THEORIE DES GRAPHES

8) ARBRES COUVRANTS DE POIDS MINIMUM

Les arbres sont des formalismes de représentation que l'on rencontre largement dans les disciplines telles l'intelligence artificielle, la recherche opérationnelle, l'analyse de données, la théorie de l'information, l'informatique, etc. Ainsi, l'étude de cette classe de graphes devient nécessaire.

8.1. Arbre

Un graphe est dit un arbre si et seulement s'il est connexe et sans cycle.

Exemple

Considérons le graphe G donné par la figure 1.

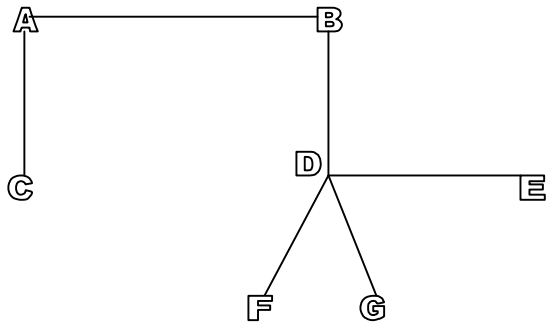


Figure 1. Un exemple d'un arbre.

Comme le graphe de la figure 1 est connexe (formé d'un seul bloc) et comme, par ailleurs, il est sans cycle, alors on en déduit que c'est un arbre.

Propriété : Soit G = [X, U] un graphe d'ordre n. Si G est **connexe**, alors il possède **au moins** n -1 arêtes.

Propriété : Soit G = [X, U] un graphe d'ordre n. Si G est **sans cycle**, alors il possède **au plus** n -1 arêtes.

Ainsi, eu égard aux deux propriétés qui précèdent, il vient qu'un arbre est un graphe qui recouvre tous les sommets et ce, en utilisant le moins d'arêtes que possible.

Théorème : Soit G = [X, U] un graphe d'ordre n = card(X) ≥ 2. Les conditions suivantes sont équivalentes pour caractériser un arbre.

1. G est connexe et sans cycle.
2. G est sans cycle et admet n -1 arêtes.
3. G est connexe et admet n-1 arêtes.
4. Tout couple de sommets est relié par une chaîne et une seule
5. G est sans cycle, et en ajoutant une arête, on créé un cycle et un seul.
6. G est connexe, et si on supprime une arête quelconque, G n'est plus connexe.

8.2. Forêt

Une forêt est un graphe sans cycle et non connexe.

Ainsi, chaque composante connexe d'une forêt est un arbre.

Exemple

Considérons le graphe G donné par la figure 2.

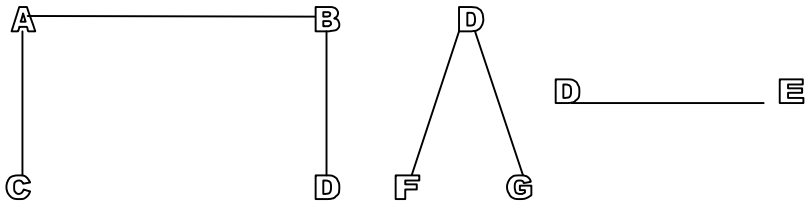


Figure 2. Un exemple d'une forêt.

Comme le graphe de la figure 2 est sans cycle et est non connexe (en fait, il possède 3 composantes connexes), on en déduit que c'est une forêt.

Par ailleurs, en appliquant le théorème qui précède, on constate que chacune des trois composantes connexes est un arbre.

8.3. Arbre couvrant

Soit: G = [X, U] un graphe.

Un arbre couvrant (ou encore recouvrant) de G est un graphe partiel de G qui est un arbre.

Autrement dit, un arbre couvrant de G est un sous graphe de G ayant le même ensemble X de sommets que G et qui est, en outre, un arbre.

Autrement dit, un arbre couvrant de G est un sous graphe de G qui est un arbre et qui recouvre tous les sommets de G.

Exemple

Considérons le graphe G donné par la figure 3.

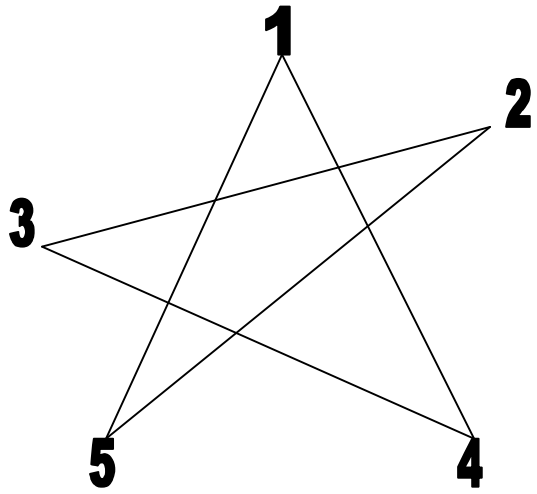


Figure 3.

Pour le graphe de la figure 3, un exemple d'arbre couvrant possible est donné par en rouge la figure3' suivante.

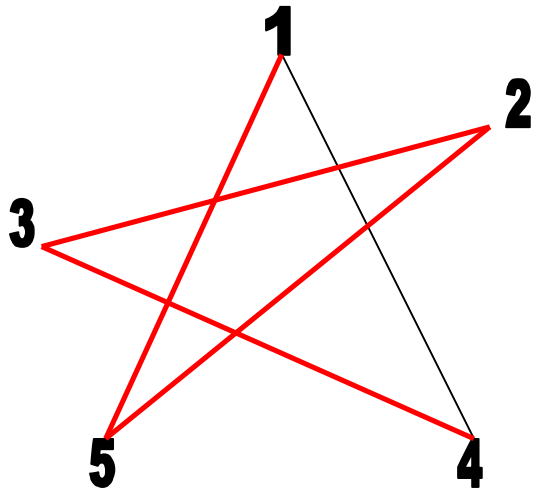


Figure 3'. Un exemple d'arbre couvrant de la figure 3.

Comme vous pouvez le constater, le graphe de la figure 3, admet d'autres arbres couvrants. Par exemple, l'arbre couvrant en rouge de la figure3'' qui suit.

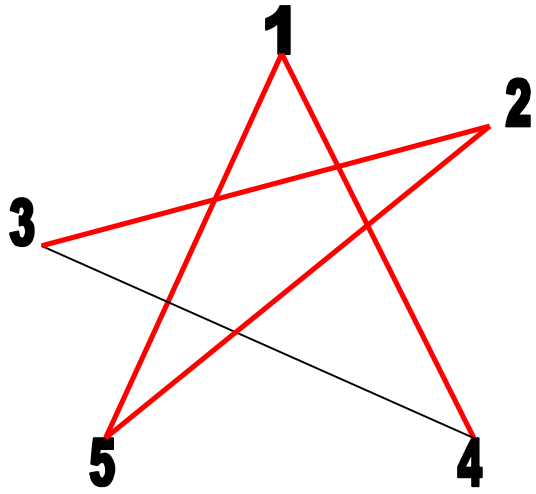


Figure 3''. Un exemple d'arbre couvrant de la figure 3.

Lemme : Un graphe admet un arbre couvrant si et seulement s'il est connexe.

Remarque

Comme le prouve l'exemple de la figure 3, un graphe connexe peut admettre plusieurs arbres couvrants.

Remarque

Quand le graphe n'est pas connexe, on parle alors de forêt couvrante.

8.4. Le problème de l'arbre couvrant de poids minimum

Soit: G = [X, U] un graphe connexe orienté ou non, et soit p une application de l'ensemble U vers l'ensemble des réels R et qui, à chaque arc ui (ou arête ui), fait correspondre p(ui) dit poids de l'arc ui (ou de l'arête ui).

Le problème de l'arbre couvrant de poids minimum consiste à chercher un arbre couvrant de G dont le poids est le plus petit parmi ceux de tous les arbres couvrants possibles de G.

Remarque

Si les arcs (ou arêtes) ont des poids tous différents, c.-à-d., l'application p est injective, **alors** l'arbre couvrant de poids minimum est unique.

Application: les problèmes d'optimisation des réseaux valués avec recouvrement de tous les sommets.

Les arbres couvrants de poids minimum entrent en jeu dans les applications que l'on peut modéliser sous forme d'un graphe valué et où l'on souhaite, d'une part, que tous les sommets du graphe puissent communiquer entre eux directement ou indirectement et, d'autre part, ne préserver que les arêtes non redondantes et ce, afin de faire des économies.

Par exemple, on pourra faire appel aux arbres couvrants de poids minimum (ou maximum) si l'on souhaite optimiser le coût d'installation d'un réseau (électrique, communication, routier, etc.) tout en passant par tous les sommets du réseau et tout en connaissant le coût de chacune des arêtes du réseau.

8.5. Algorithmes de recherche d'un arbre couvrant de poids minimum

Nous allons étudier trois algorithmes de construction d'un arbre couvrant de poids minimum dans un graphe valué, à savoir, l'algorithme de kruskal, l'algorithme de Prim et l'algorithme de Sollin.

8.5.1. Algorithme de Kruskal pour construire un arbre de poids minimum

Le principe de cet algorithme est le suivant.

1. On numérote les arêtes dans l'ordre des poids croissants, c-à-d, p(ul) < p(u2) < ... < p(um).
2. On pose U' = ∅.
3. A la ième étape (i = 1, 2, ..., m), on considère l'arête ui. Si ui ne forme pas de cycle avec les arêtes de U', on va à l'étape 4. Sinon, on va à l'étape 5.
4. On ajoute l'arête ui à U'.
5. Si i < m, on incrémente i de 1. Sinon, si i = m, on s'arrête en concluant que [X, U'] est un arbre (car G est connexe) et que [X, U'] est de poids minimum (par construction).

Procédure Kruskal (donnée G = [X, U] : graphe; donnée p : poids ; résultat U' **: ensemble des arêtes de poids minimum) ;**

{*On suppose que le graphe est connexe et que les arêtes sont numérotées dans l'ordre des poids croissants*}

U'= ∅ ;

Début

Pour i := 1 à m faire

Si [X, U' ∪ {ui}] ne forme pas de cycle alors

U' := U' ∪ {ui}

Fin Si

Fin Pour

Fin;

Exemple

Considérons le graphe G donné par la figure 4 et déterminons un arbre couvrant de poids minimum et ce, en utilisant l'algorithme de Kruskal.

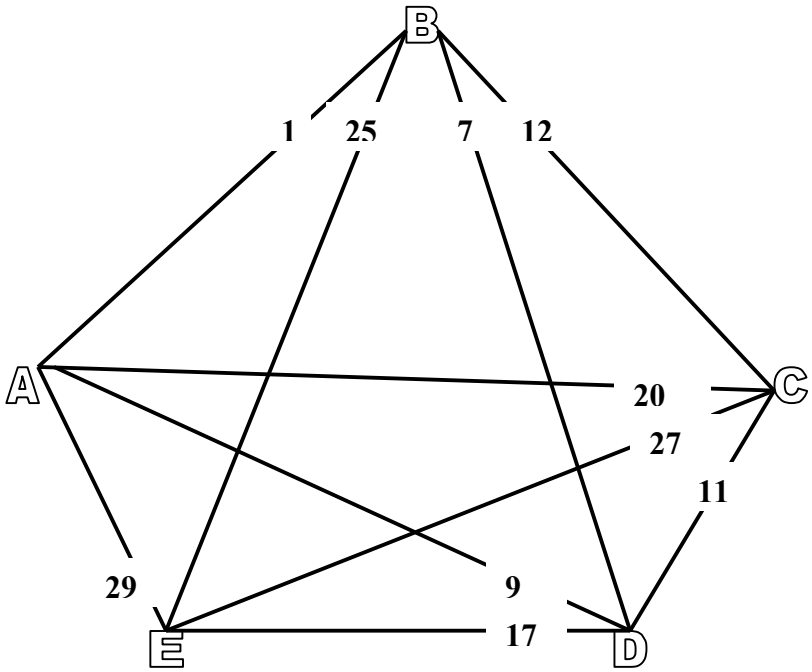


Figure 4.

✓ Tout d'abord, on va commencer par numéroter les arêtes dans l'ordre des poids croissants.

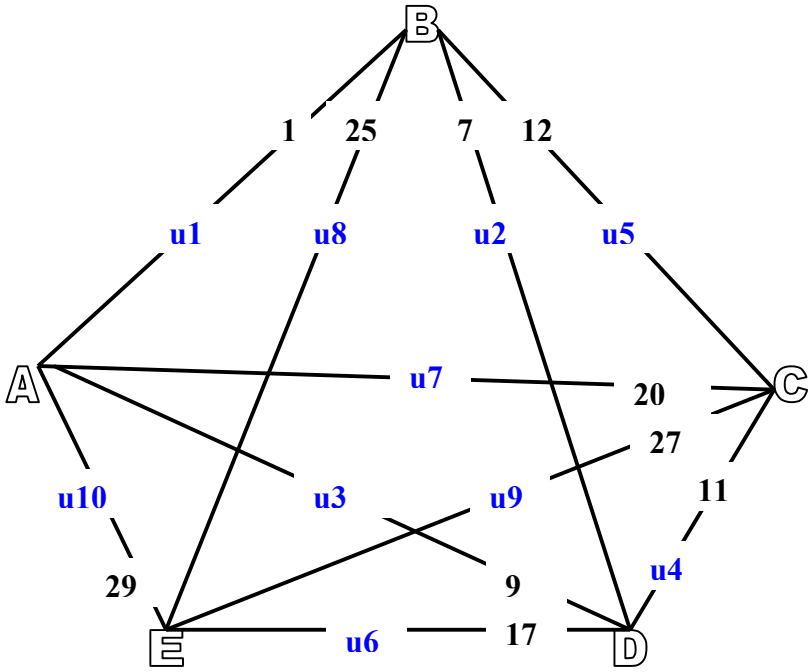


Figure 4.

✓ Ensuite, on considère les arêtes selon l'ordre des poids croissants en ne construisant que les arêtes qui ne forment pas de cycle avec celles déjà construites.

Ainsi, à la première étape de l'algorithme de Kruskal, on considère la première arête u1 et on construit u1.

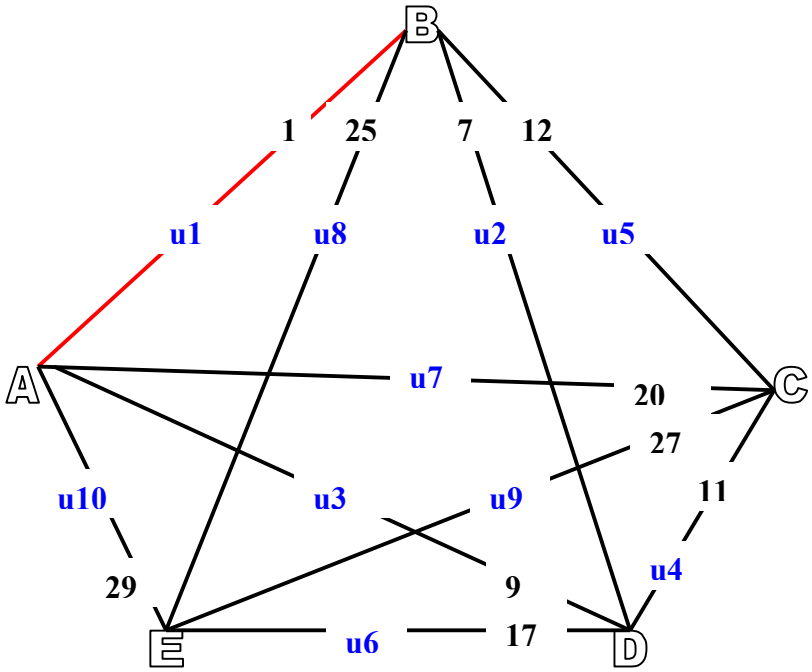


Figure 4.

A la 2ème étape de l'algorithme de Kruskal, on considère la deuxième arête u2 et on construit u2 car u1 et u2 ne forment pas de cycle.

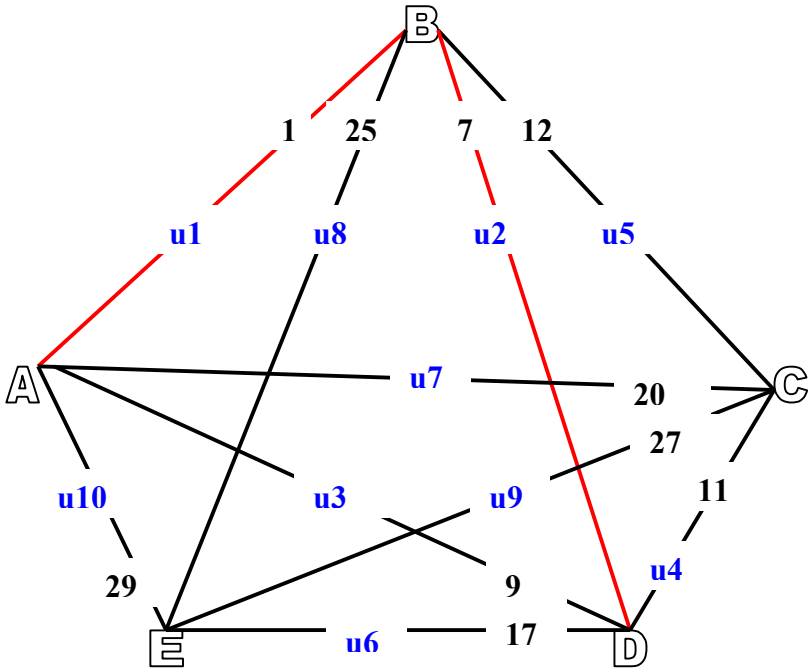


Figure 4.

A la 3ème étape de l'algorithme de Kruskal, on considère la 3ème arête u3, mais on ne la construit pas car elle forme un cycle avec les arêtes déjà construites, à savoir, avec les arêtes u1 et u2.

A la 4ème étape de l'algorithme de Kruskal, on considère la 4ème arête u4 et on construit u4 car u4 ne forme pas de cycle avec u1 et u2.

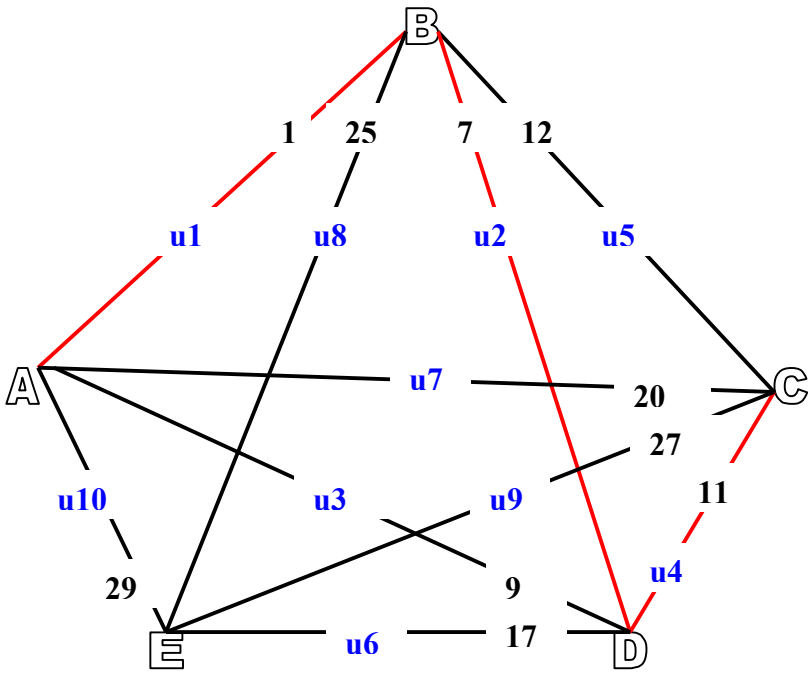


Figure 4.

A la 5ème étape de l'algorithme de Kruskal, on considère la 5ème arête u5, mais on ne la construit pas car elle forme un cycle avec les arêtes u2 et u4.

A la 6ème étape de l'algorithme de Kruskal, on considère la 6ème arête u6, et on construit u6 car u6 ne forme pas de cycle avec les arêtes déjà construites.

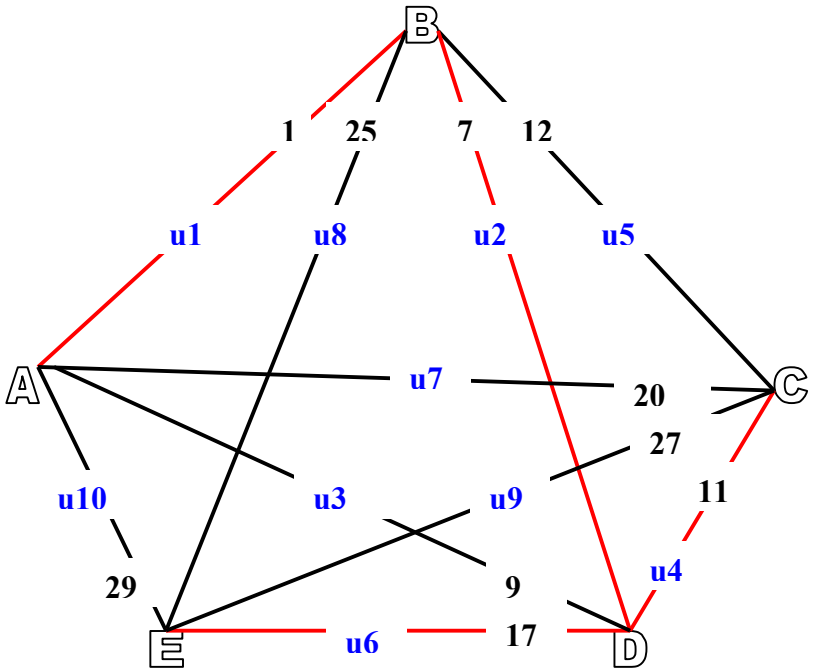


Figure 4.

A la 7ème étape de l'algorithme de Kruskal, on considère la 7ème arête u7, mais on ne la construit pas car elle forme un cycle avec les arêtes u1, u2 et u4.

A la 8ème étape de l'algorithme de Kruskal, on considère la 8ème arête u8, mais on ne la construit pas car elle forme un cycle avec les arêtes u2 et u6.

A la 9ème étape de l'algorithme de Kruskal, on considère la 9ème arête u9, mais on ne la construit pas car elle forme un cycle avec les arêtes u4 et u6.

A la 10ème étape de l'algorithme de Kruskal, on considère la 10ème arête u10, mais on ne la construit pas car elle forme un cycle avec les arêtes u1, u2 et u6.

✓ Maintenant, comme on a balayé toutes les arêtes du graphe, on s'arrête en concluant que l'on a obtenu un arbre couvrant de poids minimum. Cet arbre est celui donné en rouge par la figure 4' et son poids est de (1 + 7 + 11 + 17) = 36.

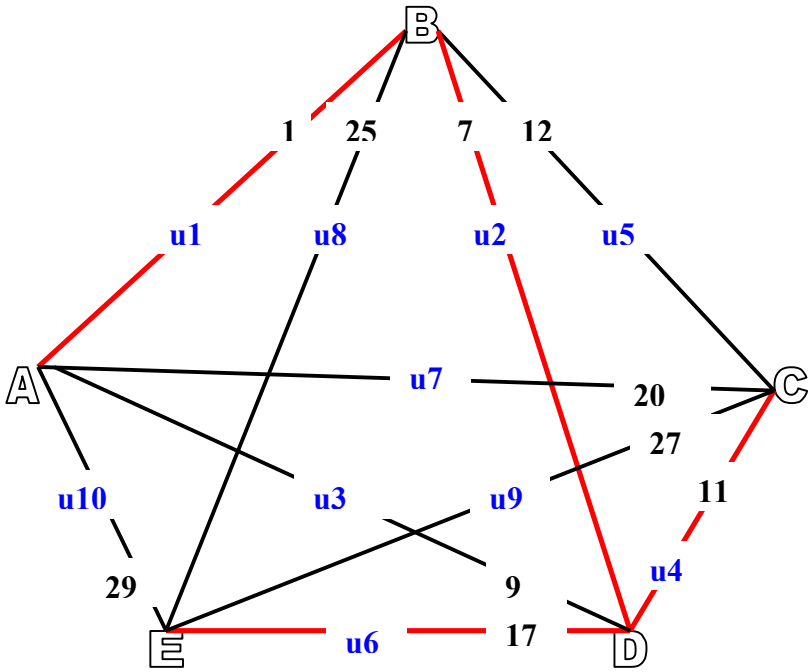


Figure 4'. L'arbre couvrant de poids minimum associé à la figure 4.

Remarque

Comme vous pouvez le constater, comme toutes les arêtes ont des poids tous différents, alors l'arbre couvrant de poids minimum associé à la figure 4 est unique. Par ailleurs, on notera, comme cela est attendu, que l'arbre engendré par Kruskal est connexe et comporte 5 sommets et 5- 1 = 4 arêtes.

Remarque

Si le graphe n'était pas connexe, alors l'algorithme de Kruskal aurait engendré une forêt couvrante de poids minimum.

8.5.2. Algorithme de Prim pour construire un arbre de poids minimum

C'est un algorithme qui consiste à construire un arbre en s'appuyant sur la propriété de connexité. Le principe de cet algorithme est le suivant.

1. On choisit un sommet x0 quelconque de départ.
2. On pose U' = ∅ et S = {x0}.
3. On ajoute, parmi toutes les arêtes d'extrémité x0, l'arête de poids minimum.
4. Tant que l'on n'a pas couvert l'ensemble des sommets du graphe, on cherche une arête ui de poids minimum parmi toutes les arêtes dont seule une extrémité ei n'est pas dans S.
5. On ajoute l'arête ui à U'.
6. On ajoute le sommet ei à S.
7. Quand tous les sommets du graphe deviennent dans S, on s'arrête en concluant que [X, U'] est un arbre.

Procédure Prim (donnée G = [X, U] : graphe; donnée p : poids ; résultat U' **: ensemble des arêtes de poids minimum ; résultat** S' **: ensemble des extrémités de U') ;**

U'= ∅ ;

S = {x0} ;

Début

Tant que S ≠ X faire

Trouver une arête ui = (x, y) de poids minimum tel que x ∈ S et y ∉ S

U' := U' ∪ {ui}

S := S ∪ {y}

Fin Tant que

Fin;

Exemple

Reprenons le graphe G donné par la figure 4 et déterminons un arbre couvrant de poids minimum et ce, en utilisant l'algorithme de Prim. Par ailleurs, supposons que le point de départ soit A.

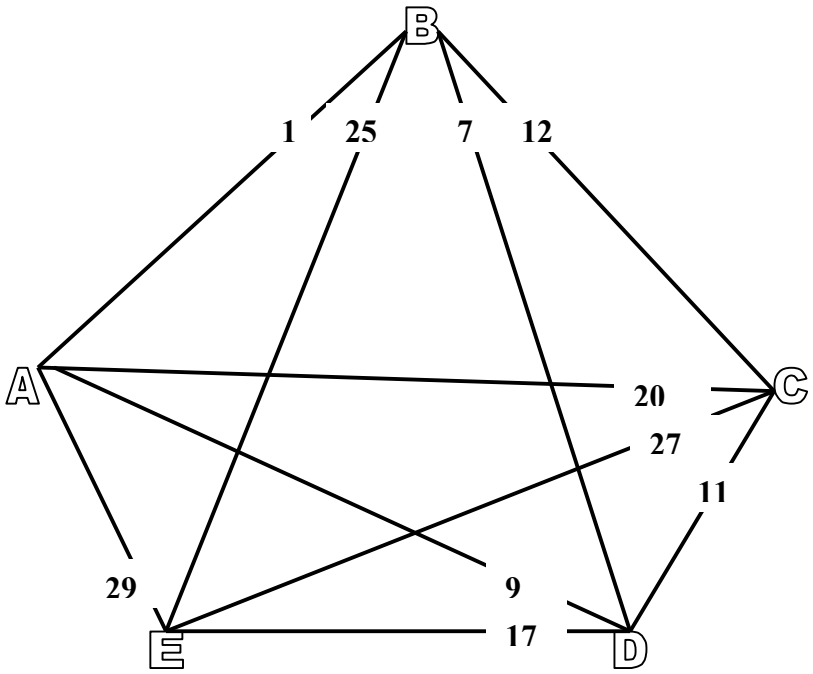


Figure 4.

La simulation de l'algorithme de Prim par rapport au graphe de la figure 4 est donnée par le tableau qui suit.

Sommet choisi	Arête retenue	Sommets marqués
A	AB	{A, B}
D	BD	{A, B, D}
C	DC	{A, B, C, D}
E	DE	{A, B, C, D, E}

Le graphe ainsi obtenu est un arbre couvrant de poids minimum. Il est donné en rouge par la figure 4'' et son poids vaut 36.

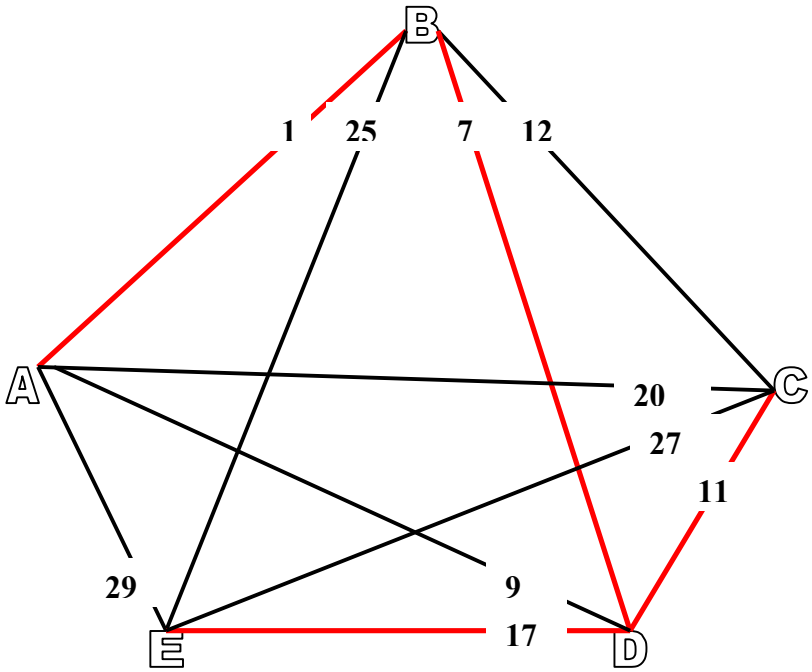


Figure 4''. L'arbre couvrant de poids minimum associé à la figure 4.

Remarque

L'arbre couvrant de poids minimum engendré par l'algorithme de Prim est le même que celui obtenu via l'utilisation de l'algorithme de Kruskal.

Remarque

Si le graphe n'était pas connexe, alors l'algorithme de Prim aurait engendré une forêt couvrante de poids minimum.

8.5.3. Algorithme de Sollin pour construire un arbre de poids minimum

L'algorithme de Sollin est une version hybride de l'algorithme de kruskal et de l'algorithme de Prim. Son principe est le suivant.

1. On considère les sommets selon l'ordre lexicographique.
2. Tant que l'on n'a pas couvert l'ensemble des sommets du graphe,
   - On choisit un sommet x non marqué.
   - On choisit l'arête (x, y) qui, d'une part, a le sommet x choisi comme une de ses extrémités et, d'autre part, est de poids minimum parmi toutes les arêtes admissibles et non encore sélectionnées.

- Si le sommet y n'est pas déjà marqué, on le marque.

3. S'il n'y a qu'une composante connexe, on s'arrête en concluant que l'on a engendré un arbre couvrant de poids minimum. Sinon, on va à l'étape 4.
4. On démarque tous les sommets du graphe.
5. On assimile chaque composante connexe à un sommet et on va à l'étape 1.

Exemple 1

Considérons le graphe G donné par la figure 5 et déterminons un arbre couvrant de poids minimum et ce, en utilisant l'algorithme de Sollin.

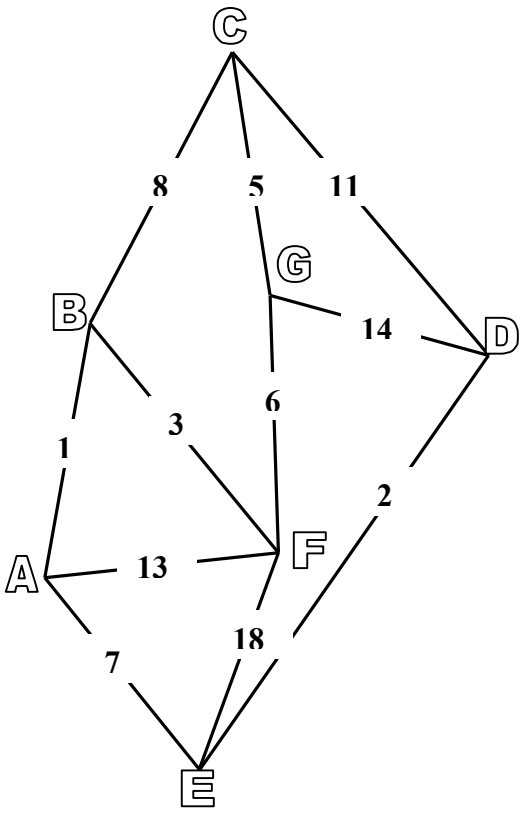
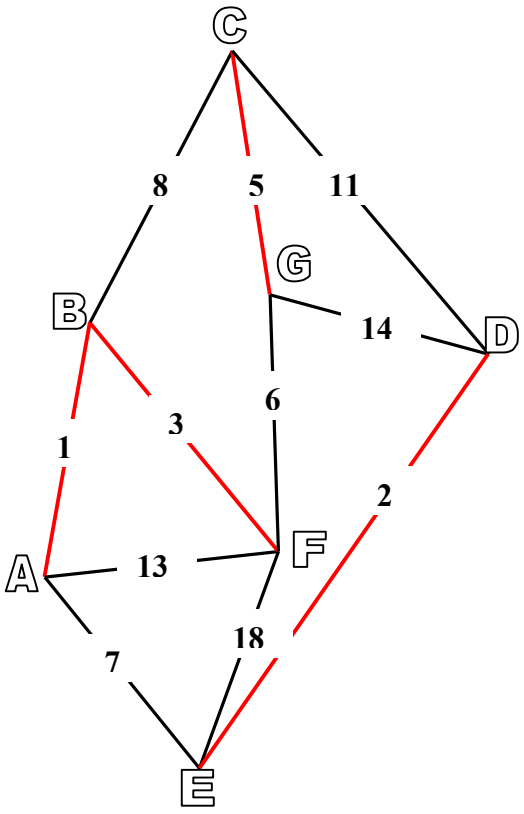


Figure 5.

La simulation de l'algorithme de Sollin par rapport au graphe de la figure 5 est donnée par le tableau qui suit.

Sommet choisi	Arête retenue	Sommets marqués
A	AB	{A, B}
C	CG	{A, B, C, G}
D	DE	{A, B, C, G, D, E}
F	FB	{A, B, C, G, D, E, F}

Le graphe ainsi obtenu et le suivant.



Comme vous pouvez le constater, par rapport au graphe engendré, on a 3 composantes connexes : C1 = {A, B, F}, C2 = {C, G} et C3 = {D, E}.

Maintenant, comme tous les sommets ont été marqués, mais que le nombre d'arêtes du graphe engendré n'est pas égal à 6 (nombre de sommets – 1), alors on démarque tous les sommets et on réitère le traitement précédent en assimilant chaque composante connexe à un sommet.

Composante connexe choisie	Arête retenue	Sommets marqués
C1	GF	{A, B, F, C, G}
C3	AE	{A, B, F, C, G, E, D}

Remarques :

- ✓ L'arête GF a été retenue, car :

  1. Une de ses extrémités est dans C1 et l'autre extrémité est dans une autre composante connexe ;

  2. Elle est de est de poids minimum parmi toutes les arêtes non encore retenues et vérifiant 1.

- ✓ Après avoir retenue l'arête GF, on marque tous les sommets des composantes connexes C1 et C2 (car F ∈ C1 et G ∈ C2).

Le graphe ainsi obtenu est celui donné en rouge par la figure 5'.

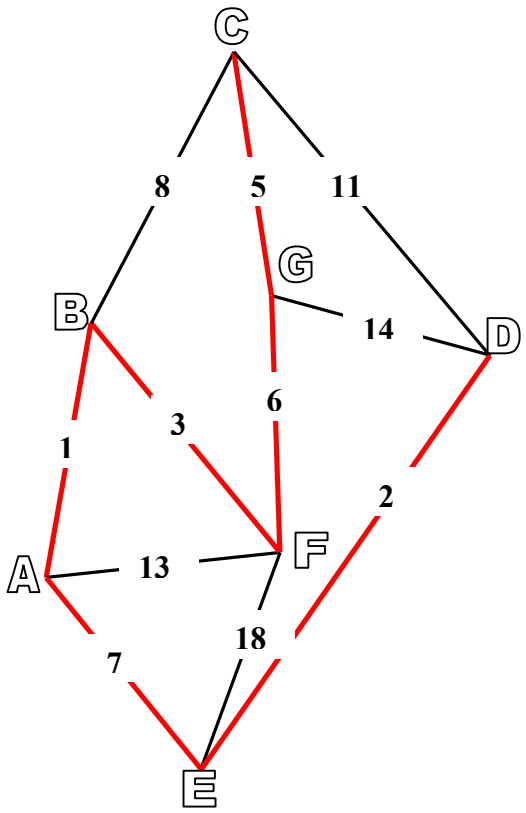


Figure 5'. L'arbre couvrant associé à la figure 5.

Maintenant, comme tous les sommets ont été marqués et qu'il n'y a qu'une composante connexe, alors on s'arrête en concluant que l'on a trouvé l'arbre couvrant de poids minimum et ce, en utilisant l'algorithme de Sollin. Le poids de cet arbre vaut 2 + 7 + 1 + 3 + 6 + 5, soit 24.

Exemple 2

Reprenons le graphe G donné par la figure 4 et déterminons un arbre couvrant de poids minimum et ce, en utilisant l'algorithme de Sollin.

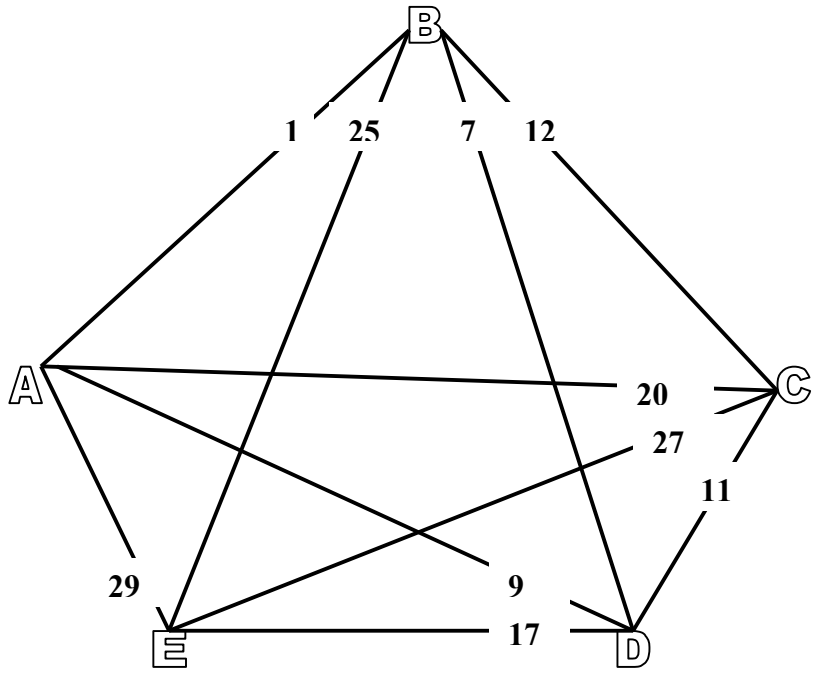
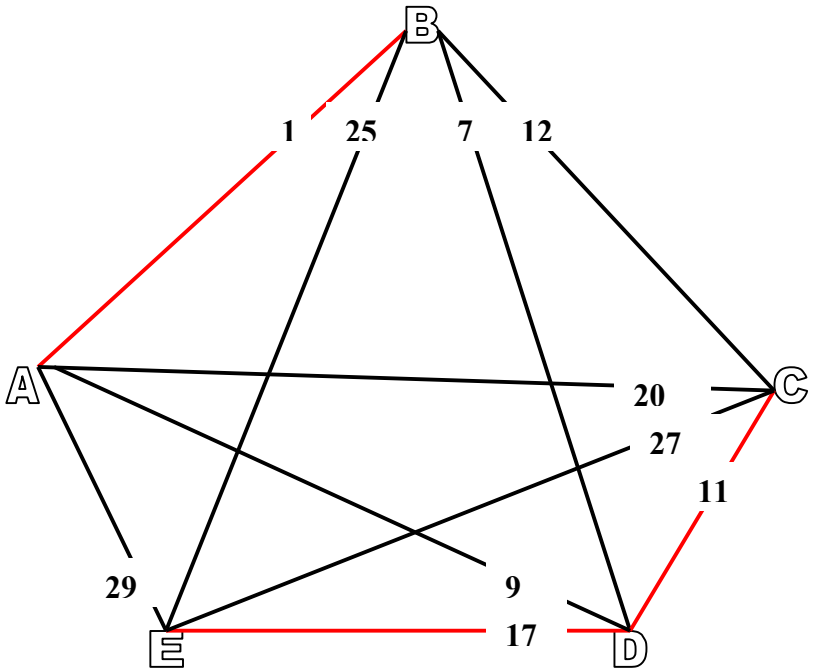


Figure 4.

La simulation de l'algorithme de Sollin par rapport au graphe de la figure 4 est donnée par le tableau qui suit.

Sommet choisi	Arête retenue	Sommets marqués
A	AB	{A, B}
C	CD	{A, B, C, D}
E	DE	{A, B, C, D, E}

Le graphe ainsi obtenu et le suivant.



Maintenant, comme tous les sommets ont été marqués et qu'il y a deux composantes connexes (C1 = {A, B} et C2 = {C, D, E}), alors on démarque tous les sommets et on réitère le traitement précédent en assimilant chaque composante connexe à un sommet.

Composante connexe choisie	Arête retenue	Sommets marqués
C1	BD	{A, B, C, D, E}

Maintenant, comme tous les sommets ont été marqués et qu'il n'y a qu'une composante connexe, alors on s'arrête en concluant que l'on a trouvé l'arbre couvrant de poids minimum et ce, en utilisant l'algorithme de Sollin. Le poids de cet arbre vaut 36.

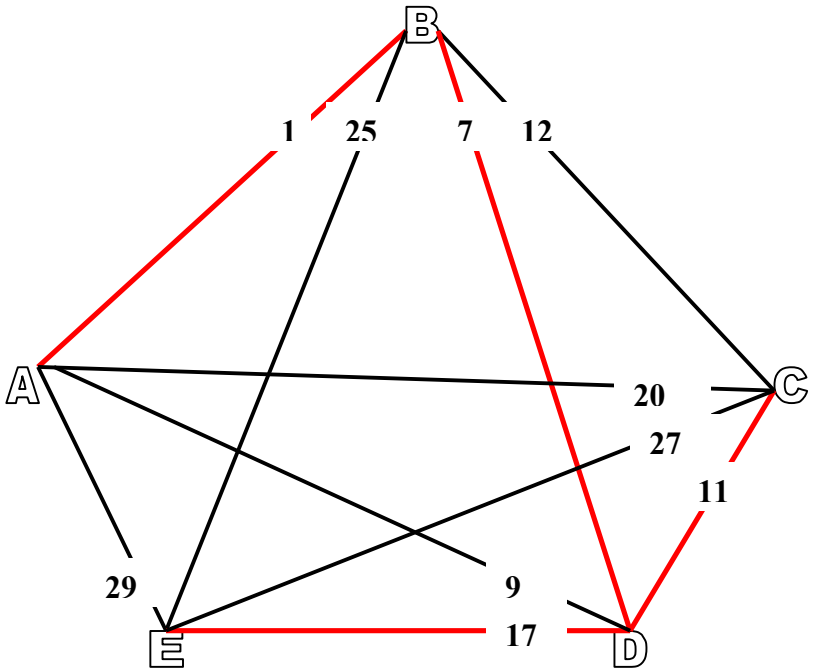


Figure 4''. L'arbre couvrant de poids minimum associé à la figure 4.

Remarque

L'arbre couvrant de poids minimum engendré par l'algorithme de sollin est le même que celui obtenu via l'utilisation de l'algorithme de Kruskal ou de l'algorithme de Prim.