Arbre de couverture optimal

1. Définition d'un arbre

Soit $G=(X,U)$ un graphe orienté d'ordre $n \geq 2$. G est un **arbre** si l'une des **six (6) propriétés** suivantes est vérifiée :

1. G est connexe et sans cycles.
2. G est sans cycles et admet n-1 arcs.
3. G est connexe et admet n-1 arcs.
4. G est sans cycle maximal (tout arc supplémentaire créé un cycle dans G).
5. G est connexe minimal (la suppression d'un arc quelconque le rend non connexe).
6. Pour toute paire de sommets ($x, y \in X, x \neq y$) Il existe dans G une chaîne et une seule joignant x à y.

Les 6 caractéristiques ci-dessus (propres aux arbres) sont équivalentes.
Ceci est valable pour un graphe non-orienté.

2. Codage de Prufer

Un arbre représenté par un graphe d'ordre n peut être représenté par séquence ordonnée de n-2 éléments. Chaque élément représente le numéro (indice) d'un sommet. Pour cela, chaque sommet du graphe doit être numéroté. Dans ce qui suit, nous présentons un algorithme de codage qui génère pour un graphe arbre le code Prufer correspondant ainsi qu'un algorithme de décodage qui génère un graphe arbre à partir d'un code de Prufer.

2.1 Algorithme de codage

Principe

A partir d'un graphe qui est arbre d'ordre n (dont les sommets sont numérotés 1, 2, …, n), l'algorithme rajoute pour chaque feuille (sommet de degré 1) ayant le plus petit numéro son voisin dans le code de Prufer et supprime cette feuille ainsi que l'arête qui lui est incidente. L'algorithme s'arrête après n-2 itérations. En d'autres termes, l'algorithme se termine après avoir inséré dans le code de Prufer n-2 sommets.

Donnée

En Entrée : Le graphe arbre $G=(X,E)$
En Sortie : Une suite ordonnée de sommets P
Autres :
x : variable sommet

L'algorithme

```
Début
      P ← ∅;
      Tant que |X|>2
          Faire
                 Choisir x dans X tel que dG(x)=1 et x minimal; // feuille de numéro minimal
                 P ← P . Adjacent(x);
                 // Rajouter le sommet adjacent à x dans la liste ordonnée P
                 X ← X - {x}; // Supprimer le sommet x
                 E ← E - {x, Adjacent(x)}; // Supprimer l'arête incidente à x
          Fait
Fin.
```

2.2 Algorithme de décodage

Principe

A partir d'un code de Prufer P qui est une suite de n-2 numéros de sommets, où n est l'ordre du graphe, l'algorithme va générer un graphe arbre d'ordre n. L'algorithme calcule d'abord les degrés des sommets à partir de P. Par la suite, pour chaque sommet qui apparaît dans P, il crée une arête vers le sommet de degré 1 puis décrémente les degrés des deux extrémités de cette arête. A la fin, les deux sommets restants ayant un degré égal à 1 seront reliés par une arête.

Donnée

En Entrée : Le code de Prufer sous forme d'une liste ordonnée *P*
En Sortie : Un graphe non-orienté qui est arbre *G=(X, E)*
Autres :
x : variable sommet
D : vecteur de n éléments. Chaque *D*[*i*] représente au début le degré de *i*.

L'algorithme

```
Début
      n ← longueur(P)+ 2;
      X ← ∅;
      E ← ∅;
      Pour i de 1 à n
            Faire
                  X ← X ∪ {i};
                  D[i] ← 1;
            Fait
      Pour chaque valeur j de P
            Faire
                  D[i] ← D[i] + 1;
            Fait
      Pour chaque valeur j de P
            Faire
                  Chercher k tel que D[k]=1 et k minimal;
                  E ← E ∪ {j,k};
                  D[j] ← D[j]-1;
                  D[k] ← 0;
            Fait
      Relier les deux sommets ayant D[i]=1;
Fin.
```

3. Graphe pondéré (Réseau)

3.1 Poids d'une arête

Soit *G=(X,E)* un graphe non-orienté,
On définit $p : E \rightarrow \mathcal{R}$ une application qui associe pour chaque arête $e \in E$ de *G* une valeur réelle *p(e)* appelée poids de l'arête *e*.

3.2 Poids d'un arc

Soit *G=(X,U)* un graphe orienté,
On définit $p : U \rightarrow \mathcal{R}$ une application qui associe pour chaque arc $u \in U$ de *G* une valeur réelle *p(u)* appelée poids de l'arc *u*.

3.3 Poids d'un arc

Soit *G=(X, U, p)* (resp. *G=(X, E, p)*) un graphe doté d'une application *p* de pondération des arcs (resp. des arêtes).
Un tel graphe est appelé graphe pondéré, graphe valué ou réseau.

4. Arbre de couverture d'un graphe

Soit *G=(X,U)* un graphe orienté d'ordre $n \geq 2$. On appelle arbre dans *G* un sous-graphe partiel de *G*, *H=(Y,V)* connexe et sans cycles. Un arbre est maximal dans *G* s'il contient le maximum possible de sommets de *G* (c'est-à-dire *Y=X* si *G* est connexe).

5. Arbre de couverture de poids optimal

5.1 Identification du problème

Soit *G=(X,U)* un graphe orienté connexe muni d'une application poids *p*.
La recherche dans *G* d'un arbre de poids optimal revient à trouver un graphe partiel *G'=(X,U')* de *G* qui soit un arbre et pour lequel la somme des poids des arcs de *G'* soit optimale (maximale ou minimale selon la situation).

5.2 Algorithme de Kruksal

Principe

Dans cet algorithme, il faut d'abord trier dans un tableau les arcs du graphe selon l'ordre croissant des poids. Le tableau trié doit être parcouru dans l'ordre. Un arc est sélectionné s'il ne forme pas avec les arcs déjà sélectionnés un cycle. Les (n-1) premiers arcs sélectionnés forment l'arbre recouvrant de poids minimal.

Donnée

En Entrée : Le graphe pondéré (réseau) $G=(X,U,p)$
En Sortie : Ensemble d'arcs H
Autres :
x : variable sommet
u : variable arc
V : tableau d'arcs trié dans un ordre de poids croissants ($V[i] \leq V[i+1]$)
n : nombre de sommets dans G, c'est-à-dire $n=|X|$

L'algorithme

```
Début
     V = tri_poids_croissant(U,p);
     H ← {V[1]};
     i ← 1 ;       j ← 1;
     Tant que (j < n-1)
     Faire
          i ← i+1; u ← V[i];
          Si (H ∪ {u} ne contient pas de cycle)
            Alors H ← H ∪ {u}; j ← j+1; // L'arc sélectionné ne doit pas créer de cycle
          fSi
     Fait
Fin.
```

Remarques

- L'algorithme a une complexité de l'ordre de $o(n^2)$.
- On peut utiliser l'algorithme sur un graphe non-orienté.
- On peut trier les arcs dans un ordre décroissant. Dans ce cas, on met tous les arcs dans H et on supprime dans l'ordre les arcs si ça ne déconnecte pas le graphe. On s'arrête quand on aura n-1 arcs.
- Il est possible d'utiliser le même algorithme pour trouver l'arbre de couverture poids maximal.

5.3 Définition (Contraction)

Soit $G=(X,U)$ un graphe et $u=(x,y) \in U$. On appelle contraction, le graphe $G_u=(X_u,U_u)$ obtenu à partir de G comme suit :

- Les sommets x et y seront remplacés par un sommet unique $c\{x,y\}$ dans G_u.
- Pour tout arc v_l dans G ayant x ou y comme extrémité initiale (resp. terminale), on lui remplace dans G_u son extrémité initiale (resp. terminale) par $c\{x,y\}$.
- On supprime l'arc u dans G_u.

5.4 Théorème 1

Soit $G=(X,U)$ un graphe et $u=(x,y) \in U$. G est un arbre si et seulement si G_u est un arbre.

5.5 Théorème 2

Soit $G=(X,U,p)$ un graphe connexe et $x \in X$.
Soit U_x un sous-ensemble de U défini comme suit : $U_x=\{u \in U / (I(u)=x \text{ ou } T(u)=x) \text{ et } u \neq (x,x)\}$
Soit $u(x)$ l'arc de U_x ayant le plus grand poids : $u(x) / p(u(x)) = min\{p(u) / u \in U_x\}$.
L'arbre $H=(X,V) / V \subset U$ et $u(x) \in V$ est un arbre de poids minimum.

5.6 Algorithme de Prim

Principe

Cet algorithme se base sur les deux théorèmes précédents. A partir d'un sommet x quelconque, on sait que l'arc adjacent à ce sommet de poids minimum ($u(x)$) fait certainement partie de l'arbre de couverture de poids minimum. A partir de cet arc là, on fait une contraction du graphe.

On répète ces opérations jusqu'à ce que le graphe soit contracté en un seul sommet. L'ensemble *V* formé des différents arcs $u(x)$ obtenus à chaque itération donnera l'arbre de poids minimal.

Donnée

En Entrée : Le graphe pondéré (réseau) *R=(X,U,p)*
En Sortie : Ensemble d'arcs *H*
Autres :
x : variable sommet
u : variable arc
u(x) : arc de poids minimal dont l'une des extrémités est *x*
G : variable graphe valué
G_u : Graphe contraction de *G* par rapport à l'arc *u*

L'algorithme

```
Début
      G ← R ;
      H ← ∅;
      Tant que |X|≥ 2
      Faire
            Choisir x dans X;
            u ← u(x); // choisir l'arc u de poids min ayant x comme extrémité
            H ← H ∪ {u};
            G ← Gu ; // Contracter G par rapport à l'arc u
      Fait
Fin.
```

Remarques

- L'algorithme a une complexité de l'ordre de $o(n^2)$.
- Il est possible d'utiliser le même algorithme pour trouver l'arbre de couverture poids maximal, en choisissant à chaque itération l'arc de poids maximal.