# 21 - Le type abstrait ARBRE

## I – Type abstrait de données : Arbre enraciné ou Arborescence

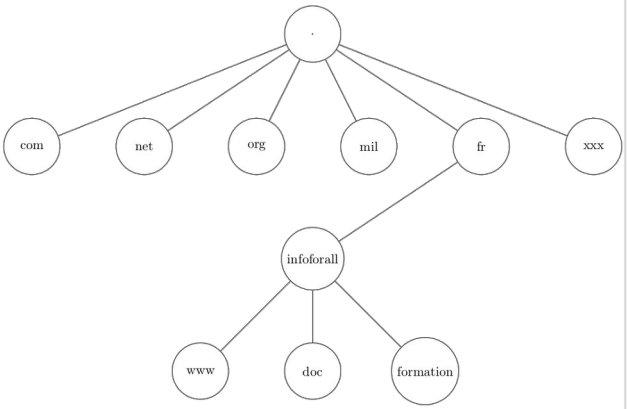
### 1.1 Qu'est-ce qui n'est pas un arbre enraciné ?

Une structure qui possède un cycle, qui n'est pas connexe ou que ne possède pas de point de départ clairement défini n'est pas un arbre enraciné.

### 1.2 Idée générale

Un **arbre enraciné** est un **type abstrait** ayant les propriétés suivantes (on parle ici des arborescences) :

- Il est composé de **noeuds** contenant au moins deux informations :
  - Une donnée d'un type donnée (un nombre, un string...);
  - Les adresses de ses noeuds-successeurs éventuels.
- On **hiérarchise** les noeuds entre eux et il en existe trois types :
  - Une **racine unique** : ce noeud est particulier car il est le seul à ne pas posséder de noeud-parent. Il est tout en haut de la hiérarchie. possède pas de fils.
  - Les **noeuds internes** : il s'agit des noeuds qui ont un noeud-parent et possède un ou plusieurs noeuds-enfants.
  - Les **feuilles** : il s'agit des noeuds qui ont un noeud-parent mais qui n'ont pas de noeud-enfant.
- Le terme noeud générique fait donc référence à la racine, aux noeuds internes et aux feuilles.




**01°** L'élément html joue un rôle particulier puisqu'il n'est pas contenu dans une autre balise. Cet élément est la racine de notre arbre. Combien de choix de direction doit-on faire pour trouver la balise div en partant de cette racine html ? Voir le code HTML sur le site.

**02°** Représenter l'arborescence de l'ordinateur dont on vous fournit la structure sur le site.

**03°** Représenter l'arbre d'une maison ou d'un appartement dont vous connaissez la structure. Voir le site.

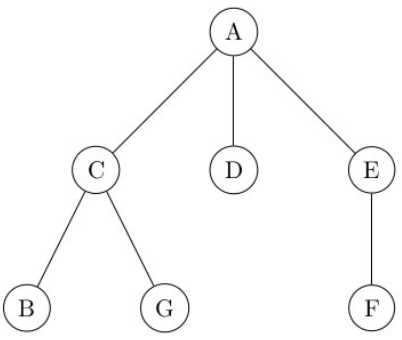
**04°** Représenter maintenant la même situation mais en changeant de racine : vous prendrez comme racine une autre salle que l'entrée principale. Obtient-on le même arbre ?

**05°** Dans le cas d'une recherche orientée à l'aide d'un arbre, le choix de la racine est-il important ? Répondre en montrant qu'aux questions 03 et 04 certaines salles n'auront pas le même éloignement de la racine.

**05°** Que donnerait la représentation directe d'une liste dont la tête est 5 et la queue 10-15-20-25-30-35 sous forme d'un arbre ?

### 1.3 Définition d'un arbre enraciné à partir des noeuds

Définition : un **arbre enraciné** est composé d'au moins un noeud particulier : une racine. Avec cette définition, on voit qu'un arbre enraciné ne peut pas être vide : il y a toujours au moins la racine.
L'arbre est une structure récursive puisqu'un arbre contient potentiellement d'autres arbres.




**06°** Combien d'arbres enracinés dans l'arbre ci-dessous ?
**07°** Peut-on considérer qu'un arbre enraciné peut-être vide ? Justifiez.

**08°** Quelle est la racine cet arbre ? Quelles sont les 4 feuilles de cet arbre ? Quels sont le ou les enfants du noeud E ?

**09°** Représentez l'arbre enraciné obtenu si on considère que la racine est plutôt le noeud D. Si on veut utiliser cette structure dans le but de faire des recherches, vaut-il mieux utiliser l'arbre enraciné en A ou l'arbre enraciné en D ?

**10° Trois questions : représenter** un arbre qui serait la représentation directe d'une liste dont : la tête est 5 et la queue 10->15->20->25->30->35. **Quelle est la caractéristique** en terme de nombre d'enfants qu'on obtient sur ce cas particulier d'arbre enraciné ? **Peut-on vraiment considére**r qu'une liste est un cas particulier d'arbre enraciné ?

## II – Type abstrait de données : Arbre binaire

### 2.1 L'idée générale de l'arbre binaire (1er partie de la définition)

**A - Définition récursive d'un Arbre Binaire** : un arbre binaire est composé

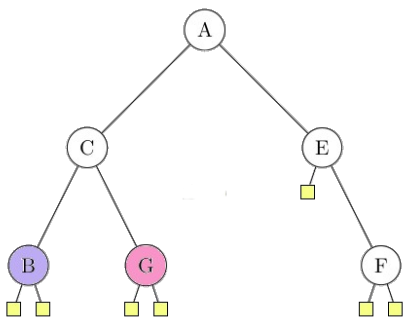
- soit d'un arbre binaire VIDE ∅ ;
- soit d'un arbre binaire NON VIDE, un triplet (r,g,d) composé d'une racine r menant à deux sous-arbres
  - le sous-arbre binaire gauche g (qui peut donc être un arbre binaire VIDE ou NON VIDE)
  - le sous-arbre binaire droite d (qui peut donc être un arbre binaire VIDE ou NON VIDE)

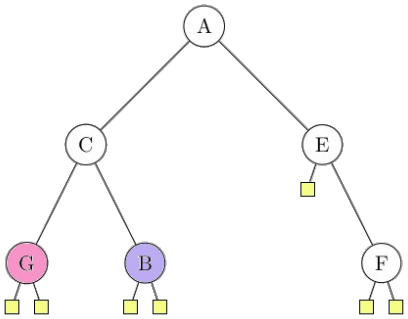
**B - Différences avec l'arbre enraciné** : trois différences majeures entre arbres enracinés et binaires :

1. L'arbre binaire peut être VIDE, alors qu'un arbre enraciné possède au moins un noeud-racine ;
2. Chaque noeud mène au maximum à deux enfants ;
3. Il doit exister une logique interne pour savoir si on place un enfant à gauche ou à droite de son parent.

**C – Exemple ;** On peut représenter ou pas l'arbre VIDE. Ci-dessous, on utilise un symbole particulier : le petit carré jaune.

Voici deux arbres binaires différents : H et G ne sont pas placés au même endroit.

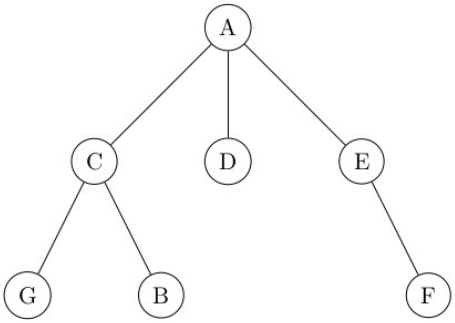


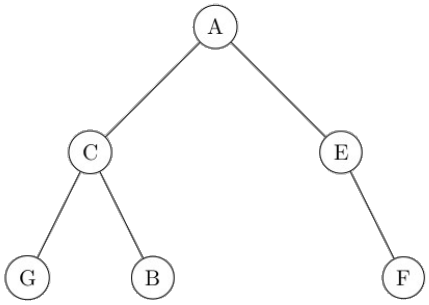




**D - Remarque importante**

Chaque noeud d'un arbre binaire non vide a au maximum deux enfants mais chaque arbre binaire a exactement deux sous-arbres binaires (puisque l'arbre binaire vide est un arbre binaire).

**11** Expliquer si l'arbre de gauche est un arbre binaire. Si l'arbre est binaire, noter les arbres vides non représentés sur l'arbre.




**12-A°** Expliquer si l'arbre de droite est un arbre binaire. Si l'arbre est binaire, noter le symbole des arbres vides non représentés sur l'arbre.

**12-B°** Entourer en rouge le sous-arbre gauche de l'arbre précédent (à droite). Entourer en bleu le sous-arbre droit. Entourer en vert le sous-arbre droit du sous-arbre gauche de l'arbre.

### 2.2 Les primitives (2e partie de la définition de l'arbre binaire) version immuable

1. **nvABV()** -> Arbre Binaire VIDE
2. **nvAB(v**:Element, **g**:Arbre Binaire, **d**:Arbre Binaire**)** -> Arbre Binaire NON VIDE
   Crée un AB dont la racine est un noeud portant v et les enfants les arbres g et d fournis.
3. **estABV(arbre**:Arbre Binaire**)** -> bool
4. **racine(arbre**:Arbre Binaire NON VIDE**)** -> Noeud
5. **contenu(noeud**:Noeud**)** -> Element
6. **gauche(arbre**:Arbre Binaire NON VIDE**)** -> Arbre Binaire
   Renvoie le sous-arbre gauche de l'arbre initial. On obtient bien un Arbre. Si vous voulez le noeud gauche, il faudra appliquer en plus la fonction racine.
7. **gauche(arbre**:Arbre Binaire NON VIDE**)** -> Arbre Binaire

**13°** Créer l'arbre de la question 12 à l'aide des primitives. On considère que le contenu est le nom du noeud.

**14°** Utiliser les primitives pour stocker dans une variable nommée valeur l'étiquette du noeud correspondant à l'enfant-gauche de la racine de l'arbre. On considère qu'on a accès initialement uniquement à la variable abr qui fait référence à l'arbre de racine A.

**15°** Utiliser les primitives pour stocker dans une variable reponse l'enfant-droite de l'enfant-gauche de la racine.

Attention : n'oubliez pas qu'on part de la racine, on doit donc lire la phrase de l'énoncé en sens inverse. Stocker ensuite la valeur associée à ce noeud dans une variable valeur. On considère qu'on a accès uniquement à la variable abr qui fait référence à l'arbre de noeud A.

## III – Caractéristique des arbres BINAIRES

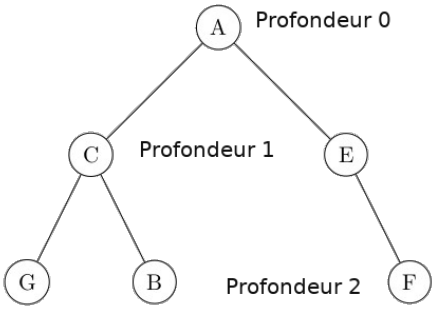
**3,1 Taille d'un arbre :** La taille d'un arbre correspond au **nombre de nœuds** présents dans l'arbre.

**3.2 Arête : une arête** est le **segment qui relie deux noeuds.**

**3.3 Profondeur d'un nœud (deux conventions possibles)**

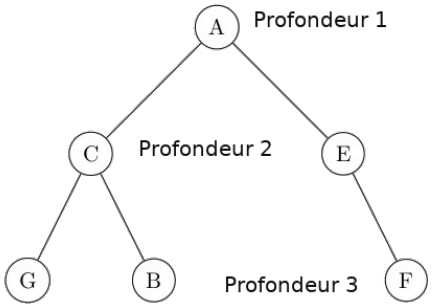
Convention profondeur 0 pour la racine

**Définition** : la profondeur d'un noeud est le nombre d'arêtes à emprunter pour atteindre ce noeud en partant de la racine.




Convention profondeur 1 pour la racine

**Définition** : la profondeur d'un noeud est alors 1 + le nombre d'arêtes à emprunter pour atteindre ce noeud en partant de la racine.




Propriété : quelque soit la convention, la profondeur d'un enfant est supérieure de 1 à celle de son parent.
Propriété : si 2 noeuds ont la même profondeur, c'est qu'ils sont à la même distance de la racine, même « étage ».

**3.4 Hauteur d'un arbre : la hauteur d'un ARBRE est liée à la profondeur maximale qu'on trouve dans l'arbre binaire.**

**Hauteur en utilisant une profondeur de 0 pour la racine**

Ici, la hauteur de l'arbre est donc 2.
Avec cette convention :

- la hauteur d'un arbre composé d'un n**oeud unique est 0**.
- la hauteur d'un arbre vide serait donc -1.

On fait souvent ce choix sur les arbres enracinés : ils ne peuvent pas être vides.

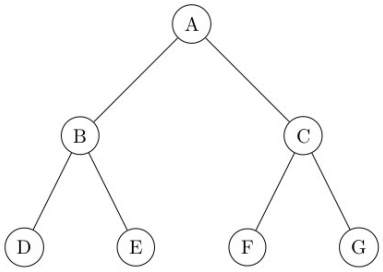
**Hauteur en utilisant une profondeur de 1 pour la racine**

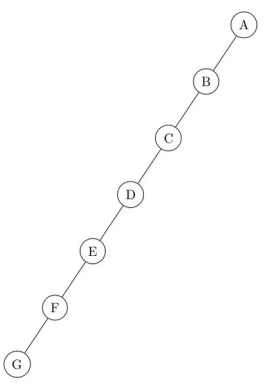
Ici, la hauteur de l'arbre est donc 3.
Avec cette convention :

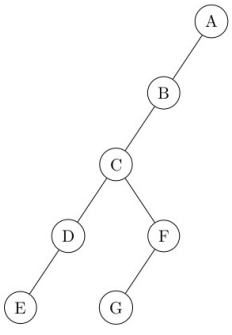
- la hauteur d'un arbre composé d'un noeud unique est 1.
- la hauteur d'un **arbre vide serait donc 0**.

On fait souvent ce choix sur les arbres binaires : ils peuvent être vides.

**16°** On considère la convention $p_R = 0$. Fournir la taille, la hauteur et le nombre d'arêtes de l'**arbre parfait** (de gauche). Fournir également la profondeur du noeud C.
**17°** Fournir la taille, la hauteur et le nombre d'arêtes de l'**arbre filiforme ci-dessous** (celui du milieu). Fournir également la profondeur du noeud C.
**18°** Fournir la taille, la hauteur et le nombre d'arêtes de l'arbre de droite. Fournir la profondeur du noeud C.



**3.5 Vocabulaire :**

**Arbre binaire parfait** : Un arbre est parfait lorsqu'il ne manque aucun noeud.

**Arbre binaire complet** : On dit qu'un est complet lorsque les noeuds manquants sont uniquement sur son dernier étage.

**Arbre binaire dégénéré** : Lorsque chaque noeud possède au maximum un seul enfant, on dit qu'il est dégénéré.

**Arbre binaire filiforme** : Lorsque l'arbre comporte uniquement des sous-arbres d'un côté identique, on dit qu'il est filiforme. Il pourrait représenter une liste.

**D'autres définitions existent ! C**e vocabulaire n'est pas si défini que cela. On trouve parfois : parfait qui est remplacé par complet ; complet qui est remplacé par presque complet**.**

## IV – Encadrement de la hauteur

**Encadrement de la hauteur avec la convention profondeur 1 pour la racine**

**19-A °** Quelle va être la hauteur d'un arbre filiforme de taille n si on prend une profondeur de 1 pour la racine ?

**19-B°** On considère la convention $p_R = 1$.

1. Calculer les tailles manquantes pour les Arbres Binaires Parfaits dont on vous donne des valeurs de hauteur h.
   On remarquera qu'à chaque nouvel étage, on rajoute le double de noeuds de l'étage précédent.
   - Hauteur h = 1 : La taille est n = 1
   - Hauteur h = 2 : La taille est n = 1 + 2 = 3
   - Hauteur h = 3 : La taille est n = 1 + 2 + 4 = 7
   - Hauteur h = 4 : La taille est n = ...
   - Hauteur h = 5 : La taille est n = ...
2. Trouver alors la formule permettant de connaître la taille n d'un Arbre Binaire Parfait dont on connaît la hauteur h.

**19-C°** En partant de la formule précédente, trouver la formule donnant la hauteur h d'un arbre binaire parfait en fonction de sa taille n : $\mathbf{n = 2^h - 1}$ (formule valable si profondeur 1 pour la racine)

**19-D°** Déduire des questions 19-A et 19-C l'encadrement de la hauteur d'un arbre binaire quelconque :

**... ≤ h ≤ ...** (formule valable si profondeur 1 pour la racine)

On ne tiendra pas compte pour le moment que les logarithmes donnent parfois des valeurs non entières.

**Encadrement de la hauteur avec la convention profondeur 0 pour la racine**

Voir le site pour les questions 20-A à 20-D.

**BILAN : estimation de la hauteur h d'un arbre connaissant sa taille n** (formules à savoir utiliser)

| | *(convention racine à p=1)* | *(convention racine à p=0)* |
|---|---|---|
| • Arbre binaire filiforme : | $\mathbf{h = n}$ | $\mathbf{h = n - 1}$ |
| • Arbre binaire parfait : | $\mathbf{h = \log_2(n+1)}$ | $\mathbf{h = \log_2(n+1) - 1}$ |
| • Arbre binaire quelconque | $\mathbf{\lceil \log_2(n+1) \rceil \leq h \leq n}$ | $\mathbf{\lfloor \log_2(n) \rfloor \leq h \leq n-1}$ |
| | | $\lceil \log_2(n+1) \rceil - 1 \leq h \leq n-1$ |
| | **+1 et arrondi sup.** | **+0 et arrondi inf.** |

**Ordre de grandeur de la hauteur d'un arbre (à connaître par coeur)**

**h dépend de n dans le cas filiforme.**
**h dépend de $\log_2(n)$ dans le cas parfait.**
**L'ordre de grandeur de la hauteur d'un Arbre Binaire est compris entre $\log_2(n)$ et n.**

Ex : Avec la première formule, 15 donne une hauteur de 4 : ($\log_2(15+1)=4$).
Cette valeur est bien un entier car l'arbre peut être parfait si il possède 15 nœuds.