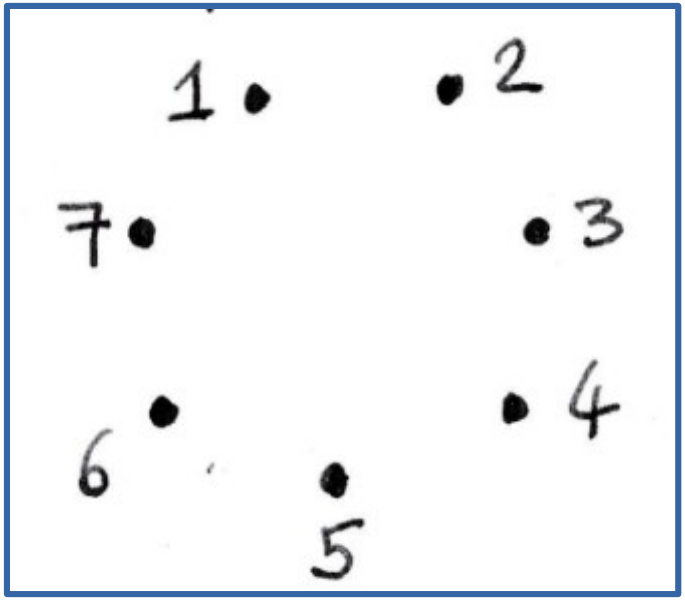
La procédure s'est bien terminée à l'étape k = **n-2** = **7-2** = **5** et la **suite de Prüfer** obtenue compte bien **n-2** éléments !

► 2) Reconstruire l' **Arbre Etiqueté** à partir d'une **suite de Prüfer**

La 2ème partie de la preuve de Prüfer du théorème de Cayley consiste à exploiter une suite de Prüfer pour en obtenir l'arbre étiqueté correspondant.

Opération **1** : Plaçons dans le plan **n**=7 points numérotés de 1 à 7. Ce seront les sommets du graphe (arbre) à construire.

Pourquoi **n**=7 ? Parce que c'est le nombre de termes de la suite de Prüfer donnée, $\mathbf{P_1}$ = (**6**,6,5,5,1) , plus 2.



Donc : à la **suite de Prüfer** de **n-2**=5 termes $\mathbf{P_1}$ = (**6**,6,5,5,1) correspond la Liste $\mathbf{L_1}$ = (1,2,3,4,5,6,7) de **n**=7 termes ...

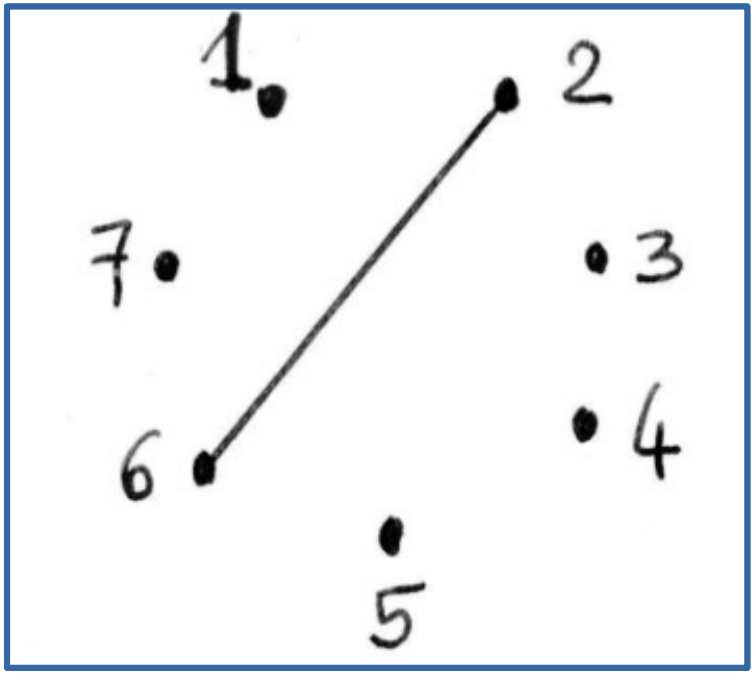
Opération **2** : On choisit :

a) dans la liste **L_1** = (1,2,3,4,5,6,7) le plus petit nombre qui ne se trouve pas *aussi* dans la suite (6,6,5,5,1). Ici, c'est : **2**

(**1** est bien le plus petit nombre de la liste **L_1**, mais il est *aussi* dans la suite !)

b) dans la suite **P_1** = (**6**,6,5,5,1), le **1er nombre** ; c'est : **6** .

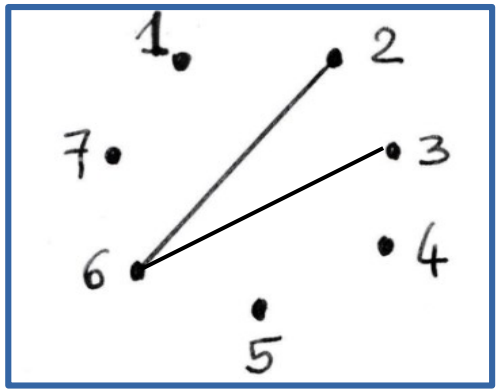
Opération **3** : On trace l'arête joignant les sommets **2** et **6** :

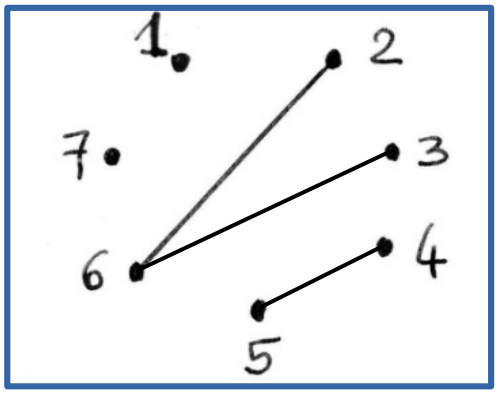
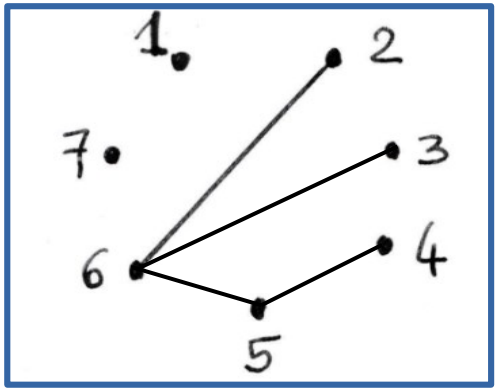


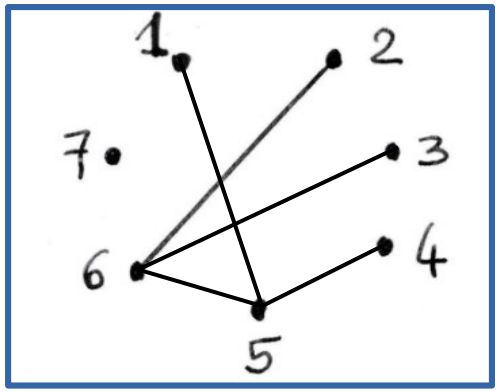
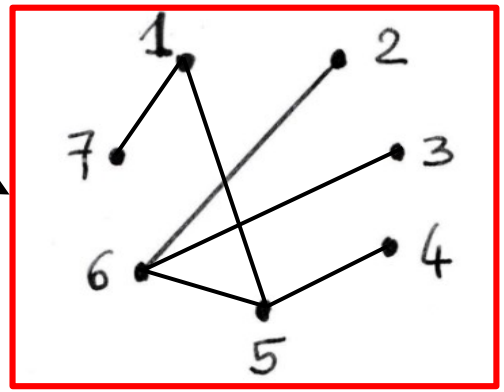
Opération **4** : On supprime le terme **2** de la liste et de la suite d'où :

- la nouvelle liste : (1,3,4,5,6,7) L_2
- la nouvelle suite : (**6**,5,5,1) P_2

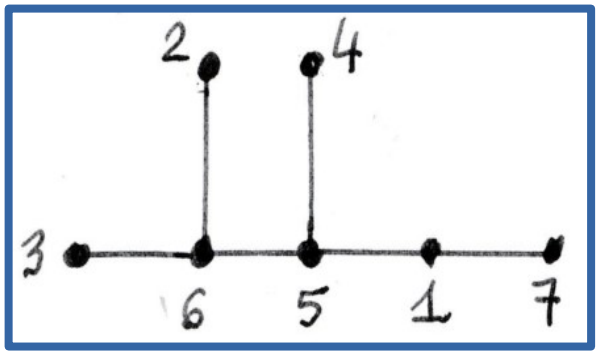
Opération **5** : A partir de la liste L_2 et de la suite P_2, on réitère les opérations **2** à **4** jusqu'à ce que la suite P_k soit vide !

k	Plus petit nombre de la liste L_k qui n'est pas dans la suite P_k	Arête correspondante de l'arbre A_k	Nouvelle liste L_k Nouvelle suite P_k
2	L_2 = (1,**3**,4,5,6,7) P_2 = (**6**,5,5,1)	(**3**,**6**) 	L_3 = (1,4,5,6,7) P_3 = (5,5,1)

3	$L_3 = (1,\mathbf{4},5,6,7)$ $P_3 = (\mathbf{5},5,1)$	$(\mathbf{4},\mathbf{5})$ 	$L_3 = (1,5,6,7)$ $P_3 = (5,1)$
4	$L_4 = (1,5,\mathbf{6},7)$ $P_4 = (\mathbf{5},1)$	$(\mathbf{6},\mathbf{5})$ 	$L_5 = (1,5,7)$ $P_5 = (1)$

5	$L_5 = (1,5,7)$ $P_5 = (1)$	(5,1) 	$L_6 = (1,7)$ $P_6 \neq (0)$
6	$L_6 = (1,7)$ $P_6 = (\emptyset)$	(1,7) A_2 	**La procédure est terminée** : - la liste L_6 ne contient plus que 2 termes ; - la suite $P_6 = (\emptyset)$

L'arbre *reconstruit* A_2 est isomorphe à l'arbre A_1 d'origine →



2-3

Les DIFFÉRENTES FAÇONS de PARCOURIR un GRAPHE

Les Types de PARCOURS d'un Graphe

Il y a plusieurs façons de **parcourir un graphe** G.

Parmi les types de parcours les plus courants, on trouve :

- le **parcours générique**,
- le **parcours en largeur**,
- le **parcours en profondeur**.

Le Parcours GÉNÉRIQUE d'un Graphe

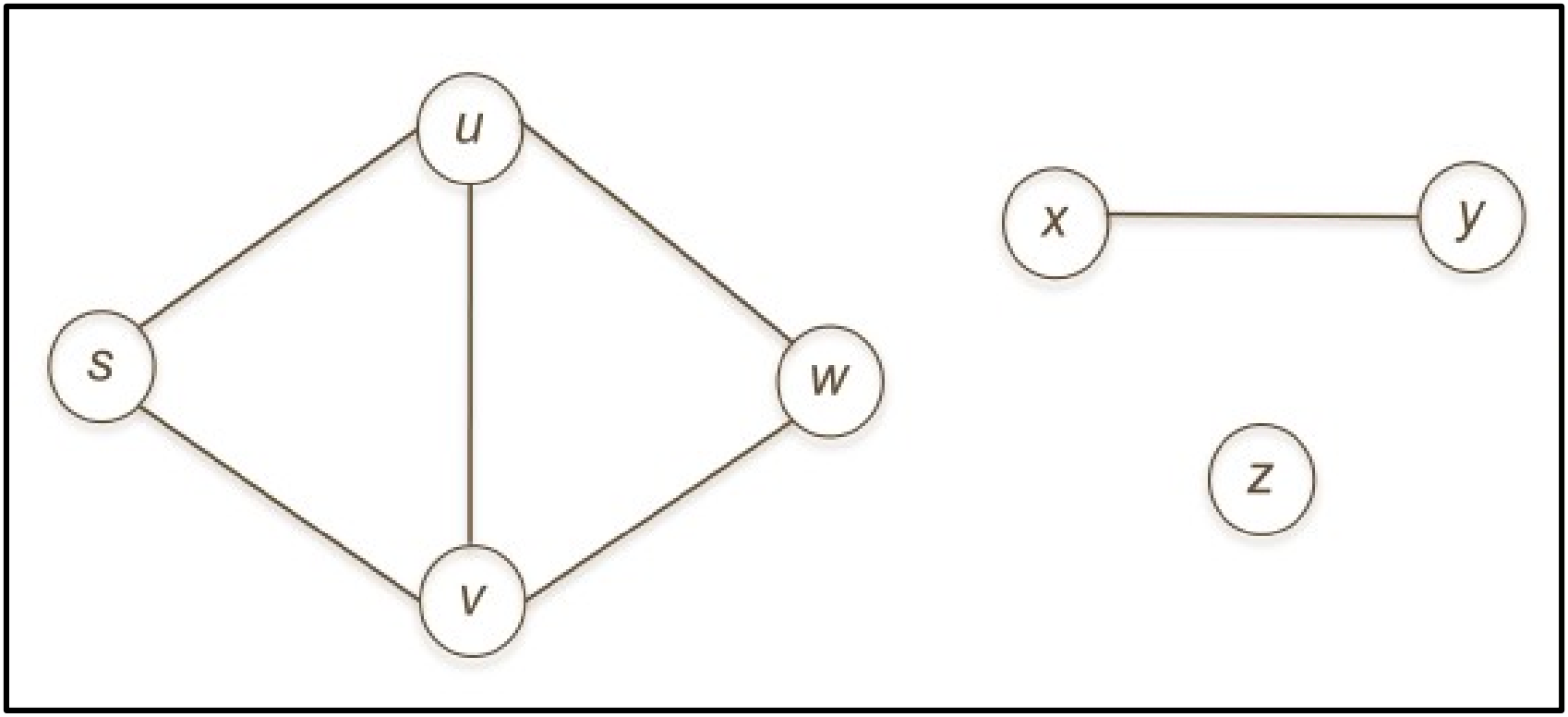
L'objectif du parcours d'un graphe est de déterminer s'il existe (au moins) un **chemin** qui, à partir d'un sommet de départ **s**, *atteint* un certain sommet **y** de **G**.

Le **parcours générique** d'un graphe consiste alors :

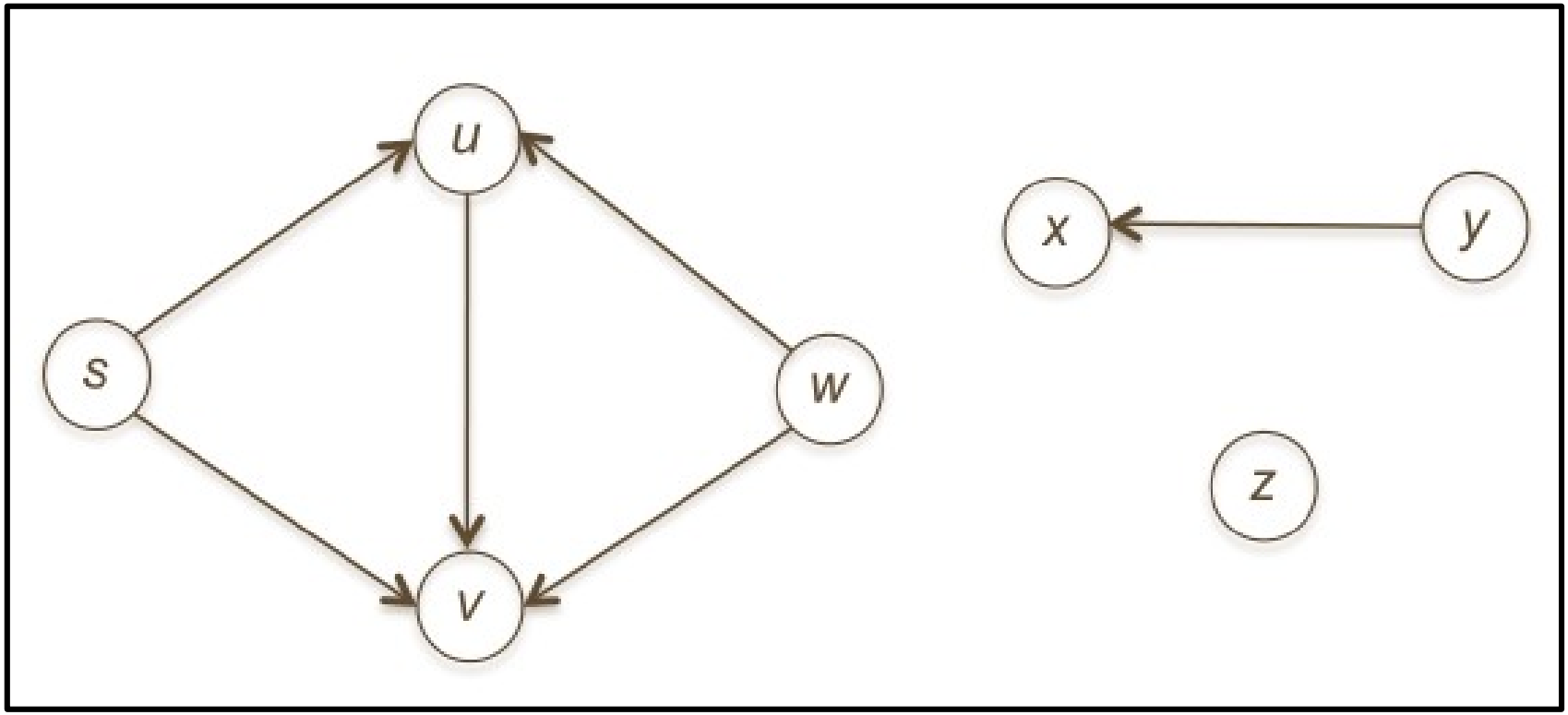
- à se donner un graphe **G**, orienté ou non,
- à choisir un sommet de départ **s** de **G**,
- à identifier les sommets de **G** « atteignables » à partir du sommet **s**.

► Caractéristiques d'un parcours générique

1) Le fait que le graphe G soit *orienté* ou non a son importance



Graphe G non orienté :
l'ensemble des sommets *atteignables* au départ de **s** est
A={s,u,v,w}



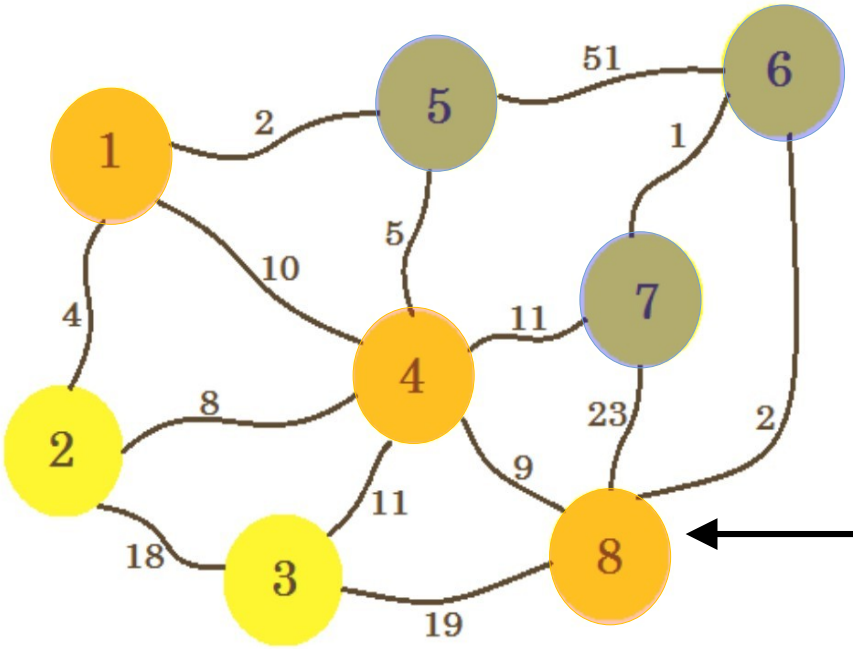
Graphe G <u>orienté</u> :
l'ensemble des sommets *atteignables* au départ de **s** est
A={s,u,v}

2) Dans un chemin obtenu par un **parcours générique** à partir du sommet de départ **s**, <u>un même sommet du graphe **G** ne doit pas figurer plus d'une fois</u>.

► Bordure d'une Partie d'un Graphe et Parcours Générique

Autre façon de représenter un **parcours générique** partant d'un sommet **s** du graphe **G** : définir la **bordure** d'une partie **E** de **G** contenant **s**.

3) Étant donnés un sommet **s** et une partie **E** non vide d'un graphe **G**, la **bordure** **ß(E)** de **E** est l'ensemble des sommets de **G-E** qui sont *adjacents* à **E**.



1er Exemple de **bordure**

G={1,2,3,4,5,6,7,8}

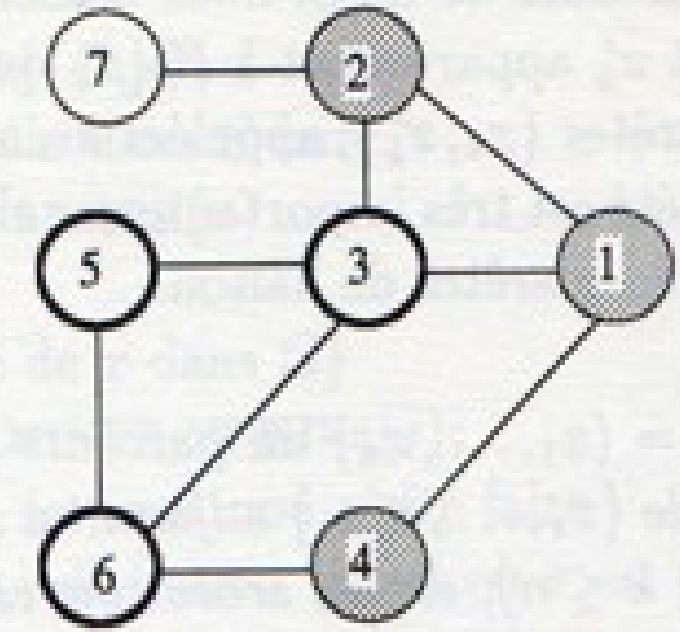
E={5,6,7}

Bordure de **E** = **ß(E)** = {1,4,8}

4) *Un* **parcours générique** d'un graphe G, au départ de l'un de ses sommets **s**, est alors une **liste** de sommets de G (donc un *sous-graphe*) obtenue par une *boucle* appliquée à la **bordure ß(E)** de la partie **E** de **G** qui contient le sommet **s**.

2ème Exemple de **bordure**

G = {1,2,3,4,5,6,7}, **E** = {3,5,6}, bordure de **E** = **ß(E)** = {1,2,4}



(Beauquier, Eléments d'Algorithmique)

Exemple de Parcours

G = {1,2,3,4,5,6,7},

E = {3,5,6},

Bordure de **E** = **ß(E)** = {1,2,4}

Sommet de départ = **s** = **3**

Parcours :

{**3**,1,2}, {**3**,2,1}, {**3**,1,4}

Le Parcours en LARGEUR d'un Graphe

En partant d'un sommet *quelconque* x, on atteint d'abord les voisins de x, ensuite les voisins des voisins (sauf ceux qui sont déjà atteints) et ainsi de suite.

Le parcours en largeur est aussi appelé **parcours concentrique**.

Il sert notamment à la recherche des **plus courts chemins** dans un graphe, i.e. de la plus courte **distance** entre deux sommets du graphe.

Le Parcours en PROFONDEUR d'un Graphe

A partir d'un sommet x :

- on tente d'avancer le plus loin possible dans le graphe, jusqu'à épuiser toutes les possibilités de progression ;

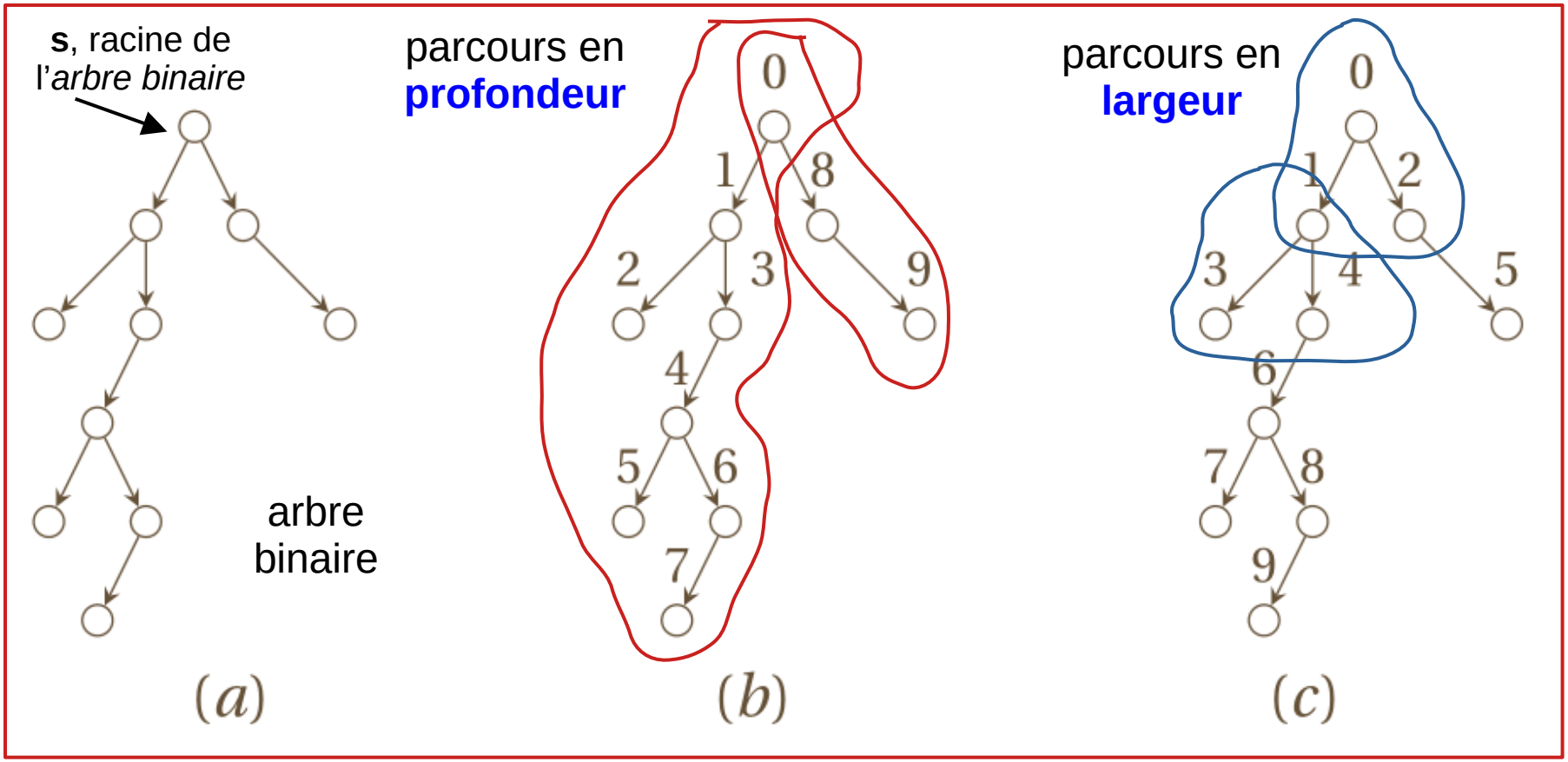
- on revient ensuite au sommet de départ pour explorer un nouveau chemin ou une nouvelle chaîne.

Le **parcours en profondeur** correspond à l'exploration d'un ***labyrinthe***. Ce type de parcours permet de résoudre efficacement certains problèmes plus difficiles :

- rechercher les composantes fortement connexes d'un graphe orienté (https://fr.wikipedia.org/wiki/Composante_fortement_connexe),
- tester la planarité, etc.

► Comparaison des Parcours en LARGEUR et en PROFONDEUR

A partir du graphe en (a), qui est un *arbre binaire*, enraciné au sommet **s**, on effectue en (b) un parcours en **profondeur** et en (c) un parcours en **largeur** :



Le parcours en **profondeur** explore d'abord la racine **s**, puis tout le sous-arbre gauche, et enfin tout le sous-arbre droit.

Le parcours en **largeur** explore d'abord la racine **s**, puis le *fils* gauche de **s**, puis le *fils* droit, et répète ces actions sur les sous-arbres.

1ère Étape

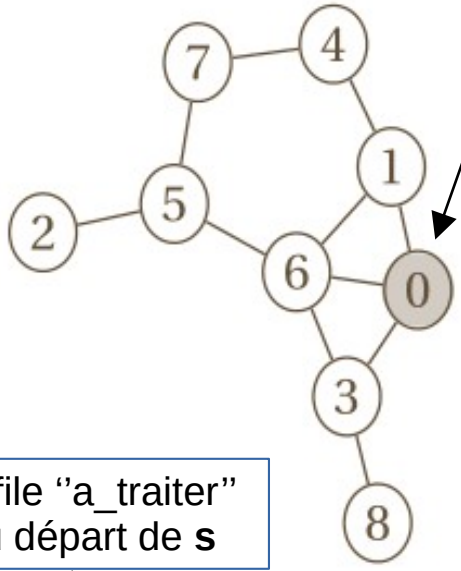
sommet de départ **s**

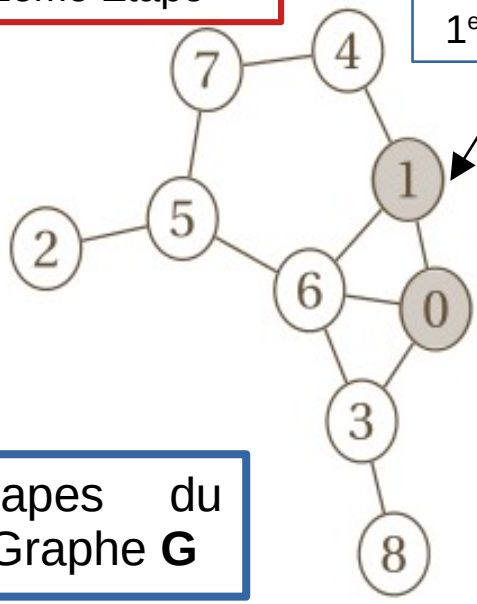
2ème Étape

Le sommet **1**, 1er voisin de **s**

Exemple de Parcours en Largeur

G





Les 2 Premières Étapes du **Parcours en Largeur** du Graphe **G**

La file ‘’a_traiter’’ au départ de **s**

a_traiter	6	3	1
résultat	0		

a_traiter	6	4	6	3
résultat	0	1		

La liste ‘’résultat’’

- Le parcours commence au sommet **s**, qui est numéroté **0** dans la liste ‘’résultat’’ ;
- les voisins de **s**, (**1**, **3** et **6**) vont dans la file ‘’a_traiter’’ ;
- le sommet **0**, qui a été traité, est ajouté à la liste ‘’résultat’’ ...

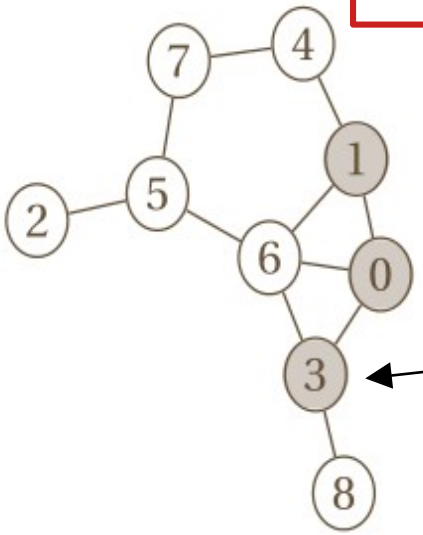
- Le parcours commence au sommet **1**, 1er voisin de **s**, qui est ajouté à la liste ‘’résultat’’ ;
- les voisins de **1** sont ajoutés dans la file ‘’a_traiter’’ (**4** et **6**) ;
- le sommet **1**, qui a été traité, est retiré de la file et ajouté à la liste ‘’résultat’’ ...

Déroulement du PARCOURS en LARGEUR

3ème Étape

4ème Étape

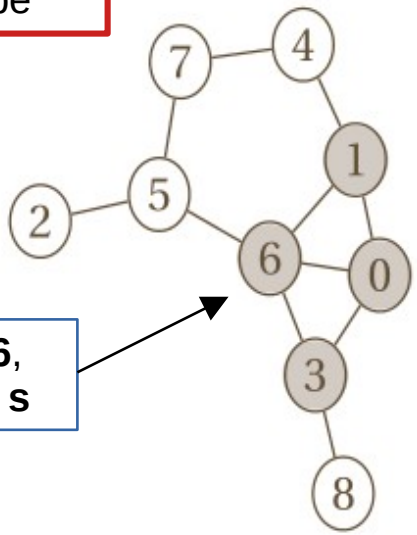
G



Les 3ème et 4ème Étapes du **Parcours en Largeur** du Graphe **G**

Le sommet **3**, 2ème voisin de **s**

Le sommet **6**, 3ème voisin de **s**



a_traiter	8	6	6	4	6
résultat	0	1	3		

a_traiter	5	8	6	6	4
résultat	0	1	3	6	

La liste ‘’résultat’’

- Le parcours se poursuit par le sommet **3**, 2ème voisin de **s**, qui est ajouté à la liste ‘’résultat’’ ;
- les voisins de **3** sont ajoutés dans la file ‘’a_traiter’’ (**6** et **8**) ;
- le sommet **3**, qui a été traité, est retiré de la file ‘’a_traiter’’ et ajouté à la liste ‘’résultat’’...

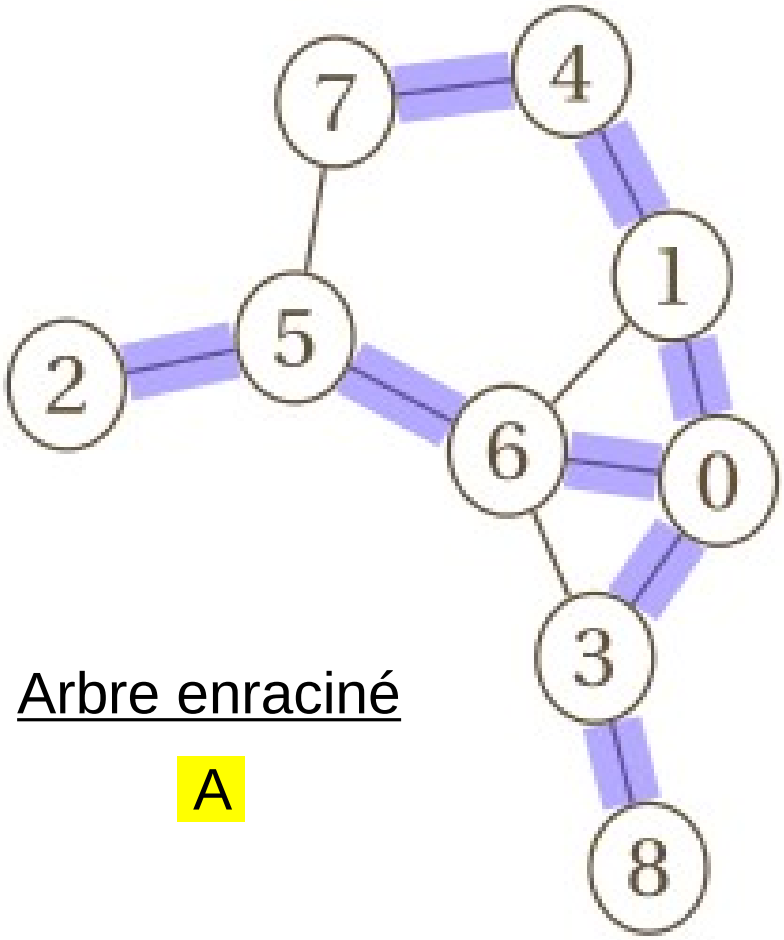
- Le parcours se poursuit par le sommet **6**, 3ème voisin de **s**, qui est ajouté à la liste ‘’résultat’’ ;
- le seul voisin de **6** est ajouté dans la file ‘’a_traiter’’ (**5**) ;
- le sommet **6**, qui a été traité, est retiré de la file ‘’a_traiter’’ et ajouté à la liste ‘’résultat’’...

Déroulement du PARCOURS en LARGEUR (suite)

► Parcourir un graphe en largeur a un double effet :

1 - Le parcours induit un **ordre** entre les sommets “rencontrés”,

2 - Le parcours crée de fait une **arborescence**, i.e. un **arbre enraciné** représentant le parcours effectué :



Arbre enraciné

A

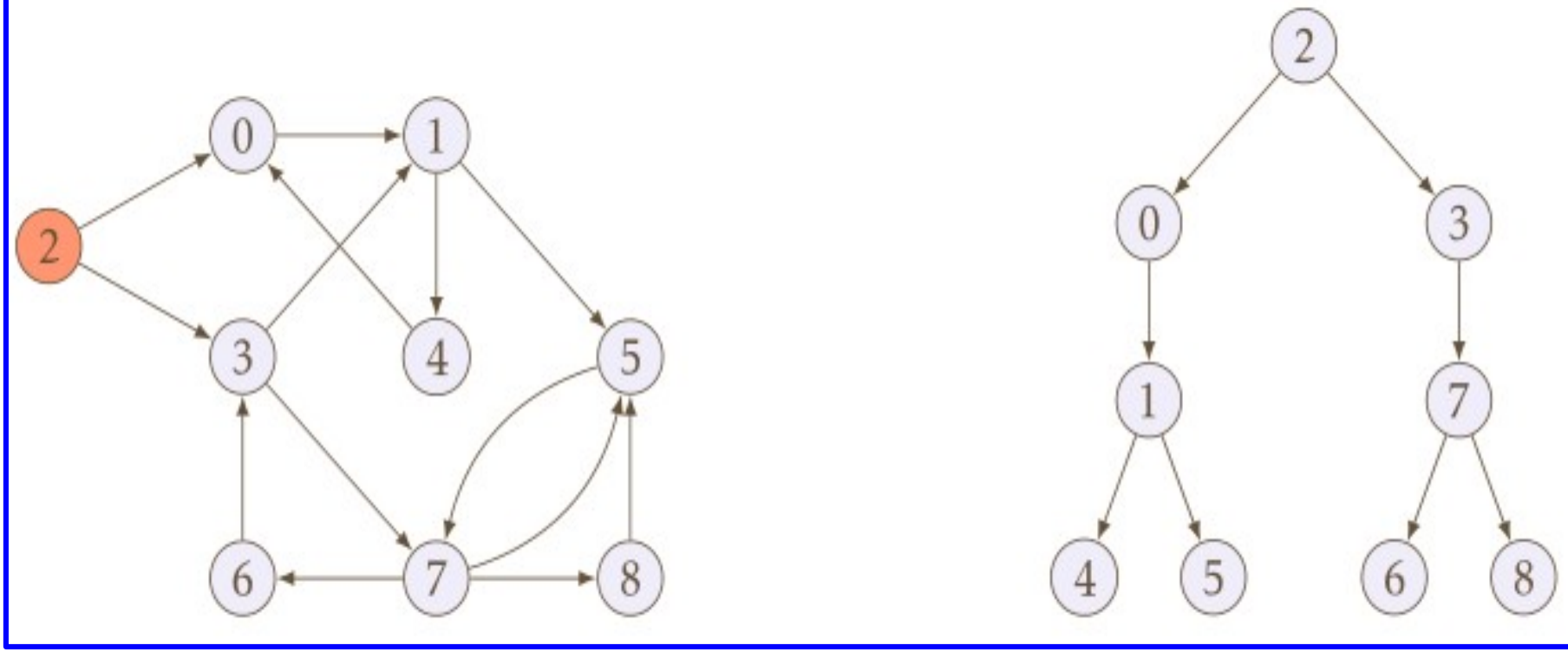
Si l’on poursuit au-delà des 4 Premières Étapes du **Parcours en Largeur** du graphe **G**, on obtient un arbre enraciné **A** :

- la racine de **A** est le sommet de départ choisi **s** = **0** ;
- de la racine partent 3 *branches*, les sous-arbres :
  - B1 = {1, 4 ,7},
  - B2 = {6, 5 ,2},
  - B3 = {3, 8}.

ARBRE résultant du PARCOURS en LARGEUR

Dans le cas général d'un *graphe* quelconque **G**, le **parcours en largeur** aboutit aussi à un **arbre** de racine **s**, correspondant au sommet de départ.

Le choix du sommet de départ **s**, parmi les **n** sommets de **G**, détermine un **arbre** (donc un **chemin**) parmi **n** possibles.



Parcours en largeur à partir du sommet **2** : graphe et **arborescence** associée

2-3

Une Façon de *Parcourir* un Graphe : l'ARBRE COUVRANT *MINIMUM*

La NOTION d'ARBRE COUVRANT

Un **arbre couvrant** (ou **recouvrant**) d'un graphe **G** *connexe*, est un **arbre A**, sous-graphe de **G**, dans lequel *tous* les sommets de **G** se retrouvent *connectés*.

Attention !

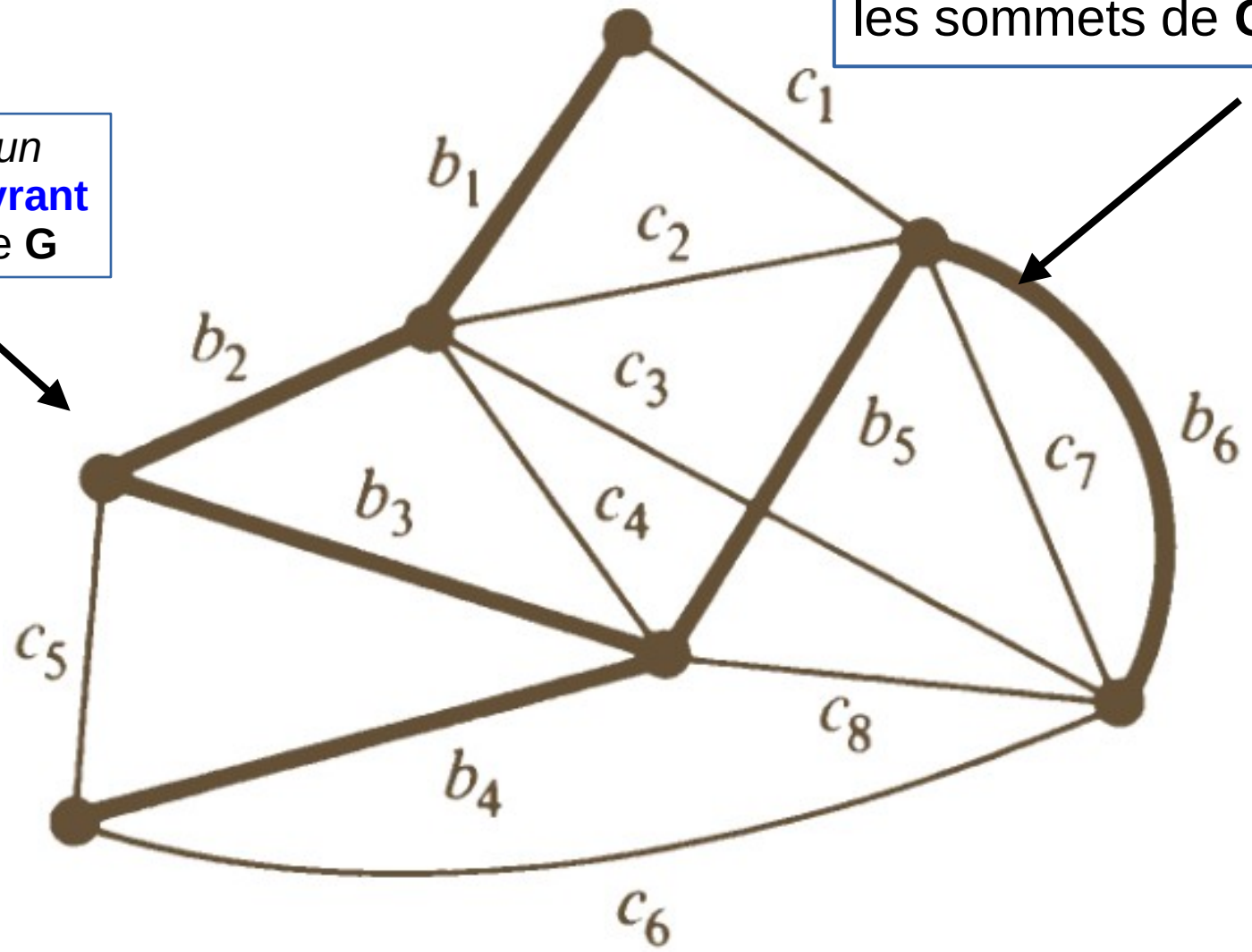
Un **arbre couvrant** n'est pas unique ! D'où le problème de *déterminer* l'***arbre couvrant minimum*** qu'on retrouve dans de nombreuses applications, notamment en optimisation ...

EXEMPLES d'ARBRES COUVRANTS

► Exemple d'**Arbre Couvrant**

Un **arbre couvrant** du graphe **G** relie *tous* les sommets de **G** …

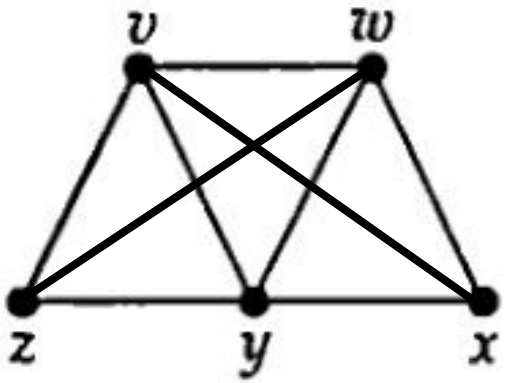
En gras *un* **arbre couvrant** du graphe **G**



G

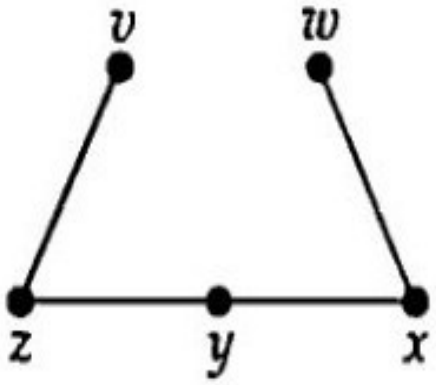
Exemple d'**arbres couvrants**

A partir du graphe **G** à **n**=5 sommets (*), on obtient les 3 arbres *couvrants* **A1**, **A2** et **A3** ci-dessous :

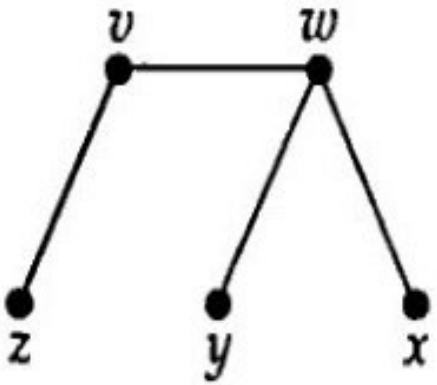


G

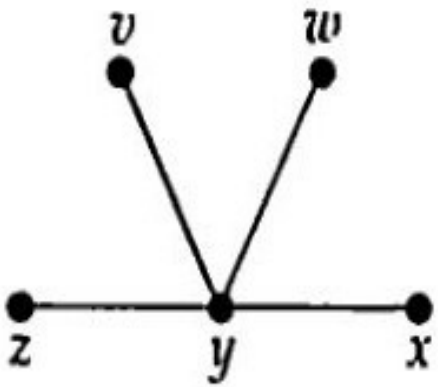
A1



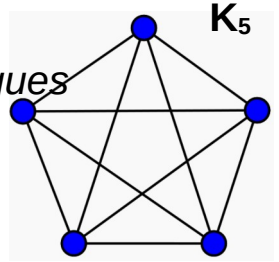
A2



A3



(*) Remarque : Le graphe **G** n'est pas complet. Un graphe est **complet** si deux *quelconques* de ses sommets sont reliés par une **arête**, i.e. sont **adjacents**, comme dans le graphe K_5 ; ce dernier est *complet* et compte **(n(n-1))/2** = 10 arêtes ; le graphe **G** n'en a que 9. La différence vient du fait que **z** et **x** ne sont pas liés par une arête, mais par un **chemin** !



Le nombre **N** d'arbres couvrants d'un graphe **G** de **n** sommets est égal au nombre d'arbres *étiquetés* de **n** sommets …

n=nombre de sommets	Nombre d'arbres non étiquetés	N=Nombre d'arbres étiquetés
1	1	1
2	1	1
3	1	3
4	2	16
5	3	125
6	6	1296
7	11	16807
8	23	262144
9	47	4782969

Dénombrement des $N = n^{n-2}$ Arbres Étiquetés d'un graphe **G** ayant **n** sommets

ALGORITHMES de l'ARBRE COUVRANT *MINIMUM*

Deux algorithmes classiques calculent
l'**arbre couvrant minimum** (= de poids minimal) d'un graphe G :

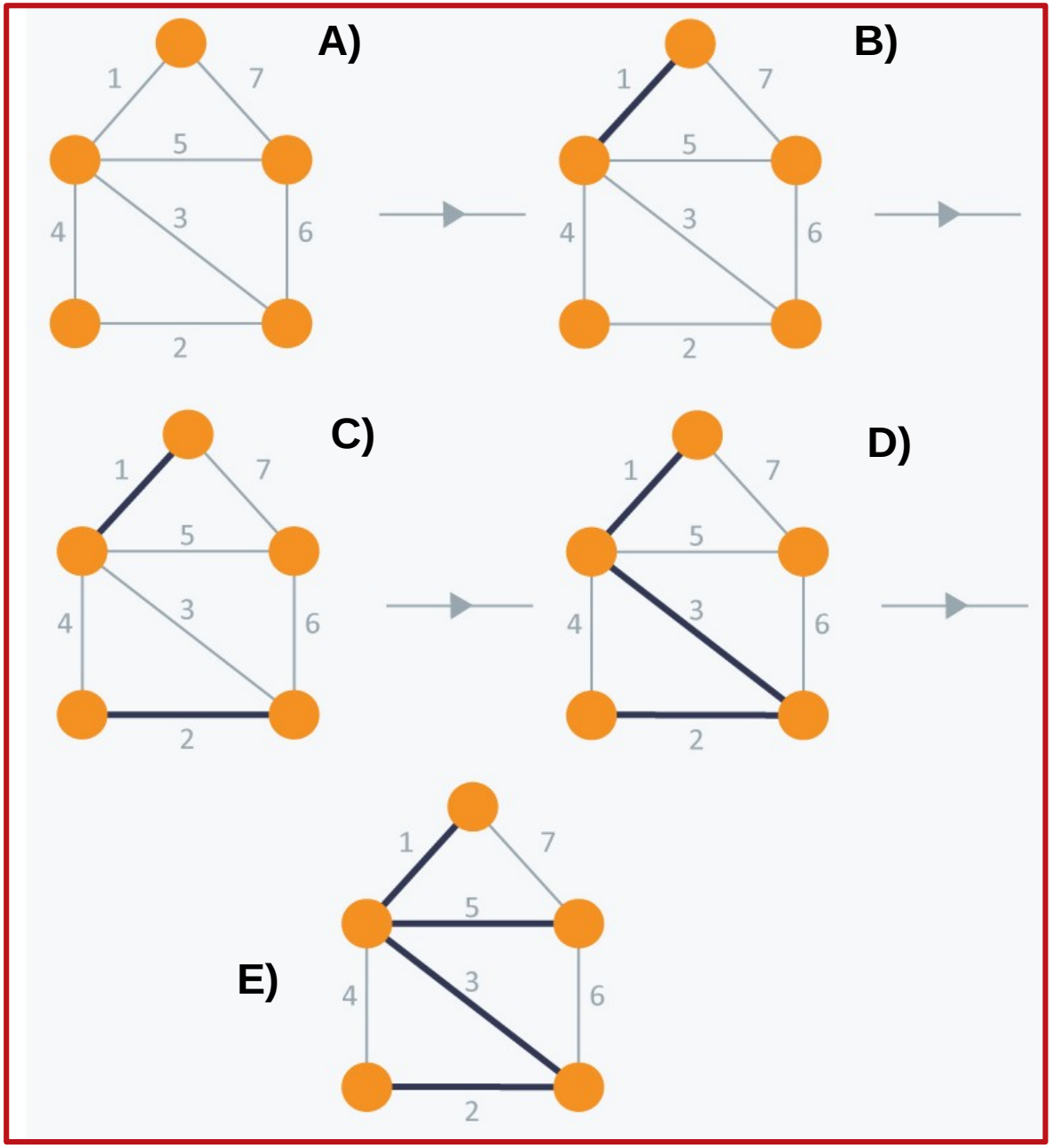
Joseph Kruskal
1928-2010

En **1956**, Joseph KRUSKAL (1928-2010) met au point un algorithme de recherche de l'arbre couvrant de poids minimal dans un graphe : c'est l'**algorithme de KRUSKAL**.

Robert C. Prim
1921-2021

En **1957**, Robert C. Prim, collègue de J. Kruskal, met au point un *autre* algorithme ayant le même objectif : c'est l'**algorithme de PRIM**.
Cet algorithme fut redécouvert en **1959** par Edsger Dijkstra.

Exemple d'application de l'algorithme de KRUSKAL



Au départ, les arbres sont :

A) les sommets distincts (1 arbre = 1 sommet)

… puis

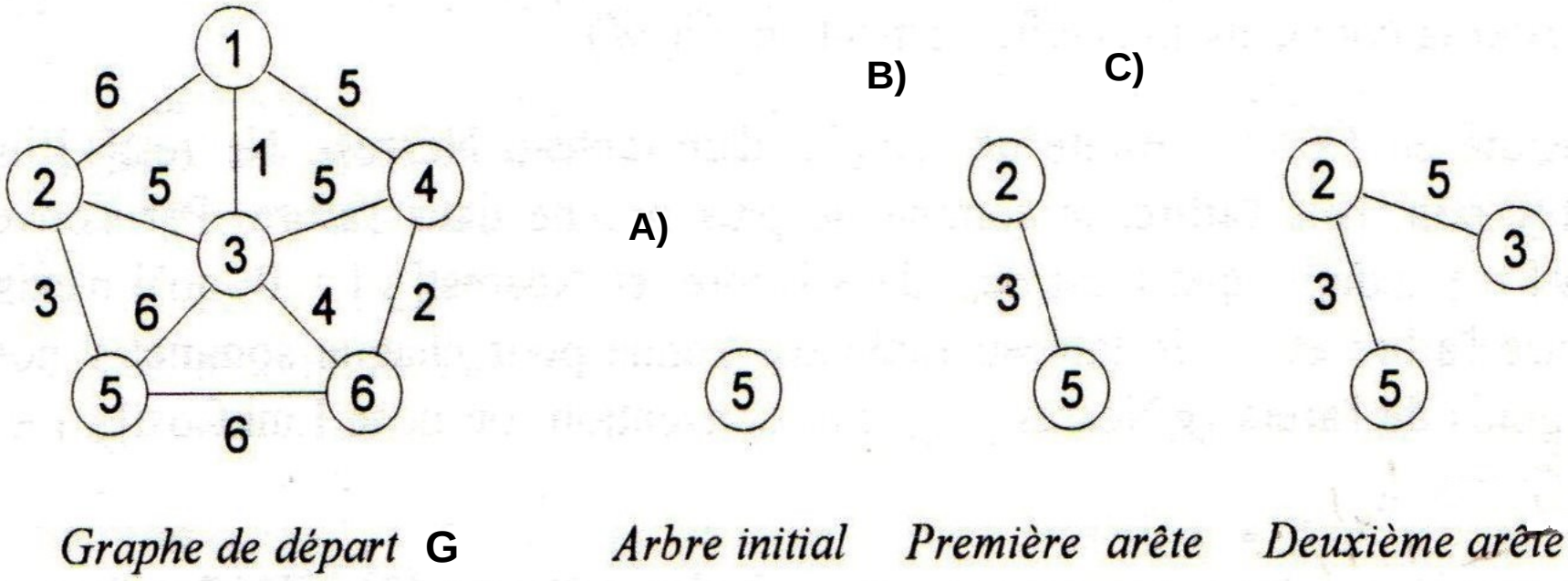
B) et **C)** les arbres formés des sommets reliés par des arcs de poids minimal,

… puis

D) les arbres formés en reliant les arbres précédents, toujours par des arcs de poids minimal et en excluant les cycles.

E) Dans l'arbre couvrant final, certains arcs du graphe de départ ne seront pas retenus.

Exemple d'application de l'**algorithme de PRIM** (début) (*)



A) Dans G, on choisit un sommet qui sera le *départ* de l'arbre couvrant : **5**

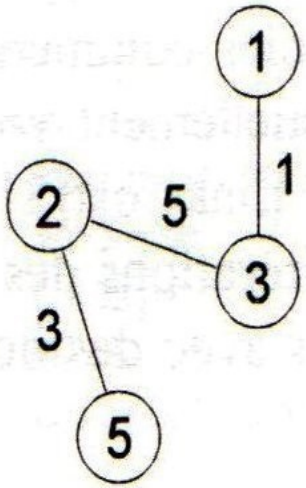
B) On relie ce sommet à l'un de ses sommets *adjacents* par une arête de poids minimal ; on obtient : **{**5,**2}**,

C) Du sommet **2**, on choisit l'arête de poids minimal jusqu'au sommet **3**, et l'on obtient : **{**5,2**,3}**,

(*) D'après Prins Christian, Algorithmes de graphes

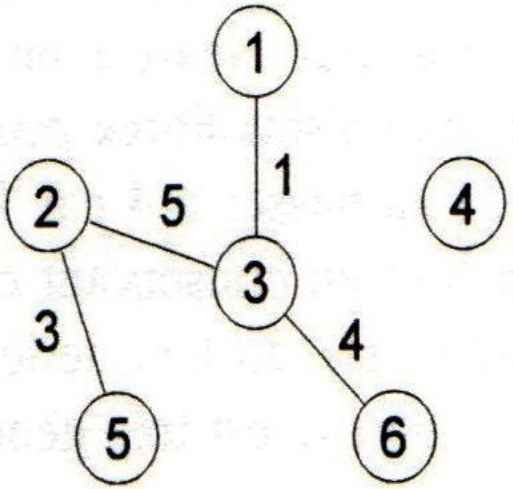
Exemple d'application de l'**algorithme de PRIM** (suite)

D)



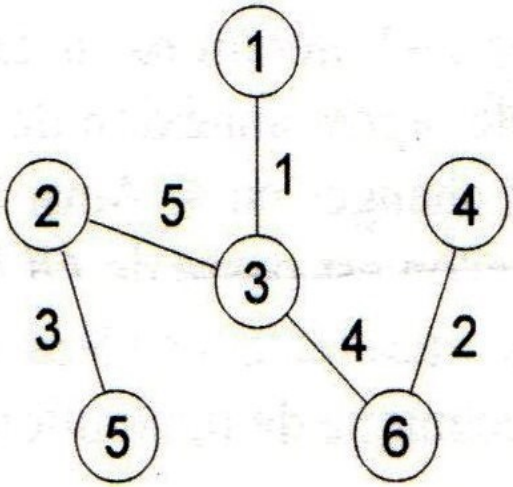
Troisième arête

E)



Quatrième arête

F)



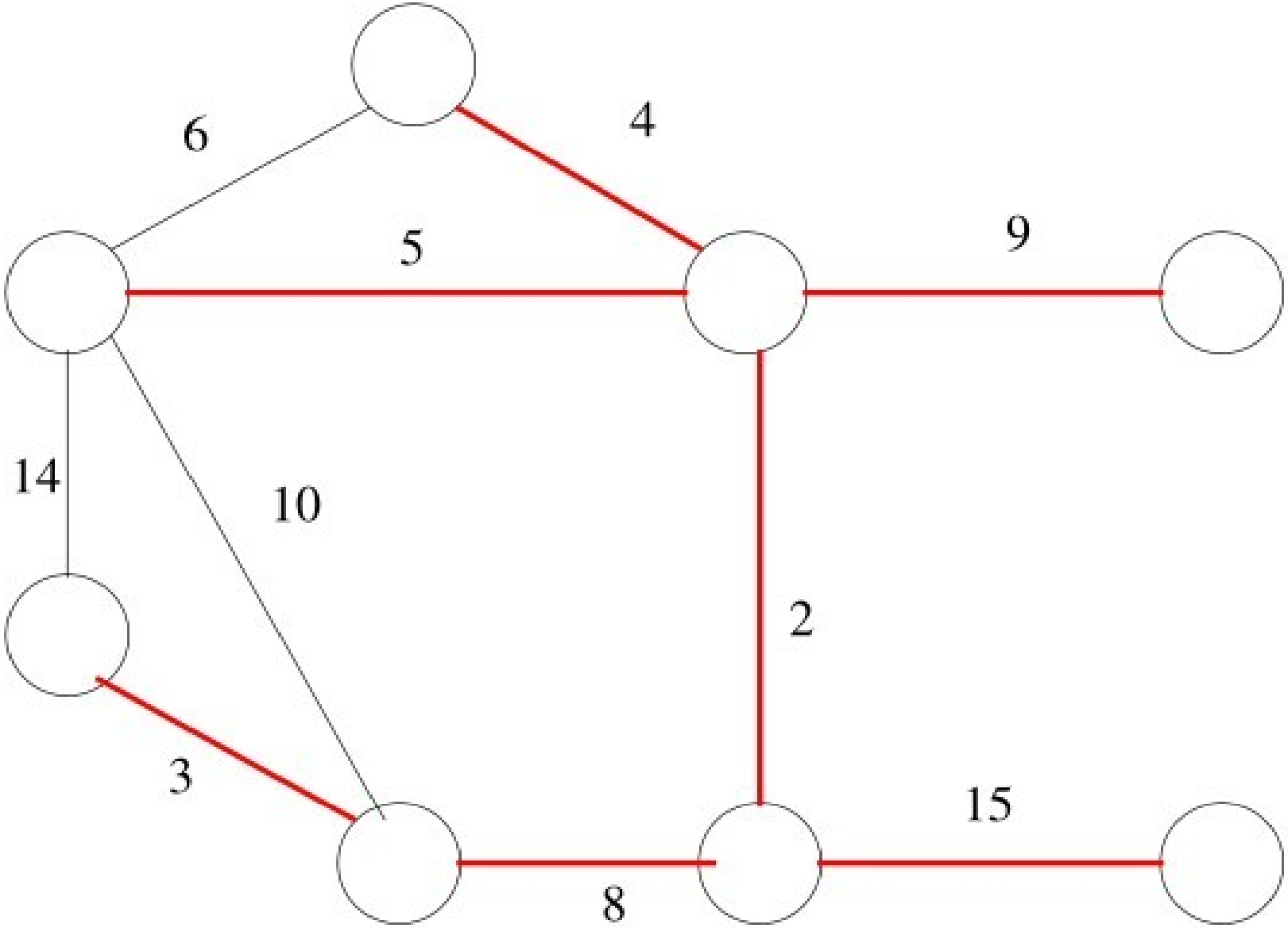
Cinquième arête

D) à **F)** D'un sommet de l'arbre de l'étape précédente, on choisit l'arête de poids minimal jusqu'à un sommet qui n'a pas été encore relié :

{5,2,3,**1}**, **{**5,2,3,1,**6}**, **{**5,2,3,1,6,**4}**.

Dans l'arbre couvrant final, certains arcs du graphe de départ ne seront pas retenus.

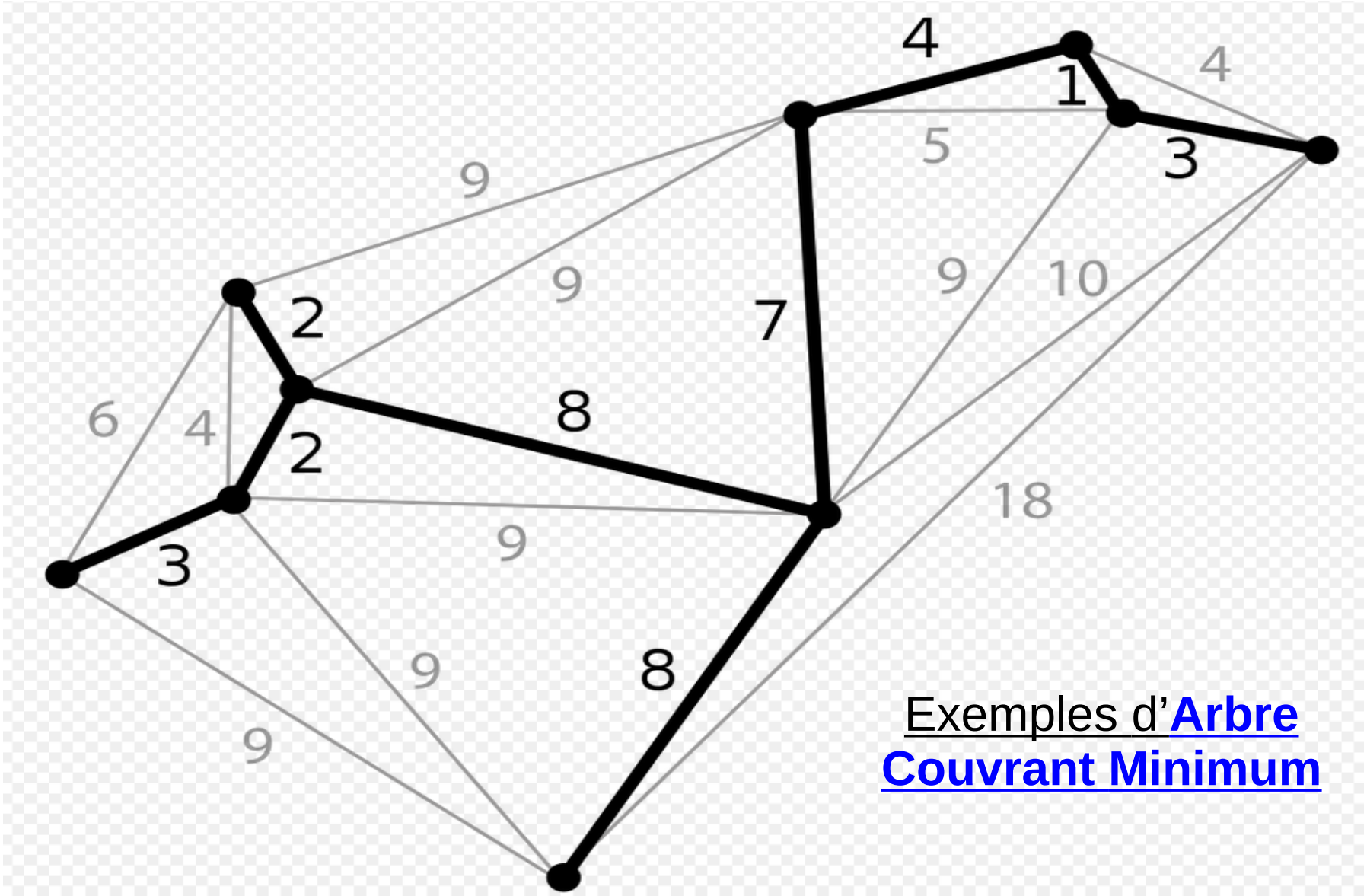
► Un exemple d'Arbre Couvrant Minimum



Voir par exemple : https://fr.wikipedia.org/wiki/Arbre_couvrant_de_poids_minimal

Voir par exemple : https://fr.wikipedia.org/wiki/Arbre_couvrant_de_poids_minimal

… Ou bien : https://fr.wikipedia.org/wiki/Arbre_couvrant_de_poids_minimal#/media/Fichier:Minimum_spanning_tree.svg



Exemples d'**Arbre Couvrant Minimum**

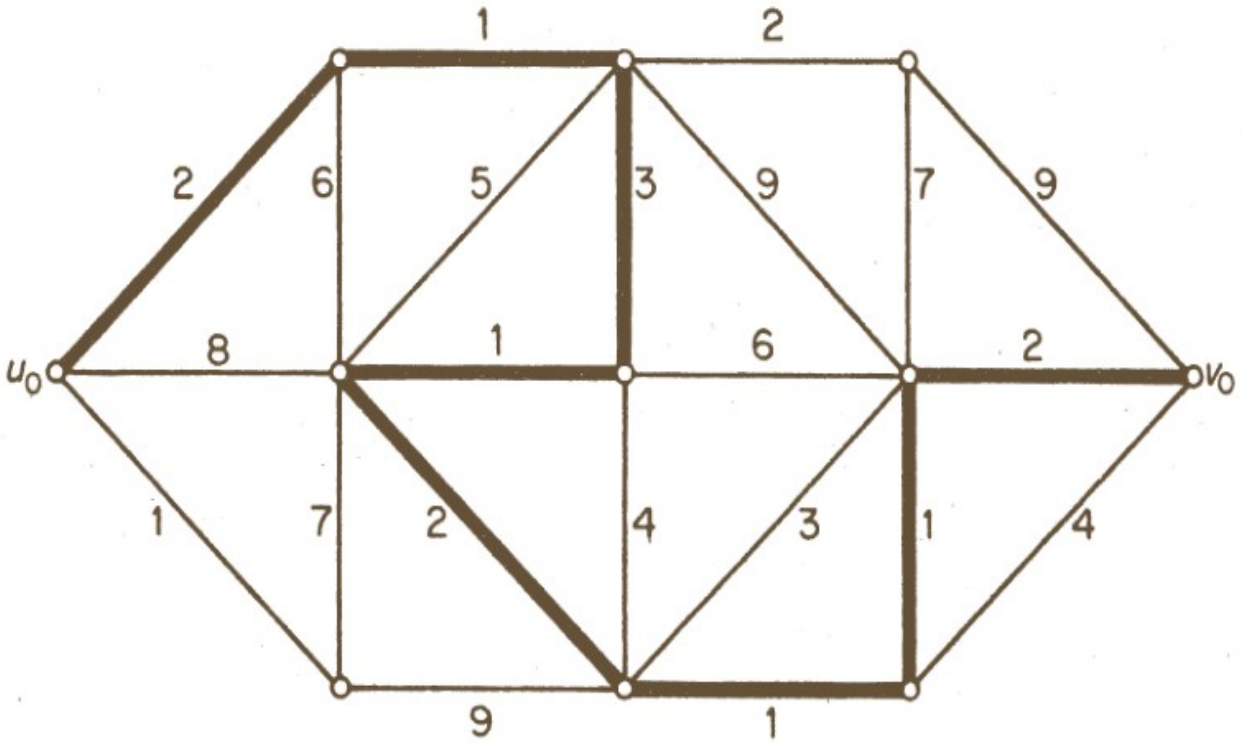
2-4

Le Problème du PLUS COURT CHEMIN

L'ALGORITHME de DIJKSTRA du PLUS COURT CHEMIN

Parmi les problèmes célèbres de la théorie des graphes, citons celui de la détermination du **plus court chemin**.

En **1959** est publié l'**algorithme de Dijkstra**, qui mesure le plus court chemin entre les sommets s_0 et s_n d'un graphe *valué*.

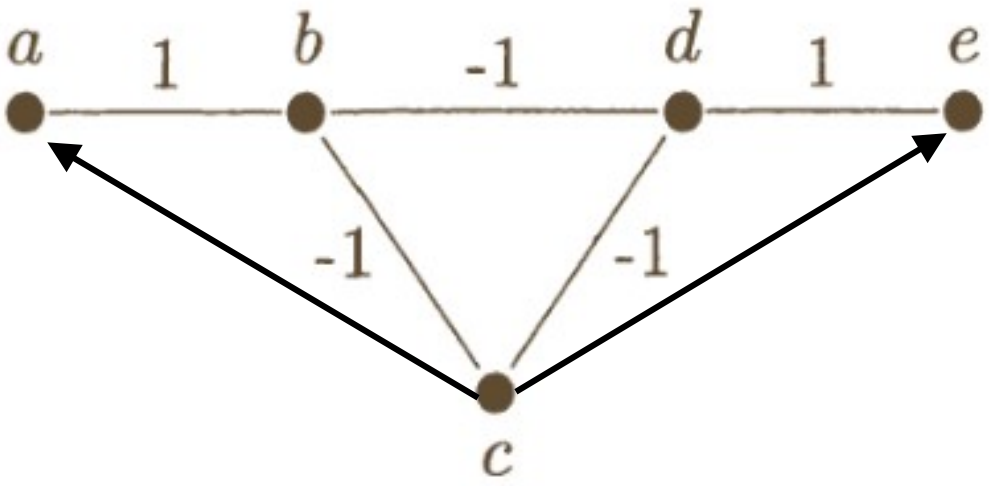


Le **plus court chemin** dans un graphe valué (Bondy JA, Murphy USR, *Graph theory with applications*)

ATTENTION ! L'**algorithme de Dijkstra** est correct <u>uniquement dans le cas où la valuation est *strictement positive* !</u>

Il ne l'est pas généralement si les valuations sont négatives, comme le montre le graphe **G** ci-dessous !

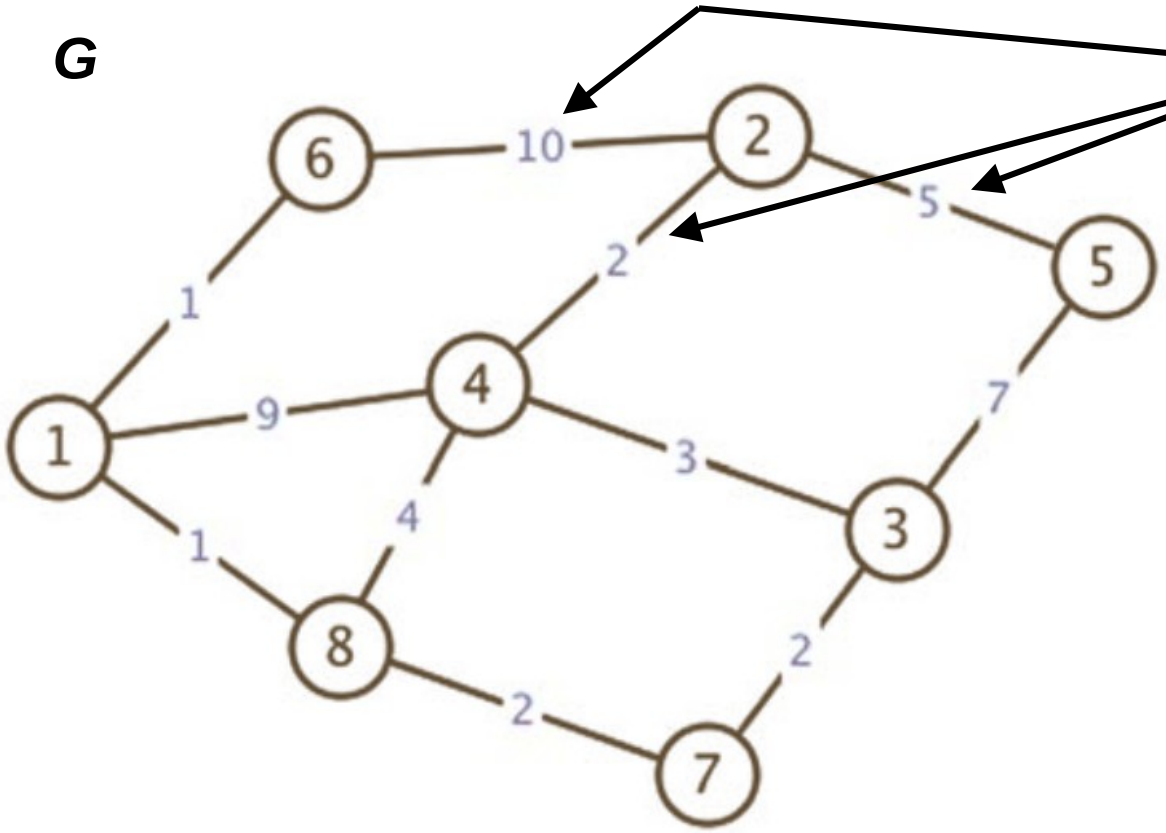
G



Quelle est la distance du sommet **a** au sommet **e** dans le graphe **G** ?

Edsger Dijkstra
(1930-2002)

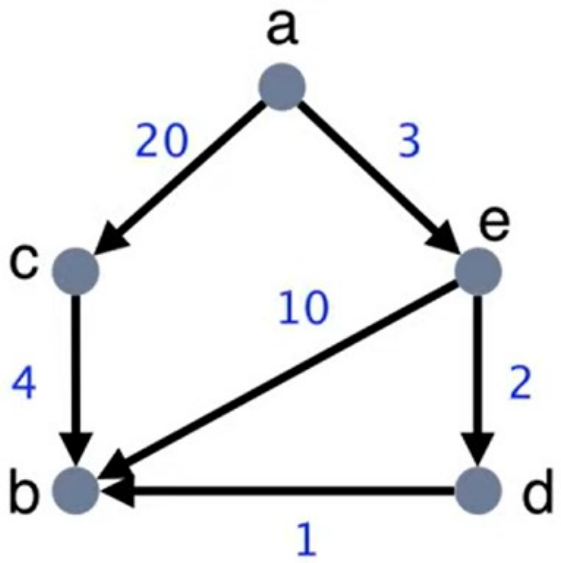
► Les *distances*, entre les sommets x de **G** et **s** , notées D[**s**,x] , sont appelées ***distances provisoires*** ou ***estimées***, parce qu'elles sont amenées à être mises à jour pendant le déroulement de l'algorithme ...



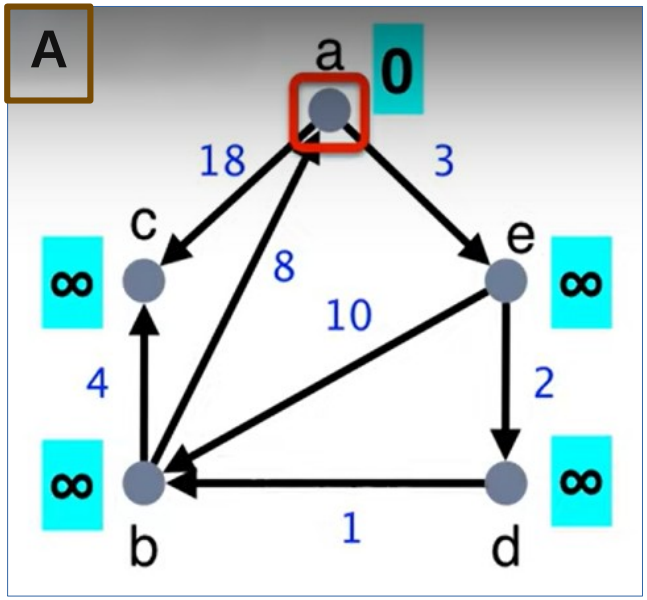
1er Exemple de Graphe Valué (Laforest Christian, ''A la découverte des graphes'')

► Un sommet de départ **s** du graphe **G** ayant été choisi, l'algorithme va examiner *tous les autres* sommets du graphe ...

► 2ème Exemple : Déroulement de l'Algorithme de Dijkstra (*)

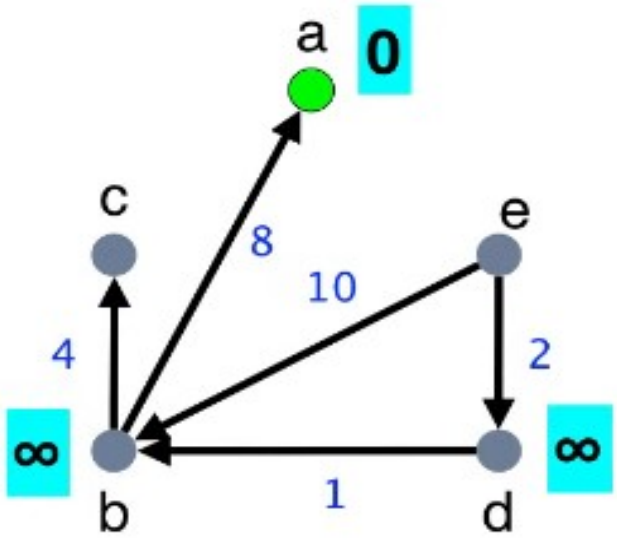


Au départ, tous les sommets du graphe ont un poids infini, ∞, sauf le sommet de départ, qui a le poids **0**.



On part du sommet **a** (poids = 0).

On indique que le sommet **a** est *traité* en le peignant en **vert**.

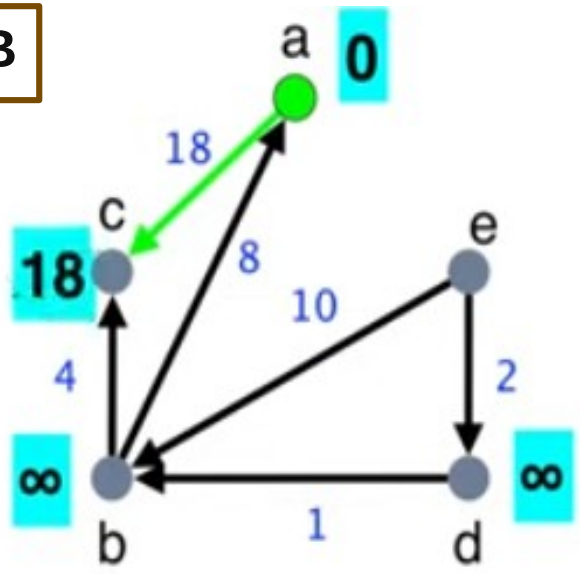


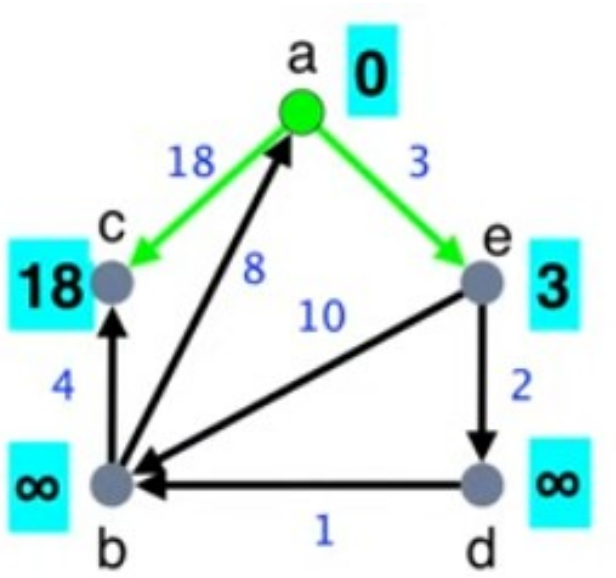
On indique que **a** a été *traité* en le marquant en **vert**

(*) D'après d'après Laforest Christian,

A partir du sommet **a**, on recherche les chemins possibles : on modifie les étiquettes des sommets **c** et **e** et l'on peint en **vert** les arêtes (**a**,c) et (**a**,e).

B

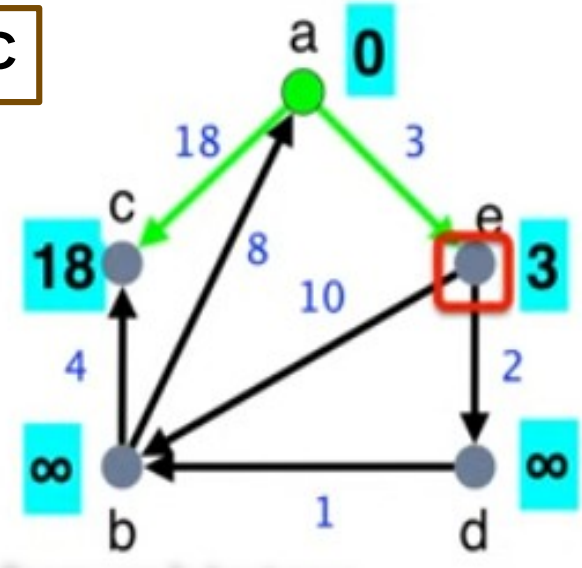


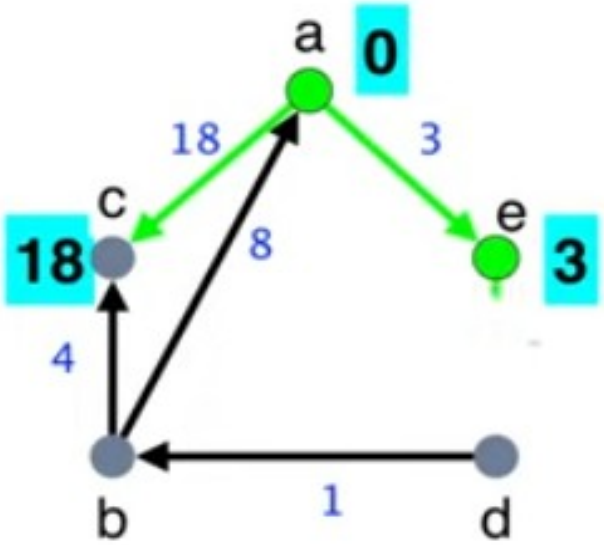


Fin du traitement du sommet **a**.

On recherche le sommet situé à la plus petite distance en partant de **a** …

C

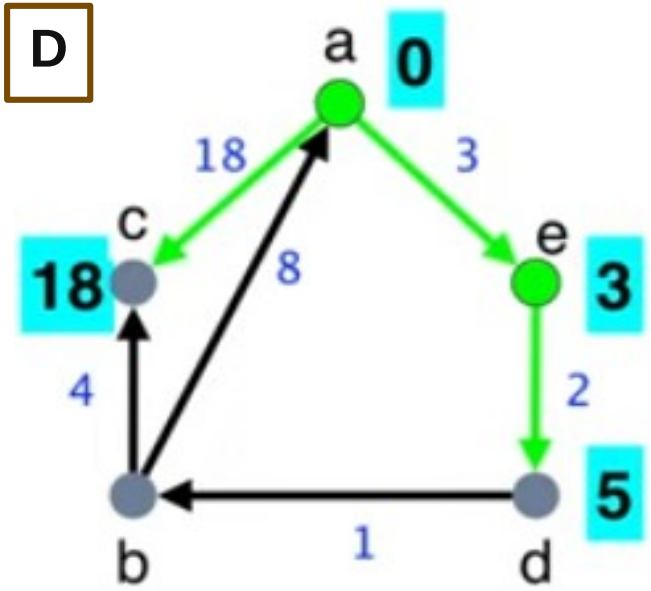


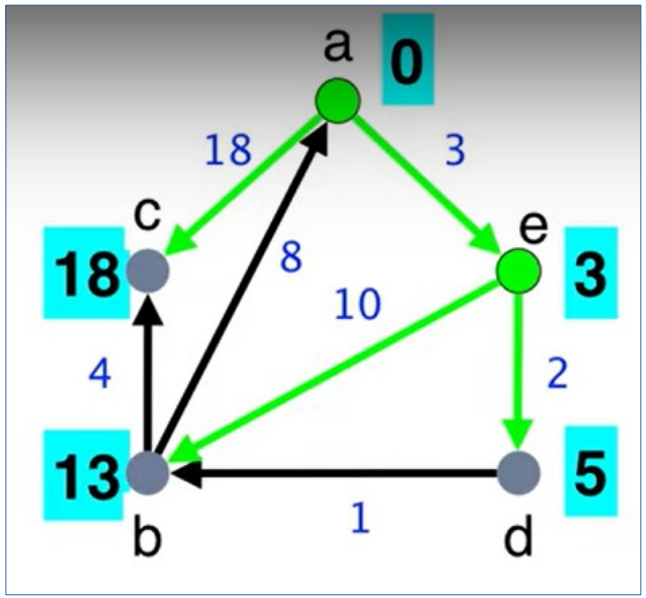


C'est **e**, que l'on peint en **vert**.

A partir du sommet **e**, on recherche les chemins possibles : on modifie les étiquettes des sommets **c** et **e** et l'on peint en **vert** les arêtes (**e**,d) et (**e**,b).

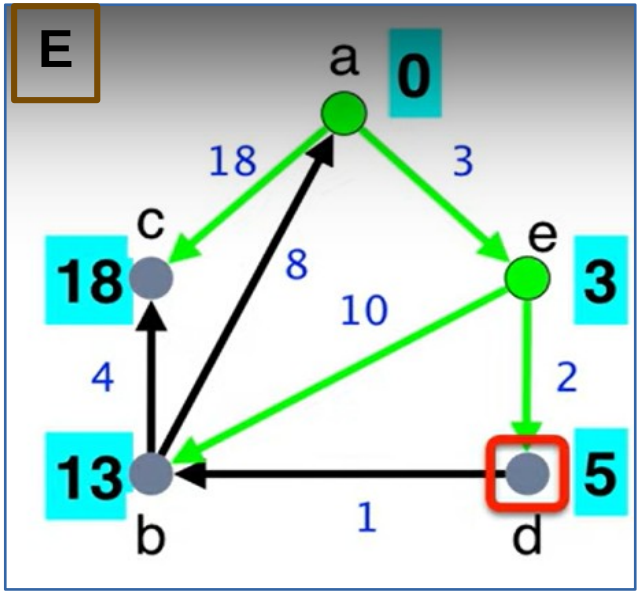
D

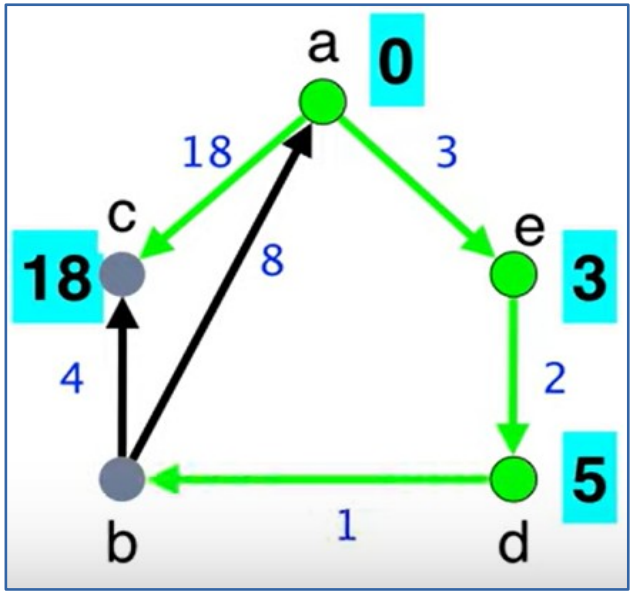


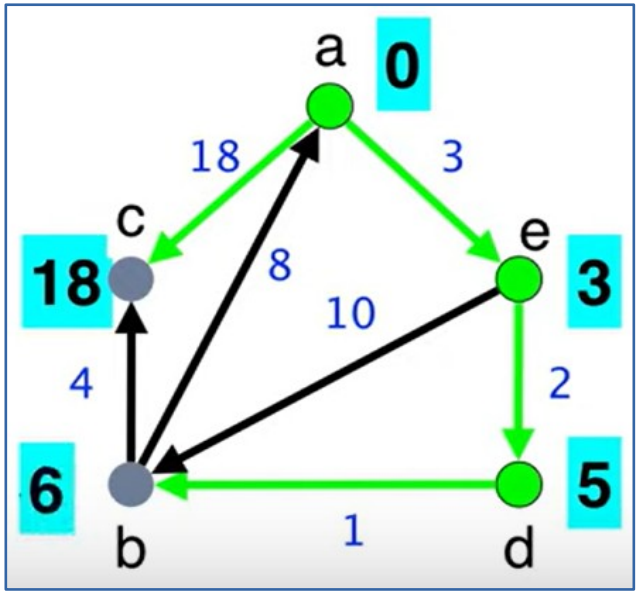


Fin du traitement du sommet **e**.

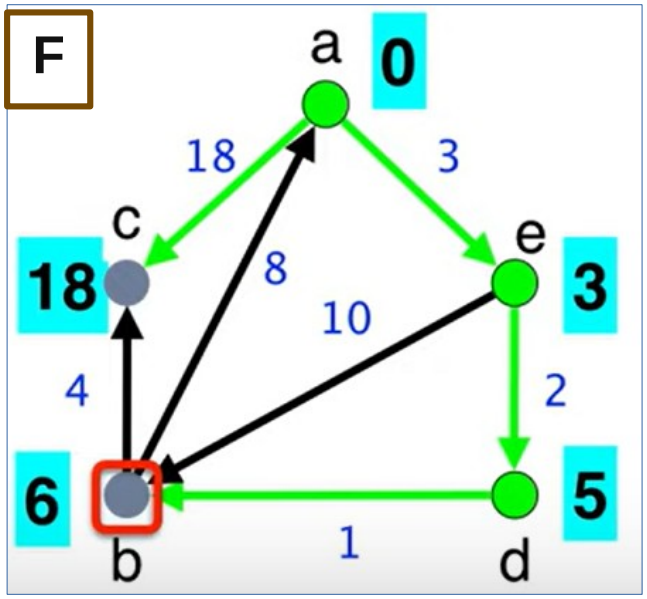
E

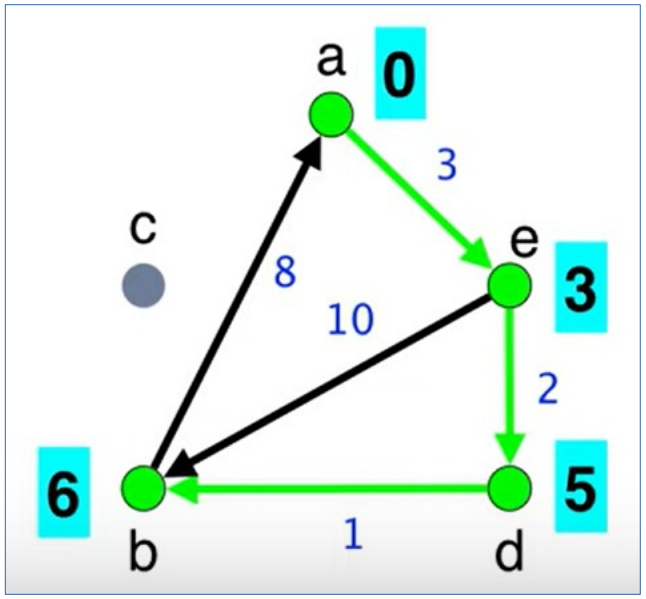




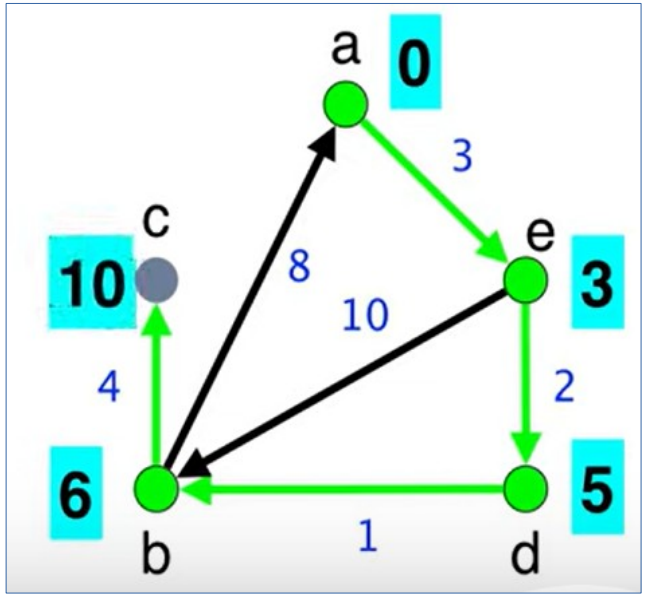


Le prochain sommet à partir de **d** est le sommet **b**, à une distance de **6** de **a** :





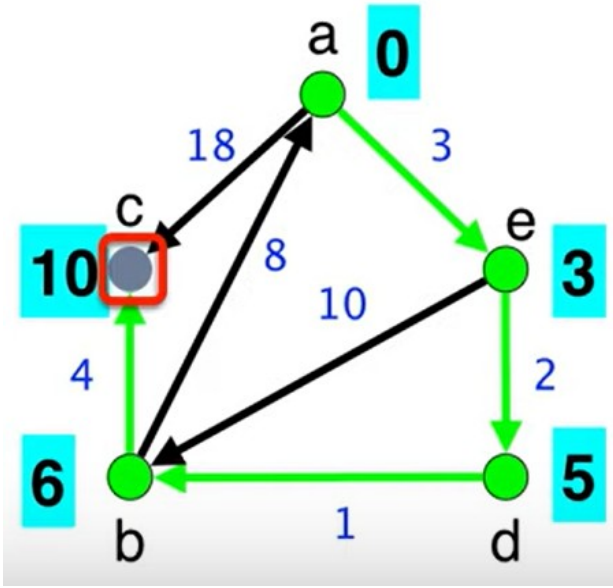
Le chemin de **b** vers **a** ne nous donne pas de parcours plus réduit ; donc, on le néglige. Au contraire du chemin de **b** à **c** nous permet de modifier l'étiquette de **c**, de **18** en **10**.

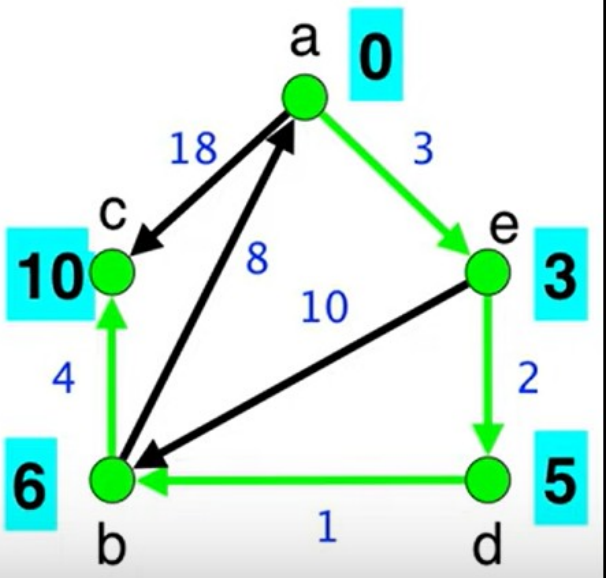


On <u>désélectionne</u> l'arc (a,c) que nous avions retenu à l'étape **B**

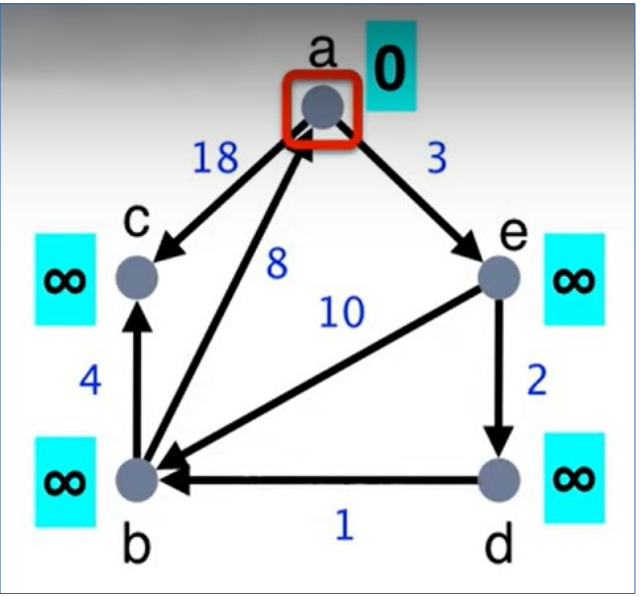
Il reste à traiter le *dernier* sommet, **c**.
Mais celui-ci n'a pas d'arc sortant (= de sommet descendant)

L'algorithme est donc terminé !

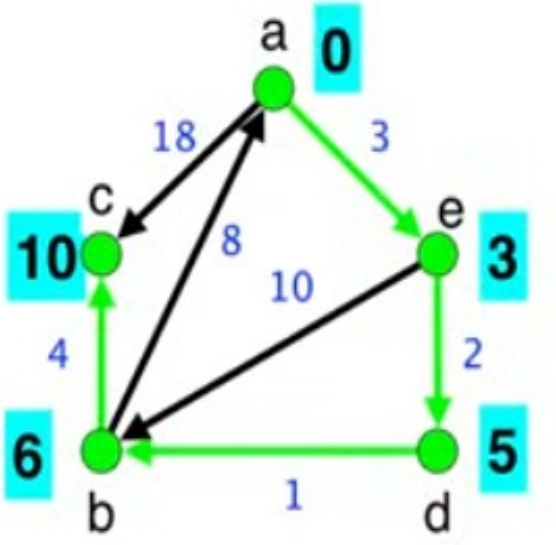




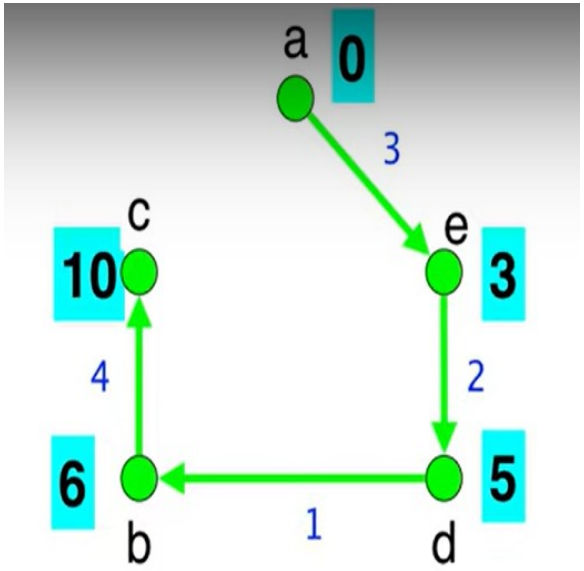
Du graphe **G** de départ ci-dessous, on a donc obtenu l'arbre final ...



dans lequel les étiquettes et les arcs de départ ont été modifiés dynamiquement …



pour donner l'arbre *minimal* ci-dessous :



Algorithme de DIJKSTRA du **plus court chemin** dans un graphe valué (fin)

L'algorithme de Dijkstra fait partie de l'**algorithmique des Chemins Optimaux**, qui traite de problèmes de ... cheminement, qui impliquent de **minimiser** ou de **maximiser** une fonction donnée.

Cas classique : traiter la longueur ou la *valeur* des arcs.
Exemple particulier : l'algorithme de Dijkstra est utilisé pour le **routage IP** (Open Shortest Path First)

D'autres algorithmes calculent aussi **le plus court chemin**.
Signalons pour ceux qui sont intéressés par l'informatique :

- l'**algorithme de Bellmam-Ford** :

https://fr.wikipedia.org/wiki/Algorithme_de_Bellman-Ford

qui s'applique aux graphes valués sans circuit ; il donne le plus court chemin d'un sommet **s** à tous les autres sommets, ainsi que son coût.

Et aussi :

- l'**algorithme de Floyd-Warshall** :

https://fr.wikipedia.org/wiki/Algorithme_de_Floyd-Warshall

qui calcule le plus court chemin entre tous les sommets du graphe à partir de la matrice d'adjacence.

Ou encore :

- l'**algorithme de Dantzig** :

https://fr.wikipedia.org/wiki/Algorithme_du_simplexe

qui repose lui aussi sur une méthode matricielle.

2-5 *Promenade* parmi divers Problèmes de Chemins et de Circuits

Le Problème du VENDEUR AMBULANT : Circuits HAMILTONIENS

Imaginons un service de livraison qui doit distribuer des colis dans **8** différentes villes (les *sommets* d'un graphe G), toutes reliées par des routes (les *arêtes* de G) : il s'agit donc ici d'un **graphe complet** ayant un total de :

8x(**8**-1)/2 = **28** arêtes.

Le camion de livraison doit visiter **toutes** les villes, dans n'importe quel ordre, et il doit les visiter **une et une seule fois** !

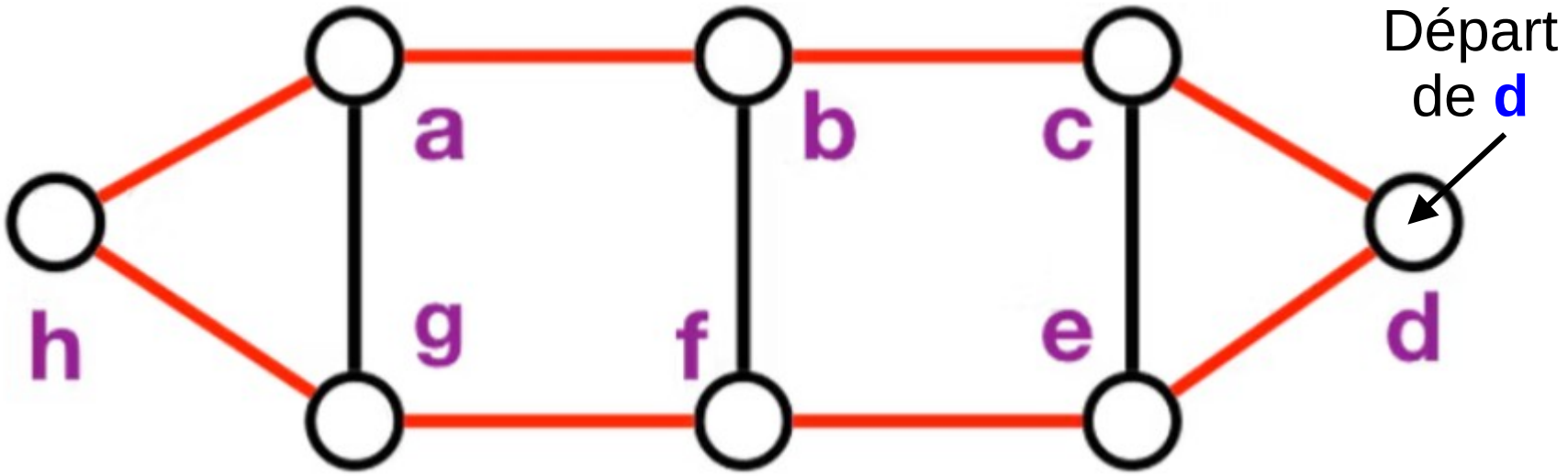
Enoncé formel du Problème :

Le problème posé en pratique revient à trouver dans **G** :

1 - soit un **chemin hamiltonien** passant **une seule fois** par *chaque* ***sommet*** (le camion ne revient pas au sommet **d**) ;

2 - soit un **cycle hamiltonien** : le chemin hamiltonien trouvé au cas 1 revient au sommet **d**.

G



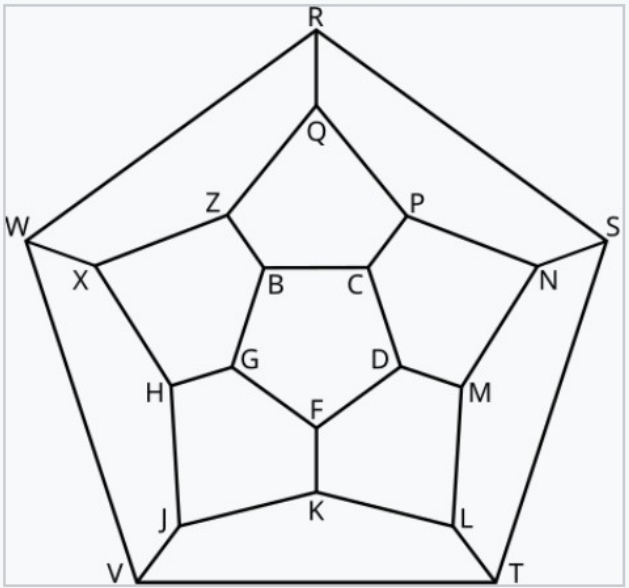
Les arêtes en **rouge** forment un cycle hamiltonien dans **G** :

L’invention par R. Hamilton du jeu appelé **icosian game** a donné lieu à de nombreux développements…

Voir :

https://fr.wikipedia.org/wiki/Graphe_hamiltonien

https://en.wikipedia.org/wiki/Icosian_game



Plateau de jeu pour le *icosian game*.

Le Problème du **POSTIER CHINOIS**

Ce problème *modélise* la tournée d'un facteur devant effectuer le plus efficacement possible sa tournée en passant au moins une fois par chaque *rue* de son secteur. Le graphe correspondant est *connexe*, *non orienté* et *passe au moins une fois par chaque arête* avant de revenir à son point de départ.

Ce problème est lié à la notion de **cycle eulérien** :

https://fr.wikipedia.org/wiki/Graphe_eul%C3%A9rien

Voir aussi notamment :

https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_postier_chinois

https://pagesperso.g-scop.grenoble-inp.fr/~stehlikm/teaching/l3miage/CM3-4.pdf

Le Problème du VOYAGEUR de COMMERCE

Un exemple simple du problème …

Lors de sa tournée commerciale, un voyageur de commerce doit se rendre de sa ville de résidence (de départ) dans **n** autres villes voisines. Il connaît la distance la plus courte en kilomètres qui sépare chaque paire de villes.

Pour **n** = 3, on aura par exemple le tableau suivant des villes et des distances, Paris étant la ville de départ :

	Paris-P	Ville-A	Ville-B	Ville-C
Paris-P	0	54	17	79
Ville-A	54	0	49	104
Ville-B	17	49	0	91
Ville-C	79	109	91	0

La distance entre deux villes, appelées P, A, B et C, pour simplifier, est la même dans les deux sens, sauf pour les trajets Ville-A → Ville-C (104 km) et Ville-C → Ville-A (109 km), qui diffèrent de 5 km

Notre voyageur veut ”**minimiser**” la distance à parcourir sachant que :

- il part de Paris,
- il y revient pour terminer sa tournée,
- il se rend une fois et une seule dans chacune des trois villes.

Il faut donc calculer la distance totale en kilomètres qui correspond à *chacun* des trajets possibles entre les villes, et déterminer ainsi la plus petite distance.

La première question est donc :

combien y-a-t-il de trajets *distincts* dans lesquels P est la ville de départ et d'arrivée et A, B et C sont "visitées" une fois et une seule ?

Comme nous avons **n**=3 villes, nous raisonnons comme suit :

- la ville de départ est **P**,
- pour le choix de la 1ère ville-étape, il y a **n**=3 possibilités,
- pour le choix de la 2ème ville-étape, il ne reste plus que **n−1**=2 villes possibles,
- pour le choix de la 3ème ville-étape (la dernière), il ne reste alors plus qu'une seule ville possible (**n−2**=1),
- enfin, la ville d'arrivée est à nouveau **P**.

Exemple :

Si l'on choisit les villes A, B et C dans cet ordre, le trajet sera : **P-A-B-C-P** ; si l'on choisit les villes-étapes dans l'ordre B, A , C, le trajet sera : **P-B-A-C-P**, etc.

On voit qu'un circuit est équivalent à une **permutation** des éléments de l'ensemble des villes-étapes V={A, B, C} ; donc le nombre de trajets est égal à **n! = 3! = 3 × 2 × 1 = 6** .

Pour chacun des **n!** = **6** trajets, on fait la somme des distances en kilomètres, ce qui donne le tableau suivant :

Trajet	Kilométrage du circuit
P-A-B-C-P	$54 + 49 + 91 + 79 = 273$
P-A-C-B-P	$54 + 104 + 91 + 17 = 266$
P-B-C-A-P	$17 + 91 + 109 + 54 = 271$
P-B-A-C-P	$17 + 49 + 104 + 79 = \boxed{249}$
P-C-A-B-P	$79 + 109 + 49 + 17 = 254$
P-C-B-A-P	$79 + 91 + 49 + 54 = 273$

Manifestement l'itinéraire le plus court est **P-B-A-C-P**, avec **249** km au total.

Comme la plupart des exemples simples, celui-ci n'est pas très réaliste !

► Dans le cas de **n** = 10 villes, le nombre de trajets possibles vaut :

n! = **10!** = 10 × 9 × 8 × 7 × 6 × 5 × 4 × 3 × 2 × 1 = **3628800**

ce qui fait beaucoup trop de trajets à calculer *sans l'aide d'un ordinateur* (à raison d'1 minute pour calculer "à la main" le kilométrage d'un des **3628800** trajets, il faudrait plus de 20 ans !)

► En ajoutant **une** ville, le nombre de chemins possibles devient :

n! = 11! = 11 × 10 × 9 × 8 × 7 × 6 × 5 × 4 × 3 × 2 × 1 = **39916800**

► Avec un circuit de **n**=25 villes-étapes, le nombre de circuits est égal à :

n! = 25! = 25 × 24 × . . . × 10 × . . . × 3 × 2 × 1 =

15511210043330985984000000

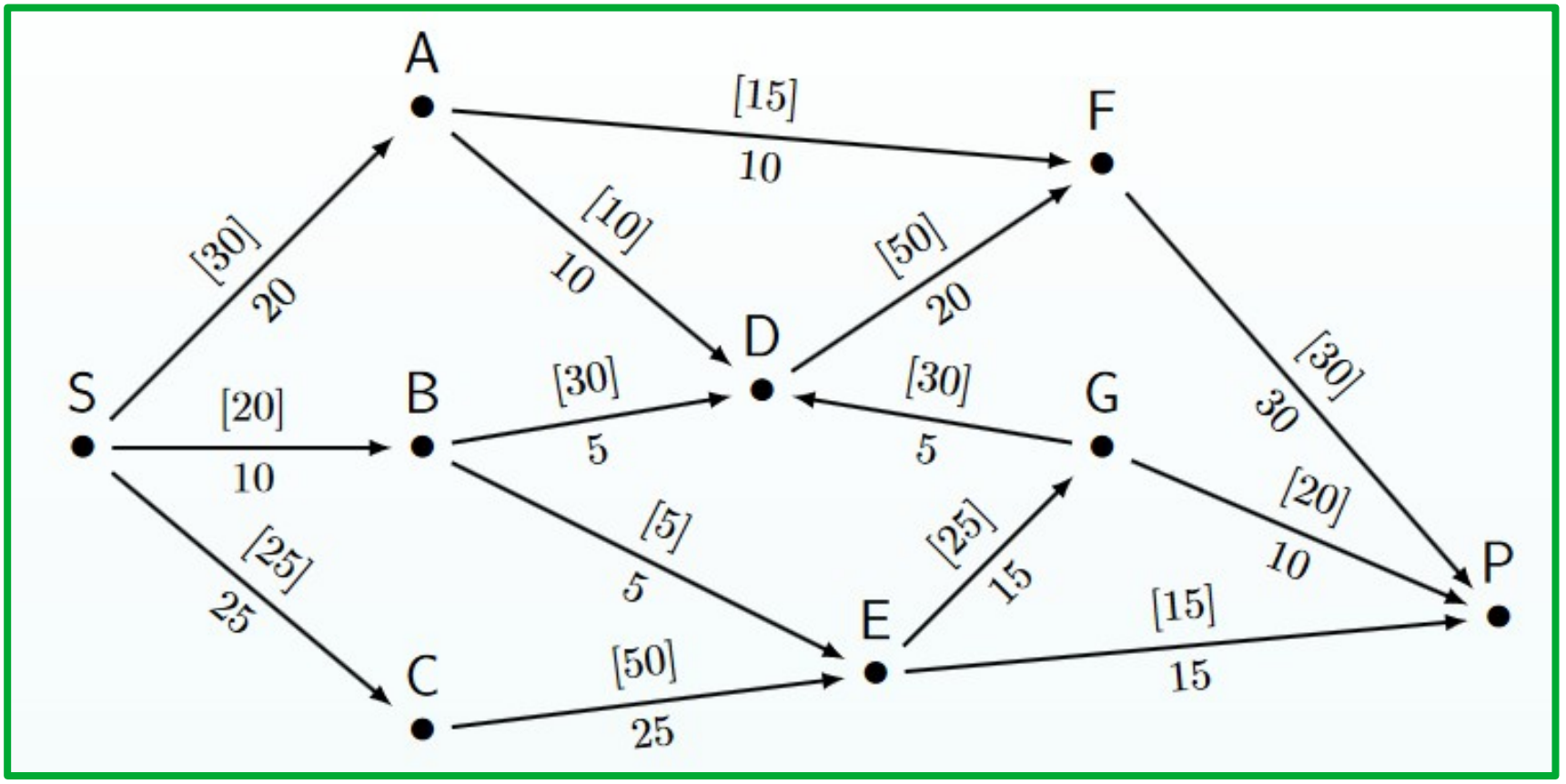
► Le problème du voyageur de commerce fait partie des problèmes dits "**NP**", qui encore aujourd'hui ne sont pas résolus !

Problèmes de FLOT - Réseaux de TRANSPORT

Les problèmes de **flot** consistent à *modéliser* et *optimiser* un réseau de transport et les contraintes qui y sont liées :

- réseau routier
- réseau électrique
- réseau de distribution d'eau, etc.

Un **réseau de transport** est un graphe **G** de **n** sommets *sans boucle*, *orienté* et *valué* (chaque arc **u** a une valuation c(u) appelée *capacité*), comprenant deux sommets x_1 (appelé *source*) et x_n (appelé *puits*) ; **G** est tel que, pour tout sommet x_k, il existe au moins un chemin allant de x_1 vers x_n et passant par x_k .



Exemple de Réseau de Flot

L'**algorithme de Ford-Fulkerson**, entre autres, permet de traiter ce type de graphe … Voir notamment :

https://fr.wikipedia.org/wiki/R%C3%A9seau_de_flot

https://fr.wikipedia.org/wiki/Algorithme_de_Ford-Fulkerson

3 BREF APERÇU de QUELQUES CONJECTURES …

(Un TRÈS bref aperçu !)

► En théorie des graphes, il existe de très nombreux **problèmes** et **conjectures**.

► Beaucoup de ceux-ci sont célèbres … Pour en savoir plus, un petit tour sur notre encyclopédie en ligne serait un bon commencement …

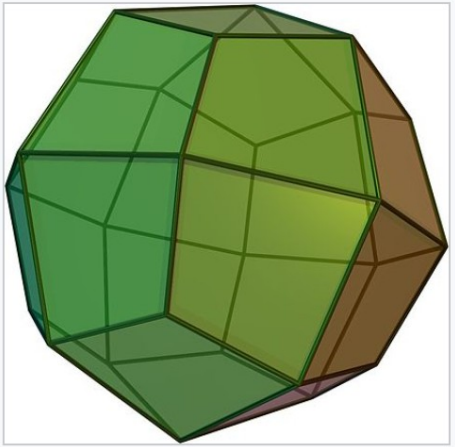
► On y distingue les **conjectures démontrées** (les *théorèmes*) des **conjectures fausses** ...

En guise d'invite à une *promenade* dans le domaine, voici quelques exemples.

3-1 Les CONJECTURES *FAUSSES*

La conjecture de HIRSCH

La **conjecture de HIRSCH**, liée à l'**algorithme du simplexe**, affirmait que deux sommets d'un *polytope* de dimension **d** ayant **n** faces pouvaient toujours être reliés par un **chemin** formé d'au plus **n – d** arêtes.



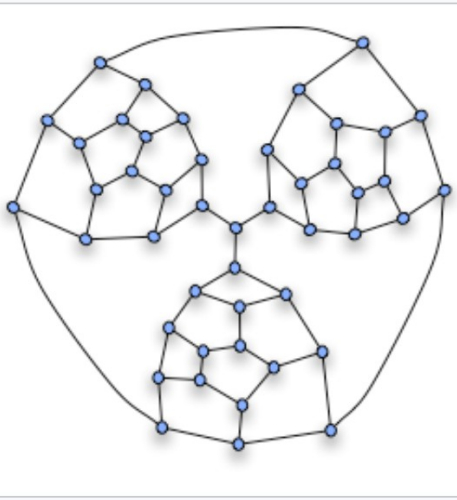
Un polytope en dimension 3.

Voir : https://fr.wikipedia.org/wiki/Conjecture_de_Hirsch

https://duckduckgo.com/?t=ffab&q=algorithme+du+simplexe&ia=web

La conjecture de TAIT

La **conjecture de TAIT** affirmait que tout *graphe cubique planaire 3-connexe* possède un **cycle hamiltonien**.



Graphe de Tutte

Cette conjecture a été un moment liée au **problème des quatre couleurs** et à la démonstration du théorème correspondant...

Voir : https://fr.wikipedia.org/wiki/Conjecture_de_Tait

3-2 Les CONJECTURES DÉMONTRÉES

Le Théorème des GRAPHES PARFAITS

Ce théorème (difficile) énonce que :

Un graphe est **parfait** si, et seulement si, ni lui, ni son complémentaire, ne contient de *sous-graphe induit* qui soit un cycle d'ordre impair ≥ 5.

Voir (pour les courageux) :

https://fr.wikipedia.org/wiki/Sous-graphe

https://fr.wikipedia.org/wiki/Graphe_parfait

https://fr.wikipedia.org/wiki/Coloration_de_graphe#D%C3%A9finition_(nombre_chromatique)

https://fr.wikipedia.org/wiki/Th%C3%A9or%C3%A8me_des_graphes_parfaits

Le Théorème de RINGEL–YOUNGS

Ce théorème (difficile), anciennement appelé **Conjecture de Heawood**, est lié à la *coloration* des *surfaces*, à la notion de *genre*, au *nombre chromatique*, et, en particulier au *théorème des 4 couleurs*.

Il donne un *minorant* pour le nombre de couleurs nécessaires pour colorer une *surface* de genre donné.

Voir (pour les courageux) :

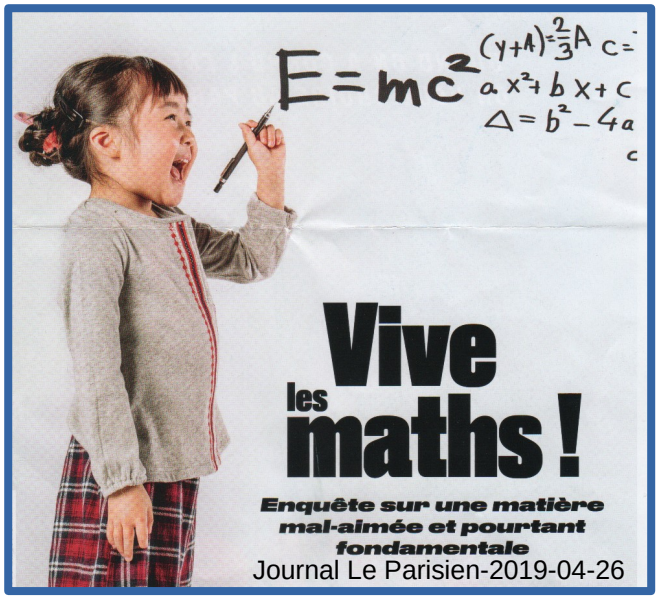
https://fr.wikipedia.org/wiki/Conjecture_de_Heawood

https://fr.wikipedia.org/wiki/Genre_(math%C3%A9matiques)

https://fr.wikipedia.org/wiki/Coloration_de_graphe#D%C3%A9finition_(nombre_chromatique

)

VOILA ! C'est FINI !



Journal Le Parisien-2019-04-26

Mais il reste beaucoup à voir ...

Les **quelques références** données dans l'exposé devraient vous occuper pendant les prochaines longues soirées d'hiver !!

COMPTER les CHEMINS et PROBLÈMES de PARCOURS dans les GRAPHES

K A F E M A T H

BIBLIOGRAPHIE

K A F E M A T H

C J U P

(**C**haque **J**our **U**n **P**eu)

14/11/2024

14 novembre 2024

MAJ : 16/11/2024

BIBLIOGRAPHIE :
Quelques Références sur les GRAPHES

- **Alavi Y**, Graph theory and applications
- **Aldous Joan**, Graphs and Applications--An Introductory Approach (Excellente introduction à la théorie, pour ceux qui lisent l'anglais)
- **Atallah Mickhail**, Algorithms and Theory of Computation Handbook (2 vol)
- **Bang-Jensen Jorgen**, Digraphs--Theory,Algorithms and Applications
- **Beutelspacher Albrecht**, Discrete Mathematik für Einsteiger
- **Biswal Purna Chandra**, Discrete mathematics and graph theory
- **Bondy J.A**., Graph Theory with Applications
- **Bondy J.A**., Graph Theory
- **Bories-Longuet Francette, Ramirez Alfonsfn Jorge,** Graphes et combinatoire-Cours avec 210 exercices corrigés (La version *lourde* du précédent, pour les courageux qui veulent aller plus loin)
- **Busser Alain**, Jeux et graphes-La théorie des graphes de 5 à 95 ans
- **Chaumartin François**, Chemin minimal (Pascalissime)
- **Cogis Olivier**, Théorie des graphes (Excellente introduction à la théorie, cette fois en français)
- **Cormen, Leiserson, Rivest, Stein**, Introduction to Algorithms--3d Edition (2009)
- **Deo Narsingh**, Graph Theory with Applications to Engineering and Computer Science
- **Diestel Reinhard**, Graph Theory (Présentation très soignée, pour ceux qui aiment les mathématiques pures, cité par Cogis Olivier)
- **Fournier Jean-Claude**, Graphs Theory and Applications
- **Gallian Joseph**, Dynamic Survey of Graph Labeling (36th ed-2023)
- **Gibbons Alan**, Algorithmic Graph Theory
- **Goodaire Edgar**, Discrete Mathematics with Graph Theory
- **Gosset Eric**, Discrete Mathematics with proofs

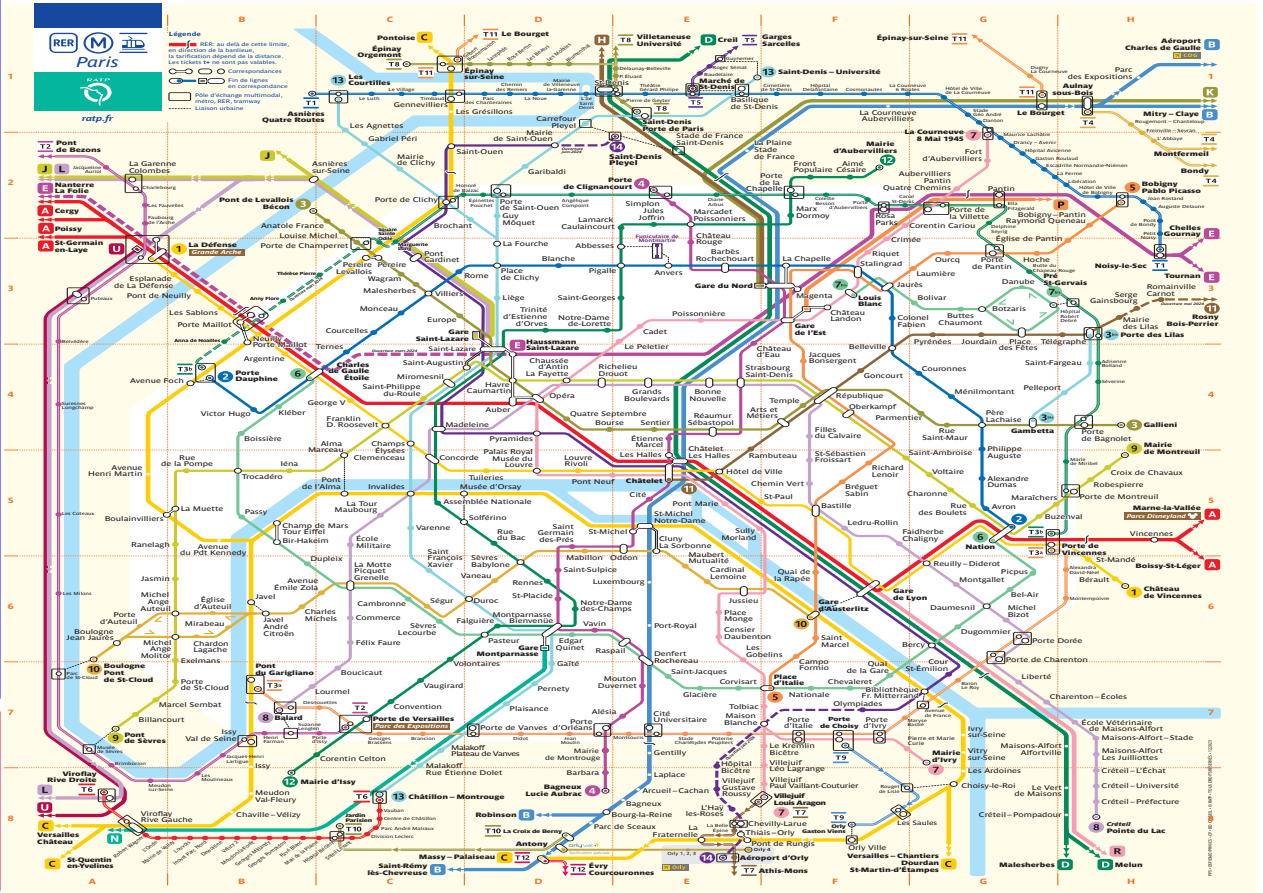
- **Gross Jonathan**, Combinatorial Methods with Computer Applications
- **Gross Jonathan**, Handbook of graph theory (1100 pages … Pour les courageux !)
- **Harary Frank**, Graph Theory (Une revue des problèmes de la théorie par un grand spécialiste)
- **Harary Frank**, Seminar of Graph Theory (A) (Démonstrations de théorèmes de la théorie des graphes)
- **Harris John**, Combinatorics and Graph Theory
- **Jukna Stasys**, On the optimality of Bellman–Ford–Moore shortest path algorithm (2020)
- **Jungnickel Dieter**, Graphs,Networks and Algorithms (4th edn)
- **Keevash Peter**, Existence of designs (the)-I
- **Keevash Peter**, Existence of designs (the)-II
- **Keevash Peter**, Ringel's tree packing conjecture in quasirandom graphs (2020)
- **Lavault Christian,** Distributed Algorithm for Constructing a Minimum Diameter Spanning Tree
- **Lavault Christian,** Distributed Algorithm for the Minimum Diameter Spanning Tree Problem
- **Le Monde est mathématique, volume 10**, Plans de métro et réseaux neuronaux-La théorie des graphes
- **Li Yifang**, Problème du voyageur de commerce (Le)
- **Li Yifang**, Problème du sac à dos (Le)
- **Liu Bolian**, Matrices in Combinatorics and Graph Theory
- **Logofatu Doina**, Algorithmen und Problemlösungen in C++ (11-Graphen)
- **Montgomery Richard**, Proof of Ringel's Conjecture (A) (2020)
- **Palermo FP**, Network Minimization Problem (A)
- **Papadimitriou Christos**, Algorithms
- **Reinelt Gerhard**, Travelling Salesman (the) (1994)
- **Rigo Michel**, Théorie des graphes
- **Ringel Gerhard**, Pearls in Graph Theory--A Comprehensive Introduction (1994)
- **Salleh Shaharuddin**, Simulation for Applied Graph Theory Using Visual C++
- **Sigward Eric**, Introduction à la théorie des graphes
- **Solnon Christine**, Théorie des graphes et optimisation dans les graphes

- **Tangente HS n°54**, Les graphes (Présentation très bien illustrée de la théorie)
- **Thomassen Carsten**, Genus Problem for Cubic Graphs (the)
- **Thomassen Carsten**, Graph Genus Problem Is NP-Complete (the)
- **Univ Cornell**, Approximation Algorithms
- **Vasudev C**, Graph Theory with Applications
- **West Douglas**, Introduction to Graph Theory (2002) (Pour les courageux, qui veulent aller plus loin qu'une introduction)
- **Wilson Robin**, Introduction to Graph Theory (Présentation très claire, avec quelques applications intéressantes, cité par Cogis Olivier)
- **Wozniak Mariusz**, Packing of graphs and permutations—a survey (2004)
- **Yap HP**, Packing of graphs—a survey (1988)

COMPTER les CHEMINS et PROBLÈMES de PARCOURS dans les GRAPHES

KAFEMATH

LEXIQUE



KAFEMATH

CJUP

(**C**haque **J**our **U**n **P**eu)

14/11/2024

14 novembre 2024

MAJ : 16/11/2024

Quelques termes de la théorie des graphes (liste thématique)

- **Sous-graphe** : $H = (Y, B)$ est un sous-graphe de $G = (X, A)$ si $Y \subseteq X$ et $B \subseteq A$.
- **Graphe partiel** : $H = (Y, B)$ est un graphe partiel de $G = (X, A)$ si $Y = X$ et $B \subseteq A$.
- **Ordre d'un graphe** : l'ordre d'un graphe est le nombre de sommets de ce graphe.
- **Chaîne** : suite finie de sommets reliés entre eux par une arête.
- **Chaîne simple** : chaîne qui n'utilise pas deux fois la même arête.
- **Chaîne eulérienne** : chaîne simple passant par toutes les arêtes d'un graphe.
- **Chaîne hamiltonienne** : chaîne simple passant par tous les sommets d'un graphe une et une seule fois.
- **Chemin** : suite de sommets reliés par des arcs dans un graphe orienté.
- **Cycle** : chaîne qui revient à son point de départ.
- **Cycle eulérien** : cycle simple passant par toutes les arêtes d'un graphe une et une seule fois.
- **Cycle hamiltonien** : cycle simple passant par tous les sommets d'un graphe une et une seule fois.

- **Graphe complet**, graphe dont chaque sommet est relié aux autres par une seule arête
- **Graphe connexe** : un graphe G est dit connexe si pour toute paire de sommets $\{x, y\}$ de G, il existe une chaîne de premier terme x et de dernier terme y.
- **Arbre** : graphe connexe sans cycle simple et sans boucle.
- **Graphe eulérien** : graphe qui possède un cycle eulérien.
- **Graphe semi-eulérien** : graphe qui possède une chaîne eulérienne.
- **Graphe hamitonien** : graphe qui possède un cycle hamiltonien.
- **Graphe semi-hamiltonien** : graphe qui possède une chaîne hamiltonienne.
- **Graphe valué** : graphe où des réels sont associés aux arêtes. Dans cet exposé, on ne considérera que des valuations positives.
- **Longueur d'une chaîne** : nombre des arêtes qui composent la chaîne.
- **Valeur d'une chaîne** : somme des valeurs des arêtes (arcs) d'une chaîne d'un graphe valué.
- **Distance entre deux sommets** : longueur de la plus courte chaîne joignant ces deux sommets.
- **Diamètre d'un graphe** : maximum des distances entre les sommets d'un graphe.

- **Indice chromatique** : nombre minimal de couleurs permettant de colorier les arêtes d'un graphe, de telle sorte que deux arêtes adjacentes n'aient pas la même couleur.
- **Nombre chromatique d'un graphe** : nombre minimal de couleurs permettant de colorier les sommets d'un graphe, de telle sorte que deux sommets adjacents n'aient pas la même couleur.