Arbres

Arbres | Définitions

On se place dans le cadre des graphes *non orientés*.

Définition (Cycle simple)

Un cycle est **simple** *s'il passe une unique fois par chacune de ses arêtes.*

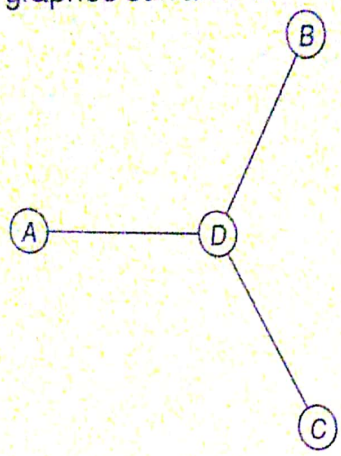
Définition (Graphe acyclique)

Un graphe non orienté est **acyclique** *s'il ne possède pas de cycles* simples.

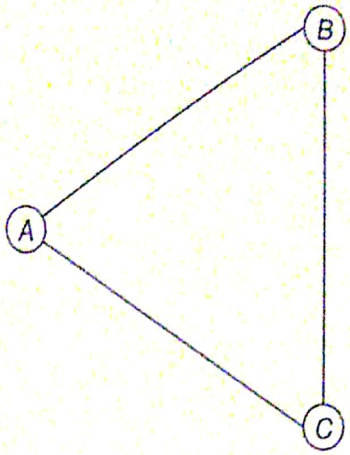
Définition (Arbre)

Un **arbre** *est un graphe non orienté qui est* **connexe** *et* **acyclique**.

Les graphes suivants sont-ils acycliques ?

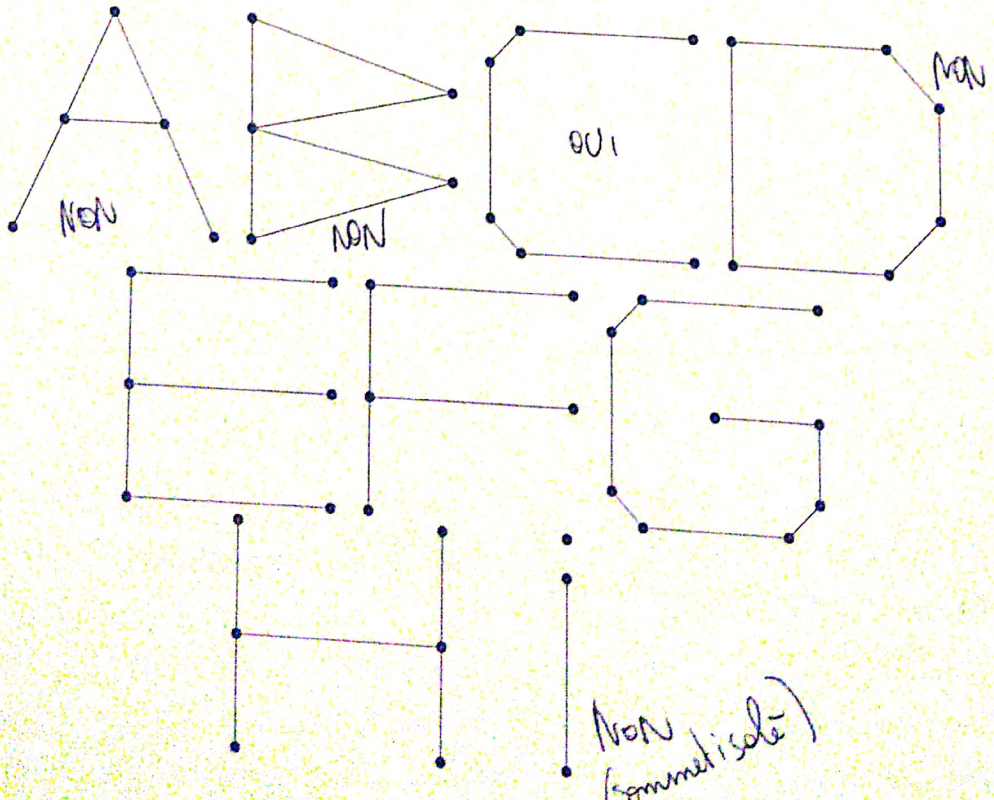


1)



2)

Parmi les graphes suivants, lesquels sont des arbres ?



Le théorème fondamental suivant donne des caractérisations des arbres :

Théorème (Caractérisation des arbres)

Soit G un graphe non orienté à n sommets. Les propositions suivantes sont équivalentes et permettent de caractériser un arbre :

1. *G est un arbre*
2. *G est acyclique et comporte n – 1 arêtes*
3. *G est connexe et comporte n – 1 arêtes*
4. *G est connexe et la suppression de n'importe quelle arête le le rend non connexe (on dit aussi que chaque arête est un pont)*
5. *G est acyclique et l'ajout d'une arête quelconque crée un cycle*
6. *deux sommets distincts de G sont reliés par un unique chemin*

Arbres | Arbre couvrant de poids minimal

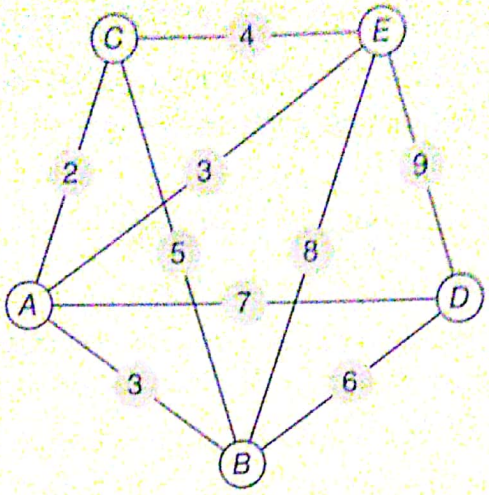
Définition (Arbre couvrant)

Soit G un graphe valué non orienté connexe.

- *Un* **arbre couvrant** *est un sous-graphe de G couvrant (i.e. contenant tous les sommets), connexe et acyclique. Son poids est la somme des valuations de ses arêtes.*
- *Un* **arbre couvrant de poids minimal** *est un arbre couvrant dont le poids est le plus petit possible parmi les arbres couvrants de G.*

Nous allons mettre en œuvre un algorithme permettant de déterminer un arbre couvrant de poids minimal dans un graphe valué G donné.

Problème. Une communauté urbaine constituée de cinq villes (A,B,C,D,E) souhaite se munir d'un réseau de connexion interne. Pour cela, il faut relier les villes avec des câbles. Une étude préliminaire a permis de déterminer les coûts (rapportés à une unité monétaire) de connexion entre les villes :



Question : Comment connecter toutes les villes à moindre coût ?

Réponse : Le coût maximal est de 47 mais il n'est pas nécessaire d'établir toutes les connexions ! L'**algorithme de Kruskal** permet de construire un *arbre couvrant* de *poids minimal* qui répond à la question posée.

Principe de l'algorithme de Kruskal : on construit l'arbre couvrant petit à petit, en s'assurant à chaque étape que l'on reste **couvrant** et **acyclique**.

Algorithme (Kruskal)

INITIALISATION :
  $(a_1, \dots, a_m)$ *: LISTE DES ARETES PAR ORDRE CROISSANT DES POIDS ;*
  $T_0 = \emptyset$;
POUR $i = 1, \dots, n-1$ *:*
  $\lambda_i = \min(\{j \in \mathbb{N},\ a_j \notin T_{i-1}$ *ET* $T_{i-1} \cup a_j$ *EST ACYCLIQUE*$\})$;
  $T_i = T_{i-1} \cup a_{\lambda_i}$;
FIN (POUR)
RESULTAT : T_{n-1} *EST UN ARBRE COUVRANT de poids minimal.*

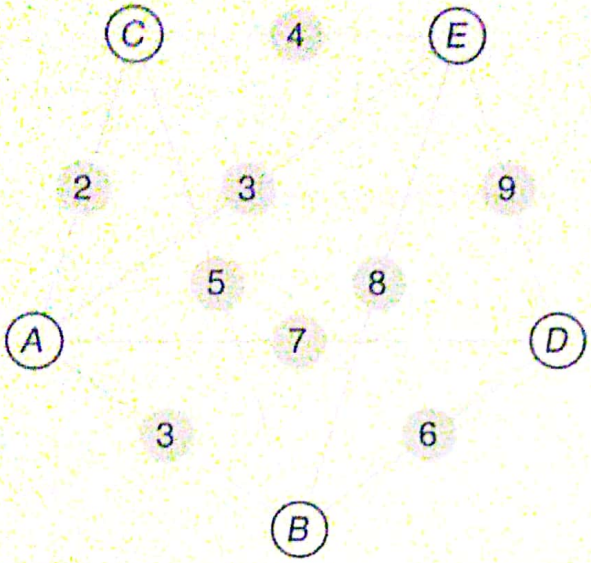
Remarque.

- À chaque étape de l'algorithme, on obtient un *sous-graphe couvrant acyclique* (qui n'est pas un arbre), qui grossit jusqu'à devenir, à l'ultime étape, un arbre : l'objet construit est donc un arbre (préservation du caractère acyclique) couvrant (par construction) avec un poids minimal (par sélection des arêtes).
- La partie la plus coûteuse est en fait le tri initial des arêtes.

- **Liste des arêtes rangée par ordre croissant des poids :**

i	1	2	3	4	5	6	7	8	9
a_i	(AC)	(AB)	(AE)	(CE)	(CB)	(BD)	(AD)	(EB)	(ED)

- **Étape d'initialisation :**

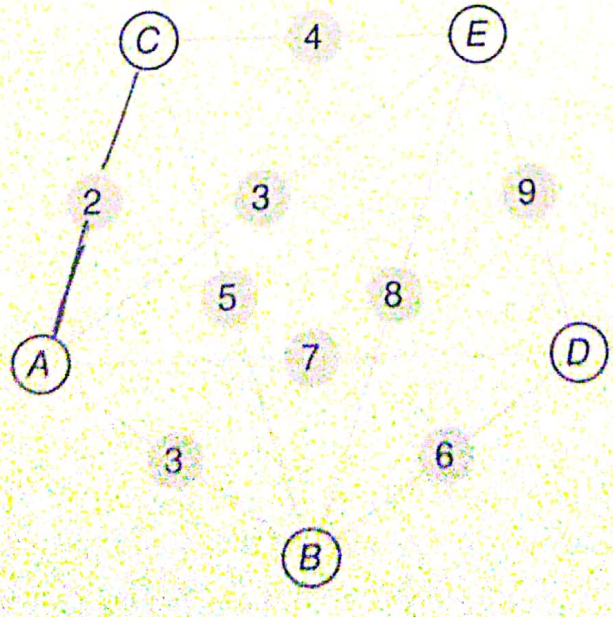


Arbres | Arbre couvrant de poids minimal

- **Liste des arêtes rangée par ordre croissant des poids :**

i	1	2	3	4	5	6	7	8	9
a_i	(AC)	(AB)	(AE)	(CE)	(CB)	(BD)	(AD)	(EB)	(ED)

- **Étape** 1 : $\lambda_1 = \min(\{j \in \mathbb{N},\ a_j \notin T_0$ ET $T_0 \cup a_j$ EST ACYCLIQUE$\}$



Détermination de l'indice :

- $\lambda_1 = 1$

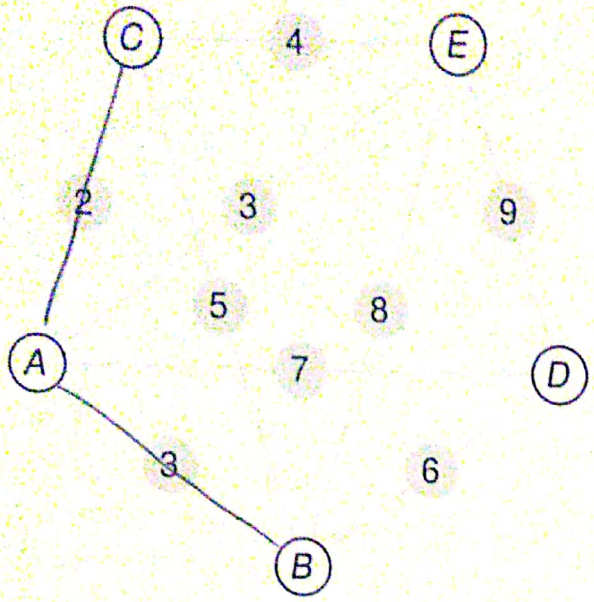
Mise à jour :

- $T_1 = T_0 \cup a_1$

- **Liste des arêtes rangée par ordre croissant des poids :**

i	1	2	3	4	5	6	7	8	9
a_i	(AC)	(AB)	(AE)	(CE)	(CB)	(BD)	(AD)	(EB)	(ED)

- **Étape** 2 : $\lambda_2 = \min(\{j \in \mathbb{N},\ a_j \notin T_1 \text{ ET } T_1 \cup a_j \text{ EST ACYCLIQUE}\}$



Détermination de l'indice :

- $\lambda_1 = 2$

Mise à jour :

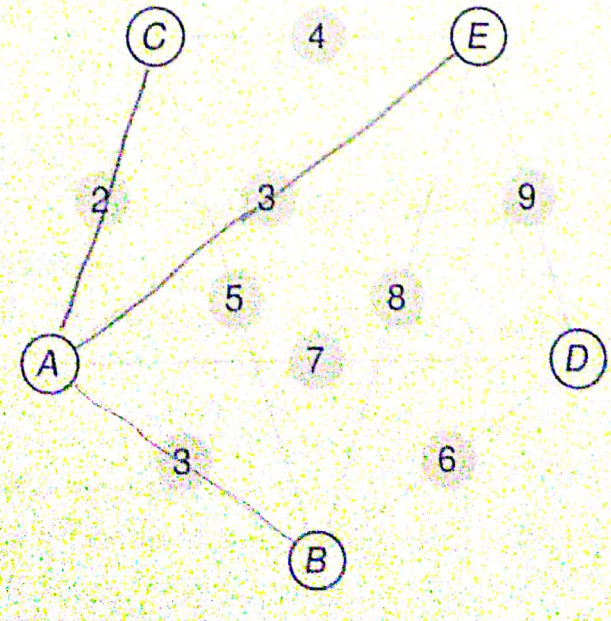
- $T_2 : T_1 \cup a_2$

Arbres | Arbre couvrant de poids minimal

- **Liste des arêtes rangée par ordre croissant des poids :**

i	1	2	3	4	5	6	7	8	9
a_i	(AC)	(AB)	(AE)	(CE)	(CB)	(BD)	(AD)	(EB)	(ED)

- **Étape** 3 : $\lambda_3 = \min(\{j \in \mathbb{N},\ a_j \notin T_2 \text{ ET } T_2 \cup a_j \text{ EST ACYCLIQUE}\}$



Détermination de l'indice :

- $\lambda_3 = 3$

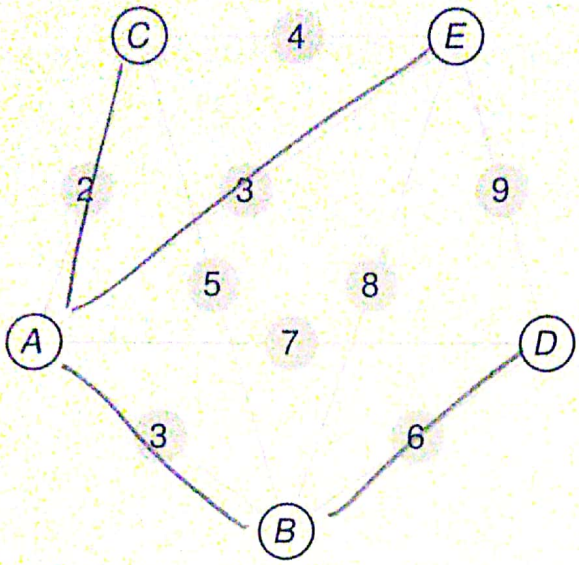
Mise à jour :

- $T_3 = T_2 \cup a_3$

- **Liste des arêtes rangée par ordre croissant des poids :**

i	1	2	3	4	5	6	7	8	9
a_i	(AC)	(AB)	(AE)	(CE)	(CB)	(BD)	(AD)	(EB)	(ED)

- **Étape** 4 : $\lambda_4 = \min(\{j \in \mathbb{N},\ a_j \notin T_3$ ET $T_3 \cup a_j$ EST ACYCLIQUE$\}$



Détermination de l'indice :

- l'indice 4 ne passe pas
- l'indice 5 ne passe pas
- $\lambda_4 = 6$

Mise à jour :

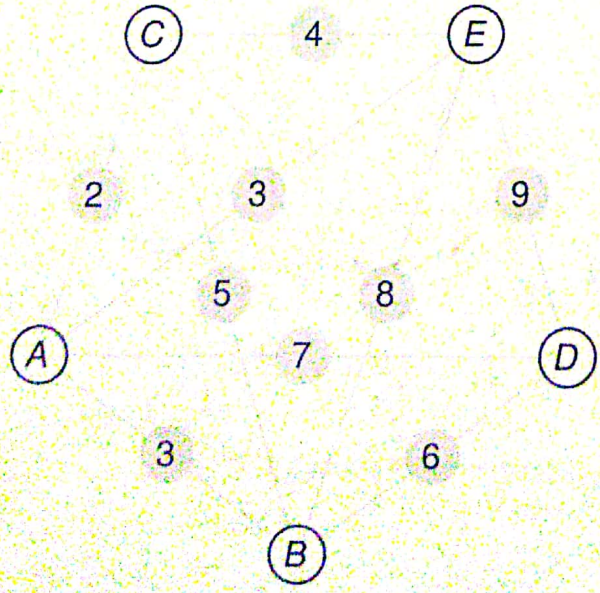
- $T_4 = T_3 \cup a_6$

Arbres | Arbre couvrant de poids minimal

- **Liste des arêtes rangée par ordre croissant des poids :**

i	1	2	3	4	5	6	7	8	9
a_i	(AC)	(AB)	(AE)	(CE)	(CB)	(BD)	(AD)	(EB)	(ED)

- **Étape** 5 : $\lambda_5 = \min(\{j \in \mathbb{N},\ a_j \notin T_4$ ET $T_4 \cup a_j$ EST ACYCLIQUE$\}$



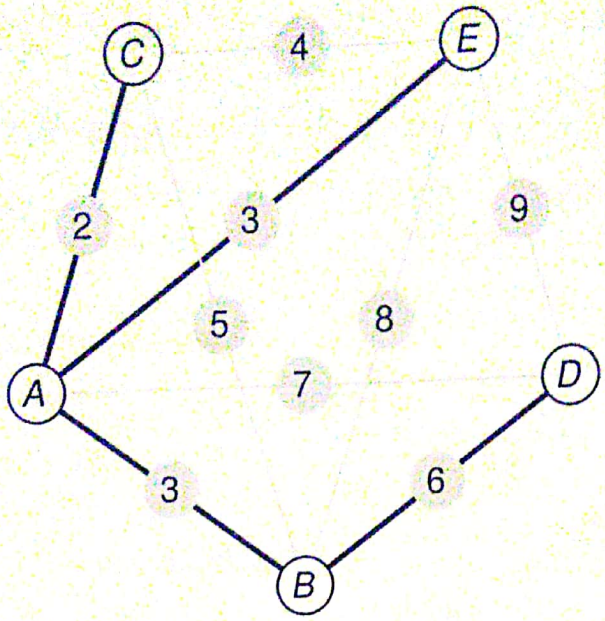
Détermination de l'indice :

- l'indice 7 ne passe pas
- 8
- 9

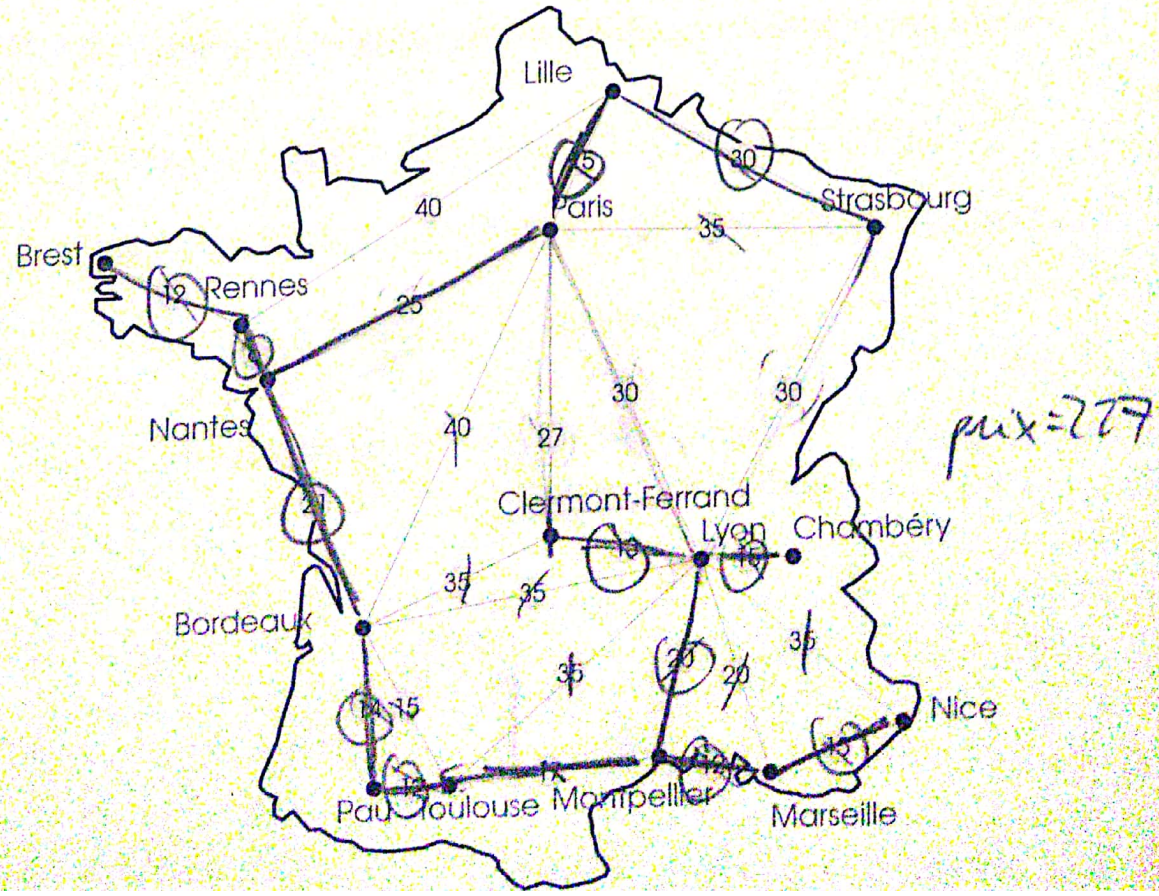
Mise à jour :

- 6

L'algorithme de Kruskal établit le coût minimal de connexion des villes :



Le coût minimal est 16

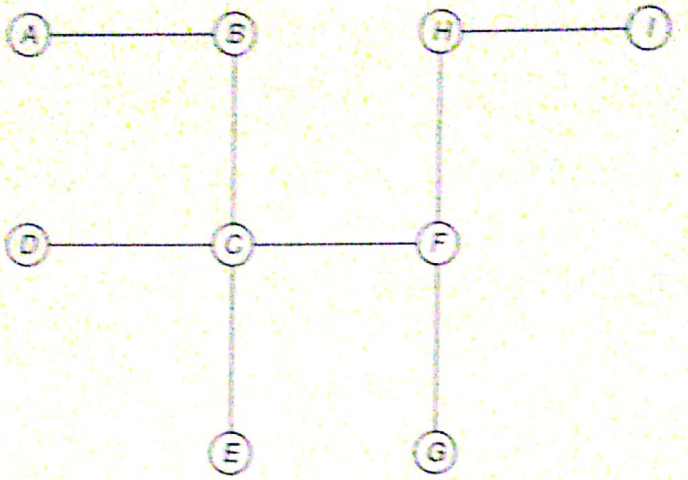


Les arbres utilisés en algorithmique ont le plus souvent une orientation et un sommet qui joue un rôle particulier : la racine.

Définition (Arbre enraciné : cas d'un graphe non orienté)

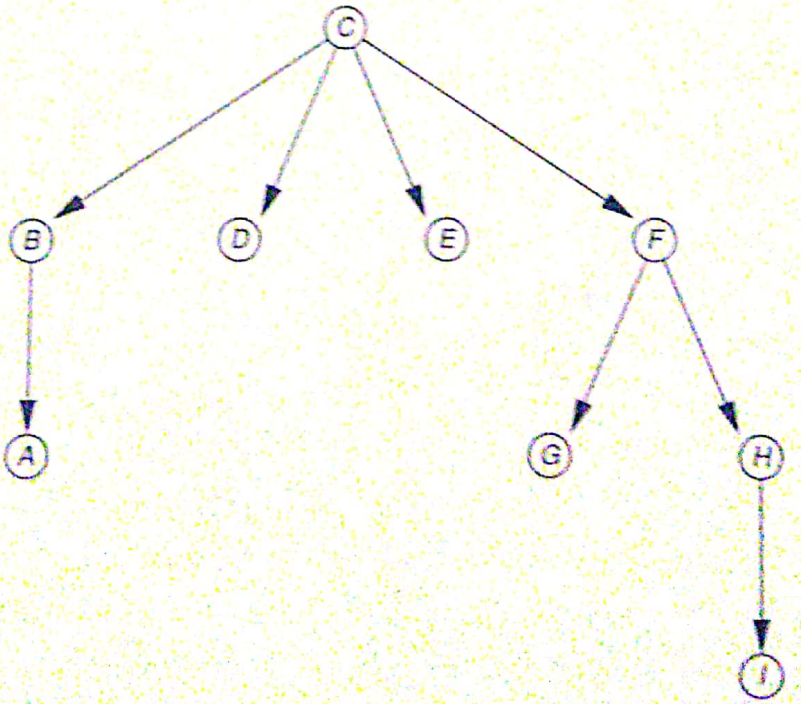
Un graphe non orienté est un **arbre enraciné** *(ou une* **arborescence***) s'il est connexe et acyclique et si un sommet particulier, la racine, a été distingué.*

Considérons le graphe non orienté suivant :

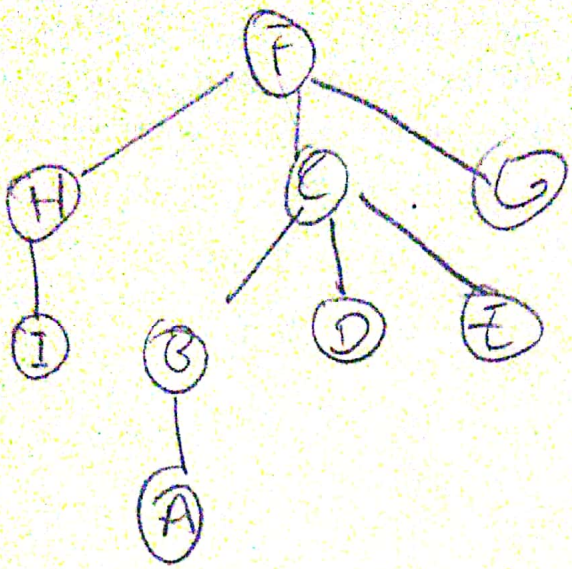


Arbres | Arbres enracinés

Si la racine est *C*, il en résulte l'arbre enraciné suivant :

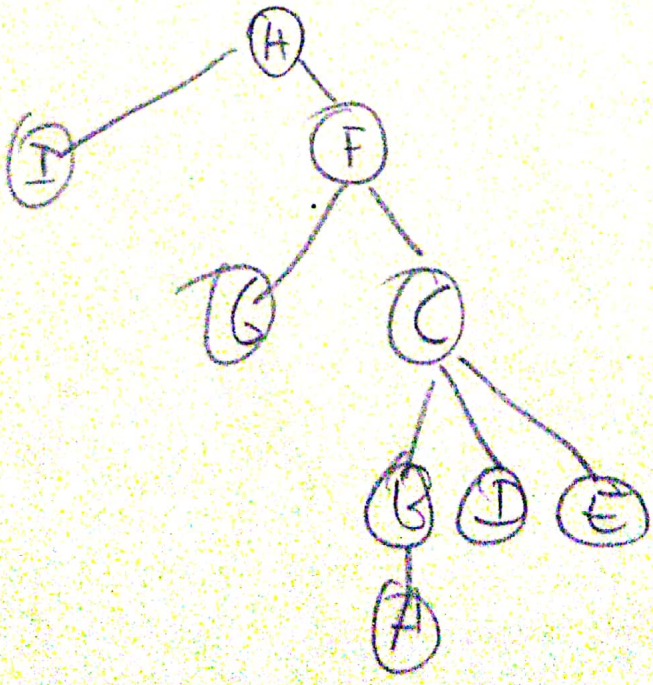


Si la racine est *F*, il en résulte l'arbre enraciné suivant :



Arbres | Arbres enracinés

Si la racine est *H*, il en résulte l'arbre enraciné suivant :



Un arbre enraciné, défini à partir d'un graphe non orienté, est muni d'une **orientation naturelle** : on oriente chaque arête de telle sorte qu'il existe un chemin de la racine à tout autre sommet.

Un arbre enraciné, défini à partir d'un **graphe non orienté** permet donc de définir un **graphe orienté** qui est aussi appelé **arbre enraciné** ou plus simplement **arbre**.

Cela permet de généraliser la notion d'arbre au cas des graphes orientés :

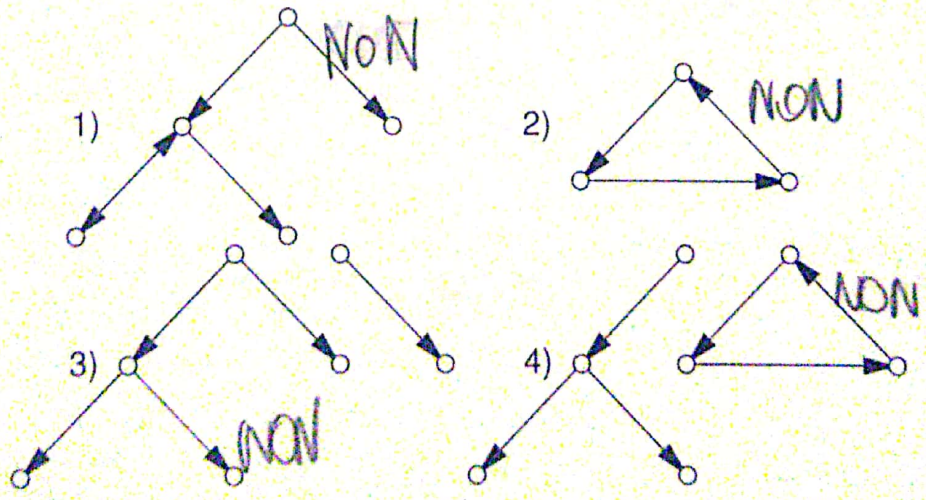
Définition (Arbre enraciné : cas d'un graphe orienté)

Un graphe orienté est un arbre enraciné si, et seulement si,

- *il est connexe,*
- *il a un unique sommet sans prédécesseur (la racine), et tous ses autres sommets ont exactement un prédécesseur.*

Arbres | Arbres enracinés

Parmi les graphes suivants, lesquels sont des arbres ?

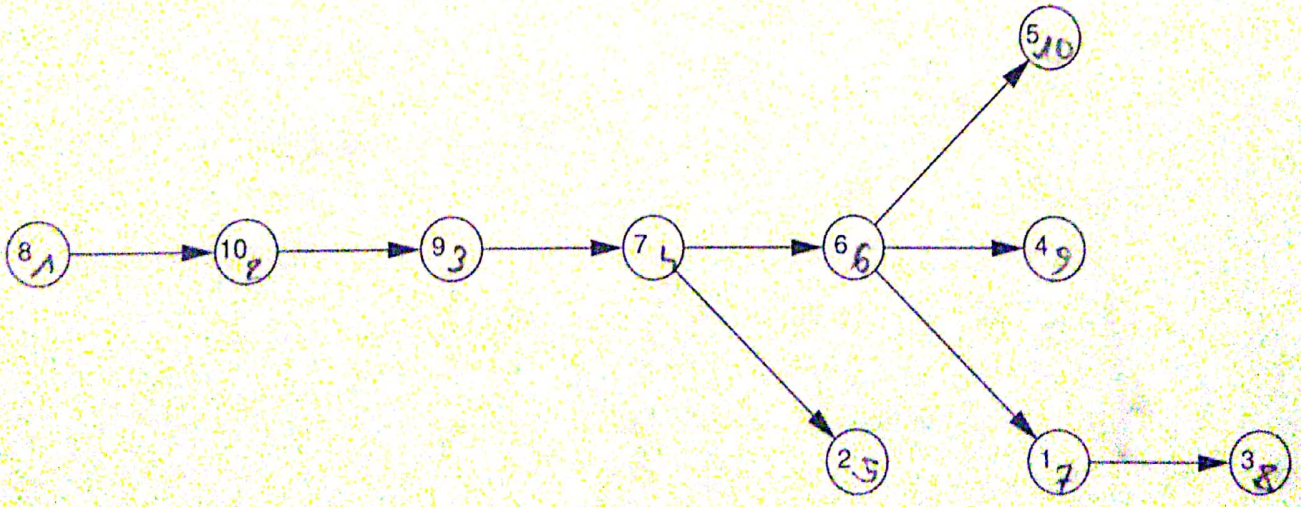


Définition (Numérotation préfixe et postfixe)

On appelle **numérotation préfixe** *et* **numérotation postfixe** *les numérotations des sommets d'un arbre correspondant à l'ordre de traitement des sommets de l'arbre lors d'un* **parcours en profondeur** *suivant qu'on numérote un sommet* **avant** *ou* **après** *le traitement de ses successeurs.*

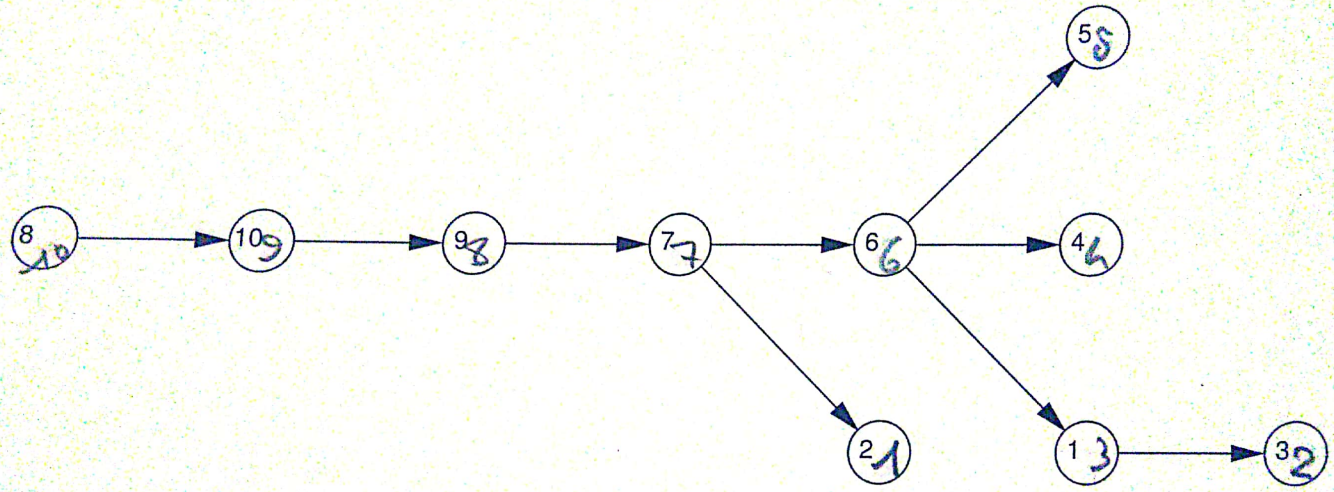
Arbres | Numérotation des sommets

Considérons l'arbre suivant :



numérotation préfixe [1] = 8, 10, 9, 7, 2, 6, 4, 3, 4, 15

Pour le graphe présenté précédemment :



numérotation postfixe[2] = 2, 3, 1, 4, 5, 6, 7, 8, 9, 10, 8

2. avec la convention du « plus petit choix possible »