En informatique, toutes les structures de données ne sont pas linéaires. Il existe bien d'autres structures. **Nous allons étudier ici les structures hiérarchiques, aussi appelés arbres**.

1 Exemple : arborescence de fichiers

Les arbres servent à décrire des arborescences comme par exemple la structure des fichiers sur votre ordinateur. Sur Unix/Linux, le sommet de l'arborescence est désigné par `/`. On trouve en dessous un certain nombre de dossiers principaux comme `bin, lib, home, usr` etc... chacun ayant un rôle prédéfini : `home` sert par exemple à stocker les répertoires personnels de tous les utilisateurs du système. Pour avoir la liste de ce qui se trouve à la racine du système, on tape la commande :

```
$ ls -l
total 176
drwxr-xr-x   2 root root 12288 avril 15 19:14 bin
drwxr-xr-x   3 root root  4096 avril 17 09:35 boot
drwxrwxr-x   2 root root  4096 oct.  21  2017 cdrom
drwxr-xr-x  20 root root  4940 avril 17 08:53 dev
drwxr-xr-x 202 root root 12288 avril 16 19:19 etc
drwxr-xr-x   8 root root  4096 juin  14  2019 home
etc...
```

Les dossiers sont indiqués par le caractère **d** en début de ligne. L'arborescence se déroule de manière analogue pour chacun de ces dossiers, par exemple, dans le dossier `/usr` on retrouve une autre arborescence :

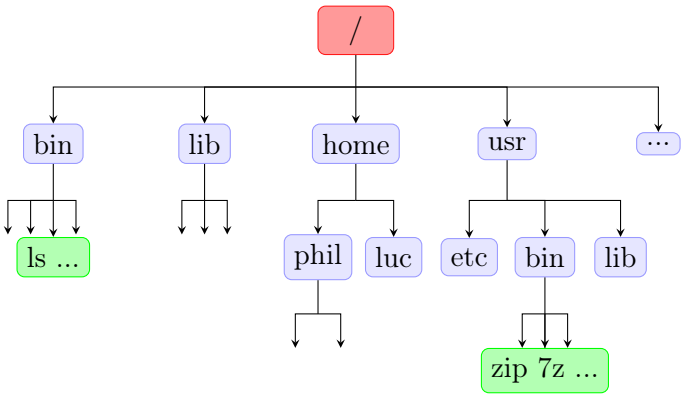
```
$ ls -l /usr
total 248
drwxr-xr-x   2 root root 131072 avril 16 19:18 bin
drwxr-xr-x   3 root root   4096 janv. 20  2018 etc
drwxr-xr-x   2 root root   4096 avril  5 20:22 games
drwxr-xr-x 116 root root  20480 avril  8 19:04 include
drwxr-xr-x 197 root root  20480 avril 16 19:17 lib
etc...
```

et ainsi de suite jusqu'à un moment où on ne trouve plus de dossiers, mais uniquement des fichiers. C'est le cas dans `/usr/bin` par exemple.

```
$ ls -l /usr/bin
total 1424264
-rwxr-xr-x 1 root root          96 avril  7 14:05 2to3-2.7
-rwxr-xr-x 1 root root       10104 avril 23  2016 411toppm
-rwxr-xr-x 1 root root          39 août   9  2019 7z
-rwxr-xr-x 1 root root          40 août   9  2019 7za
etc...
```

Nous n'avons plus ici que des fichiers et l'arborescence se termine.

On peut représenter ceci à l'aide d'un arbre dont voici un extrait :



Identifiez des exemples concrets dans lesquels la structure d'arbre intervient.

2 Vocabulaire sur les arbres

Reprenons l'exemple précédent :

- On retrouve la **racine** de l'arborescence en rouge : c'est là où l'arbre commence.
- On identifie les dossiers et sous dossiers en bleu : on les appelle des **nœuds**.
- Les arêtes qui relient les nœuds sont appelées **branches** de l'arbre.
- Les nœuds à un même niveau sont dits **frères**. Les dossiers descendent d'un nœud sont les **fils** de ce nœud.
- On a de même le nœud **père** qui désigne le nœud dont un nœud descend.
- Les fichiers en vert représentent les terminaisons de l'arbre. Lorsqu'un nœud n'a pas de descendants, on parle de **feuilles**.

On peut ajouter deux notions complémentaires : la **taille** et la **hauteur** de l'arbre.

La **taille** d'un arbre est le nombre de nœuds de l'arbre.

On définit la **profondeur** d'un nœud comme étant le nombre de nœuds traversés pour aller du nœud à la racine de l'arbre. Avec cette définition, la profondeur de la racine est 1.

La **hauteur** de l'arbre est la **profondeur maximale** des nœuds de l'arbre.

Remarque : On rencontre en fonction des ouvrages une autre définition de la profondeur ! Dans ce cas, on définit la profondeur d'un nœud comme étant le nombre de **branches** traversées pour aller du nœud à la racine de l'arbre. Dans ce cas la profondeur (et la hauteur) vaut 1 de moins que la définition précédente. Dans tous les cas, la profondeur augmente de 1 pour tous les descendants d'un nœud.

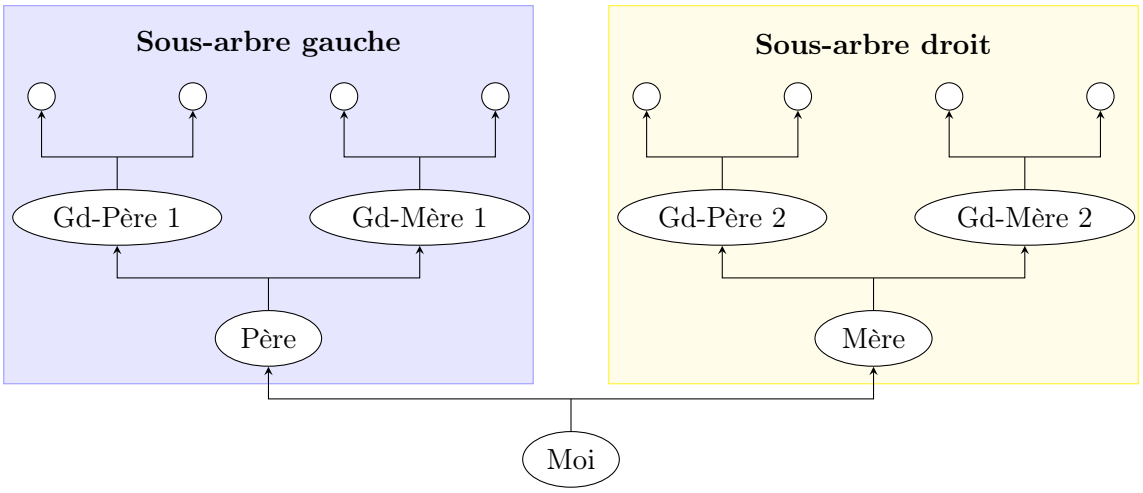
(*Le jour du BAC, la définition à appliquer sera rappelée*).

3 Les arbres binaires

Les **arbres binaires** sont un cas particulier d'arbre dans lequel de chaque nœud partent **au plus deux branches**.

Très clairement la structure d'arborescence de fichiers vue ci dessus n'est pas un arbre binaire. Si on dessine l'arbre de nos descendants, ce n'est pas non plus un arbre binaire car une personne peut avoir plus que 2 enfants.

Par contre, si on s'intéresse à nos ascendants, nous avons alors un arbre binaire :



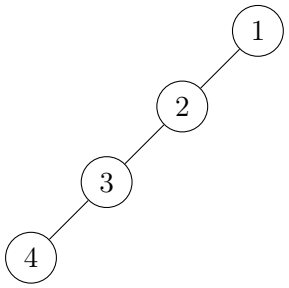
Il est intéressant de remarquer que dans un arbre binaire, les sous-arbres formés en prenant comme racine les fils d'un nœud qui n'est pas une feuille sont aussi des arbres binaires. On les nommes **sous-arbre gauche** et **sous-arbre droit** du nœud considéré. Sur l'illustration, nous avons représenté les sous arbres gauche et droit de la racine.

Un arbre binaire (non vide) est donc essentiellement un nœud racine qui contient un élément et deux arbres binaires (au maximum) : le sous-arbre gauche et le sous-arbre droit.

4 Relation taille et hauteur

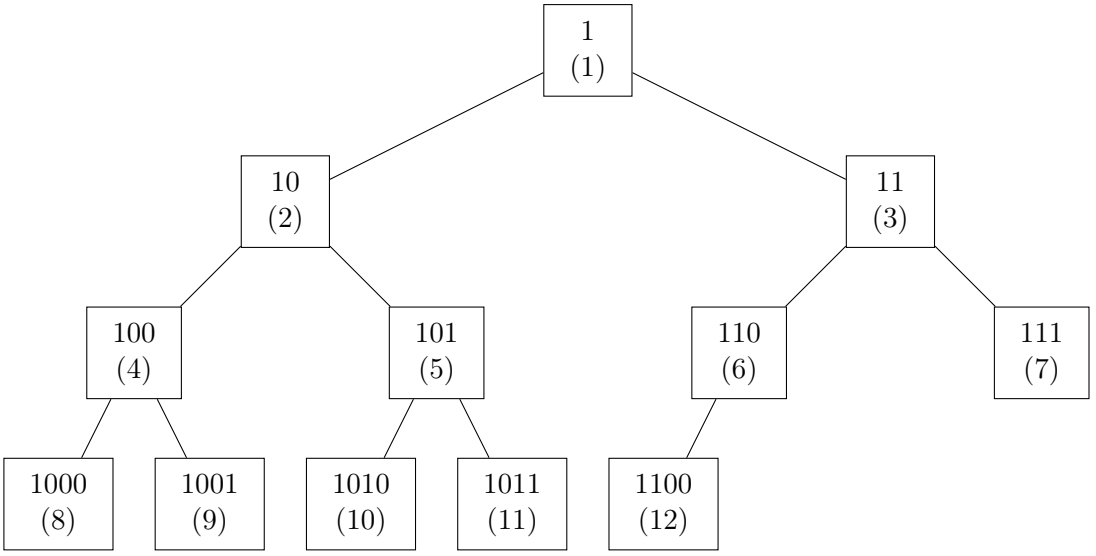
On peut considérer les deux arbres binaires de structures extrêmes suivants :

1. *Arbre dégénéré* :



Dans cet exemple, chaque nœud possède un seul fils (arbre dégénéré) ce qui conduit à une hauteur maximale h_{max} pour une taille n donnée : $\boldsymbol{h_{max} = n}$.

2. *Arbre tassé* :



Dans ce 2ème exemple, la hauteur est au contraire minimale pour une taille donnée. On dit que l'arbre est **tassé**.
L'étiquetage choisi correspond à la numérotation binaire des nœuds. Ainsi, pour une taille n, le dernier nœud est à la hauteur h, et s'écrit avec h bits en binaire.
Or le nombre de bits pour écrire un entier est égal à 1 + la partie entière du logarithme binaire de ce nombre. Donc pour cet arbre, on a $\boldsymbol{h = 1 + \lfloor \log_2(n) \rfloor}$.

Exemples :

n (dec)	n (bin)	$\log_2(n)$	$1 + \lfloor \log_2(n) \rfloor$
4	100	2	3
5	101	2,32	3
6	100	2,58	3
7	100	2,81	3
8	1000	3	4
9	1001	3,17	4

On a donc pour n'importe quel arbre binaire de hauteur h et de taille n les inégalités :

$$\boldsymbol{1 + \lfloor \log_2(n) \rfloor \leq h \leq n}$$

Remarque : Avec la convention d'une hauteur nulle pour un seul nœud, on aurait :
$\lfloor \log_2(n) \rfloor \leq h \leq n - 1$

En remarquant que le plus grand nombre qu'on peut écrire à une hauteur h est $2^h - 1$, on a aussi :

$$\boldsymbol{h \leq n \leq 2^h - 1}$$

Remarque : Avec la convention d'une hauteur nulle pour un seul nœud, on aurait :
$h + 1 \leq n \leq 2^{h+1} - 1$

5 Implémentation

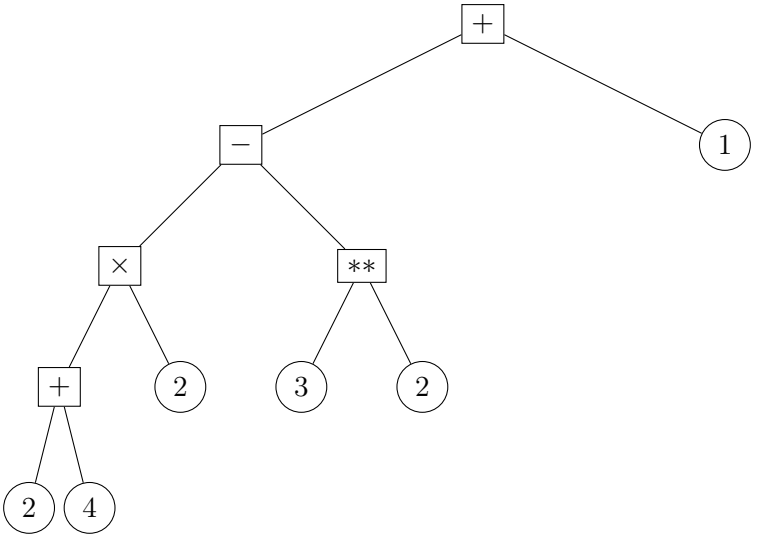
Nous verrons en TP comment implémenter les arbres en Python.

Comme pour de nombreuses structures de données, il y a plusieurs manières de procéder : on peut par exemple utiliser de simples **tableaux**, mais on peut aussi définir notre propre type arbre en le construisant avec une **classe** en programmation orientée objet.

6 Un exemple d'arbre binaire : l'arbre de calcul.

On considère l'expression suivante : $A = (2 + 4) \times 2 - 3^2 + 1$

On peut représenter cette expression par un arbre binaire dans lequel les nœuds sont les opérations et les feuilles, les nombres. Cela présente beaucoup d'avantages pour calculer l'expression car une fois l'expression écrite sous forme d'arbre, l'algorithme permettant de l'évaluer est aisé. Nous aurons l'occasion d'y revenir un peu plus tard (partie algorithmique).



Pour obtenir cet arbre, l'idée est de parcourir l'expression à calculer à la recherche de l'opération de moins forte priorité. Celle-ci prend alors place dans un nœud. On sépare alors l'expression en deux sous expressions : gauche et droite, permettant chacune de construire un sous arbre gauche et droit en répétant la même méthode de construction. Lorsqu'il n'y a plus d'opération, nous sommes au niveau d'une feuille et la construction du sous-arbre s'arrête.

1. Quelle est la hauteur de l'arbre ci-dessus ?
2. Quelle est la feuille de moindre profondeur ?
3. Quelles sont les feuilles qui sont à la même hauteur que 3 ?
4. Écrire un arbre binaire correspondant à l'expression $B = 3 - 2\sqrt{x + 12} - x$
5. Écrire tous les arbres binaires à 3 nœuds.
6. Écrire les 14 arbres binaires à 4 nœuds.