Chapitre 3
Arbres et Arborescences

3.1 Introduction

Soit une entreprise répartie sur plusieurs villes. Chacun de ses villes veut pouvoir communiquer avec les autres, soit directement, soit en passant par l'intermédiaire d'autres villes. La situation est représentée comme suit :

Les villes sont les sommets et chacune des liens (arêtes) représente une ligne de communication entre les deux villes.

Par exemple :

Un réseau comporte des villes A, B, C, D, et E qui doivent pouvoir communiquer entre elles. Les liaisons envisagées sont représentées par le graphe de la figure 3.1 (les arêtes sont étiquetées par la distance entre les villes).

Question : Comment câbler le réseau à moindre coût ?
Réponse : Il s'agit d'enlever des arêtes au graphe de façon qu'il reste connexe, et que la somme des pondérations des arêtes soit le plus petit possible.
Remarquons que le graphe partiel recherché est sans cycle.

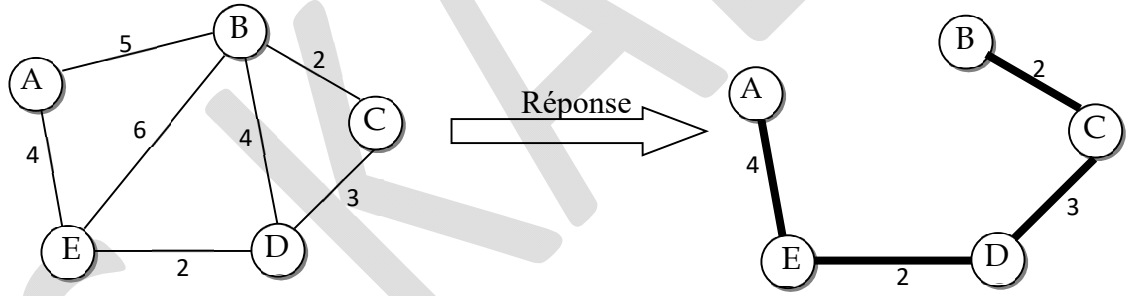


FIG 3.1 Arbre

Ce problème se pose, par exemple, lorsqu'on désire relier des villes par un réseau routier de coût minimum. Les sommets du graphe représentent les villes, les arêtes, les tronçons qu'il est possible de construire et les poids des arêtes correspondent aux coûts de construction du tronçon correspondant.

Le problème est ramené au problème suivant : Le problème de l'arbre recouvrant de poids minimal est celui qui consiste à déterminer un arbre qui soit un graphe partiel d'un graphe G simple connexe et dont le poids total est minimal

3.2 Propriétés

3.2.1 Nombre d'arcs dans un graphe

Soit un graphe G = (X,U) d'ordre *n* « n=|X| » et *m* arcs/arêtes « m=|U| ».

- Si G est connexe, m ≥ n-1.
- Si G est sans cycle, m ≤ n-1.

3.2.2. Arbre

Définition 3.1 : Un **arbre** est un graphe non orienté connexe sans cycle, il a donc «n-1» arêtes. On peut donc dire qu'un arbre est un graphe qui connecte tous les sommets entre eux avec un minimum d'arêtes.

Définition 3.2 : Un **arbre** est un graphe connexe tel que, quels que soient les deux sommets distincts i et j, il existe une seule chaîne joignant i à j.

Remarques

- L'ajout du moindre arête supplémentaire dans un arbre crée un cycle.
- Chaque graphe connexe possède un graphe partiel qui est un arbre.
- Les arbres sont des graphes particuliers, très populaires en algorithmiques et en informatique.

Les cinq graphes suivants sont des arbres :

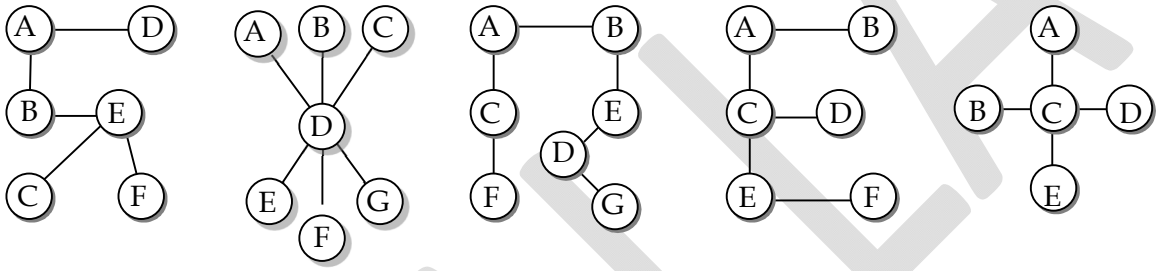


FIG 3.2 5 Arbres

Théorème :

Soit G un graphe d'ordre n, les propositions suivantes sont équivalentes :

- G est un arbre,
- G est connexe et sans cycle,
- G est sans cycle et possède n-1 arêtes/arcs,
- G est connexe et possède n-1 arêtes/arcs,
- G est sans cycle et l'ajout de toute arête crée un seul cycle et un seul,
- G est connexe et la suppression de n'importe quelle arête le déconnecte

3.2.3. Forêt

On appelle **forêt** un graphe dont chaque composante connexe est un arbre.

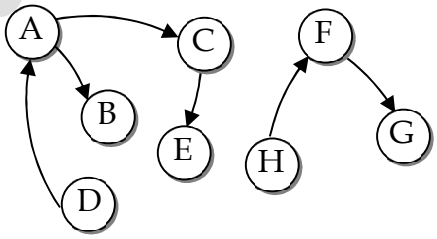


FIG 3.3 Forêt avec 2 arbres

3.2.4. Racine, Anti-racine

Souvent, pour manipuler un arbre, nous particularisons un sommet du graphe que nous appelons racine. Dans le cas des graphes non orientés, le choix d'une racine « r » dans l'arbre est arbitraire. Dans le cas des graphes orientés, la racine est définie de manière unique comme le sommet sans prédécesseur de l'arbre.

Le choix d'une racine revient dans un certain sens à orienter l'arbre, la racine apparaissant comme l'ancêtre commun à la manière d'un arbre généalogique. Le vocabulaire de la théorie des graphes s'en inspire directement : on parle de fils, de père, de frère, ...

Un sommet «a» d'un graphe G est **une racine** de G s'il existe un chemin joignant «a» à chaque sommet du graphe G.

Un sommet «a» d'un graphe G est **une anti-racine** de G s'il existe un chemin joignant chaque sommet du graphe G à «a».

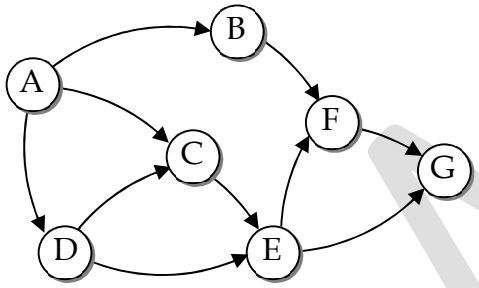


FIG 3.4 Racine, Anti-racine

Pour le graphe de la figure 3.4 on a :

- Le sommet « A » est une racine du graphe.
- Le sommet « G » est une anti-racine du graphe.

3.3.3. Arborescence, anti-arborescence

Un graphe orienté G est une arborescence de racine « a » si et seulement si :

- G est un arbre
- « a » est une racine.

Un graphe orienté G est une anti-arborescence d'anti-racine « a » si et seulement si :

- G est un arbre.
- « a » est une anti-racine.

Remarques :

- Un sommet (nœud) peut avoir 0 ou plusieurs fils (successeurs).
- Un sommet (sauf la racine) a exactement un père (prédécesseurs).
- Une feuille est un sommet qui n'a pas de fils (successeurs).
- Un sommet intérieur est un sommet qui a un seul père (prédécesseur) et a au moins un fils (successeur). Donc, Tout sommet (sauf la racine) de l'arbre est :
  - + Soit une feuille.
  - + Soit un sommet intérieur.
- Un sommet intérieur et tous ses descendants est un « sous-arbre ».
- Pour tout sommet du graphe, il existe un chemin unique qui part de la racine et l'atteint.
- Sous Windows, l'organisation des disques est basée sur une structure arborescente des répertoires.

Dans l'arbre généalogique :

- La racine est un ancêtre de tous les sommets.
- Chaque sommet est un descendant (successeur) de la racine.
- Les sommets ayant le même père sont appelés « frères ».

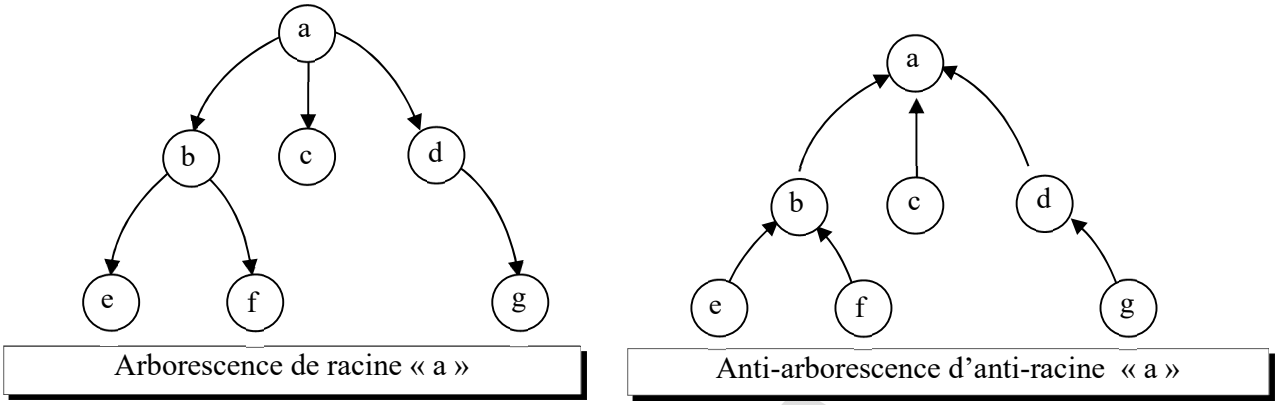


FIG 3.5 Arborescence, Anti-arborescence

3.3 Problème de l'arbre de poids minimal

3.3.1 Arbre Couvrant et Arbre Couvrant de Coût Minimum

On considère le réseau routier représenté par le graphe de la figure 3.6. Ces routes (arêtes) sont souvent enneigées en hiver et les responsables des villes concernées décident de déneiger un nombre minimal de routes de telle sorte que deux villes (sommets) quelconques du réseau (graphe) soient toujours reliées par une route (arête) déneigée. Les valeurs d'arêtes représentent la durée, en heures, de déneigement de la route correspondante.

Le problème consiste à construire un arbre couvrant pour ces villes tout en minimisant la durée totale de routes (arêtes) à déneiger. L'arbre couvrant contient 8 sommets, et doit donc contenir 7 arêtes. Il s'agit donc de construire un arbre couvrement de coût minimum (ACCM) du graphe comme le montre la figure 3.6.

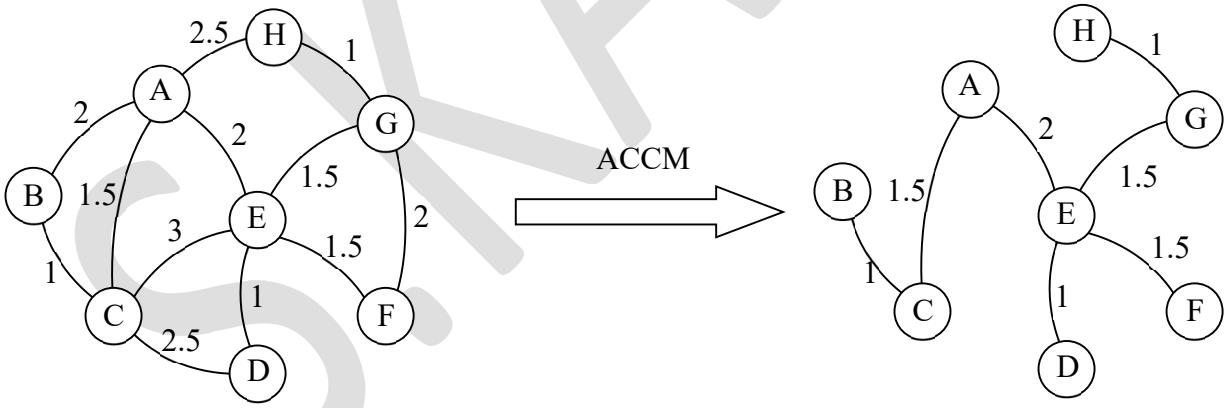


FIG 3.6 Arbre couvrant de cout minimum

Notre problème s'énonce donc : Etant donné un graphe G = (X, U), déterminer un graphe **partiel connexe** ACCM = (X, U') de coût **minimum**.

Il existe des algorithmes pour résoudre le problème de l'arbre couvrant de coût minimum.

Chacun de ces algorithmes utilise plus particulièrement l'une des caractérisations des arbres pour trouver un « ACCM ».

3.3.2. Définitions

Définition 3.1 : Un **arbre couvrant** (AC) pour un graphe G=(X,U) est un arbre construit uniquement à partir de l'ensemble des arêtes/arcs U et qui connecte ("couvre") tous les sommets de X. Un arbre couvrant AC d'un graphe G est donc un arbre et un graphe partiel de G.

Définition 3.2 : Un **arbre couvrant de coût minimum** (ACCM) pour un graphe value G=(X,U,c) correspond à la recherche d'un graphe ACCM tel que :

- Le graphe ACCM est un arbre couvrant AC.
- Le graphe ACCM est de coût minimum (somme des coûts de ses arêtes/arcs de l'AC).

3.3.3. Algorithme de KRUSKAL

Principe : Pour déterminer un Arbre Couvrant de Coût Minimum par l'algorithme de KRUSKAL :

- On trie, tout d'abord, les arcs (arêtes) selon un ordre croissant de leurs coûts.
- Ensuite, dans cet ordre, les arcs sont ajoutés un par un dans un graphe G' (initialement est vide ; c-à-d ne contient que l'ensemble de sommets X) pour construire progressivement l'arbre. Un arc est ajouté seulement si son ajout dans le graphe G' ne crée pas de cycle, autrement dit, si G' reste un arbre. Sinon, on passe à l'arc suivant dans l'ordre du tri.

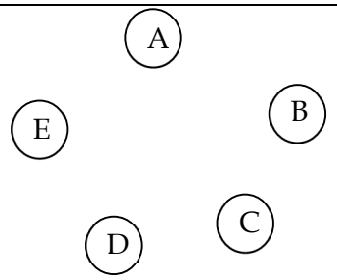
Algorithme KRUSKAL
Entrées : G(X,U) un graphe connexe d'ordre n et m arcs/arêtes.
Sortie : U' un ensemble d'arcs/arêtes
Début
Trier les arcs/arêtes de G dans l'ordre croissant des coûts ; on les notera ($u_1, u_2, \ldots, u_m$)
U'←∅ ; i ←1 ; j ←1 ;
Tant Que [(i≤m) et (j≤n-1)] **Faire**
Si (Le graphe (X,U'∪{u_i}) est acyclique) **Alors** U' ←U'∪{u_i} ; j ← j+1 ;**FSi**
i ← i+1 ;
FTQ
Fin

En appliquant l'algorithme de KRUSKAL, pour le graphe de la figure 3.6 qui comporte les villes A, B, C, D, et E qui doivent pouvoir communiquer entre elles, pour obtenir l'arbre couvrant de coût minimum de la figure 3.1:

Etape 1

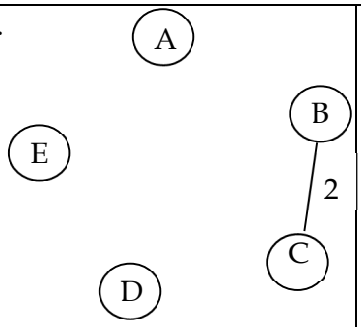
Trie des arêtes :
Les arêtes sont triées dans l'ordre :
= {(B,C), (D,E), (D,C), (B,D), (A,E), (B,A), (E,B)}= {2,2,3,4,4,5,6}
U'←∅ ;
i ←1 ;
j ←1 ;

le graphe est initialement vide contient que des sommets X

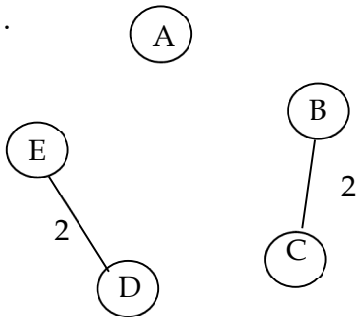


Etape 2 : (construction de l'arbre couvrant de coût minimum).

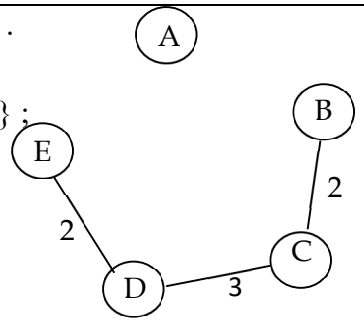
Itération 1 : (i=1) .
U' ←U'∪{(B,C)}
= {(B,C)} ;
j ← j+1
=1+1=2.



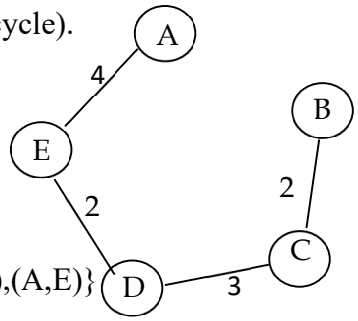
Itération 2 : (i=2) .
U'←U'∪{(D,E)}
={(B,C),(D,E)};
j ← j+1
=2+1=3.



Itération 3 : (i=3) .
U'←U'∪{(D,C)}
={(B,C),(D,E),(D,C)} ;
j ← j+1
=3+1=4.



Itération 4 : (i=4) pas de changements (l'arc (B,D) crée un cycle).



Itération 5 : (i=5).
U'←U'∪{(A,E)}
= {(B,C),(D,E),(D,C),(A,E)}
j ← j+1 =4+1=5.

j>n-1 (5>5) Fin d'algorithme

Remarques :

- Si le graphe est connexe, alors on obtient l'ACCM en «n-1» itérations.
- Si pendant une itération i <n-1 on ne trouve pas d'arête (x,y) avec x dans l'arbre et y n'appartenant pas à l'arbre, alors le graphe G n'est pas connexe.
- L'Arbre Couvrant de Coût Minimum n'est pas forcement unique.
- Si le coût de toutes les arêtes est égal à 1, le problème revient à chercher un arbre couvrant quelconque (de coût n-1).