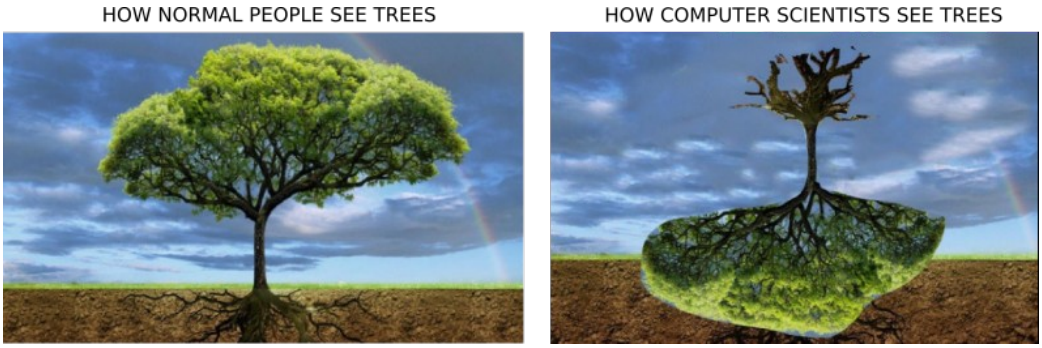
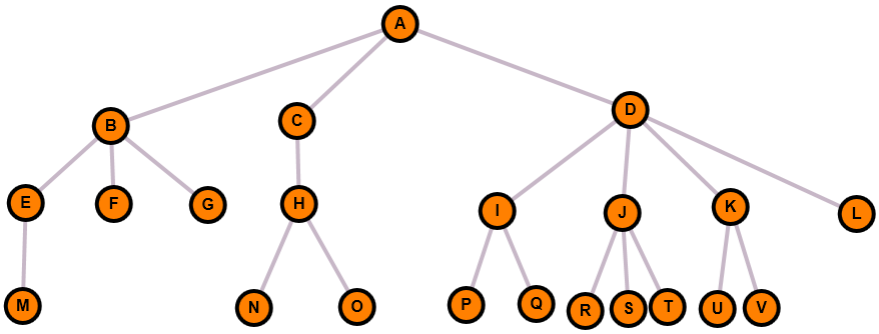
Exercices
Les arbres



Exercice 1 : Répondre aux questions suivantes à l'aide du graphe fourni.



1. Quelle est la racine de cet arbre ?
2. Quelles sont les feuilles de cet arbre ?
3. Quel est le père de :

a) L ? b) U ? c) K ? d) E ?

4. Quels sont les enfants de :

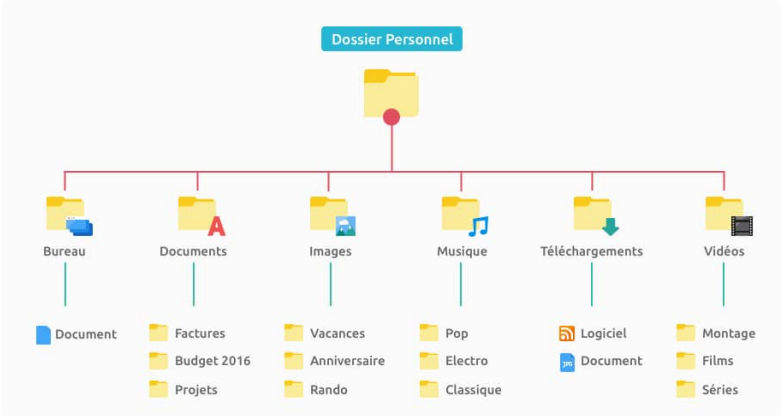
a) D ? b) A ? c) R ? d) T ? e) H ?

5. Dessiner le sous-arbre de racine C.
6. Quelle est la taille de l'arbre ci-contre ?
7. Quelle est la hauteur de cet arbre ?
8. Quelle est la profondeur du nœud F ?

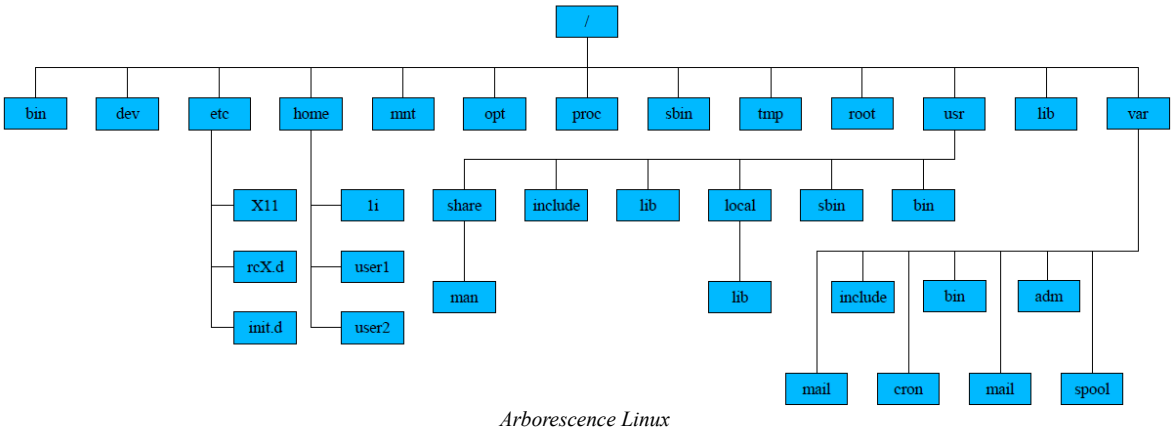
Exercice 2 : *Dans cet exercice, on travaillera avec des* ***arbres binaires***.

1. Dessiner tous les arbres binaires à 2 nœuds. Attention, on distinguera bien les fils droits des fils gauches.
2. Dessiner tous les arbres binaires à 3 nœuds.
3. Combien existe-t-il d'arbres binaires à 4 nœuds ? On pourra raisonner en s'appuyant sur le nombre de sous-arbres binaires possibles.
4. Même question avec le nombre d'arbres binaires à 5 nœuds.

Exercice 3 : Voici deux arborescences de fichiers. Pour chacune d'elles, donner la taille et la hauteur de l'arbre correspondant.



Arborescence Windows



Arborescence Linux

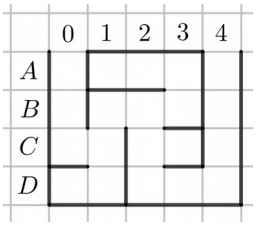
Exercice 4 : On appelle DOM (Document Object Model) l'arbre représentant l'organisation d'un document XML ou HTML. Chaque nœud peut être un élément (une balise), un attribut (par exemple `href` pour une balise `<a>`) ou du texte (contenu dans la balise). Dessinez le DOM de cette page HTML.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8"/>
      <title>Ma super page</title>
   </head>
   <body>
      <header>
         <h1>Plantez des arbres !</h1>
         <img src="arbre.png" alt="un arbre binaire" />
      </header>
      <section>
         <h2>C'est bon pour la planète</h2>
         <p>Parce que ça capte le carbone.</p>
         <h2>C'est bon pour la complexité</h2>
         <p>Parce que le logarithme c'est mieux !</p>
      </section>
   </body>
</html>
```

Exercice 5 : On souhaite réaliser un correcteur orthographique. Pour cela, on a besoin d'un dictionnaire contenant les différents mots de la langue française. Pour l'instant, on dispose d'un dictionnaire contenant quelques mots : **arbre**, **arbitre**, **arbitrer**, **binaire**, **binette**, **bio**, **empiler**, **exact**.

1. On souhaite stocker ces mots par ordre alphabétique dans une `list` Python de `str`. Chaque nom/adjectif qualificatif sera présent au singulier et au pluriel, les adjectifs qualificatif seront également présents au féminin et les verbes seront également conjugués au présent de l'indicatif. Quelle sera la taille de cette liste ? En déduire le nombre de tests nécessaires (mot à mot et pas lettre à lettre) pour trouver le mot « *exactes* » en effectuant une recherche linéaire.
2. Représenter ce dictionnaire rudimentaire sous la forme d'un arbre dans lequel chaque nœud est une lettre. La racine et la fin des mots seront notées avec le symbole *. Combien de tests sont nécessaires pour trouver le mot « *exactes* » ?

Exercice 6 : Les chemins d'un labyrinthe peuvent être représentés par un graphe orienté acyclique ou DAG (*Directed acyclic graph*). Dans le cas de labyrinthes parfaits (sans boucles, ilôts ou cellules inaccessibles) ce graphe est un arbre. Dessinez le DAG du labyrinthe ci-contre (entrée en A0 et sortie en A4).



Exercice 7 :

1. Dessiner (en notant sa taille) un arbre binaire de hauteur 3 :
   a) ayant la taille minimale ; b) ayant la taille maximale (on parle d'arbre binaire parfait) ; c)quelconque.
2. Mêmes questions avec un arbre binaire de hauteur 4.
3. En notant n la taille d'un arbre binaire et h sa hauteur, donner un encadrement de n en fonction de h.
4. En déduire que pour un arbre binaire parfait, $h = log_2(n+1)$.
5. Quelle est la taille maximale d'un arbre binaire de hauteur 8 ?

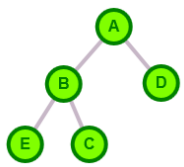
```
def mystere(x:int)->None:
   if x>0:
      mystere(x-1)
      print("*")
      mystere(x-1)
   else:
      print("-")
```

Exercice 8 : ***Retour sur la récursivité et l'arbre des appels*** : Combien d'étoiles et de tirets seront affichés par le programme ci-contre pour `x=10` ?

Exercice 9 : On souhaite rédiger un algorithme récursif qui affiche un arbre binaire de la manière suivante :

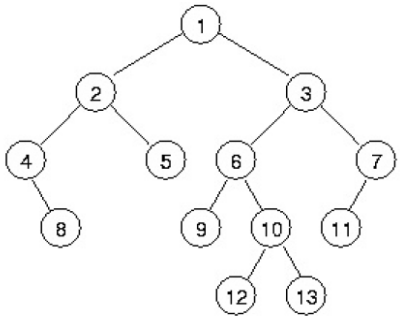
- pour un arbre vide, on n'affiche rien ;
- pour un nœud, on affiche une parenthèse ouvrante, son sous-arbre gauche, une flèche ← , sa valeur, une flèche →, son sous-arbre droit et enfin un parenthèse fermante.

1. Que donnerait l'affichage de l'arbre ci-conte ?
2. Quel arbre correspond à l'affichage : (←1→((←2→)←3→)) ?
3. Rédiger cet algorithme en pseudo-code.
4. Combien faut-il écrire d'appels récursifs au sein de l'algorithme ?
5. Quelle différence y a-t-il entre l'affichage de la parenthèse ouvrante, celui de la parenthèse fermante et celui des flèche et de la valeur ?



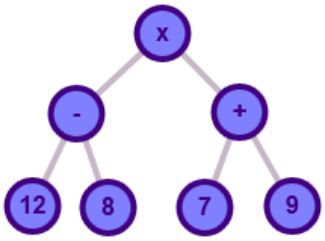
Remarque *: pour les instructions exécutées avant le 1er appel récursif, on parle de traitement* ***préfixe****. Pour celles exécutées après le 2ème appel, on parle de traitement* ***postfixe****. Pour les instructions exécutées entre le 1er et le 2nd appel, on parle de traitement* ***infixe****.*

Exercice 10 : Écrire les nombres contenus dans cet arbre binaire en le parcourant :

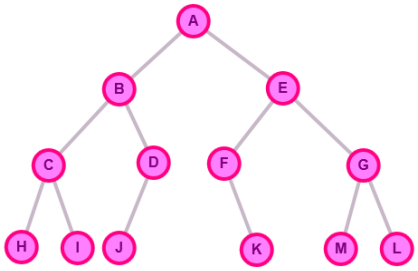


1. en **largeur** (la racine, puis tous les nœuds de profondeur 1, puis ceux de profondeur 2…) ;
2. en **profondeur** (on descend dans l'arbre, puis on remonte…) avec un traitement :
   a) **préfixe** (on affiche avant de traiter le sous-arbre gauche) ;
   b) **postfixe** (on affiche après avoir traité le sous-arbre droit) ;
   c) **infixe** (on affiche après avoir traité le sous-arbre gauche et avant de traiter le sous-arbre droit) ;

Exercice 11 : ***Arbre arithmétique***. L'arbre ci-contre permet de modéliser un calcul.



1. Écrire ce calcul « normalement » et effectuez-le.
2. Écrire le contenu de cet arbre en le parcourant :
   a) en largeur,
   b) en profondeur avec traitement préfixe,
   c) en profondeur avec traitement postfixe,
   d) en profondeur avec traitement infixe.
3. Quel parcours permet d'afficher le calcul presque normalement (parenthèses exceptées).
4. A quelle notation (déjà vue cette année) correspond le parcours postfixe ? En déduire le nom de la notation correspondant au parcours préfixe.

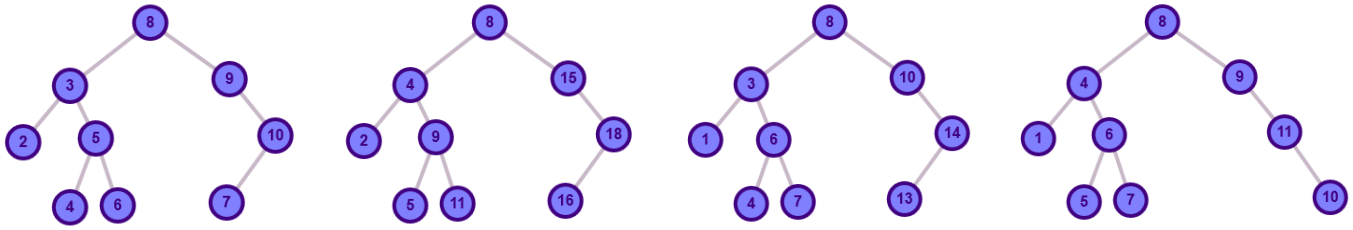


Exercice 12 : En généalogie, on synthétise parfois les arbres généalogiques avec la méthode de ***numérotation de Sosa-Stradonitz***, aussi appelée ***numérotation Ahnentafel***, ***Eyzinger*** ou ***Kekulé***. Pour nos arbres binaires, cela consiste à stocker les valeurs contenues dans chaque nœud dans un tableau indexé à partir de 0 en parcourant l'arbre en largeur.

1. Dessiner un tel tableau en notant ? les valeurs absentes et en indexant le tableau.
2. Quelle lettre est située à l'indice $i = 3$?
3. Quel est l'indice de son fils gauche ? De son fils droit ?
4. Vérifier que l'indice du fils gauche est toujours égal à $2 \times i+1$ et celui du fils droit à $2 \times i+2$.
5. Reprendre le tableau en stockant la racine à l'indice 1, de manière à utiliser la case d'indice 0 pour enregistrer la taille de l'arbre. Par quelles opérations obtient-on les fils gauche et droit ?

Remarque *: cette méthode sera utilisée lorsque nous aborderons le tri par tas.*

Exercice 13 : Parmi les arbres ci-dessous, lequel est un ABR, arbre binaire de recherche. Expliquez votre réponse.



Exercice 14 : En vous basant sur l'ABR de l'exercice 11 :

1. ajouter le nombre 2 ;
2. ajouter le nombre 5 ;
3. ajouter le nombre 12 ;
4. décrire le cheminement (tests et décisions) pour accéder au nombre 4 ;
5. décrire le cheminement pour rechercher le nombre 11.
6. En vous basant sur les deux questions précédentes, déterminer si le coût de la recherche d'un élément dans un ABR est proportionnel à sa taille n ou à sa hauteur h.
7. En déduire la complexité d'un algorithme de recherche en fonction de h puis en fonction de n.

Exercice 15 : En partant d'un ABR vide, on ajoute les nombres suivants dans cet ordre : 20, 8, 30, 40, 25, 5, 15, 18, 13, 4, 28. Dessiner l'ABR obtenu.

Exercice 16 : Quel type de parcours d'un ABR permet d'afficher son contenu en ordre croissant ?