

# Algorithmique Avancée - TD 4 - 8 novembre 2011

Université Paris 7 — BioInformatique — M1

## Arbres couvrants de poids minimum

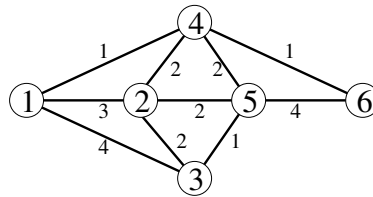
### 1 La Minute Historique

Le problème de trouver un arbre couvrant de poids minimum est un problème très ancien. Et aussi bizarre que cela puisse paraître, c'est un problème qui est même plus vieux que l'informatique elle-même. En effet en 1926, Otakar Borůvka fut un des premiers (si ce n'est le premier) à fournir un algorithme pour trouver "efficacement" un arbre couvrant de poids minimum. La motivation d'alors était de concevoir un réseau de distribution électrique pour la Moravie du Sud. Avec la contrainte évidente : que cela coûte le moins possible. Ce n'est qu'au milieu des années 50 et l'avènement de l'informatique que ce problème est revenu à la mode. C'est à Joseph B. Kruskal (1956) et à Robert Prim (1957) que nous devons les algorithmes que nous allons étudier aujourd'hui.

### 2 Échauffement

On considère des graphes non-orientés *pondérés* : à chaque arête  $(u, v)$  est associé un nombre (réel)  $p(u, v)$ . Étant donné un graphe  $G$ , un arbre couvrant de poids minimum de  $G$  est un arbre

1. qui a les même sommets que le graphe
2. dont chaque arête (avec son poids) est une arête du graphe
3. tel que la somme des poids est la plus petite parmi tous les arbres vérifiant les propriétés 1 et 2



**Exercice 1** Trouvez *tous* les arbres couvrant de poids minimum du graphe ci-dessous

**Exercice 2** Rappeller pourquoi tout graphe admet au moins un arbre couvrant de poids minimum

### 3 Algorithme de Kruskal

On donne l'algorithme de Kruskal :

KRUSKAL(G)

F : ensemble vide.

Trier les arêtes de E par ordre croissant de poids.

Pour toute arête (u,v) de E prise dans l'ordre croissant faire

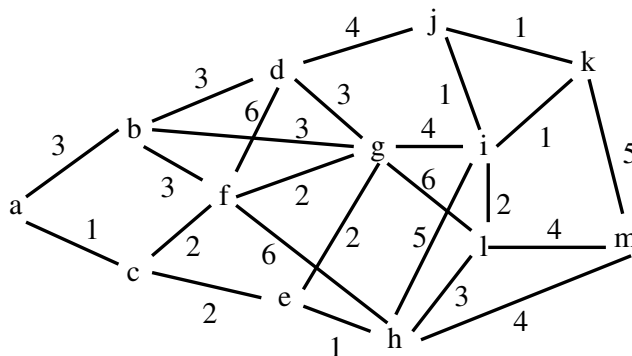
  Si ajouter (u,v) à F ne crée pas de cycle

    ajouter (u,v) à F

  Fin du si

Fin du pour

Retourner F



**Exercice 3** Faire tourner l'algorithme de Kruskal sur le graphe ci-dessus

On suppose que l'on dispose d'une structure de données stockant toutes les composantes connexes de A à un instant donné. Plus précisément :

- Une fonction booléenne `Find(u,v)` dit si deux sommets sont dans la même composante
- Une procédure `Union(u,v)` modifie la structure de données en remplaçant la composante contenant u et la composante contenant v par une nouvelle composante qui les contient tous les deux.

**Exercice 4** Réécrire l'algorithme en faisant appel à `Union` et à `Find`

**Exercice 5** Combien d'appels à `Union` et à `Find` faut-il faire ?

**Exercice 6** Proposez une implémentation de la structure de données des composantes

**Exercice 7** Déduisez-en la complexité d'un appel à `Union`, d'un appel à `Find` et de l'algorithme en général

## 4 Algorithme de Prim

On donne l'algorithme de Prim :

PRIM( $G, s$ )

Clé : tableau d'entiers indexé par  $V$  initialisé à  $[+\infty, +\infty, \dots, +\infty]$

Père : tableau de sommets indexé par  $V$  initialisé à  $[\text{null}, \text{null}, \dots, \text{null}]$

Marque : tableau de booléens indexé par  $V$  initialisé à  $[\text{faux}, \text{faux}, \dots, \text{faux}]$

Clé[s] = 0

Tant qu'il existe des sommets non-marqués faire

  Soit  $u$  un des sommets de Clé minimale

  Marquer  $u$

  Pour tout sommet  $v$  voisin de  $u$  faire

    Si  $v$  est non-marqué et  $p(u, v) < \text{Clé}[v]$

      Père[v] =  $u$

      Clé[v] =  $p(u, v)$

    Fin du si

  Fin du Pour

Fin du tant que

Retourner Père

- Exercice 8**
1. Faire tourner l'algorithme de Prim sur le graphe ci-dessus en partant du sommet 1.
  2. Proposer une structure de données pour stocker les clés. Quelle est la complexité des opérations
    - Trouver la clé minimale?
    - Modifier la clé?
  3. Réécrire l'algorithme de Prim avec cette structure de données
  4. En déduire la complexité de l'algorithme de Prim
  5. Comparez avec Kruskal

## 5 Extensions

**Exercice 9** [Arbre couvrant de poids max.] On se propose maintenant d'étudier le problème de trouver l'arbre couvrant de poids maximum.  $T$  est toujours couvrant mais cette fois pour tout arbre couvrant  $T'$  de  $G$  on cherche  $T$  tel que  $\mathbb{P}(T) \geq \mathbb{P}(T')$ .

Est-ce que ce problème est relié à l'ACPM?

Donnez un algorithme pour trouver un tel arbre.

Quelle est sa complexité?

**Exercice 10** [Arbre couvrant de poids min,  $\times$ .] On se propose maintenant d'étudier le problème de trouver l'arbre couvrant de poids minimum quand la fonction de valuation est défini comme :

$$\mathbb{P}(G) = \prod_{(u,v) \in E(G)} \mathbb{P}(u, v)$$

Est-ce que ce problème est relié à l'ACPM?

Donnez un algorithme pour trouver un tel arbre.

Quelle est sa complexité?