

Exercices

Les arbres

HOW NORMAL PEOPLE SEE TREES

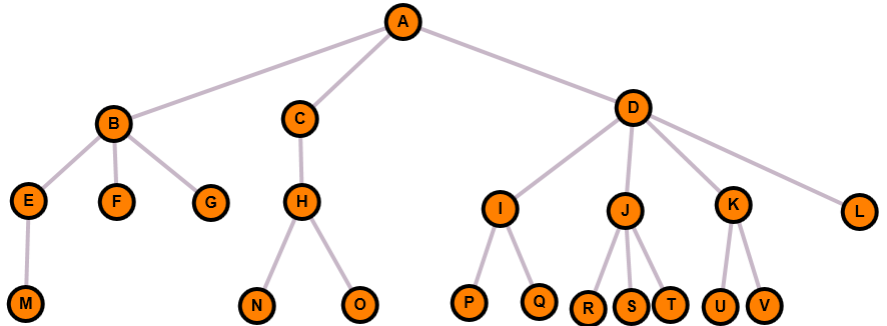


HOW COMPUTER SCIENTISTS SEE TREES



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

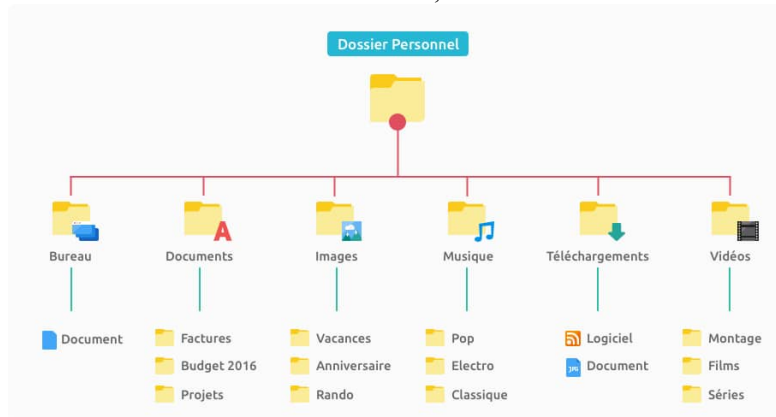
1. Quelle est la racine de cet arbre ?
2. Quelles sont les feuilles de cet arbre ?
3. Quel est le père de :
 - a) L ?
 - b) U ?
 - c) K ?
 - d) E ?
4. Quels sont les enfants de :
 - a) D ?
 - b) A ?
 - c) R ?
 - d) T ?
 - e) H ?
5. Dessiner le sous-arbre de racine C.
6. Quelle est la taille de l'arbre ci-contre ?
7. Quelle est la hauteur de cet arbre ?
8. Quelle est la profondeur du nœud F ?



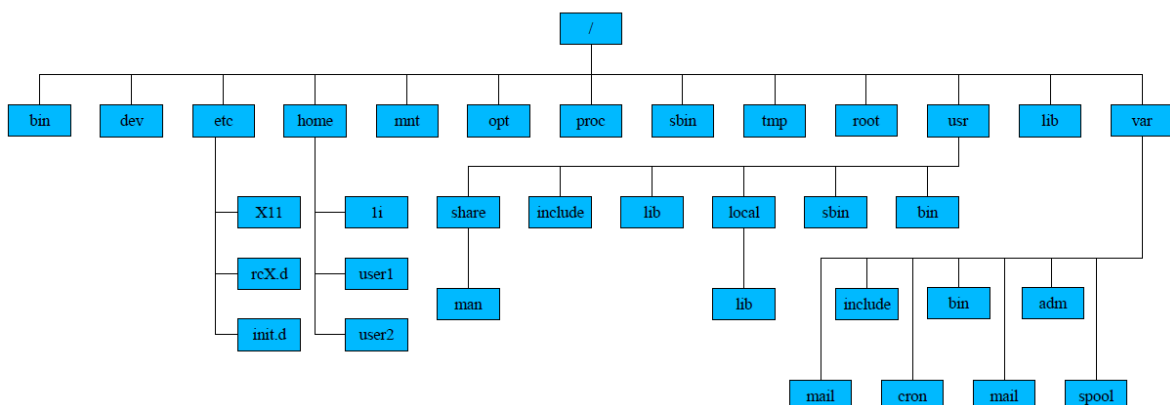
Exercice 2 : Dans cet exercice, on travaillera avec des **arbres binaires**.

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

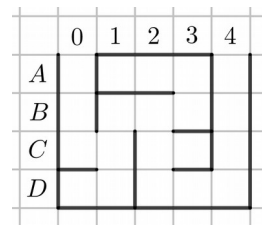
Exercice 4 : On appelle DOM (Document Object Model) l'arbre représentant l'organisation d'un document XML ou HTML. Chaque nœud peut être un élément (une balise), un attribut (par exemple href pour une balise <a>) ou du texte (contenu dans la balise). Dessinez le DOM de cette page HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Ma super page</title>
  </head>
  <body>
    <header>
      <h1>Plantez des arbres !</h1>
      
    </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**.

- 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.
- 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, îlots ou cellules inaccessibles) ce graphe est un arbre. Dressez le DAG du labyrinthe ci-contre (entrée en A0 et sortie en A4).



Exercice 7 :

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

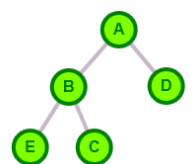
```
def mystere(x:int)->None:
    if x>0:
        mystere(x-1)
        print("***")
        mystere(x-1)
    else:
        print("-")
```

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 $\leftarrow$, sa valeur, une flèche $\rightarrow$, son sous-arbre droit et enfin une parenthèse fermante.

- Que donnerait l'affichage de l'arbre ci-contre ?
- Quel arbre correspond à l'affichage : $(\leftarrow 1 \rightarrow ((\leftarrow 2 \rightarrow) \leftarrow 3 \rightarrow))$?
- Rédiger cet algorithme en pseudo-code.
- Combien faut-il écrire d'appels récursifs au sein de l'algorithme ?
- 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 1^{er} appel récursif, on parle de traitement **préfixe**. Pour celles exécutées après le 2^{ème} appel, on parle de traitement **postfixe**. Pour les instructions exécutées entre le 1^{er} et le 2nd appel, on parle de traitement **infixe**.