Algorithmique — Maîtrise
TD 7
Arbres couvrants minimaux

Isabelle Fagnot, Mihaela Sighireanu, Yan Jurski, Olivier Serre

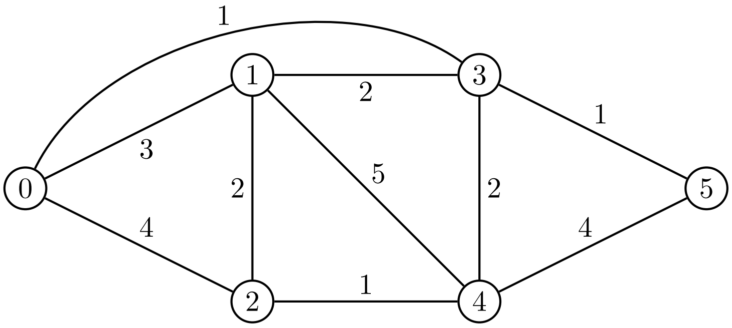

FIG. 1 – Graphe valué non orienté

1 Arbres couvrants minimaux : Algorithme générique

Soit un graphe valué non orienté $G = (V, E)$ et soit $A \subseteq E$, un sous-ensemble d'arêtes incluses dans un arbre couvrant minimal de G. Une arête (u, v) est *sûre* pour A si l'ensemble $A \cup \{(u, v)\}$ est inclus dans un arbre couvrant minimal de G.

On a alors l'algorithme générique suivant de calcul d'un arbre couvrant minimal :

```
Generique-MST(G,w)
  A ensemble vide
  Tant que A ne forme pas un arbre couvrant minimal faire
     Choisir une arête sure (u,v) pour A
        A = A union {(u,v)}
  Retourner A
```

<u>Exercice</u> 1 :

1. Proposer un invariant pour cet algorithme et prouver sa terminaison.
2. Une *coupe* est une partition $(S, V \setminus S)$ de V. Une arête *traverse* une coupe $(S, V \setminus S)$ si elle part de S et arrive en $V \setminus S$. Une coupe respecte un sous-ensemble A d'arêtes s'il n'y a pas d'arête de A qui la traverse. Une arête est *claire* si elle traverse la coupe et est de poids minimal.
   Soit un sous-ensemble A de E qui est inclus dans un arbre couvrant minimal, soit $(S, V \setminus S)$ une coupe qui respecte A et soit (u, v) une arête claire pour la coupe. Montrer que (u, v) est sûre pour A.
3. Soit un sous-ensemble A de E qui est inclus dans un arbre couvrant minimal, et soit $C = (V_C, E_C)$ une composante connexe dans la forêt $G_A = (V, A)$. Montrer que si (u, v) est une arête claire reliant C à une autre composante connexe de G_A, alors (u, v) est sûre pour A.

2 Arbres couvrants minimaux : Algorithme de Prim

On donne l'algorithme de Prim :

```
MST-PRIM(G,w,s)
  clé : tableau d'entiers indexé par V
  Pi : tableau de sommets indexé par V
  Pour tout sommet u faire
     cle[u] = infini
     Pi[u] = NIL
  Fin du pour
  cle[s] = 0
  Q = file de priorité indexée par cle et initialisée à V
  Tant que Q est non vide faire
     u = extract_min(Q)
     Pour tout v dans Adj(u) faire
        Si v est dans Q et w(u,v) < clé[v]
           Pi[v] = u
           clé[v] = w(u,v)
        Fin du si
     Fin du Pour
   Fin du tant que
   Retourner Pi
```

<u>Exercice</u> 2 :

1. Faire tourner l'algorithme de Prim sur le graphe de la figure 1 en partant du sommet 0.
2. Prouver la validité de l'algorithme de Prim.
3. Quelle est la complexité de l'algorithme de Prim ?

3 Arbres couvrants minimaux : Algorithme de Kruskal

On donne l'algorithme de Kruskal :

```
MST-KRUSKAL(G,w)
  A : ensemble vide.
  Pour tout sommet u faire
      Make-Set(u)
  Trier les arêtes de E par ordre décroissant de poids.
  Pour toute arête (u,v) de E pris dans l'ordre croissant faire
      Si Find-Set(u) <> Find-Set(v) faire
            Ajouter (u,v) à A
            Union(u,v)
      Fin du si
  Fin du pour
  Retourner A
```

La procédure `Make-Set` crée un singleton contenant son argument. La procédure `Find-Set` retourne le représentant de l'ensemble qui contient son argument. Enfin, `Union` fusionne les ensembles contenant ses deux arguments.

Exercice 3 :

1. Faire tourner l'algorithme de Kruskal sur le graphe de la figure 1.
2. Prouver la validité de l'algorithme de Kruskal.
3. Quelle est la complexité de l'algorithme de Kruskal ?

4 Divers

Exercice 4 :

1. Étant donné un graphe G et un arbre couvrant minimal T pour G, peut-il exister une arête (u, v) de G de poids minimal et qui n'appartienne pas à T ? Donner un exemple le cas échéant.
2. Soit (u, v) une arête de poids minimal dans un graphe G. Montrer que (u, v) appartient à *un* arbre couvrant minimal de G.