Arbre

Définition . Un **arbre** est un graphe non orienté connexe et acyclique (i.e. sans cycle).

Proposition

1. Tout arbre est un graphe simple
2. Toute chaîne élémentaire est un arbre.

Arbre

Proposition

1. Tout arbre est un graphe simple
2. Toute chaîne élémentaire est un arbre.

Démonstration.

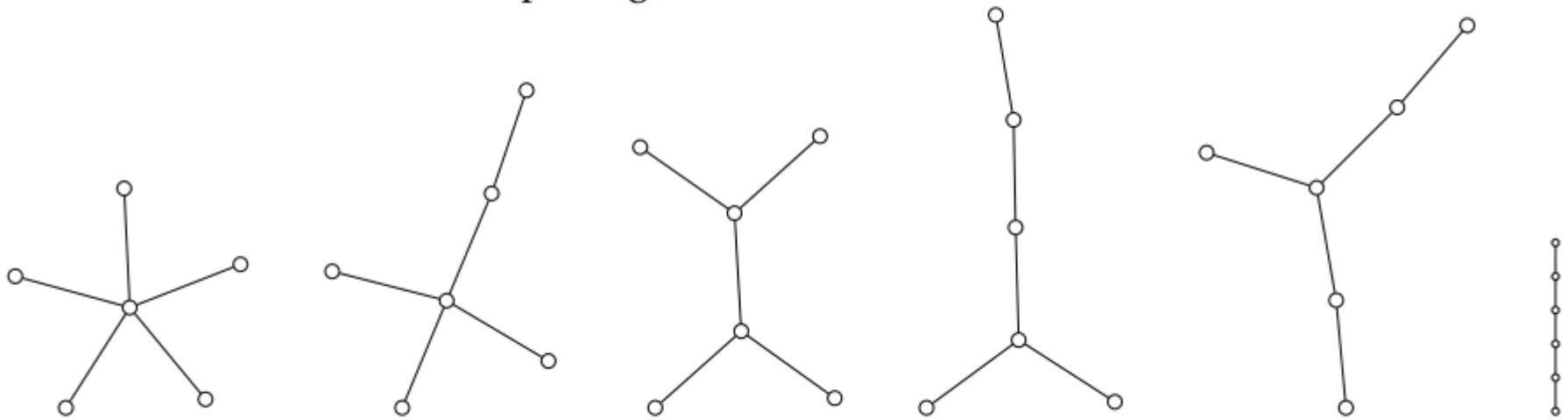
1. Toute boucle est un cycle et deux arêtes multiples induisent un cycle.
   Donc un arbre ne peut contenir ces éléments : il est donc simple.
2. Soit $C = (v_0, v_1, \ldots, v_p)$ une chaîne élémentaire :
   — comme tous les sommets sont reliés, C est connexe ;
   — comme tous les sommets sont distincts, C est acyclique.
   Donc C est un arbre.

Arbre

Exercice *Déterminer tous les arbres à six sommets.*

Arbre

Exercice On classe les arbres par degré maximal décroissant :



Arbre

Proposition (Existence de sommets de degré un)**.** Soit $T = (V, E, \gamma)$ un arbre tel que card $V \geq 2$.
Alors il existe au moins deux sommets de degré un.

Théorème (Nombre de sommets et d'arêtes dans un arbre)**.** Soit $T = (V, E, \gamma)$ un arbre.
Alors card $E =$ card $V - 1$.

Arbre

Proposition (Unicité de chaîne élémentaire)**.** Soit $T = (V, E, \gamma)$ un arbre.
Alors pour tout couple $(v, v') \in V^2$, il existe une unique chaîne élémentaire d'extrêmités v et v'.

Arbre

Définition (Isthme)**.** Soit $G=(V,E,\gamma)$ un graphe non orienté.
Un **isthme de G** est une arête e telle $G-e$ ait une composante connexe de plus que G.

Proposition Soit $G=(V,E,\gamma)$ un graphe non orienté et $e \in E$.
Alors e est un isthme de G si et seulement si ses extrémités ne sont reliés par aucune chaîne de $G-e$.

Propositior (Arbre et isthmes)**.** Toute arête d'un arbre est un isthme.

Lemme. Soit $G=(V,E,\gamma)$ un graphe non orienté et $e \in E$.
e est un isthme de G si et seulement si elle n'appartient à aucun cycle de G

Arbre

Caractérisation d'un arbre

Théorème (Caractérisations d'un arbre)**.** Soit $G = (V, E, \gamma)$ un graphe non orienté.
Les propositions suivantes sont équivalentes.

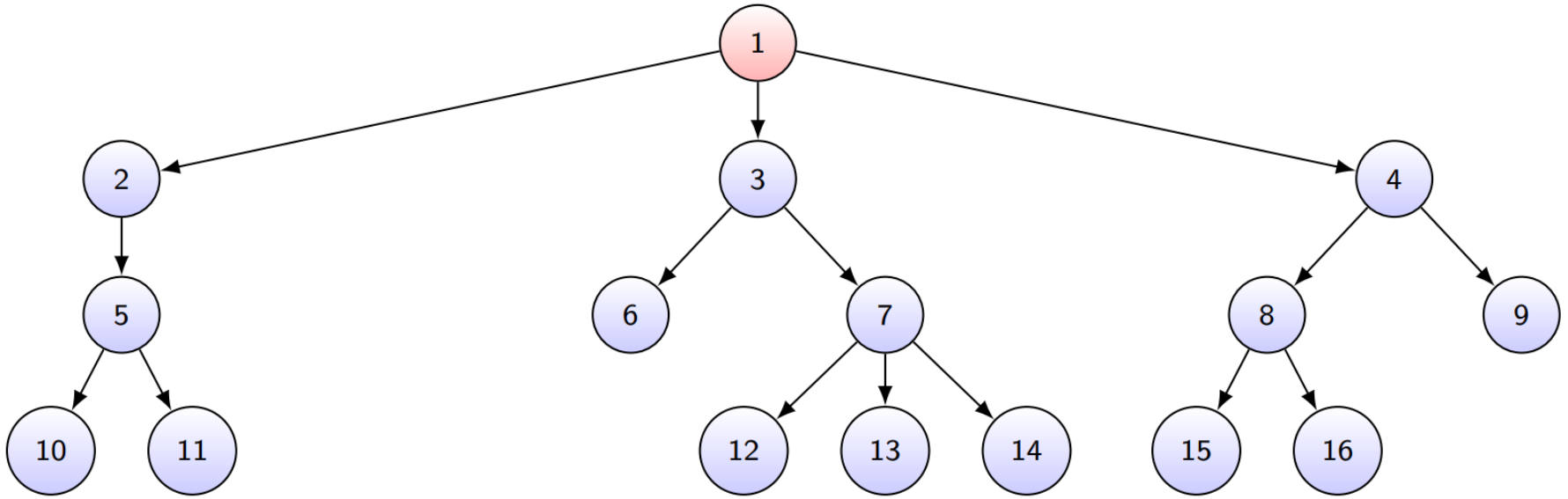
1. G est un arbre.
2. G est connexe et $\operatorname{card} E = \operatorname{card} V - 1$.
3. G est acyclique et $\operatorname{card} E = \operatorname{card} V - 1$.
4. G est connexe et toute arête est un isthme.
5. Pour tout couple $(v, v') \in V^2$, il existe une unique chaîne élémentaire d'extrêmités v et v'.

Parcourir un arbre

- Beaucoup d'algorithmes sur les arbres nécessitent de parcourir (traiter) tous les sommets. Par exemple :
  - ❑ Tester l'existence d'une valeur particulière dans un arbre.
  - ❑ Afficher un arbre.
- On peut parcourir un arbre de gauche à droite, ou de droite à gauche.
- On distingue les parcours :
  - ❑ En largeur
  - ❑ En profondeur

Parcourir un arbre en largeur

- On explore les nœuds un par un sur un même niveau. On passe ensuite sur le niveau suivant, et ainsi de suite (on va vers les frères avant de parcourir les fils).



Parcours : 1, 2, 3,4, 5,6,7,8,9,10,11,12,13,14,15,16

Parcourir un arbre en largeur

- Représenter un arbre par un dictionnaire python.
- Donner une fonction python pour afficher un arbre avec un parcours en largeur.
- Donner la complexité de votre algorithme.

Parcourir un arbre en profondeur

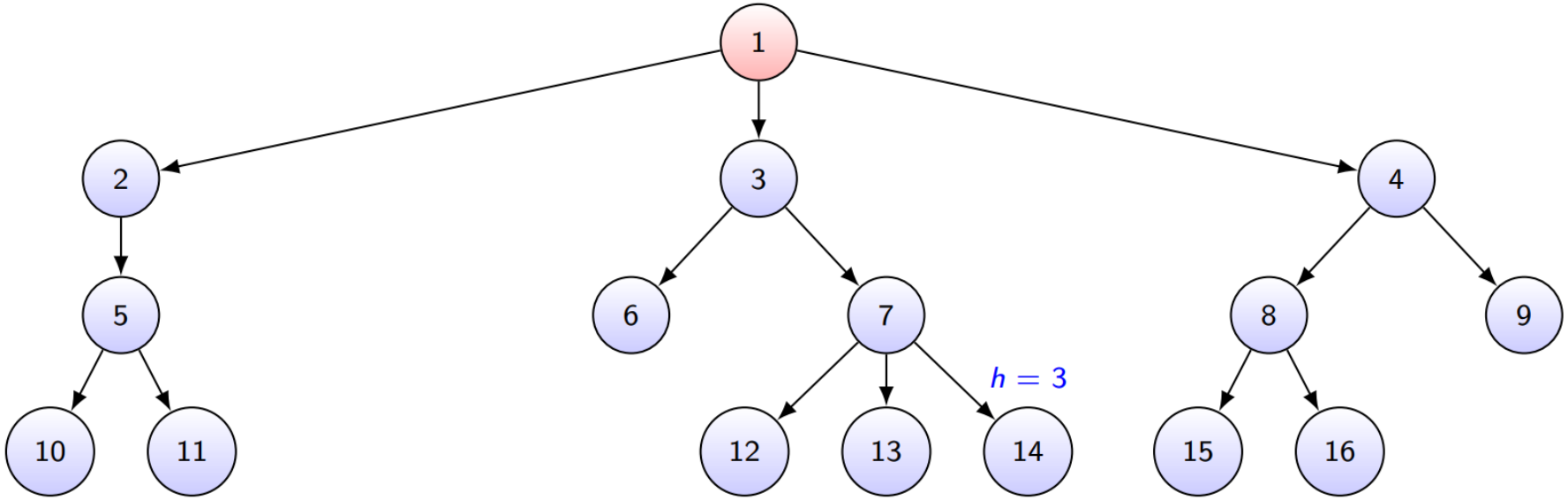
- Le parcours en profondeur d'un arbre se déroule naturellement de manière récursive. On explore jusqu'au bout une branche pour passer à la suivante (on va vers les fils avant d'aller vers les frères). Un pseudo algorithme pour ce parcours pourrait ressember à ceci :

```
explorer(graphe G, sommet s)
        afficher(s)
        pour tout sommet t voisin du sommet s
                        explorer(G, t);
```

Vocabulaire pour les arbres

- **Taille d'un arbre :** c'est le nombre de nœuds de l'arbre.
- **Profondeur d'un nœud :** c'est la distance du chemin qui relie la racine au nœud.
- **Hauteur d'un arbre :** c'est la profondeur maximale de ses nœuds.

Vocabulaire pour les arbres



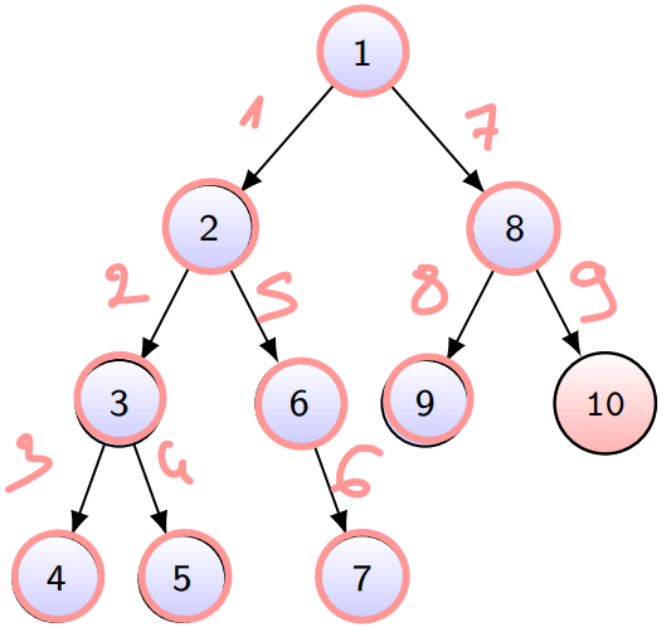
- taille = ?
- profondeur du noeud 12= ?
- hauteur = ?

Parcours en profondeur: préfixe

Préfixe : chaque nœud est visité avant leurs fils.

1. On visite la racine de l'arbre
2. Parcours préfixe des sous-arbres de gauche à droite

Dans un arbre binaire, on l'appelle parcours R G D (Racine Gauche Droit).



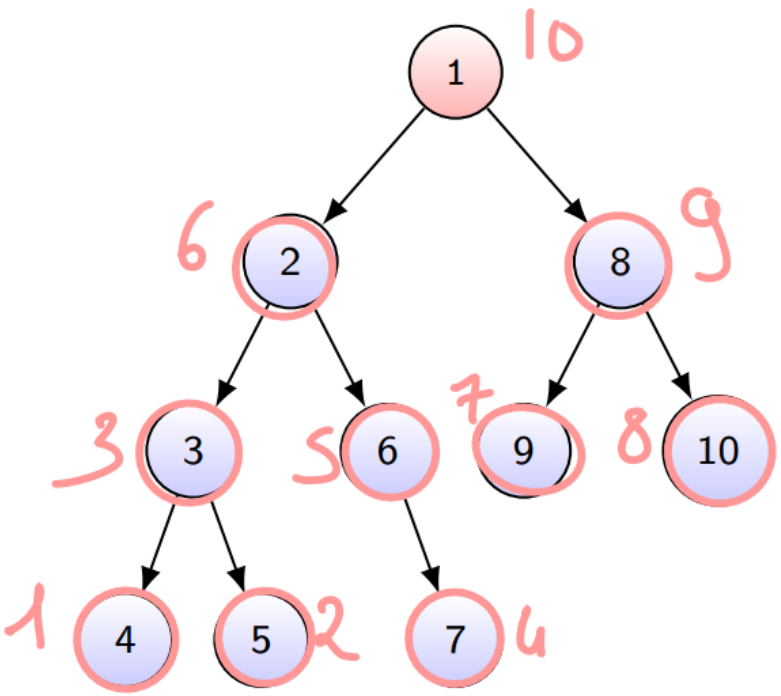
Parcours : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Parcours en profondeur: postfixe

Postfixe (ou suffixe) : on visite chaque nœud après avoir visité chacun de ses fils.

1. Parcours postfixe des sous-arbres de gauche à droite
2. On visite la racine de l'arbre

Dans un arbre binaire, on l'appelle parcours G D R (Gauche Droit Racine).



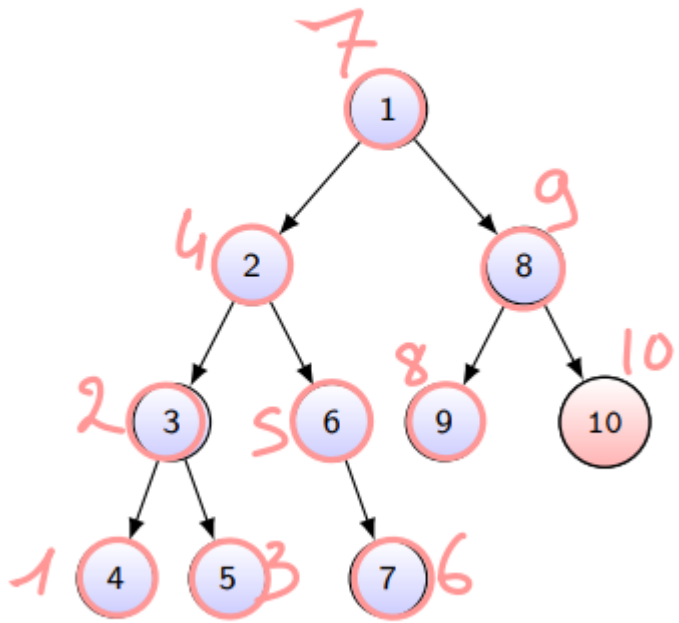
Parcours : 4, 5, 3, 7, 6, 2, 9, 10, 8, 1

Parcours en profondeur: infixe

Infixe (ou symétrique) : (dans un arbre binaire seulement) on visite chaque nœud après son fils à gauche et avant son fils à droite.

1. Parcours infixe du sous-arbre gauche
2. On visite la racine de l'arbre
3. Parcourt infixe du sous-arbre droit

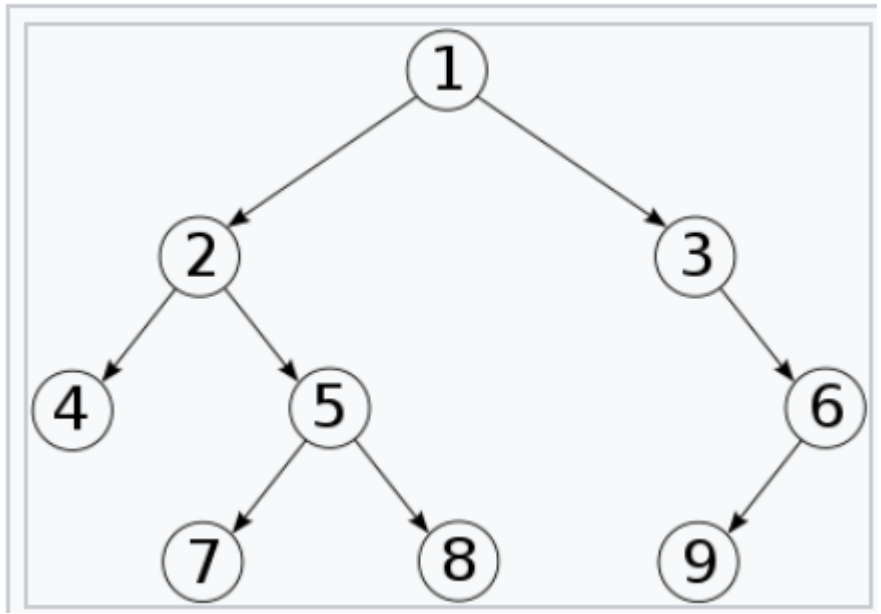
On l'appelle parcours G R D (Gauche Racine Droit).



Parcours : 4, 3, 5, 2, 6, 7, 1, 9, 8, 10

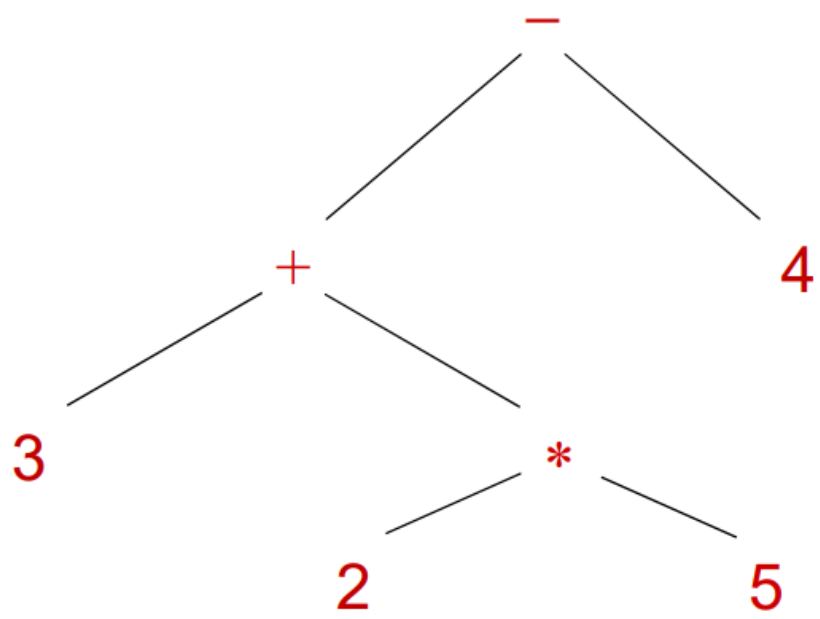
Arbre binaire

- Un arbre binaire est un arbre qui possède au maximum deux sous-arbres (d'où le binaire).



Arbre binaire

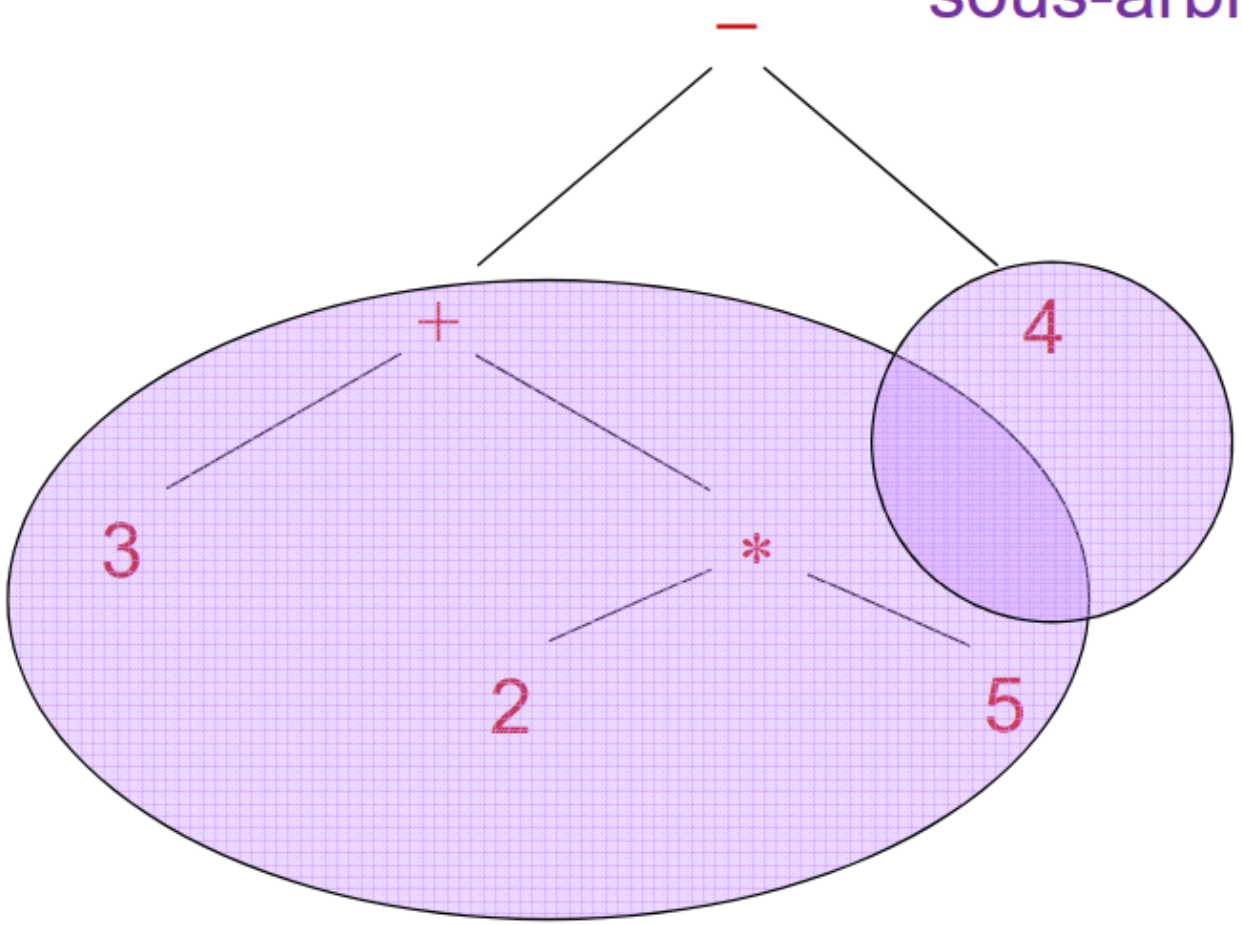
- Expressions mathématiques : 3 + 2*5 - 4



Arbre binaire

sous-arbre gauche de « – »

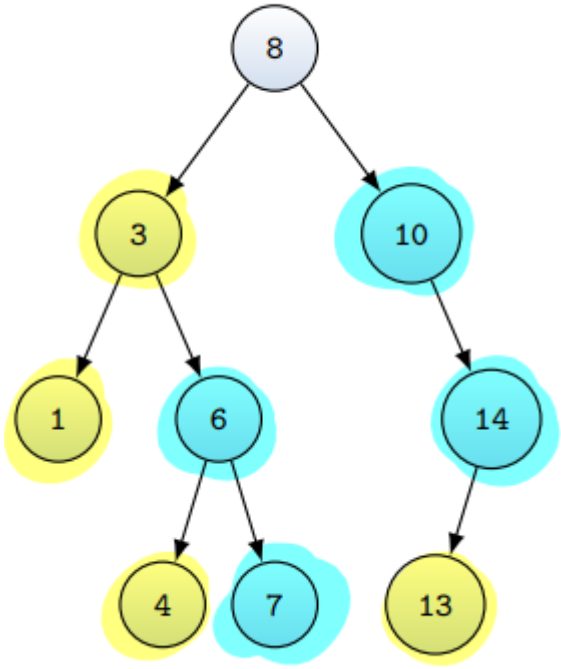
sous-arbre droit de « – »



Arbre binaire ABR

Un arbre binaire *a* est appelé **arbre binaire de recherche (ABR)** ssi :

- les valeurs du sous-arbre de gauche sont $<$ *racine(a)*
- les valeurs du sous-arbre de droite sont $>$ *racine(a)*
- tout sous-arbre de *a* est un ABR



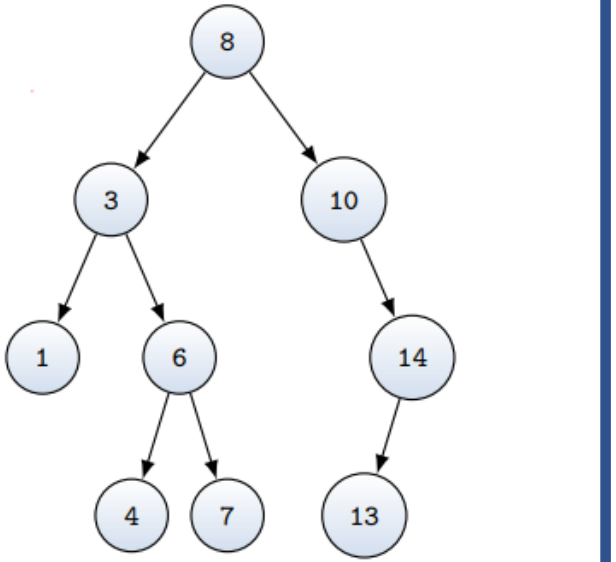
Remarque. On travaillera dans notre cours avec des arbres à **éléments uniques**.

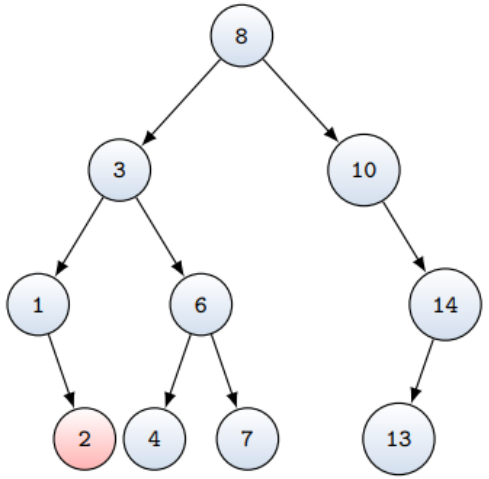
Opérations sur les ABR

- Réaliser les fonctions suivantes pour un ABR :
  - ❑ chercher une valeur dans un ABR
  - ❑ insérer une valeur dans un ABR
  - ❑ supprimer une valeur dans un ABR

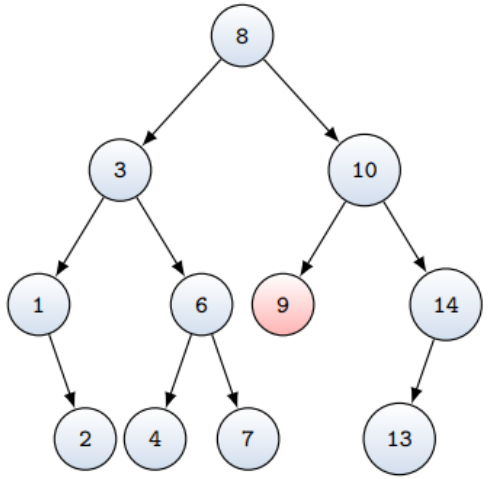
- Calculer la complexité de vos algorithmes

Opérations sur les ABR (ajout)



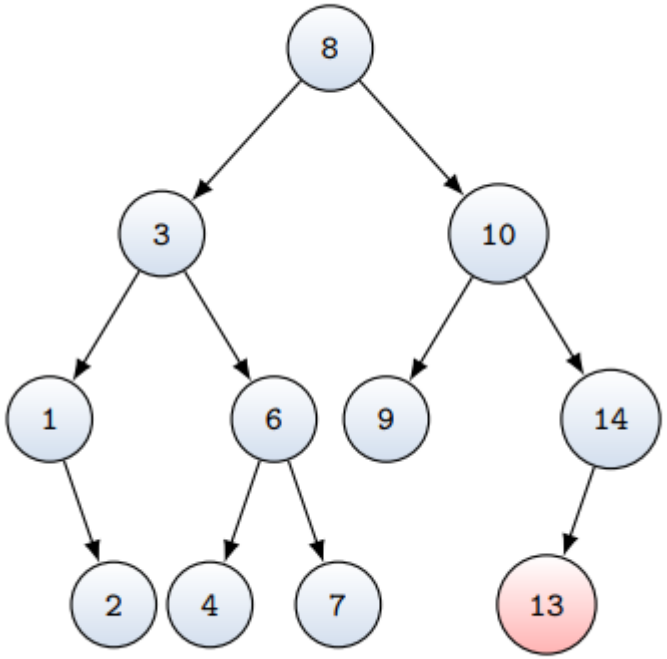


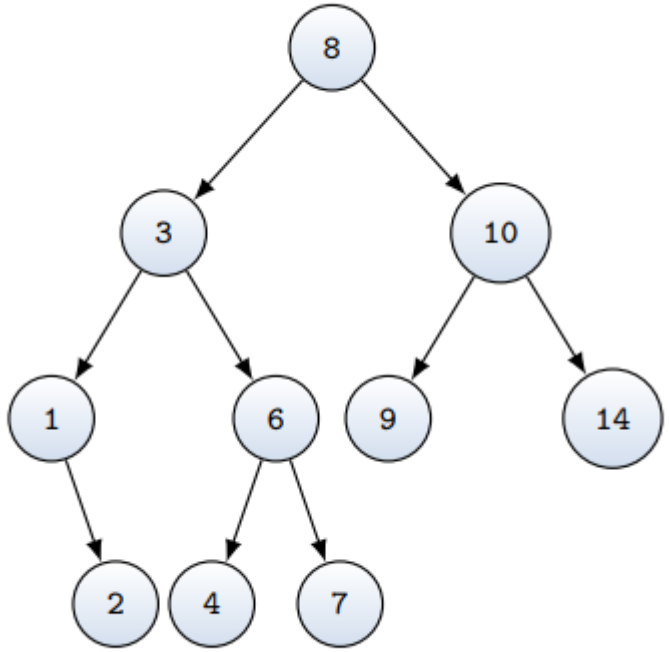
Ajout de l'élément 2



Ajout de l'élément 9

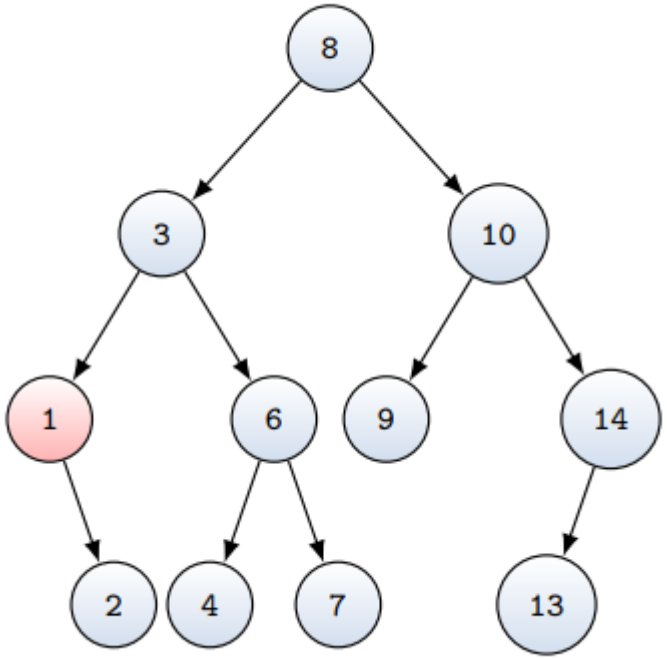
Opérations sur les ABR (suppression d'une feuille)

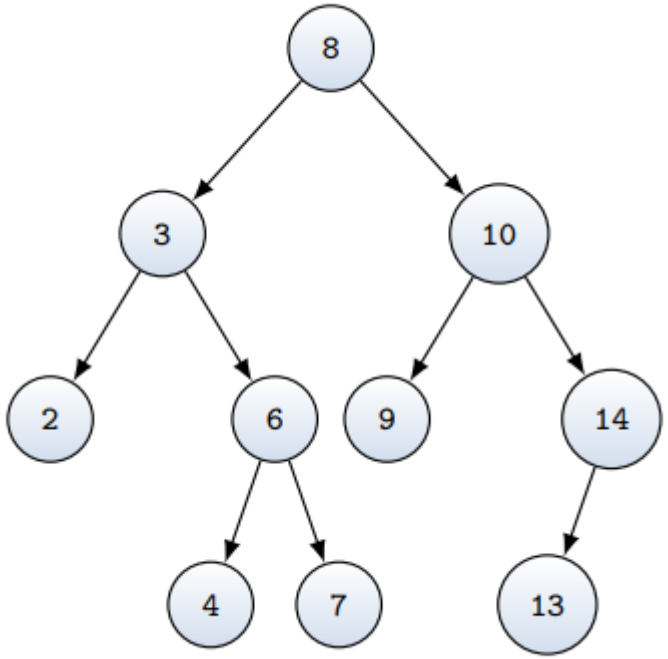




Cas trivial

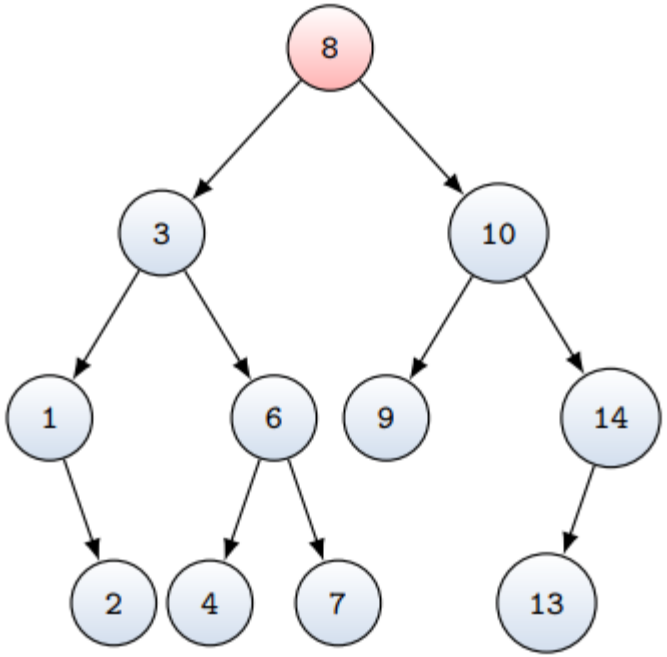
Opérations sur les ABR (suppression un noeud avec un seul fils)

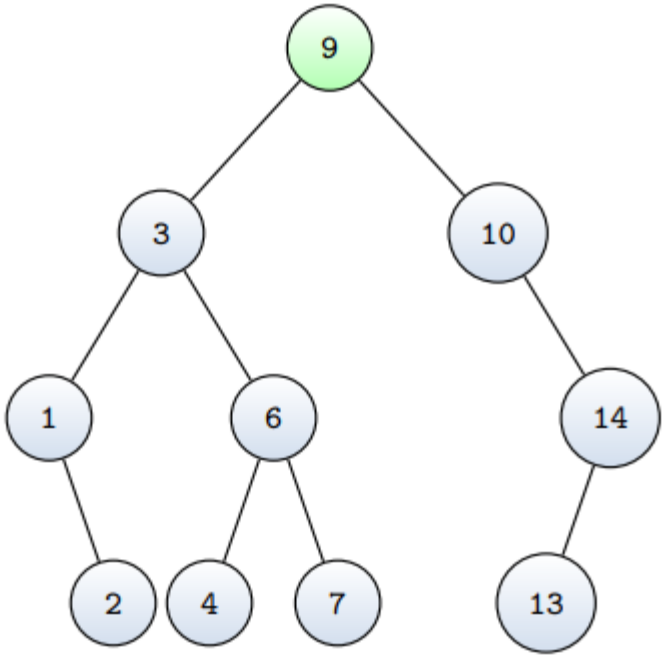




Remplacer le noeud par son fils

Opérations sur les ABR (suppression un noeud avec un deux fils)





Remplacer le nœud par son successeur (ou prédécesseur) dans son sous-arbre droit (ou gauche).

Arbre couvrant

Définition (Arbre couvrant)**.** Soit $G = (V, E, \gamma)$ un graphe non orienté.

Un sous-graphe T est un **arbre couvant de G** si et seulement si T est un arbre et un graphe couvrant de G.

Proposition (Existence d'arbre couvrant)**.**

Tout graphe fini connexe admet au moins un arbre couvrant.

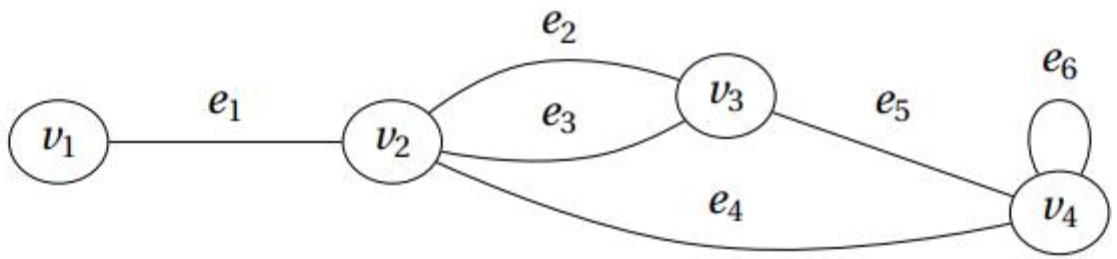
Démonstration. Soit $G = (V, E, \gamma)$ un graphe non orienté fini.

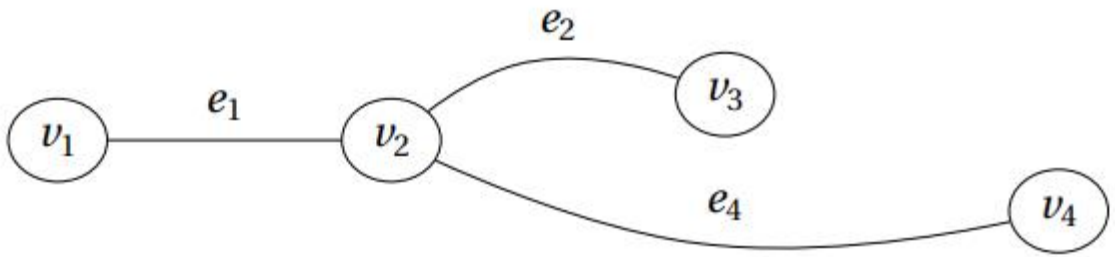
Tant qu'il reste des arètes qui ne sont pas isthmes, on en supprime une : la terminaison de ce processs est assurée par la finitude de E.

Le graphe ainsi obtenu est couvrant et ne contient que des isthmes : donc c'est un arbre.

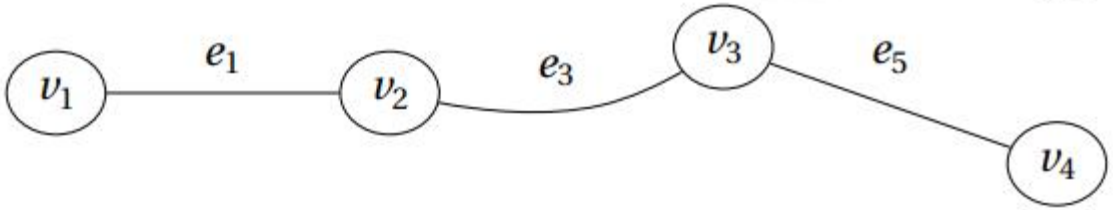
Arbre couvrant

Exemple. $G =$





et



sont des arbres couvrants de G.

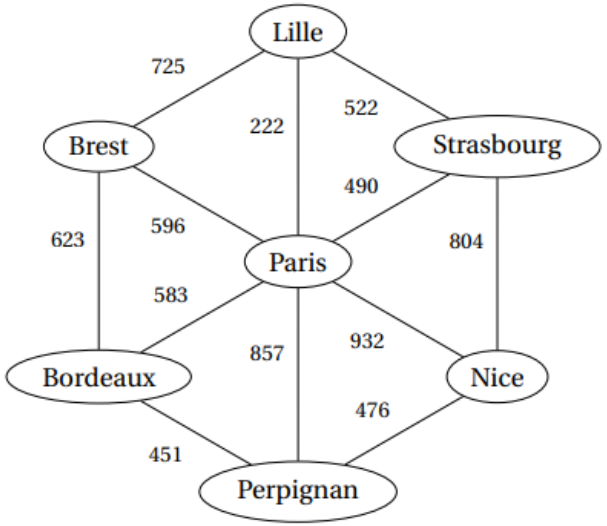
Arbre couvrant minimum

Définition (Valeur d'un arbre couvrant)**.**
Soit $G=(V,E,\gamma,w)$ un graphe non orienté valué et $T=(V,E',\gamma_{|E'})$ un arbre couvrant de G.
On appele la **valeur de T**, notée $w(T)$ le réel défini par $w(T)=\sum_{e\in E'} w(e)$.

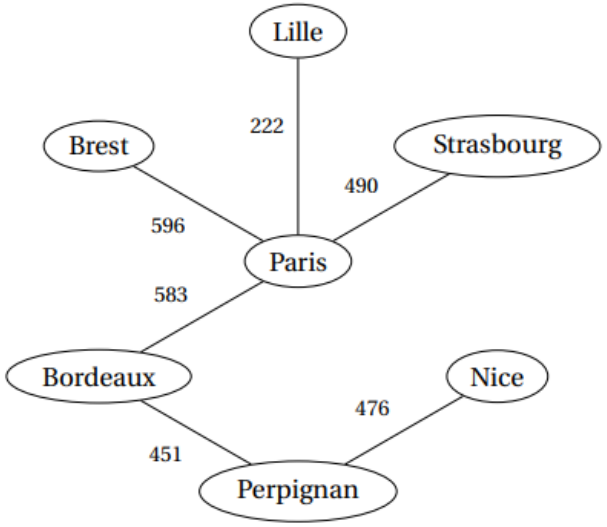
Définition (Arbre couvrant minimum)**.** Soit $G=(V,E,w)$ un graphe valué simple et connexe.
Un **arbre couvrant minimum de G** est un arbre couvrant de G dont la valeur est minimale parmi tous les arbres couvrants de G.

Arbre couvrant minimum

Exemple. $G =$



Un arbre couvrant minium de G est



Remarque. Un arbre couvrant minimum n'est pas toujours unique.

Arbre couvrant minimum

Recherche d'arbre couvrant minimum

Une première solution peut consister en une recherche exhaustive, c'est-à-dire :

- recenser tous les arbres couvrants ;
- pour chacun d'entre eux, calculer sa valeur ;
- en choisir un parmi ceux qui ont la valeur la plus faible.

Cette solution a une complexité exponentielle : elle est donc inutilisable en pratique car trop coûteuse.

Arbre couvrant minimum

Une autre solution consiste en l'application d'un algorithme de type « glouton » : l'algorithme de KRUSKAL.

1. On initialise le graphe couvrant avec aucune arête.
2. Soit e l'arête non encore testée de plus faible valuation : si e n'induit pas un cycle, on l'ajoute au graphe couvrant en cours.

Arbre couvrant minimum

Une autre solution consiste en l'application d'un algorithme de type « glouton » : l'algorithme de KRUSKAL.

1. On initialise le graphe couvrant avec aucune arête.
2. Soit e l'arête non encore testée de plus faible valuation : si e n'induit pas un cycle, on l'ajoute au graphe couvrant en cours.

Arbre couvrant minimum

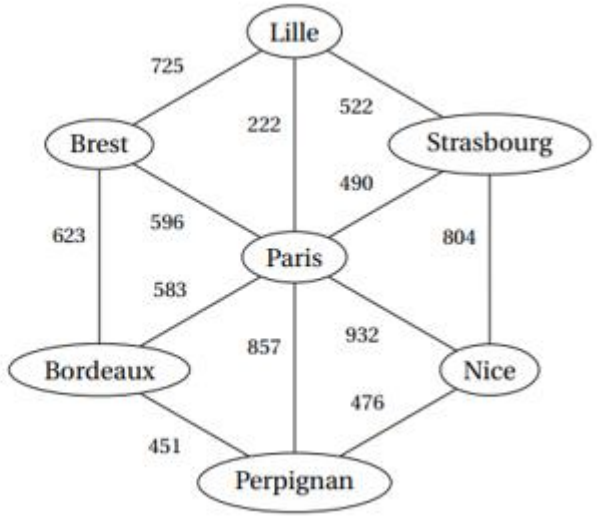
préconditions : $G = (V, E, \gamma, w)$ graphe valué simple et connexe
postconditions : $G(F)$ est un arbre couvrant minimal de G
$E^c \leftarrow E$
$F \leftarrow \emptyset$
tant que card $F <$ card $V - 1$ **faire**
  e est choisi dans argmin$\{w(e), e \in E^c\}$
  $E^c \leftarrow E^c \backslash \{e\}$
  si $G(F \cup \{e\})$ est acyclique **alors**
    $F \leftarrow F \cup \{e\}$
  fin si
fin tant que

Algorithme 1: Algorithme de KRUSKAL

Théorème 2.5 (Terminaison et complexité de l'algorithme de KRUSKAL)**.**
L'algorithme de KRUSKAL converge vers un arbre couvrant minimum.
Sa complexité est $O\big(\text{card } E \cdot \log(\text{card } V)\big)$.

Arbre couvrant minimum

Exemple. $G =$



Initialisation $F = \emptyset$

Itération n° 1 $e = \text{Paris} - \text{Lille} \Rightarrow F = F \cup \{e\}$

Itération n° 2 $e = \text{Bordeaux} - \text{Perpignan} \Rightarrow F = F \cup \{e\}$

Itération n° 3 $e = \text{Perpignan} - \text{Nice} \Rightarrow F = F \cup \{e\}$

Itération n° 4 $e = \text{Paris} - \text{Strasbourg} \Rightarrow F = F \cup \{e\}$

Itération n° 5 $e = \text{Lille} - \text{Strasbourg} \Rightarrow F$ inchangé

Itération n° 6 $e = \text{Paris} - \text{Bordeaux} \Rightarrow F = F \cup \{e\}$

Itération n° 7 $e = \text{Paris} - \text{Brest} \Rightarrow F = F \cup \{e\}$

Terminaison $\text{card}\, F = 6 = \text{card}\, V - 1 \Rightarrow$ `STOP`