Chapitre 3: Arbres et Arborescences

3.1. Définition d'un arbre, Co-arbre

- Un arbre est un graphe non orienté connexe et sans cycles

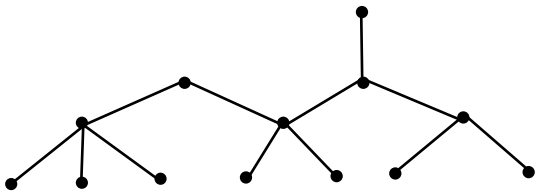
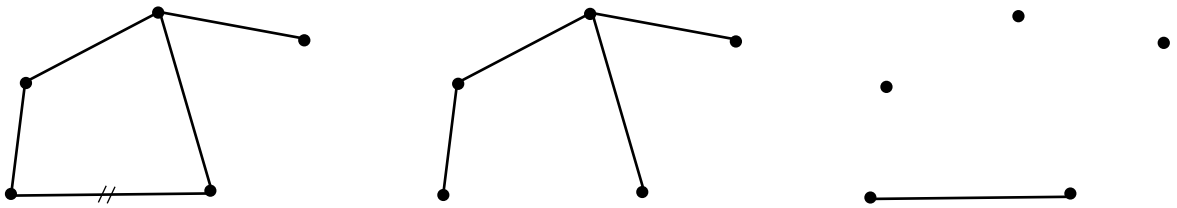


Fig. 3.1: Exemple d'un arbre

- Si G est un graphe non orienté, connexe, l'arbre T du graphe G est un graphe partiel du G connexe et sans cycles.
- Le co-arbre T'associé à T est le graphe partiel complémentaire de T par rapport à G.

Exemple 3.1: Construction d'un arbre, co-arbre à partir d'un graphe

Soit le graphe G suivant :



(G) **l'arbre T associé au graphe G** **Le co-arbre T'associé**

Fig. 3.2: Graphe, arbre et Co-arbre associés

3.2. Définition d'une forêt :

- Est un graphe non orienté, sans cycle (la connexité n'est pas nécessaire)

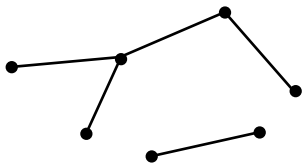
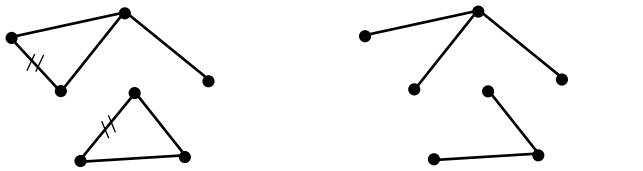


Fig.3.3: Définition d'une forêt

- Une forêt d'un graphe G est un graphe partiel, non connexe de G, sans cycle.
- Une forêt n'est pas un arbre.



Un graphe G **Une forêt associée au graphe G**

Fig.3.4: Différence entre un arbre et une forêt

Remarques

Dans un arbre, Co-arbre, forêt on distingue deux types de sommets :

- Les sommets ayant plusieurs arêtes incidentes (d(x) >1) (arêtes entrantes ou sortantes)
- Les sommets ayant une seule arête incidente, ces derniers sont appelés sommets pendants (sommets de degré 1).

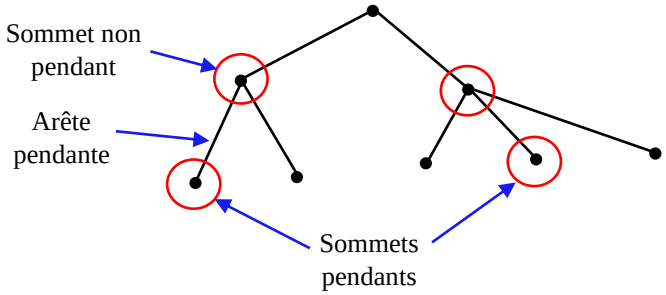


Fig. 3.5: Sommets pendants et sommets non pendants

3.3. Quelques propriétés d'un arbre

- Un arbre d'ordre ≥ 2 admet au moins deux sommets pendants.
- Tout graphe connexe G admet un arbre comme graphe partiel (en éliminant les cycles)
- Un arbre d'ordre *n* est de taille *(n-1)*

- Si un graphe G non orienté connexe d'ordre *n* et de taille *m*, alors il admet un arbre T d'ordre *n* et de taille (*n-1*) et un co-arbre d'ordre *n* et de taille *m-(n-1)*
- La suppression d'une arête d'un arbre disconnecte l'arbre (donne deux composantes connexes)
- Toute arête d'un arbre est un isthme (relie 2 composantes connexes), c'est une arête qui n'est contenue dans aucun cycle.
- Toute pair de sommets x et y de l'arbre T est reliée par une chaîne unique.
- Le rajout d'un arc/une arête à l'arbre crée un cycle unique.

3.4. Définition d'une arborescence

Soit G un graphe orienté, on appelle arborescence un arbre orienté de sorte qu'il existe un sommet *r* particulier, tel qu'il existe un chemin de *r* à tout autre sommet de G, *r* est appelé la racine de l'arborescence.

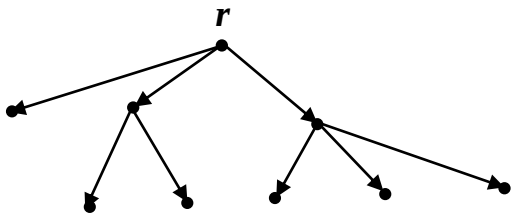


Fig. 3.6: Définition d'une arborescence

Observation : une arborescence est un arbre orienté admettant une racine.

3.5. Propriétés d'une arborescence

- Une arborescence d'ordre ≥ 2 admet au moins un sommet pendant.

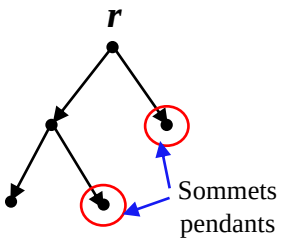


Fig. 3.7: Définition de sommets pendants

- La suppression d'un sommet pendant d'une arborescence d'ordre ≥ 2 donne une arborescence.

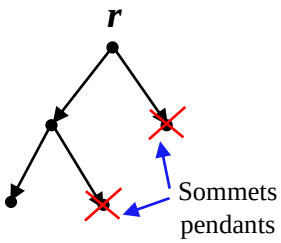


Fig. 3.8: Résultat de suppression d'un sommet pendant

- Dans une arborescence de racine *r*, il existe un chemin unique de *r* à tout sommet x≠r. (si ∃ 2 chemins ⇒ ∃ cycle ⇒ contradiction avec la définition (n'est pas arborescence))

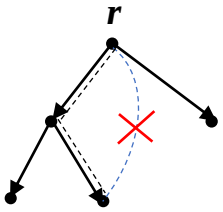


Fig. 3.9: Unicité de chemin dans une arborescence

- Dans une arborescence A de racine *r*, $d^-(r)=0$ et $d^-(x)=1$ pour tout sommet x≠r

- Si A est une arborescence, l'ensemble des descendants de x $\Gamma^+(x)$ (les successeurs de x) engendre une arborescence A' de racine *x* pour tout sommet *x*.

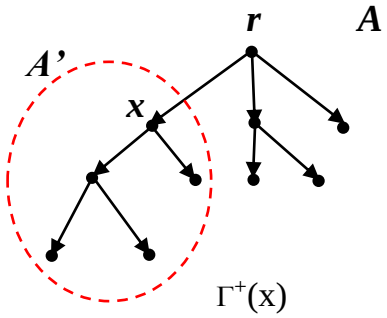


Fig. 3.10: Successeurs d'un sommet dans une arborescence

- La suppression d'un arc d'une arborescence A, donne deux (02) sous arborescences disjointes A_1, A_2.

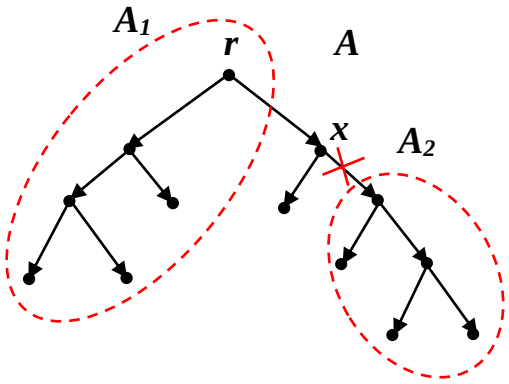


Fig. 3.11: Suppression d'un arc d'une arborescence

- Si G est un graphe orienté admettant une racine *r*, alors il existe dans G une arborescence A de racine *r*, qui soit un graphe partiel de G.

3.6. Représentation d'une arborescence

Une arborescence peut être représentée par les tables (vecteurs et matrices) vues dans chapitres II. Cependant, elle admet une représentation plus efficace en utilisant les listes chaînées (représentation dynamique) comme suit :

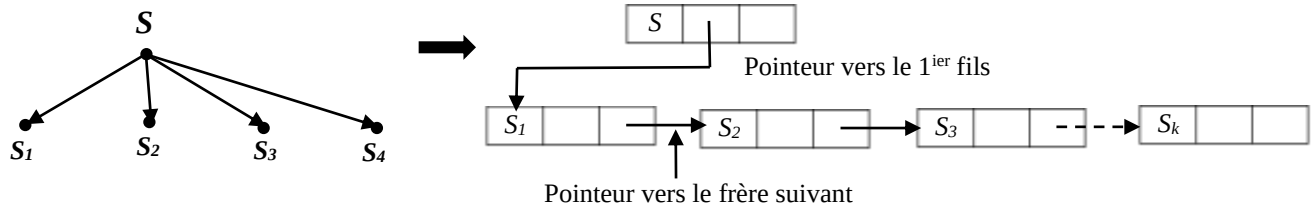


Fig. 3.12: Représentation dynamique d'une arborescence

Exemple 3.2: Représentation dynamique d'une arborescence

Soit l'arborescence suivante :

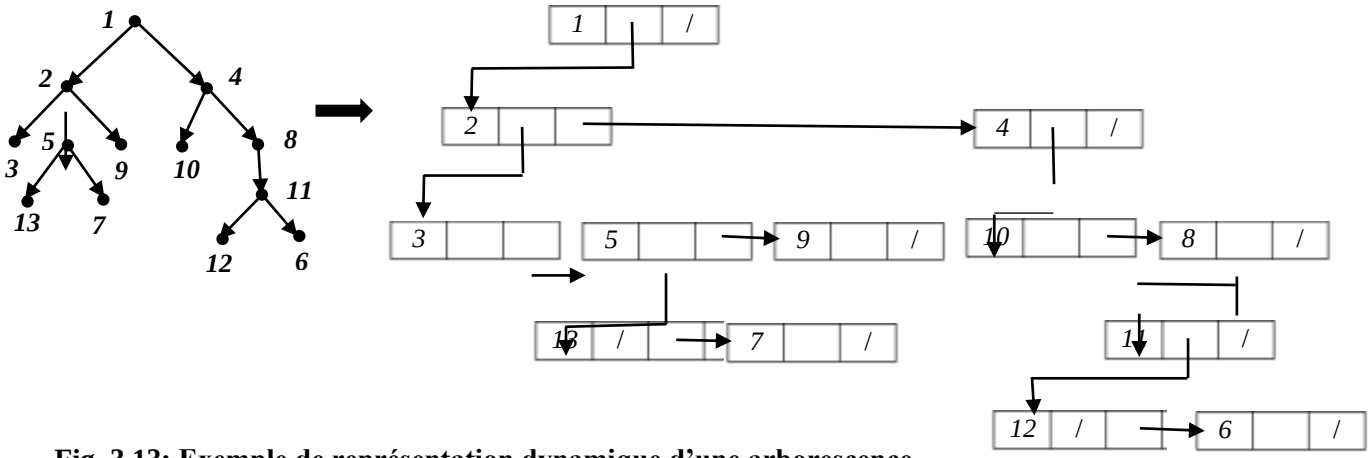


Fig. 3.13: Exemple de représentation dynamique d'une arborescence

3.7. Parcours d'une arborescence :

Plusieurs modes de parcours sont possibles :

1. Parcours en largeur : 1, 2, 4, 3, 5, 9, 10, 8, 13, 7, 11, 12, 6
2. Parcours en profondeur : 1, 2, 3, 5, 13, 7, 9, 4, 10, 8, 11, 12, 6

Remarque :

Trois types de parcours sont aussi utilisés : préordre, in-ordre, post-ordre (préfixé, infixé, postfixé)

3.8. Cycles et vecteurs associés :

Soit G (X, U) un graphe orienté d'ordre |X| = n et de taille |U| = m. On peut faire correspondre à tout cycle c un vecteur $v_c = (v_1, v_2, ….., v_m)$, avec :

$$\begin{cases} +1 \ \ si\ l'arc\ v_i \ \in v_c\ et\ dans\ le\ sens\ du\ parcours\ de\ c \\ -1 \ \ si\ l'arc\ v_i \in v_c\ et\ dans\ le\ sens\ inverse\ du\ parcours\ de\ c \\ 0\ si\ l'arc\ v_i \ \ n'est\ pas\ dans\ v_c \end{cases}$$

Remarque : on parle ici d'un cycle au lieu d'un circuit, car les vecteurs composants ont des sens différents.

Exemple 3.3: Recherche de vecteurs associés à des cycles

Soit le graphe G défini comme suit :

X = {a, b, c, d, e, f, g, h} U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Soit le cycle c_1 = (1, 3, 5, 9, 8, 2)

Le vecteur associé est le suivant :

v_{c1} = (+1, -1, +1, 0, -1, 0, 0, -1, +1, 0)

Soit le cycle c_2 = (1, 3, 10, 8, 2)

Le vecteur associé est le suivant :

v_{c2} = (+1, -1, +1, 0, 0, 0, 0, -1, 0, +1)

Soit le cycle c_3 = (5, 9, 10)

Le vecteur associé est le suivant :

v_{c3} = (0, 0, 0, 0, -1, 0, 0, 0, +1, -1)

Soit le cycle c_4 = (4, 10, 8)

Le vecteur associé est le suivant :

v_{c4} = (0, 0, 0, -1, 0, 0, 0, -1, 0, +1)

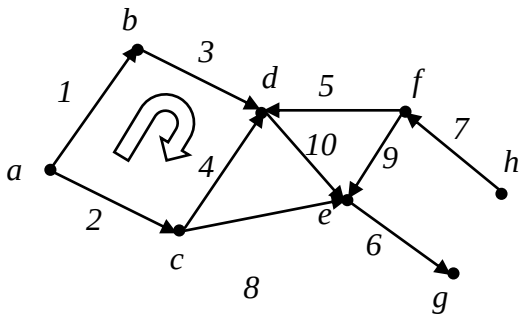


Fig. 3.14: Cycles et vecteurs associés

3.9. Cycles indépendants :

On dit que les cycles c_1, c_2, …, c_k sont indépendants si les vecteurs correspondants v_{c1}, v_{c2}, …, v_{ck} sont linéairement indépendants.
c.à.d, il existe une relation de la forme :
$\alpha_1 v_{c1} + \alpha_2 v_{c2} + \ldots\ldots\ldots + \alpha_k v_{ck} \neq \vec{0}$ avec : α_1, α_2, …., α_k sont des nombres réels non tous nuls, sinon (c.à.d, $\alpha_i = 0 \ \forall i = 0, 1, \ldots$) ils sont dits dépendants.

Par exemple, les cycles c_1, c_2, c_3 sont indépendants car on a :
$v_{c1} - v_{c2} - v_{c3} \neq \vec{0}$ ($\alpha_1 = -1$, $\alpha_2 = -1$, $\alpha_3 = +1$)

Vérification :
(-1)*(+1, -1, +1, 0, -1, 0, 0, -1, +1, 0) + (-1) (+1, -1, +1, 0, 0, 0, 0, -1, 0, +1) + (+1) (0, 0, 0, 0, -1, 0, 0, 0, +1, -1) = (-2, +2, -2, 0, 0, 0, 0, 0, +2, -2) $\neq \vec{0}$

Théorème :
Soient c_1, c_2, c_3, …, c_k des cycles, si chaque cycle contient un arc que les autres ne contiennent pas, alors les cycles c_1, c_2, c_3, …, c_k forment un ensemble de cycles indépendants.

3.10. Définition d'une base de cycles :

Une base de cycles est un ensemble minimal de cycles c_i indépendants correspondants à des vecteurs v_{ci} (linéairement indépendants), tel que tout vecteur du graphe G puisse s'exprimer en fonction de cette base (des vecteurs de cette base)

Exemple 3.4: Définition d'une base de cycles dans un graphe

$c_1 = (1, 6, 2) \rightarrow v_{c1} = (+1, -1, 0, 0, 0, +1)$
$c_2 = (1, 6, 3) \rightarrow v_{c1} = (+1, 0, +1, 0, 0, +1)$
$c_3 = (2, 3) \rightarrow v_{c1} = (0, -1, +1, 0, 0, 0)$
$c_4 = (1, 4, 5, 2) \rightarrow v_{c1} = (+1, -1, 0, +1, -1, 0)$
$c_5 = (6, 5, 4) \rightarrow v_{c1} = (0, 0, 0, +1, -1, +1)$
$c_6 = (1, 4, 5, 3) \rightarrow v_{c1} = (+1, 0, +1, +1, -1, 0)$

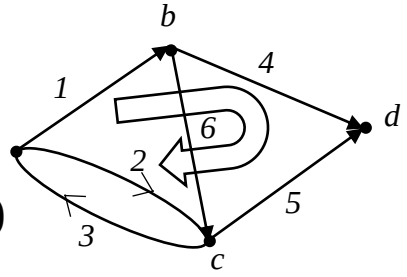


Fig. 3.15: Définition d'une base de cycles

On constate que c_1, c_3, c_5 sont indépendants, car chaque cycle contient un vecteur que les autres ne contiennent pas $c_1(1)$, $c_3(3)$, $c_5(5/4)$ ===> forment une base de cycles BC.

Théorème : Soit G(X, U) un graphe d'ordre *n* et de taille *m*, constitué de *p* composantes connexes distinctes.
La dimension de la base de cycles de ce graphe est donnée par la relation :

$$\boldsymbol{V(G) = m - n + p}$$

V(G) est appelé le nombre cyclomatique de G.

3.11. Définition d'une base de cocycles

G(X, U) un graphe, $U = \{u_1, u_2, \ldots, u_3\}$, $|X| = n$, $|U| = m$
Soit A un sous-ensemble de sommets de X, $A \subset X$
Un cocycle θ est l'ensemble des arcs liant A et (X – A), c.à.d,
$\theta = w(A) = w^+(A) \cup w^-(A)$
On associe au cocycle θ le vecteur $v_\theta = (\theta_1, \theta_2, \ldots, \theta_m)$ défini comme suit :

$$\begin{cases} +1 \ si \ u_i \in \ w^+(A) \\ -1 \ si \ u_i \in \ w^-(A) \\ \ 0 \ ailleurs \end{cases}$$

Exemple 3.5: Définition d'une base de cocycles dans un graphe

Soit le graphe G suivant :

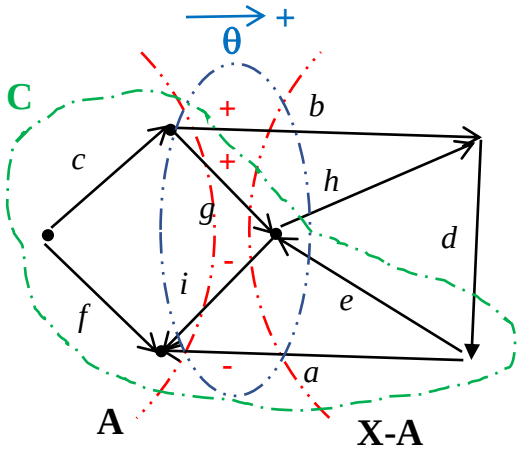


Fig. 3.16: Définition d'une base de cocycles

Soit le cycle c = {a, e, g, c, f} et le cocycle θ = {b, g, i, a}

Les vecteurs associés au cycle *c* et au cocycle θ seront comme suit :

Table. 3.1: Vecteurs associés aux cycles et aux cocycles

	a	b	c	d	e	f	g	h	i	
Vc	+1	0	+1	0	-1	-1	+1	0	0	
Vθ	-1	+1	0	0	0	0	+1	0	-1	
V_c . V_θ	-1	0	0	0	0	0	+1	0	0	**0**

$$===> \ \sum \vec{V_c} . \overrightarrow{V_\theta} = 0$$

OBS : le produit scalaire d'un cycle et d'un cocycle est nul

L'ensemble de cocycles linéairement indépendants forme une base de cocycles pour le graphe G
La dimension de cette base λ(G) est donné par la relation :

$$\lambda(G) = n - 1$$

λ(G) est aussi appelé le nombre cocyclomatique de G.

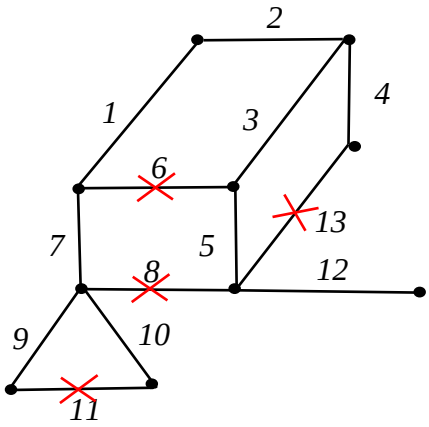
3.12. Algorithme de recherche d'une base de cycles d'un graphe connexe

Soit G(X, U) un graphe connexe

1. Chercher un arbre maximal T dans le graphe G (qui contient tous les sommets)
2. L'ajout d'un arc $\vec{u}$ du coarbre T'à T crée un cycle unique c_u orienté dans le sens de $\vec{u}$
3. Ecrire tous les cycles uniques obtenus ==> forment la base cherchée (sont linéairement indépendants)

Exemple 3.6: Recherche d'une base de cycles dans un graphe

Soit le graphe suivant :



n = 10 ; m = 13
On a : n = 10 sommets dans l'arbre ==>
la taille de l'arbre résultant T est :
m' = n – 1 = 10 – 1 = 9 arêtes

∃ une seule composante connexe (p = 1)

Le nombre cyclomatique (la dimension de la base de cycle) est V(G) = m – n + p = 13 – 10 + 1 = 4

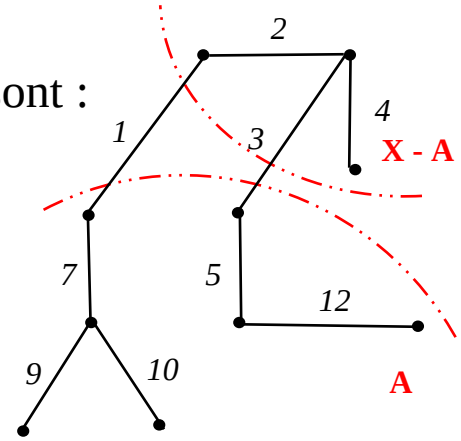
Les cycles qui forment la base de cycle sont :

C_6 = (1, 2, 3, **6**)

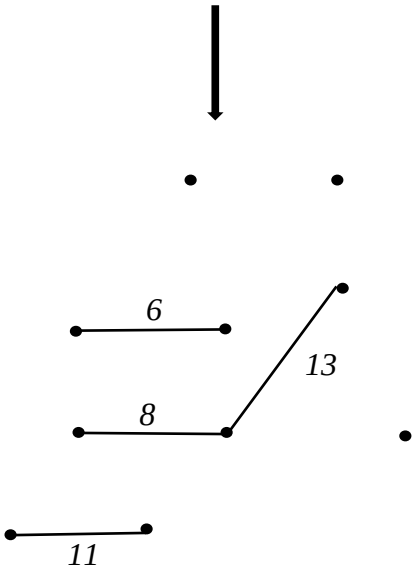
C_8 = (1, 2, 3, 5, **8**, 7)

C_{11} = (9, 10, **11**)

C_{13} = (3, 4, **13**, 5)



L'arbre maximal obtenu



Le coarbre obtenu

Fig. 3.17: Démarche de recherche d'une base de cycles

OBS : Si c'est nécessaire, on donne une orientation arbitraire aux différentes arêtes, et au cycles le même sens de l'arête ajoutée (qui ferme le cycle)

3.13. Recherche d'une base de cocycles

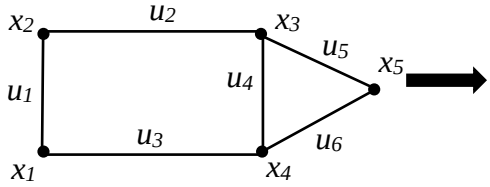
1. Répartir X en A et (X – A)
2. Les arcs V dont I(V)∈A et T(V)∈(X – A) forment un cocycle θ_V
3. L'ensemble des cocycles uniques obtenus (chaque cocycle contient un arc que les autres ne contient pas) forment la base cherchée (sont Linéairement indépendants)

3.14. Algorithme de recherche d'un arbre maximal

G(X, U) est un graphe connexe d'ordre n et de taille m.
Un arbre maximal de G est un graphe partiel T de G connexe, sans cycle d'ordre *n* et de taille *(n – 1)*.
L'algorithme de construction de l'arbre T du graphe G consiste à prendre les (n – 1) arêtes qui ne ferment pas des cycles (On garde tous les sommets et on supprime des arêtes)

Exemple 3.7: Recherche d'un arbre maximal

Soit G(X, U) avec : |X| = n =5 ; |U| = m = 6



L'algorithme

Arête	u_1	u_2	u_3	u_4	u_5	u_6
Décision	OK	OK	OK	Non	OK	Non

L'arbre maximal

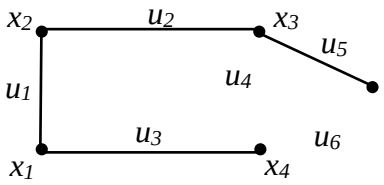


Fig. 3.18: Démarche de recherche d'un arbre maximal dans un graphe

3.15. Algorithme de recherche d'un arbre maximal de poids minimum (Kruskal 1)

Soit G(X, U, L) un graphe valué, non orienté, connexe et dont toutes les longueurs d'arêtes soient différentes (si u ≠ v ==> L(u) ≠ L(v))

(i) On représente le graphe par la liste des arcs triés selon leurs poids croissants.

(ii) Partons d'un graphe vide, et on prend successivement les (n – 1) premières arêtes qui ne fermant pas des cycles avec celles déjà prises.

(iii) Les (n – 1) arêtes retenues forment un arbre maximal de poids minimum.

OBS : l'arbre maximal de poids minimum est constitué de (n – 1) arêtes)

Exemple 3.8: Recherche d'un arbre maximal de poids minimum par l'application de KRUSKAL 1

Exemple : Soit G = (X, U, L)

Avec : X = {1, 2, 3, 4, 5, 6, 7, 8, 9}

U = {a, b, c, d, e, f, g, h, i, j, k, l, p, q}

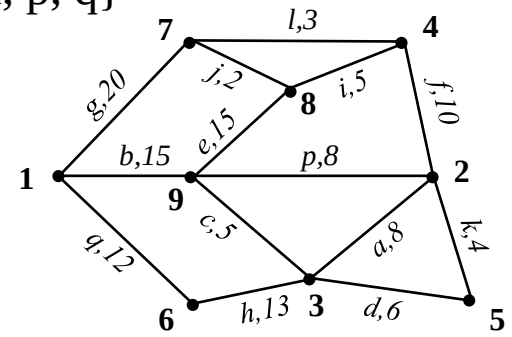


Fig. 3.19: recherche d'un arbre maximal de poids minimum dans un graphe valué

Application de l'algorithme :

Liste d'arêtes non triées

Table. 3.2: Liste triée d'arêtes représentant un graphe

Arête	*a*	b	c	d	e	f	*g*	*h*	*i*	*j*	*k*	*l*	*p*	*q*
Ext.Init	3	1	3	3	8	4	1	6	4	7	2	7	9	1
Ext.Ter	2	9	9	5	9	2	7	3	8	8	5	4	2	6
Poids	8	15	5	6	15	10	20	13	5	2	4	3	8	12

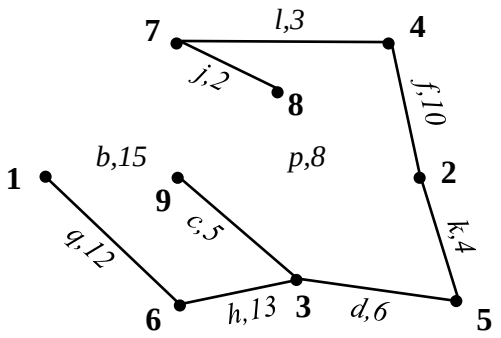
Après le tri on aura:

Liste des arêtes triées selon leurs poids :

Table. 3.3: Liste d'arêtes triée selon le poids représentant un graphe

Arête	*j*	*l*	*k*	c	*i*	d	*a*	*p*	f	*q*	*h*	b	e	*g*
Ext.Init	7	7	2	3	4	3	3	9	4	1	6	1	8	1
Ext.Ter	8	4	5	9	8	5	2	2	2	6	3	9	9	7
Poids	2	3	4	5	5	6	8	8	10	12	13	15	15	20
Décision	ok	ok	ok	ok	X	ok	X	X	ok	ok	ok	X	X	X

On s'arrête ici, car on a la taille de l'arbre maximal est n -1 = 9 – 1 = 8 arêtes (déjà prises)



Arbre maximal de poids minimum
Nb.arêtes = 9 – 1 =8 ; Poids.Tot = 55

Fig. 3.20: Démarche de recherche d'un arbre maximal de poids minimum dans un graphe valué

3.16. Algorithme de Kruskal 2

(i) Représenter le graphe par la liste des (arcs/arêtes) obtenue lors de la saisie

(ii) Partons d'un graphe vide, on prend successivement les arcs et dès que l'arc en cours de traitement forme un cycle avec ceux déjà pris, on retire l'arc du poids maximum du cycle

(iii) Les arcs retenus sont ceux d'un arbre de poids minimum

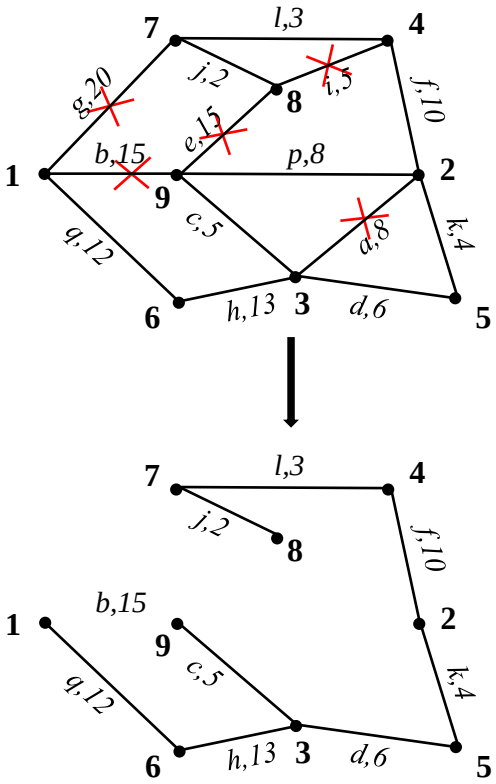
Application sur l'exemple précédent :

Table. 3.4: Liste d'arêtes non triées représentant un graphe

Arête	*a*	b	c	d	e	f	*g*	*h*	*i*	*j*	*k*	*l*	*p*	*q*
Ext.Init	3	1	3	3	8	4	1	6	4	7	2	7	9	1
Ext.Ter	2	9	9	5	9	2	7	3	8	8	5	4	2	6
Poids	8	15	5	6	15	10	20	13	5	2	4	3	8	12
Décision	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	ok	X	ok
Déc. Finale	X	X	ok	ok	X	ok	X	ok	X	ok	ok	ok	X	ok

Remarque :

En appliquant les mêmes algorithmes on peut construire l'arbre maximal de poids maximum.



Arbre maximal de poids minimum

Fig. 3.21: Recherche d'un arbre maximal de poids minimum par application de KRUSKAL 2