M23 : Théorie des Graphes

M. Zimmermann

IUT Robert Schuman, Département Informatique

2022/23

Chapitre 4

Arbres

Remarque :

Avertissement aux lecteurs les plus sensibles : le vocabulaire de ce chapitre est sujet, encore une fois, à certains soucis de normalisation. Ici, le souci vient souvent du fait que l'informatique utilise beaucoup la notion d'arbre mais avec des propriétés supplémentaires.
Des choix ont donc été faits en privilégiant le vocabulaire mathématique au vocabulaire informatique
(je fais ce que je veux).
Il en va de même en anglais. Il faut donc toujours connaître la définition exacte des auteurs quand ils utilisent ce genre de notions.

Section 1

Retour sur la connexité et l'acyclicité

Proposition : connexité et nombre d'arêtes

Soit un graphe $G = (S, A)$.
Si G est connexe, alors $\|G\| \geqslant |G| - 1$.
(pour les plus lents, le nombre d'arêtes est au moins égal à l'ordre moins 1)

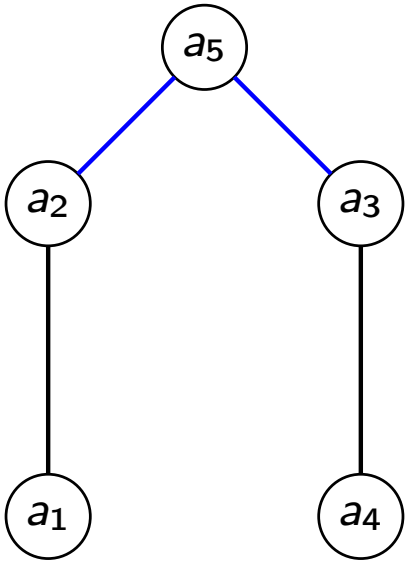
Démonstration :
Un graphe d'ordre 1 n'a pas d'arête.
Pour chaque sommet supplémentaire, il faut au moins une arête pour le connecter au graphe, il faut donc au moins $|G| - 1$ arêtes pour avoir un graphe connexe.

■

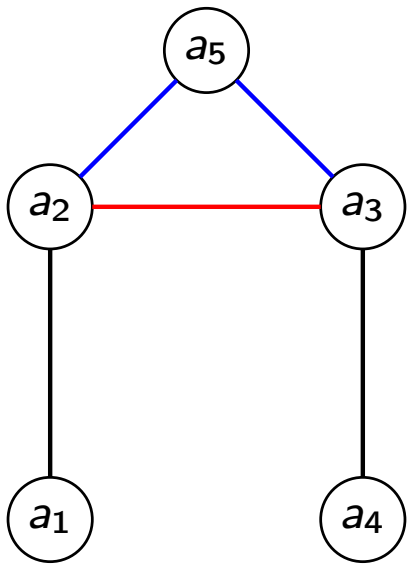
Proposition : acyclicité et nombre d'arêtes

Soit un graphe $G = (S, A)$.
Si G est acyclique, alors $\|G\| \leqslant |G| - 1$.
(pour les plus lents, le nombre d'arêtes est au plus égal à l'ordre moins 1)

Démonstration :
Remarquons que lorsqu'on ajoute une arête incidente à deux sommets déjà connectés, on crée un cycle. En effet, puisqu'ils sont déjà connectés, il existe une chaîne (simple) les reliant. En ajoutant la nouvelle arête à cette chaîne, on crée donc bien un cycle.



Démonstration :
Remarquons que lorsqu'on ajoute une arête incidente à deux sommets déjà connectés, on crée un cycle. En effet, puisqu'ils sont déjà connectés, il existe une chaîne (simple) les reliant. En ajoutant la nouvelle arête à cette chaîne, on crée donc bien un cycle.



Démonstration :
Donc on peut ajouter des arêtes uniquement entre deux composantes connexes distinctes et donc, à chaque fois, on fait baisser le nombre de composantes connexes de 1.
Au maximum, je peux donc ajouter $|G| - 1$ arêtes à un graphe sans arêtes pour passer de $|G|$ composantes connexes à une seule.

■

Section 2

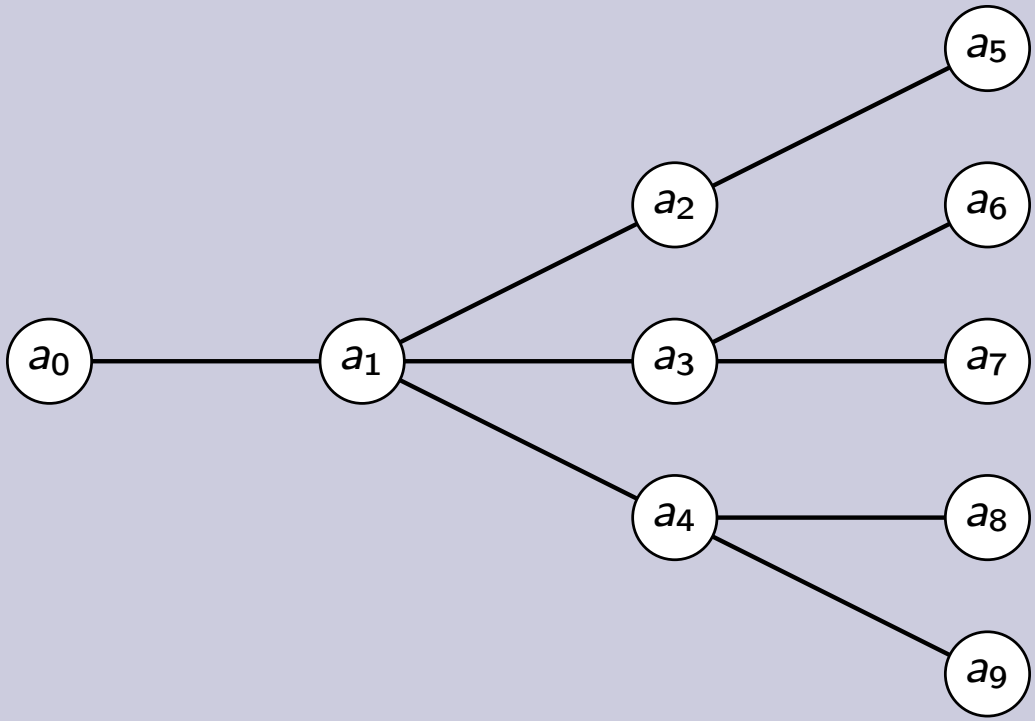
Définitions sylvestres

Définition : Forêt, Arbre et feuilles

Soit un graphe $G = (S, A)$.

1) Une *forêt* est un graphe acyclique.
2) Un *arbre* est un graphe acyclique connexe.
3) Une *feuille* d'un arbre est un sommet de degré 1.

Exemple :



Ordre : 10
Nombre d'arêtes : 9
Nombre de feuilles : 6

Caractérisation des arbres

Soit un graphe $G = (S, A)$.

1) Si G est un arbre, alors :

$$\|G\| = \left|G\right| - 1$$

(pour les plus lents : le nombre d'arêtes est égal à l'ordre de l'arbre moins 1).

2) Si G est connexe et $\|G\| = \left|G\right| - 1$, alors G est un arbre.

3) Si G est acyclique et $\|G\| = \left|G\right| - 1$, alors G est un arbre.

Démonstration :

1) Par récurrence :
Initialisation : C'est vrai à l'ordre 1.
Hérédité : Supposons que la propriété soit vraie au rang $N \in \mathbb{N}^*$: tout arbre d'ordre N admet $N-1$ arêtes.
Remarquons d'abord que tout arbre T d'ordre $N+1$ contient un arbre T' d'ordre N obtenu en enlevant une feuille et son arête incidente de T (pourquoi un arbre a-t-il toujours au moins une feuille ?).
Or, d'après l'hypothèse de récurrence, T' admet $N-1$ arêtes, donc T en admet N.
Conclusion : La propriété est vérifiée pour tout ordre $n \in \mathbb{N}^*$.

Démonstration :

2) Supposons que G admette un cycle alors qu'il est connexe et que $\|G\| = |G| - 1$.
Il faut donc enlever au moins une arête pour casser le cycle mais sans briser la connexité pour obtenir un arbre. Or, il aurait moins de $|G| - 1$ arêtes, tout en étant connexe, ce qui est absurde (diapo 5).

3) Supposons que G soit non connexe alors qu'il est acyclique et que $\|G\| = |G| - 1$.
Il faut donc ajouter au moins une arête pour connecter les composantes mais sans créer de cycle. Or, il aurait plus de $|G| - 1$ arêtes, tout en étant acyclique, ce qui est absurde (diapo 7).

■

Remarque :

On peut donc voir les arbres comme des graphes connexes minimaux en terme de nombre d'arêtes ou des graphes acycliques maximaux en terme de nombre d'arêtes.

Section 3

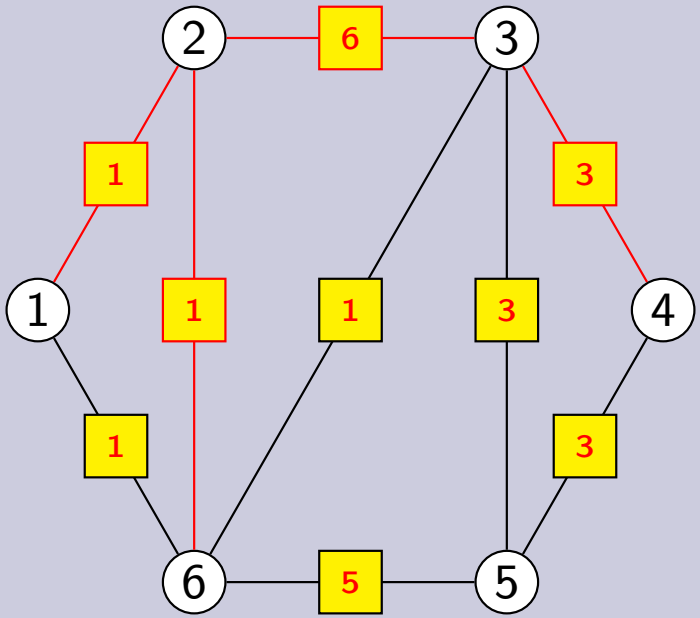
Arbres couvrants

Définition : Arbre couvrant

Soit un graphe $G = (S, A)$ connexe.

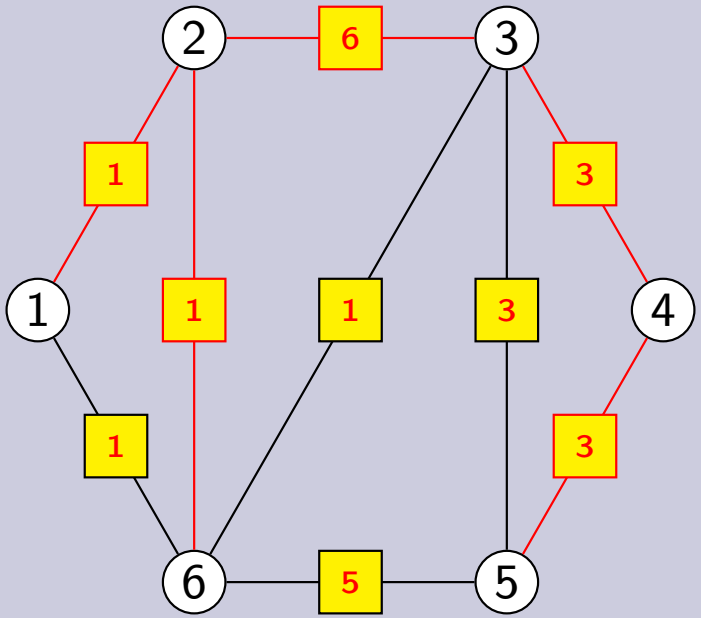
1) Un *arbre couvrant* de G est un sous-graphe T qui est maximal dans l'ensemble des arbres étant des sous-graphes de G (*i.e.* on ne peut lui ajouter de sommet et/ou d'arête supplémentaire sans qu'il ne perde sa propriété d'arbre étant un sous-graphe G).
2) Si G est un graphe valué, un *arbre couvrant de poids minimal* (ou *ACPM*) de G est un arbre couvrant ayant un poids total minimal parmi tous les arbres couvrants de G.

Exemple :



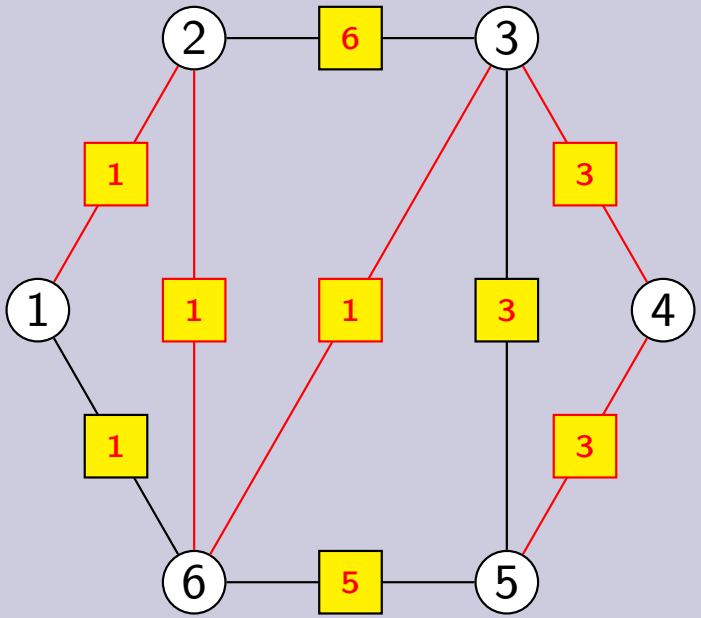
On a représenté un sous-arbre qui n'est pas couvrant.

Exemple :



On a représenté un arbre couvrant qui n'est pas de poids minimal.

Exemple :



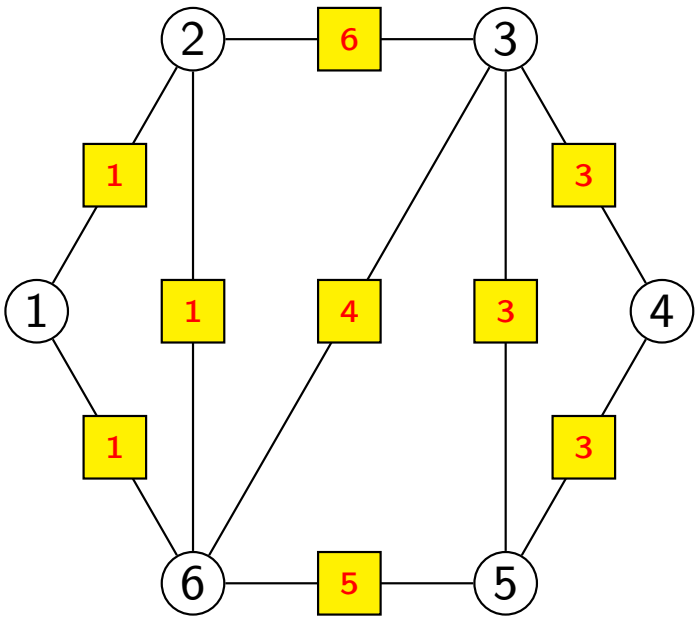
On a représenté un arbre couvrant de poids minimal. D'autres ACPM sont-ils possibles ?

Deux algorithmes célèbres pour trouver un ACPM : Kruskall et Prim.

Algorithme : Kruskall

Données : Un graphe valué connexe G
Ordonner les arêtes suivant leurs valuations
Créer un graphe vide T
Tant que T n'est pas un arbre couvrant :
    Ajouter la plus petite arête (selon la valuation) ne créant pas de cycle
Retourner T

Algortihme de Kruskall

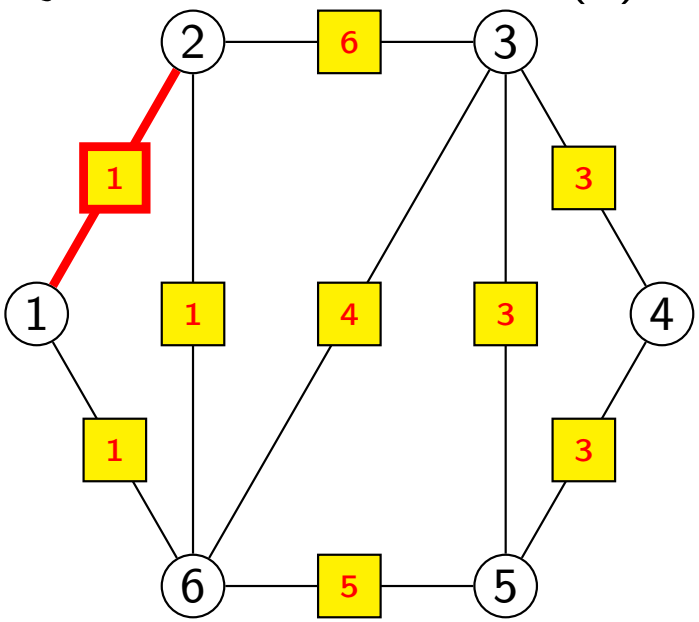


Liste des arêtes dans l'ordre :

$1 - 2$ (1)
$1 - 6$ (1)
$2 - 6$ (1)
$3 - 4$ (3)
$3 - 5$ (3)
$4 - 5$ (3)
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Kruskall

Ajout de l'arête $1 - 2$ (1)



Liste des arêtes dans l'ordre :

~~$1 - 2$ (1)~~ *

$1 - 6$ (1)

$2 - 6$ (1)

$3 - 4$ (3)

$3 - 5$ (3)

$4 - 5$ (3)

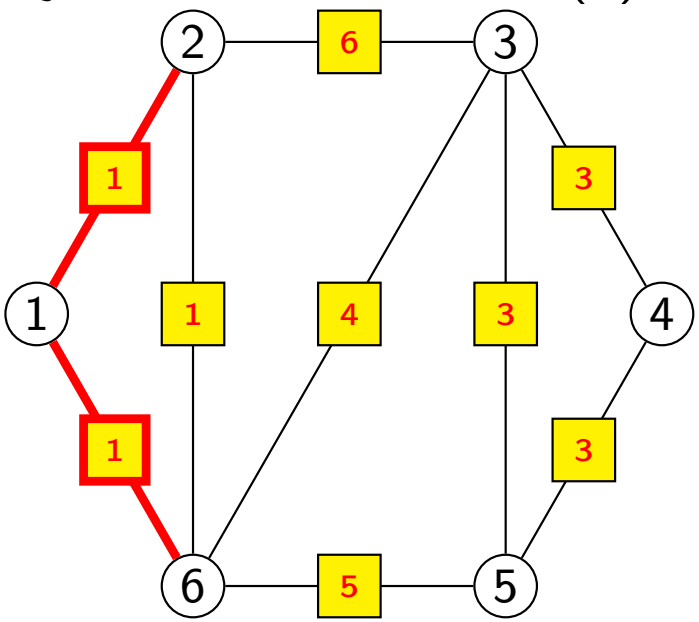
$3 - 6$ (4)

$5 - 6$ (5)

$2 - 3$ (6)

Algortihme de Kruskall

Ajout de l'arête $1 - 6$ (1)

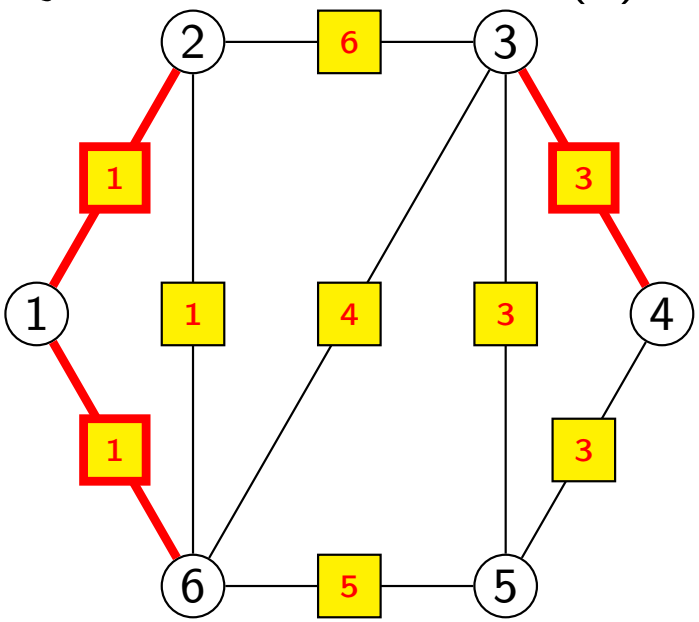


Liste des arêtes dans l'ordre :

~~$1 - 2$ (1)~~ *
~~$1 - 6$ (1)~~ *
~~$2 - 6$ (1)~~
$3 - 4$ (3)
$3 - 5$ (3)
$4 - 5$ (3)
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Kruskall

Ajout de l'arête $3 - 4$ (3)

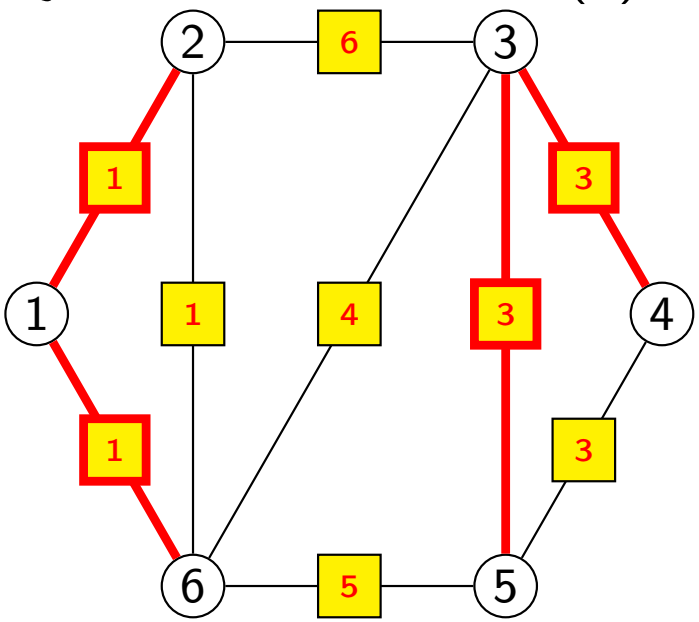


Liste des arêtes dans l'ordre :

~~$1 - 2$ (1)~~ *
~~$1 - 6$ (1)~~ *
~~$2 - 6$ (1)~~
~~$3 - 4$ (3)~~ *
$3 - 5$ (3)
$4 - 5$ (3)
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Kruskall

Ajout de l'arête $3 - 5$ (3)

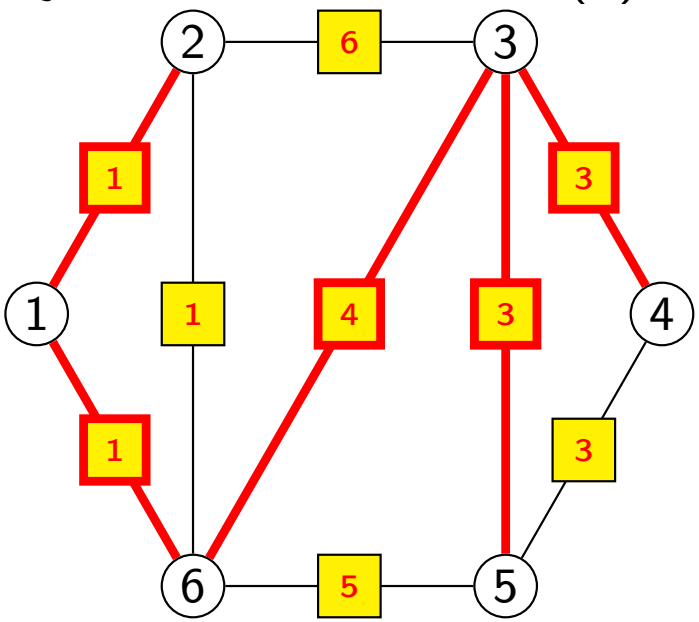


Liste des arêtes dans l'ordre :

~~$1 - 2$ (1)~~ *
~~$1 - 6$ (1)~~ *
~~$2 - 6$ (1)~~
~~$3 - 4$ (3)~~ *
~~$3 - 5$ (3)~~ *
~~$4 - 5$ (3)~~
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Kruskall

Ajout de l'arête $3 - 6$ (4)



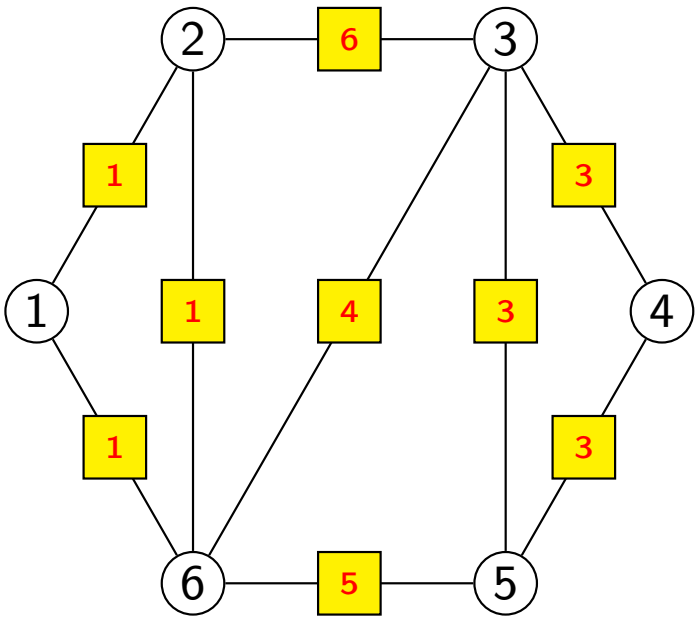
Liste des arêtes dans l'ordre :

~~1 — 2 (1)~~ *
~~1 — 6 (1)~~ *
~~2 — 6 (1)~~
~~3 — 4 (3)~~ *
~~3 — 5 (3)~~ *
~~4 — 5 (3)~~
~~3 — 6 (4)~~ *
~~5 — 6 (5)~~
~~2 — 3 (6)~~

Algorithme : Prim

Données : Un graphe valué connexe G et un sommet s
Ordonner les arêtes suivant leurs valuations
Créer un graphe T ayant pour unique sommet s
Tant que T n'est pas un arbre couvrant :
    Ajouter la plus petite arête (selon la valuation) ayant un sommet dans T et un sommet qui n'est pas dans T
Retourner T

Algortihme de Prim au départ de 4

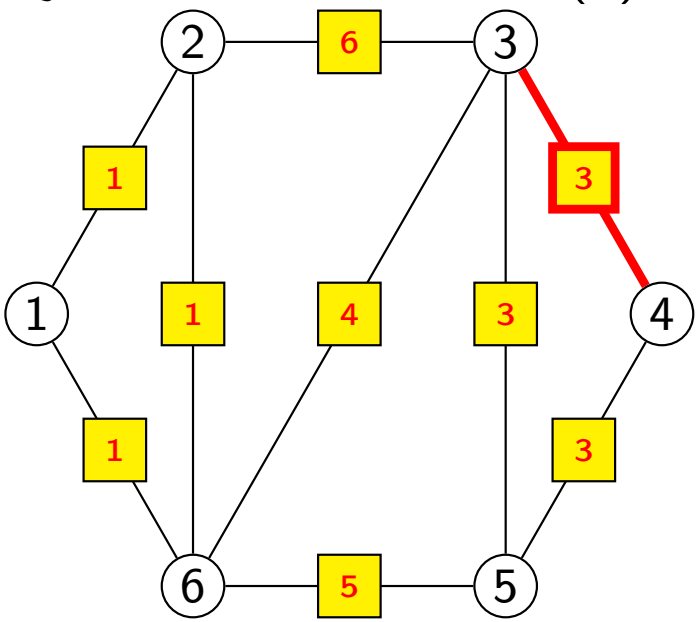


Liste des arêtes dans l'ordre :

$1 - 2$ (1)
$1 - 6$ (1)
$2 - 6$ (1)
$3 - 4$ (3)
$3 - 5$ (3)
$4 - 5$ (3)
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Prim au départ de 4

Ajout de l'arête $3 - 4$ (3)

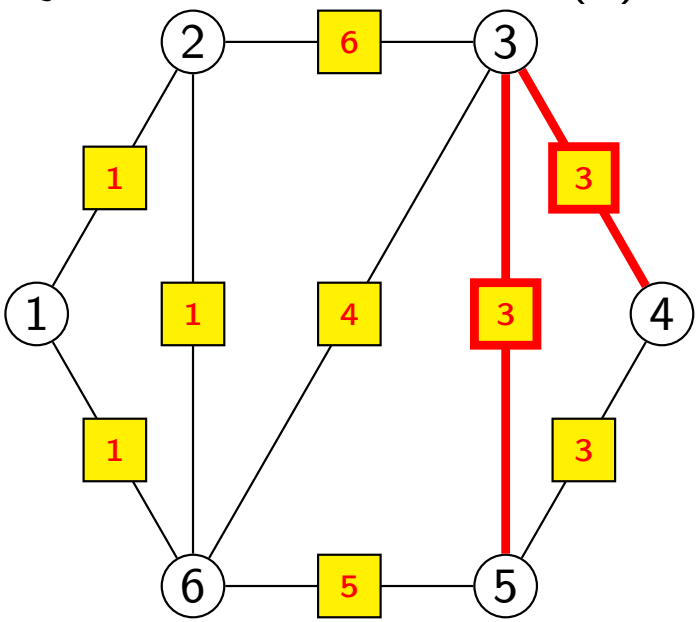


Liste des arêtes dans l'ordre :

$1 - 2$ (1)
$1 - 6$ (1)
$2 - 6$ (1)
~~$3 - 4$ (3)~~ *
$3 - 5$ (3)
$4 - 5$ (3)
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Prim au départ de 4

Ajout de l'arête $3 - 5$ (3)

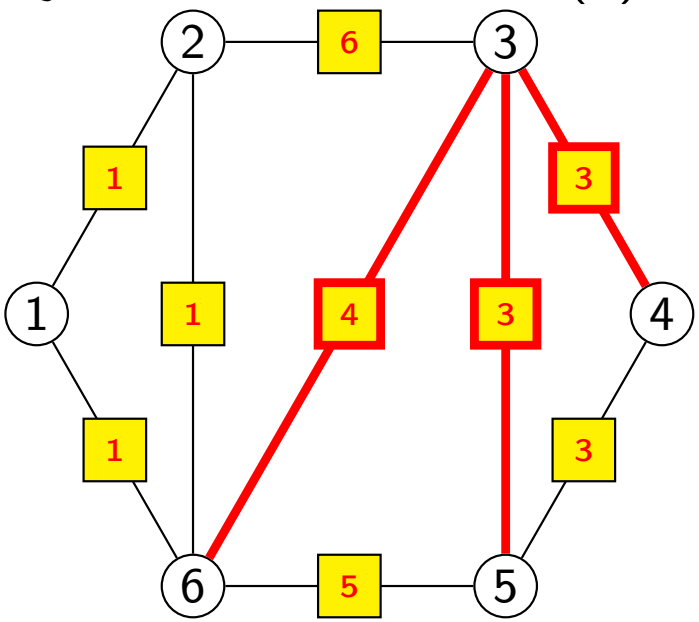


Liste des arêtes dans l'ordre :

$1 - 2$ (1)
$1 - 6$ (1)
$2 - 6$ (1)
~~$3 - 4$ (3)~~ *
~~$3 - 5$ (3)~~ *
~~$4 - 5$ (3)~~
$3 - 6$ (4)
$5 - 6$ (5)
$2 - 3$ (6)

Algortihme de Prim au départ de 4

Ajout de l'arête $3 - 6$ (4)

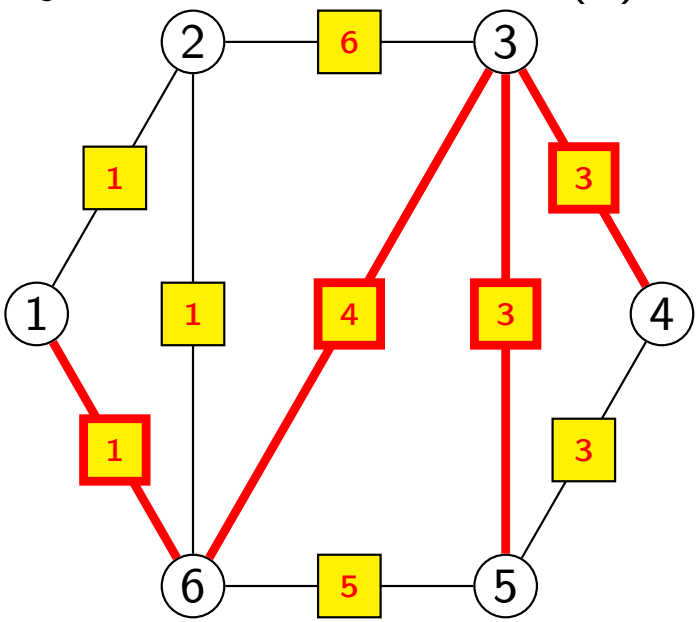


Liste des arêtes dans l'ordre :

$1 - 2$ (1)
$1 - 6$ (1)
$2 - 6$ (1)
~~$3 - 4$ (3)~~ *
~~$3 - 5$ (3)~~ *
~~$4 - 5$ (3)~~
~~$3 - 6$ (4)~~ *
~~$5 - 6$ (5)~~
$2 - 3$ (6)

Algortihme de Prim au départ de 4

Ajout de l'arête $1 - 6$ (1)



Liste des arêtes dans l'ordre :

$1 - 2$ (1)

~~$1 - 6$ (1)~~ *

$2 - 6$ (1)

~~$3 - 4$ (3)~~ *

~~$3 - 5$ (3)~~ *

~~$4 - 5$ (3)~~

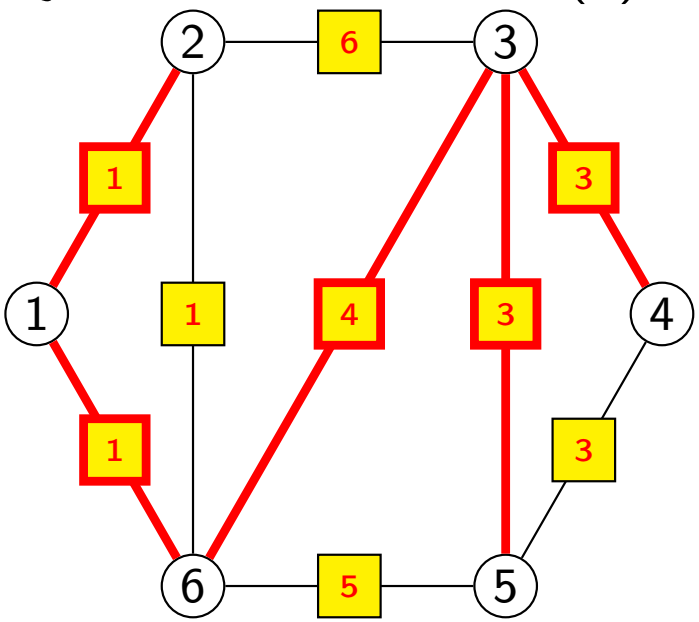
~~$3 - 6$ (4)~~ *

~~$5 - 6$ (5)~~

$2 - 3$ (6)

Algortihme de Prim au départ de 4

Ajout de l'arête $1 - 2$ (1)



Liste des arêtes dans l'ordre :

~~1 — 2 (1)~~ *
~~1 — 6 (1)~~ *
~~2 — 6 (1)~~
~~3 — 4 (3)~~ *
~~3 — 5 (3)~~ *
~~4 — 5 (3)~~
~~3 — 6 (4)~~ *
~~5 — 6 (5)~~
~~2 — 3 (6)~~

Remarques :

1) Les boucles *tant que* de ces deux algorithmes pourraient se faire remplacer par des boucles *pour* sachant qu'on connaît le nombre d'arêtes d'un arbre d'ordre n (diapo 13).
2) Que se passerait-il pour ces algorithmes si les arbres en entrée n'étaient pas connexes ?
3) Un autre algorithme que vous connaissez donne un arbre couvrant (le *spanning tree protocol*). Est-il de poids minimal ?

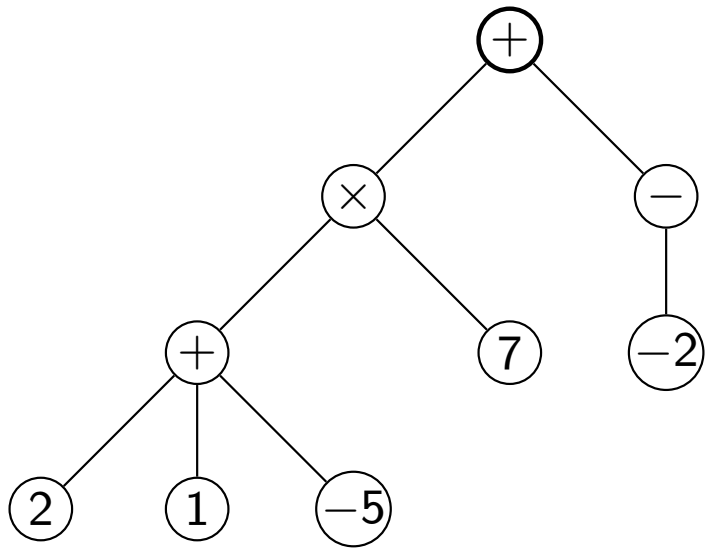
Section 4

Arbres enracinés

Pour la culture : en informatique, on retrouve des arbres de types particuliers dans des structures de données (arbres binaires de recherche par exemple, arbres syntaxiques pour manipuler les expressions algébriques par exemple).
On va donc en donner quelques définitions supplémentaires ici sans approfondir le sujet.

Définition : Arbres enracinés

Un *arbre enraciné* est un arbre pour lequel on désigne un sommet particulier appelé *racine*.
Un arbre enraciné admet une orientation naturelle allant de la racine vers ses feuilles.
On parle aussi d'*arborescence*.



Cet arbre syntaxique représente l'expression :

$$\Big(\big(2+1+(-5)\big)\times 7\Big)+\big(-(-2)\big)$$