

# Algorithmique - ROB3

## TD7 - Algorithme de Prim

### - Exercice 1 (Algorithme de Prim)

Appliquer l'algorithme de Prim pour trouver un arbre couvrant de poids minimum du graphe  $G$  représenté sur la figure 1.

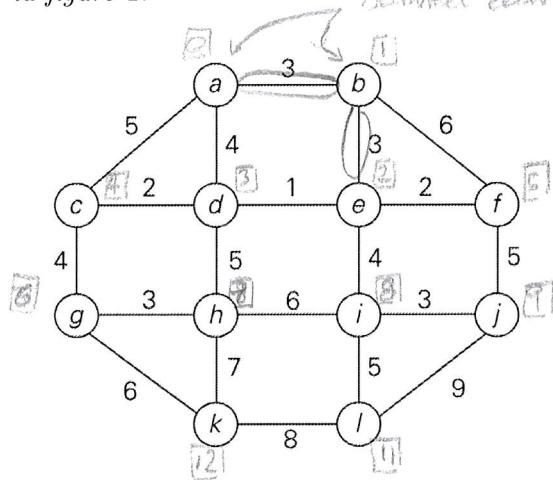


FIGURE 1 – Graphe  $G$

### - Exercice 2 (Vrai ou faux ?)

Indiquer si chacune des propositions suivantes est vraie (dans ce cas là, le prouver), ou fausse (donner un contre-exemple). On supposera que  $G = (S, A, v)$  est un graphe non-orienté connexe, avec  $n = |S|$  et  $m = |A|$ .

- Si le graphe  $G$  a plus de  $n - 1$  arêtes, et s'il y a une seule arête de coût maximum, alors cette arête ne peut pas faire partie d'un arbre couvrant de poids minimum.*
- Si  $G$  contient un cycle avec une unique arête  $e$  de coût maximum, alors  $e$  ne peut pas faire partie d'un arbre couvrant de poids minimum.*
- Soit  $e \in A$  une arête de coût minimum. Alors  $e$  appartient à un arbre couvrant de poids minimum de  $G$ .*
- Si l'arête de coût minimum de  $G$  est unique, alors elle fait nécessairement partie de tout arbre couvrant de poids minimum.*
- Si  $G$  contient un cycle avec une unique arête  $e$  de coût minimum, alors  $e$  fait partie de tout arbre couvrant de poids minimum.*

- f) La plus courte chaîne entre deux sommets fait nécessairement partie d'un arbre couvrant de poids minimum.
- g) L'algorithme de Prim est valide même quand les coûts des arêtes sont négatifs.

**Exercice 3 (Arbre couvrant de coût minimum sous contrainte)**

Soit un graphe valué  $G = (V, E, c)$  avec  $n = |V|$  et  $m = |E|$ , et un sous-ensemble  $U \subseteq V$ . Proposer un algorithme en  $O(m \log n)$  pour déterminer un arbre couvrant de coût minimum parmi les arbres couvrants dont tous les nœuds de  $U$  sont des feuilles.

**Exercice 4 (Arbre couvrant minimax)**

Dans cet exercice, on appelle coût maxitif  $c_{\max}(T)$  d'un arbre couvrant  $T$  le coût maximal d'une arête de  $T$ , et coût additif  $c(T)$  d'un arbre couvrant  $T$  la somme des coûts des arêtes de  $T$  (cette dernière fonction de coût correspond à celle vue en cours). Plus formellement, on a  $c_{\max}(T) = \max_{a \in T} c_a$  (où  $c_a$  le coût d'une arête  $a$ ) et  $c(T) = \sum_{a \in T} c_a$ . Par exemple, si on considère l'arbre couvrant en gras sur la figure ci-dessous, son coût maxitif est 2 et son coût additif est  $1 + 2 = 3$ .



On appelle arbre couvrant minimax un arbre couvrant de coût maxitif minimum, et arbre couvrant minimum un arbre couvrant de coût additif minimum (dans ce dernier cas, cela correspond à un arbre couvrant minimum tel que défini en cours). Sur l'exemple, l'arbre couvrant en gras est à la fois un arbre couvrant minimax et un arbre couvrant minimum.

**Question 4.1** “Tout arbre couvrant minimax est également un arbre couvrant minimum” : prouver cette assertion (dans le cas général) ou donner un contre-exemple.

**Question 4.2** “Tout arbre couvrant minimum est également un arbre couvrant minimax” : prouver cette assertion (dans le cas général) ou donner un contre-exemple.

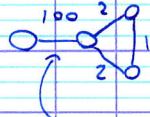
**Question 4.3** Etant donné un graphe  $G$  non orienté à  $n$  sommets et  $m$  arêtes, ainsi qu'une valeur  $b$ , donner un algorithme en  $O(n + m)$  afin de déterminer si la valeur d'un arbre couvrant minimax est inférieure ou égale à  $b$ .

## TD 7

### Exercice 2

a)

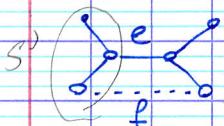
Fausse, contre-exemple:



Cette arête fais partie du tout arbre couvrant.

b)

Vrai: Par l'absurde: Soit  $T$  ACFM (Arbre Couvrant du paid  $\mathbb{N}^n$ ) qui contient  $e$ .



En supprimant  $e$  de  $T$  on crée un cocycle entre les 2 composantes connexes obtenue ( $S'$  et  $S \setminus S'$ )

Dans le cocycle obtenu, il existe nécessairement une arête  $f \neq e$  qui est dans  $C$ . En remplaçant  $e$  par  $f$ , on obtient un nouvel arbre couvrant  $T \cup \{f\} / \{e\}$

$$\text{Or } C(T \cup \{f\} / \{e\}) = C(T) - C(e) + C(f) < C(T)$$

car  $C(f) < C(e)$ .

Dans  $T$  on'est pas un ACFM. Contradiction.

$\Rightarrow$  Règle des cycles.

c)

Vrai. Soit  $\{u, v\}$  une arête du poids minimum dans  $G$ . Elle est en particulier de poids minimum dans le cycle entre  $\{u\}$  et  $S \setminus \{u\}$ . La règle des cocycles assure le résultat.

d)

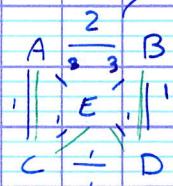
Vrai. Soit  $T$  un arbre qui ne comporte pas l'arête de poids minimum unique.  $T \cup \{e\}$  comporte un cycle. Soit  $f \in C \setminus \{e\}$

Alors  $T \cup \{e\} \setminus \{f\}$  a pour coût :  $C(T) - C(f) + C(e) < C(T)$   
 car  $C(e) < C(f)$ .

Donc un arbre couvrant de poids minimum ne comportant pas e ne peut être de poids min.

e)

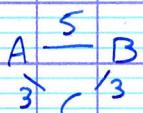
Faux



→ De poids minimum dans le cycle mais n'appartient pas à l'arbre couvrant de poids min.

f)

Faux :

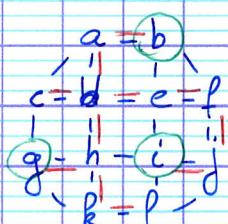


A-B est une plus courte chaîne entre A et B mais ne fait pas partie d'un arbre couvrant de poids minimum.

g)

Vrai : Prim est valide avec des poids positifs, et on peut ajouter une même valeur aux arêtes d'un graphe sans changer l'ACT.

### Esercizio 3

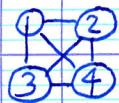


$$U = \{b, g, i, j\}$$

(cf eserc 1)

Soit un arbre couvrant  $T$  du coût min sur  $G$  parmi les arbres couvrants dont les noeuds  $U$  sont des feuilles. Si on considère le sous-graphe induit  $G[V \setminus U]$ , l'arbre  $T$  est

nécessairement du coût min. sur  $G[V \cup U]$  (en utilisant prim par exemple), puis à sélectionner une arête adjacente de poids min pour chaque sommet de  $U$ .



16 arbres couvrants.

$n^{n-2}$  arbres couvrants

K ↗ ↘ ↘ ↘

E ↗ ↘ ↗ ↗

Z ↙ N ↙ N

X ↗ X ↗ X

chemin commerce  
sans cycle

### TD 7 - Exercice 1

Examiner un sommet : sélectionner l'arête de poids minimum qui la relie à un sommet couvert.

Flâj des priorités des voisins non couverts du ce sommet.

Complexité (sans utiliser de tas) :

-  $n$  examen des sommets.

↳ identification d'un sommet de priorité min dans la bordure en  $O(n)$

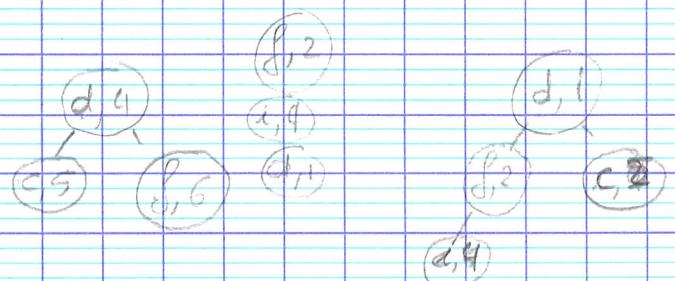
→  $O(m(n+m))$

Complexité (en utilisant un tas) :

→  $O((n+m)\log(n))$

$$2T(m/2) + \Theta(1)$$

$$\Theta(n)$$



Algorithme de Kruskal.

Trier les arêtes précédentes par poids croissant.

Examiner les arêtes dans cet ordre, en sélectionnant une arête si elle ne crée pas de cycle avec les arêtes

déjà sélectionnés.

On stoppe quand  $(n-1)$  amis sont sélectionnés.

Tester la création du cycle:

Union - Find pour maintenir les informations sur les composantes connexes du graphe partiel des amis sélectionnés:



Tester si une amitié  $\{a, b\}$  ne crée pas de cycle:

On identifie la racine  $\pi_1$  de l'arbre auquel appartient  $a$  dans union - find

On identifie la racine  $\pi_2$  .. "

"  $b$  ) dans union - find.

Si  $\pi_1 \neq \pi_2$  on sélectionne  $\{a, b\}$

sinon on ne sélectionne pas.

Union par défi: La racine comportant le moins de noeuds devient fils de la racine de l'arbre comportant le plus de noeuds.

Complexité en  $O(n \log n)$