Numérique et science informatique
Classe de Terminale

Lycée hoche

année scolaire 2025-2026

Contents

1 Notion de graphe

Remarque: Nous présentons ici quelques définitions de base sur les graphes, notions qui seront utiles pour introduire la notion d'arbre.
Nous reviendrons dans un chapitre ultérieur et de manière plus développée à la notion de graphe.

Graphe non orienté

On appelle graphe non orienté la donnée d'un ensemble $G(S, A)$, où S est un ensemble fini d'éléments appelés sommets, et où A est un multi-ensemble de paires non ordonnées d'éléments de S appelées arêtes.

Remarques

- Le nombre de sommets est appelé **l'ordre** du graphe et le nombre d'arêtes est appelé la **taille** du graphe.
- Dans une représentation graphique, les sommets sont représentés par des cercles et les arêtes par des traits.
- Une arête entre deux sommets u et v est notée indifféremment par le couple (u, v) ou par le couple (v, u) .
- Une arête reliant un sommet à lui-même est appelée une **boucle**.
- Deux sommets peuvent être reliés par plusieurs arêtes.
- Si le graphe n'a pas de boucle et a au plus une arête entre deux sommets quelconques, on parle de graphe simple non orienté. Dans le cas contraire, on parle de multi-graphe non orienté ou de pseudo-graphe non orienté.

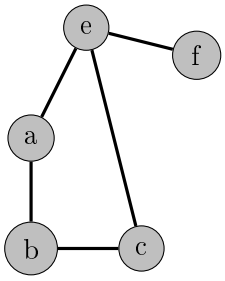
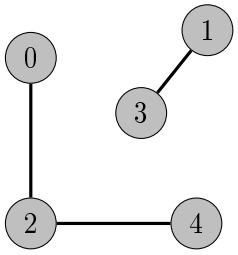


fig (2a)



fig(2b)

1.1 Chemin entre deux sommets

Quelques définitions

Définition 1

Dans un graphe, orienté ou non, on appelle :

- **Chemin** : Toute liste ordonnée de sommets telle que chaque sommet de la liste (hormis le premier) soit adjacent au précédent ;
- **chemin simple**: Un chemin constitué d'arêtes (ou d'arcs) toutes distinctes ;
- **Chemin fermé** : Un chemin dont le premier et le dernier sommet sont confondus ;
- **Cycle** : Un chemin simple fermé, c'est-à-dire un chemin fermé composé d'arêtes (ou d'arcs) toutes distinctes.

Définition 2

Dans un graphe on appelle :

- **Longueur** d'un chemin le nombre d'arêtes (ou d'arcs) qui le composent .
- **Distance** d'un sommet u à un sommet v : la plus petite des longueurs des chemins partant de et arrivant à .
- **Diamètre** du graphe : La plus grande distance entre deux sommets quelconques.

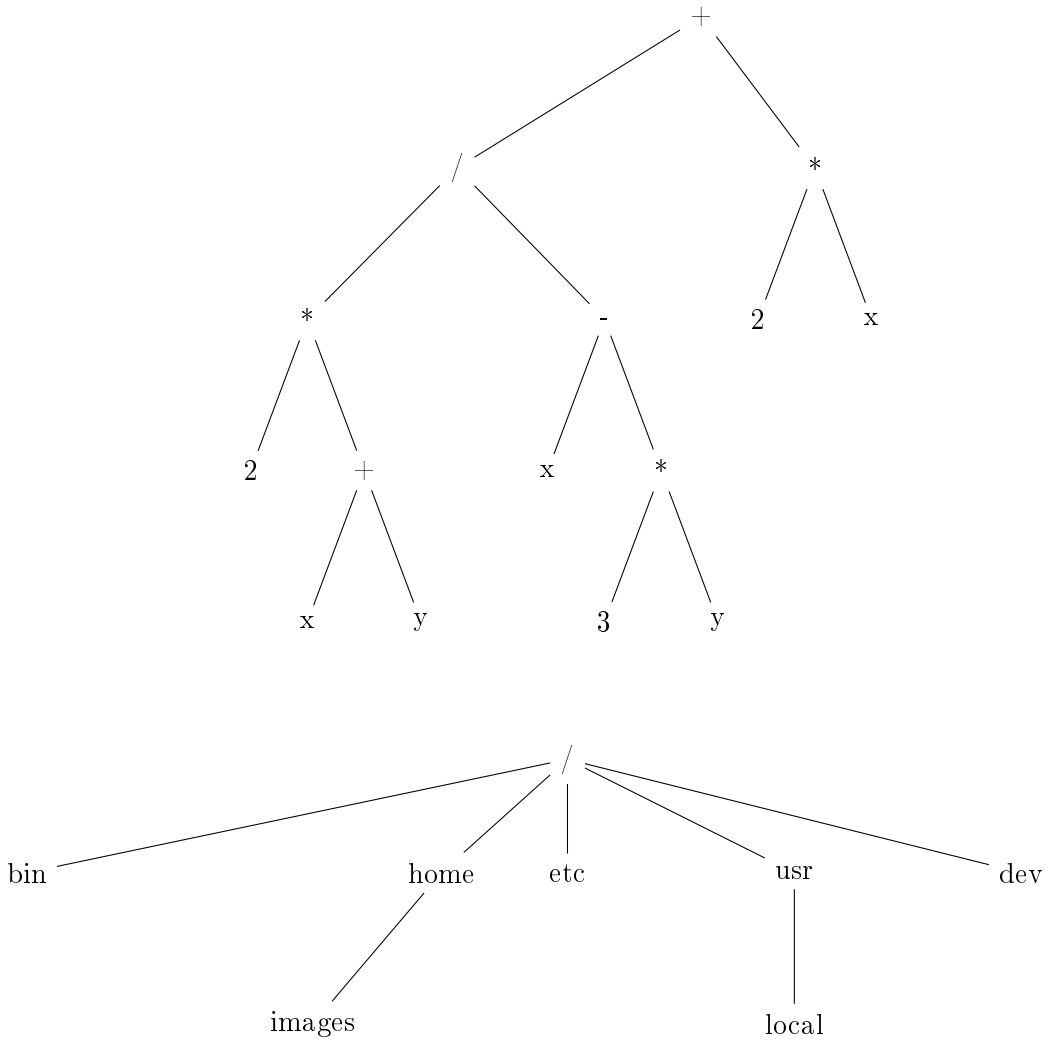
1.2 Connexité

Un graphe non orienté est dit connexe s'il existe un chemin entre deux sommets quelconques.

2 Arbres

Les arbres sont des graphes possédant une structure relationnelle particulière appelée structure hiérarchique. Ils sont parmi les structures de données les plus importantes et les plus utilisées en informatique, car ils interviennent dans de très nombreux problèmes. On les classe généralement en plusieurs catégories suivant ce qu'ils modélisent :

- arbres de classification permettant la représentation d'une arborescence de fichiers, du DOM d'une page web... ;
- arbres généalogiques descendants ;
- arbres syntaxiques permettant la représentation d'expressions arithmétiques, l'analyse d'une phrase... ;
- arbres de parcours résultants d'un parcours en largeur ou en profondeur d'un graphe ;
- arbres couvrants servant à« capturer »une structure particulière d'un graphe connexe ;
- arbres lexicographique permettant de représenter un dictionnaire... ;

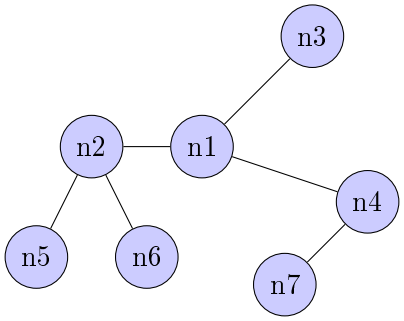

+
/
*
*
-
2
x
2
+
x
*
x
y
3
y
/
bin
home
etc
usr
dev
images
local

Définition

On appelle **arbre libre** un graphe A non orienté, connexe et acyclique. Les sommets de A sont communément appelés nœuds. Le nombre de nœuds est appelé la taille de A et est noté taille(A).

Exemple 1:

Le graphe suivant a pour taille 7



Théorème 1

Soit A un graphe non orienté. Les propriétés suivantes sont équivalentes :

1. A est un arbre libre.
2. Deux sommets quelconques de A sont reliés par un unique chemin simple.
3. A est connexe, mais ne l'est plus si on enlève n'importe laquelle de ses arêtes.
4. A st connexe et son nombre d'arêtes est égal à son nombre de sommets moins 1.
5. A est acyclique, mais ne l'est plus si on ajoute n'importe quelle arête.
6. A est acyclique et son nombre d'arêtes est égal à son nombre de sommets moins 1.

Pour prendre en compte toute la spécificité des arbres par rapport aux graphes généraux, et notamment pour stocker plus facilement des informations, il est souvent utile (et c'est ce que nous ferons dans la suite) de particulariser un sommet qui va servir "d'ancre"à l'arbre. On a alors la définition suivante :

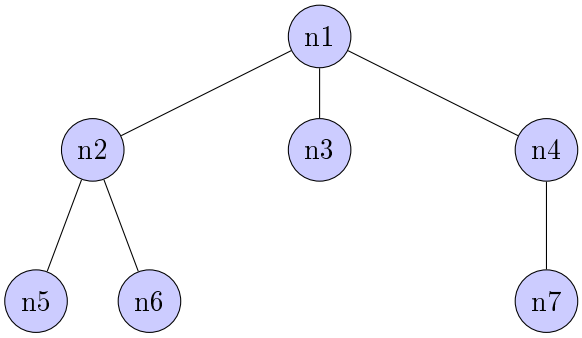
2.1 Vocabulaire

Définition 2

On appelle arbre **enraciné**, ou simplement arbre, un arbre libre dans lequel l'un des sommets se distingue des autres. Ce sommet particulier est appelé la racine de l'arbre.Pour dessiner un arbre, on procède de la façon suivante : on place sa racine en haut, puis tous les nœuds situés à distance i de la racine à la ligne numéro i. Autrement dit, en informatique, les arbres poussent de haut en bas...

Exemple 2

En prenant comme racine le sommet n1 dans l'arbre libre de l'exemple 1, on obtient l'arbre suivant :

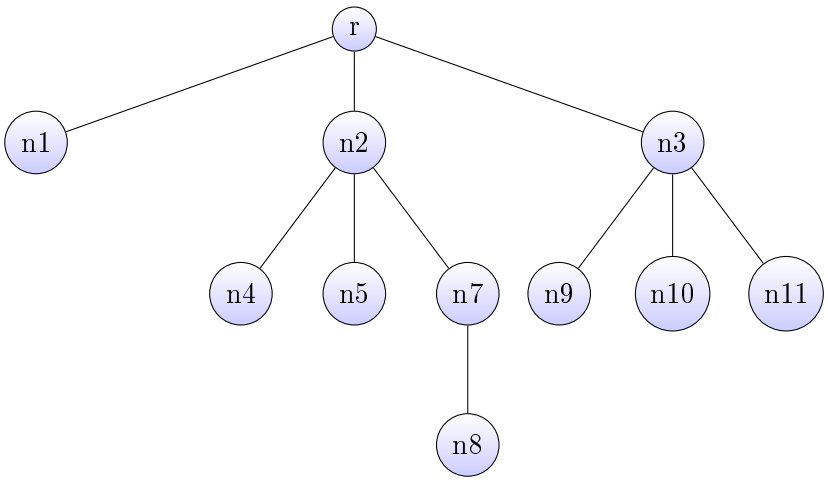


Exemple 3

• L'arborescence des fichiers d'un système Unix est un arbre dont la racine est le dossier Root.
• Le DOM d'une page web est un arbre dont la racine est l'élément html.
• Un arbre lexicographique permettant de construire un dictionnaire de mots commençant par la lettre A est un arbre de racine A.

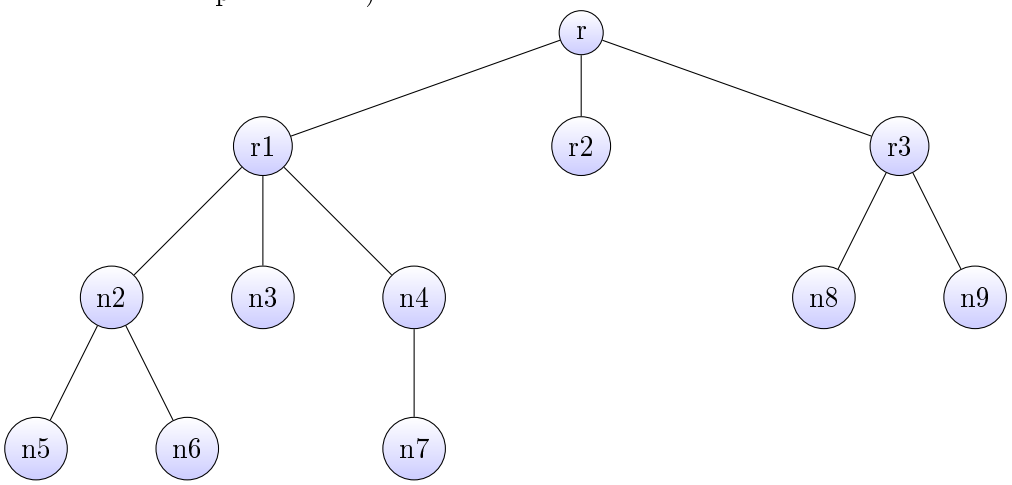
Définition 3

Il est souvent utile de voir un arbre (enraciné) comme une structure de données récursive :
• cas de base: un nœud unique r est un arbre de racine r;
• récurrence: un arbre avec au moins deux nœuds est constitué d'une racine r et d'une suite de sous-arbres A1,... Ak dont les racines r1,..., rk sont les successeurs de r.



Exemple 4

La figure suivante est un arbre de racine r constitué de trois sous-arbres A1, A2 et A3 dont les racines r1, r2 et r3 sont les successeurs de r. L'arbre A2 correspond au cas de base, car il est constitué d'un nœud unique. L'arbre A3 est constitué de deux sous-arbres possédant chacun un nœud unique (ce qui termine la description de A3).



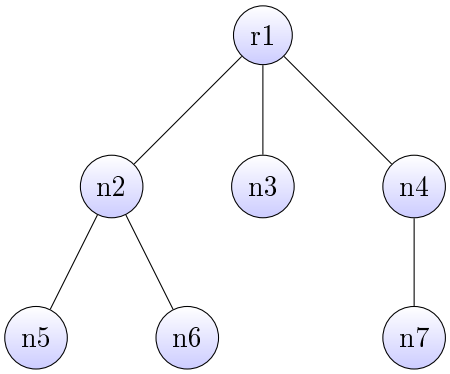
Exercice : Décrire récursivement l'arbre A1 de l'exemple 4.

Définition 4

Père,fils et frères

- Un noeud est le père d'un noeud y si y est un successeur de x.
- Deux noeuds sont frères s'ils ont le même père.

Exemple 5



Remarques:

- La racine est le seul nœud de l'arbre n'ayant pas de père.
- Un nœud sans fils est appelé nœud externe ou **feuille**. Le nombre de feuilles d'un arbre A est noté nf(A).
- Un nœud qui n'est pas une feuille est appelé nœud interne.
- La racine peut être une feuille ou un nœud interne suivant les arbres.

Exercice : Donner les feuilles et les nœuds internes de l'arbre de l'exemple 5.

Définition 5

Ancêtres-descendants et sous arbres

On appelle ancêtres d'un noeud x tous les nœuds distincts de x trouvés sur le chemin simple allant de la racine à x.
- Un noeud y est appelé descendant d'un noeud x si x est un ancêtre de y.
- On appelle sous-arbre de racine x l'arbre induit par les descendants de x.

Exemple 6:
Dans l'arbre de l'exemple 5 : r1 et n2 sont les ancêtres de n6, n5 et n6 sont les descendants de n2. Le sous-arbre de racine n2 est l'arbre constitué des nœuds n2, n5 et n6.

3 Attributs

Un noeud n d'un arbre possède différents attributs permettant de stocker diverses données. On utilise principalement :

- n.clé pour désigner la "valeur"contenue dans n ;
- n.père pour désigner le père de n;
- n.fils1 pour désigner un fils de n;
- n.fils2 pour désigner un deuxième fils de n;
- ...Il peut aussi être utile dans certains cas d'utiliser directement un tableau n.fils pour stocker tous les fils de n. Bien sûr, suivant ce que l'on désire stocker dans le nœud, d'autres attributs comme des couleurs peuvent être utilisés.

Remarque:

Lorsqu'un attribut est vide, on lui attribue la valeur particulière null.
Exercice 4

Dans l'arbre de l'exemple 5 , compléter:

- r1.père = ...
- n2.père = ...
- n3.père = ...
- r1.fils1 = ...
- n5.père = ...
- n2.fils2 = ...
- n4.fils = ...
- n7.fils = ...

3.1 Implémentation

Puisqu'un arbre est un graphe particulier, on peut utiliser en Python une implémentation sous forme de dictionnaire où les clés sont les "valeurs" des différents nœuds et où les valeurs associées sont des listes regroupant les attributs nécessaires (père, fils, etc.). Pour certains types d'arbres, on utilise une implémentation plus pertinente fondée sur la définition récursive d'un arbre et utilisant la programmation objet (étude ultérieure).

3.2 Quelques mesures:Profondeur-Niveau-taille-hauteur

Soit A un arbre de racine r. On appelle :

- Profondeur d'un nœud n la longueur du chemin simple entre r et n ;
- Niveau :l'ensemble de tous les nœuds de même profondeur ;
- Hauteur d'un nœud n la longueur du chemin simple le plus long qui relie n à une feuille dont il est un ancêtre ;
- hauteur de A la hauteur de la racine r, c'est-à-dire la longueur du chemin simple le plus long qui relie r à une feuille de l'arbre.

3.3 Cheminement et profondeur moyenne

Définition 6

Soit A un arbre de racine r.

On appelle :

• longueur de cheminement de A la somme LC(A) des longueurs de tous les chemins issus de r ;
• longueur de cheminement externe de A la somme LCE(A) des longueurs de tous les chemins issus de r et aboutissant à une feuille.

Figure 1: A picture of the universe!

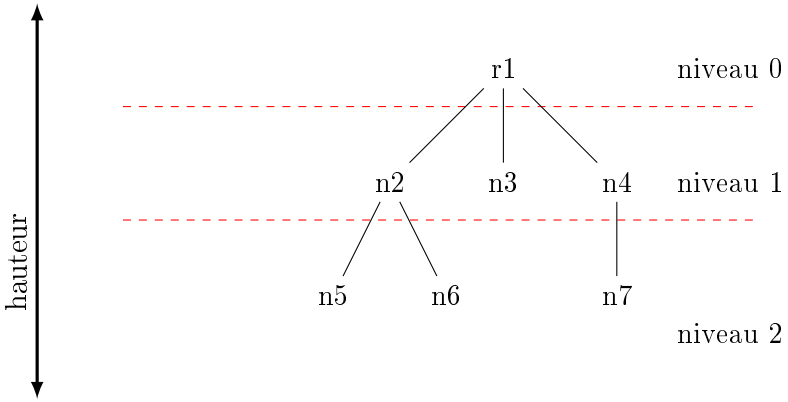


Figure 2: A picture of the universe!

Définition 7

Soit A un arbre de racine r. On appelle :

• profondeur moyenne d'un nœud de A la moyenne PM(A) des profondeurs de tous les nœuds de A :

$$PM(A) = \frac{LC(A)}{taille(A)}$$

• profondeur moyenne externe d'une feuille de A la moyenne PME(A) des profondeurs de toutes les feuilles de A

$$PME(A) = \frac{LCE(A)}{nf(A)}$$

Exercice

1. Déterminer la taille, le nombre de feuilles et le nombre de nœuds internes de A.
2. Quels sont les nœuds de niveau 3 ?
3. Quelles sont la profondeur et la hauteur des nœuds 12, 10 et 3 ?
4. Quelle est la hauteur de l'arbre A?
5. Quelle est la longueur de cheminement de A? En déduire la profondeur moyenne d'un nœud de A.
6. Quelle est la profondeur moyenne externe d'une feuille de A?

4 Algorithmes de parcours d'un arbre

4.1 Algorithmes de parcours en profondeur

Parcours infixe
L'algorithme récursif affiche la clé de la racine d'un sous-arbre entre les clés du sous-arbre gauche et du sous- arbre droit

Algorithme 1 : Parcours-infixe

```
fonction PARCOURS-INFIXE(x) ;
si x ≠ null alors
    PARCOURS-INFIXE(x.gauche) ;
    afficher x.clé ;
    PARCOURS-INFIXE(x.droite) ;
fin
```

Parcours préfixe
L'algorithme récursif affiche la clé de la racine du sous-arbre avant les clés du sous-arbre gauche et du sous- arbre droit .

Algorithme 2 : Parcours-préfixe

```
fonction PARCOURS-PREFIXE(x) ;
si x ≠ null alors
    afficher x.clé ;
    PARCOURS-PREFIXE(x.gauche) ;
    PARCOURS-PREFIXE(x.droite) ;
fin
```

Parcours suffixe

L'algorithme récursif affiche les clés du sous-arbre gauche et du sous- arbre droit avant la clé de la racine du sous-arbre .

Algorithme 3 : Parcours-suffixe

```
fonction PARCOURS-SUFFIXE(x) ;
si x ≠ null alors
    PARCOURS-SUFFIXE(x.gauche) ;
    PARCOURS-SUFFIXE(x.droite) ;
    afficher x.clé ;
fin
```

4.2 Algorithme de parcours en largeur

Algorithme 4 : Parcours en largeur

```
f est une file vide ;
enfiler(T.racine, f) ;
tant que f est non vide faire
    x ← defiler(f);
    afficher x.clé ;
    si x.gauche != nil alors
        T.g ←x.gauche ;
        enfiler(Tg.racine, f) ;
    fin
    si x.droit != nil alors
        T.d ←x.droite ;
        enfiler(Td.racine, f) ;
    fin
fin
```