Séquence 7 – Les arbres

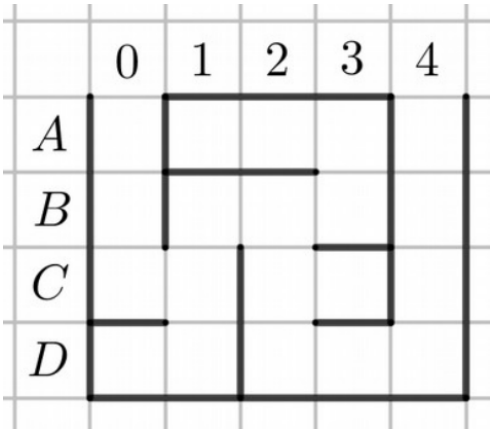
Objectifs

1. Identifier des situations nécessitant une structure de données hiérarchique, arborescente
2. Définir un arbre en tant que structure hiérarchique (nœuds, racines, feuilles, sous-arbres)
3. Évaluer la taille, la hauteur, ... d'un arbre.

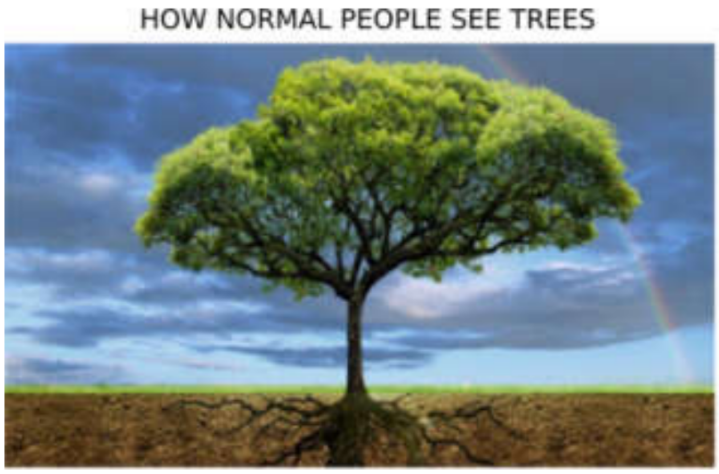
1 Exemple introductif

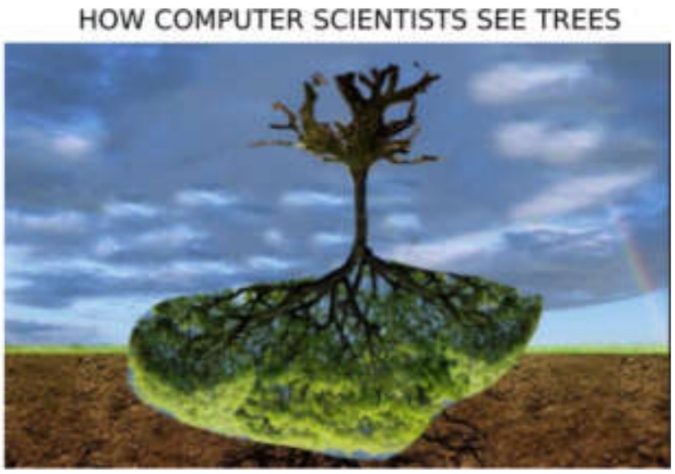
A faire vous même 1.

L' entrée de ce labyrinthe est en A0 et la sortie en A4. Décrivez tous les chemins possibles permettant de le parcourir.

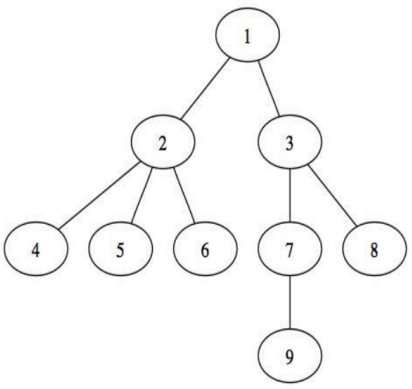


2 Les arbres

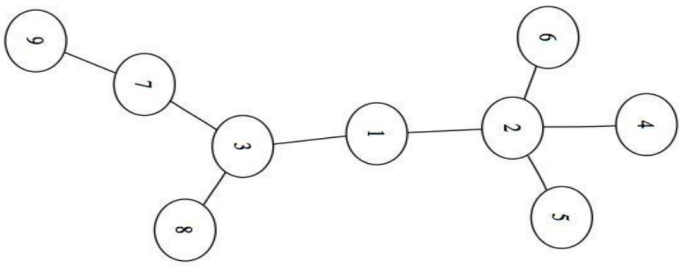




2.1 Arbre enraciné vs. Arbre non enraciné



Arbre enraciné (fig. 1)



Arbre non enraciné (fig. 2)

Un arbre enraciné (fig.1) est un arbre hiérarchique dans lequel on peut établir des niveaux (exemple : un arbre généalogique).
Dans un arbre non enraciné (fig.2), il n'y a pas de relation d'ordre ou de hiérarchie entre les éléments qui composent l'arbre.

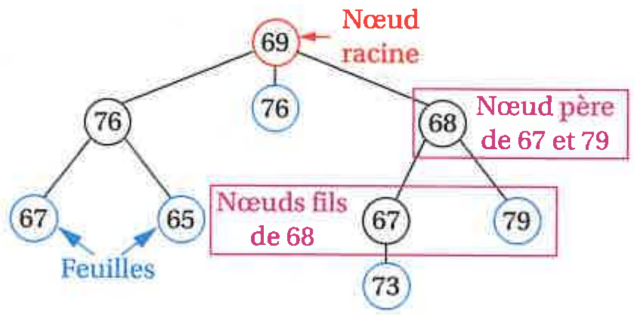
Toutefois, on peut passer d'un arbre non enraciné (fig.2) à un arbre enraciné (fig.1). Il suffit de choisir un élément comme sommet de l'arbre et de l'organiser de façon à obtenir un arbre en-raciné (ce qui est le cas en passant de la fig.2 à la fig.1).

A faire vous même 2.

Lire cours P. 122-123

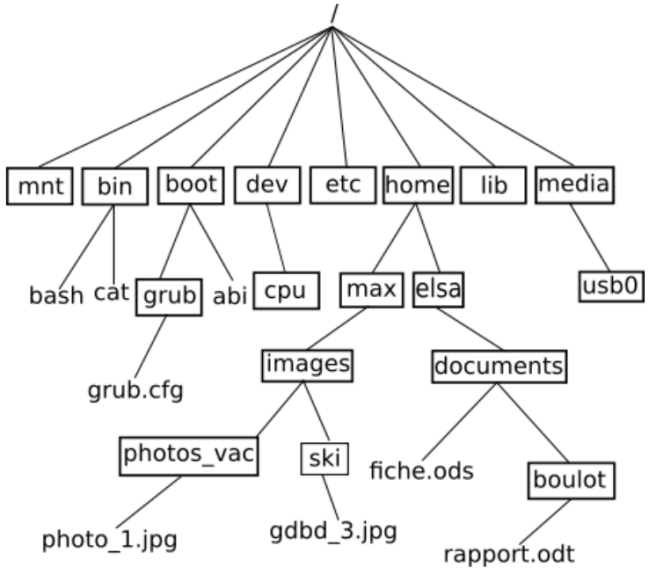
2.2 Définition

2.3 Caractéristiques/vocabulaire

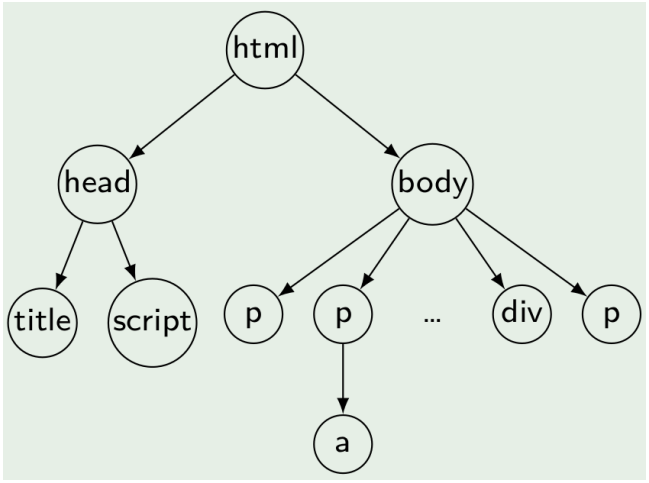


2.4 Exemples

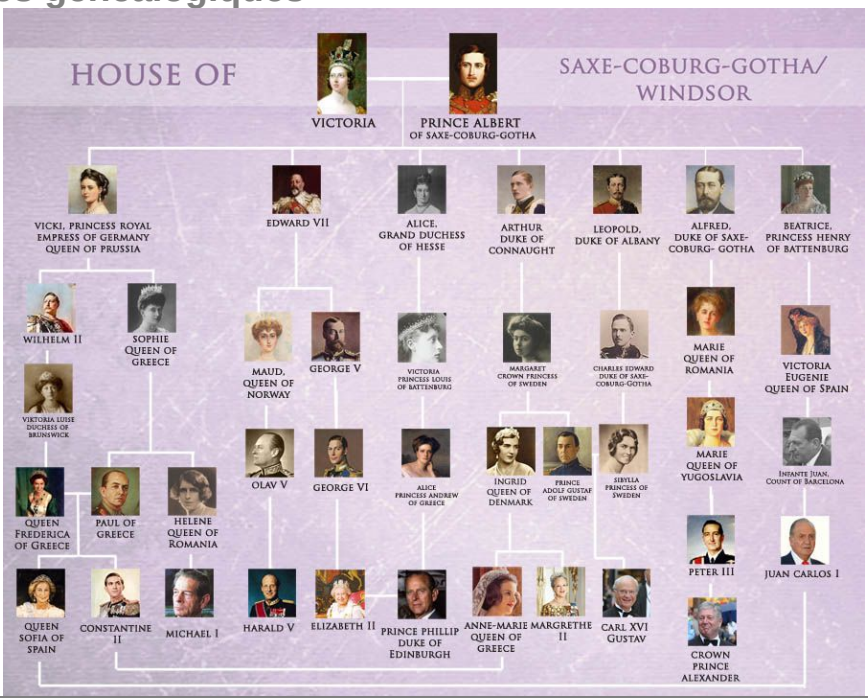
2.4.1 Structure de système d’ exploitation



2.4.2 Page web (DOM)



2.4.3 Arbres généalogiques



A faire vous même 3.

Trouvez deux caractéristiques à enlever afin que l’ arbre généalogique ci-dessus soit bien un arbre enraciné.

2.4.4 Arbres syntaxiques

Lire P. 127 - Arbres syntaxiques et reproduire l' arbre syntaxique du code suivant :

```
k = 3
for i in range(4) :
    k = k*2
print(k)
```

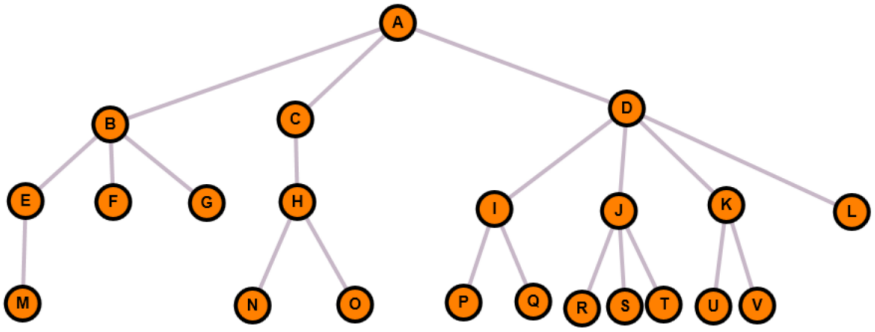
2.5 Exercices

2.5.1 Exercice

P. 130 ex 1

2.5.2 Exercice

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 ?

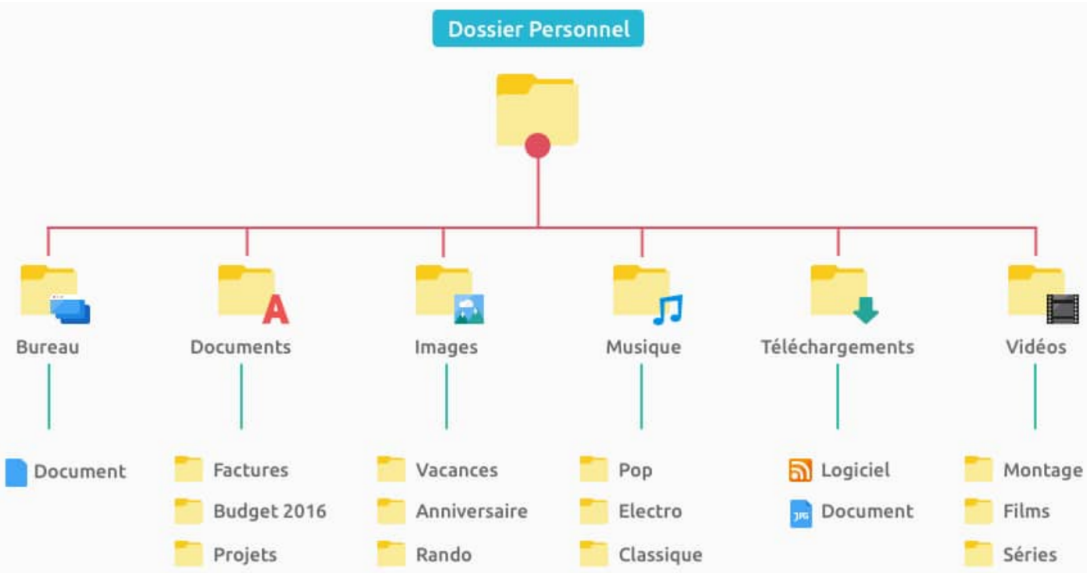
4. Quels sont les enfants de :

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 ?

2.5.3 Exercice – Arborescence de fichiers

Voici une arborescence de fichiers. Donnez la taille et la hauteur de l'arbre correspondant.



2.5.4 Exercice - HTML et DOM

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>
```

2.5.5 Exercice dictionnaire

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 » ?

3 Modélisation grâce au langage python

3.1 Graphviz et pygraphviz

A faire vous même 4.

Pour créer nos arbres à l' aide de python, nous allons installer :

- Graphviz : https://graphviz.org/
- Pygraphviz : https://pygraphviz.github.io/

Voir le tutoriel pygraphviz : https://pygraphviz.github.io/documentation/stable/tutorial.html

- Graphs

  To make an empty pygraphviz graph use the AGraph class:

  ```
  >>> import pygraphviz as pgv
  >>> G = pgv.AGraph()
  ```

  You can use the strict and directed keywords to control what type of graph you want. The default is to create a strict graph (no parallel edges or self-loops). To create a digraph with possible parallel edges and self-loops use

  ```
  >>> G = pgv.AGraph(strict=False, directed=True)
  ```

  You may specify a dot format file to be read on initialization:

  ```
  >>> G = pgv.AGraph("Petersen.dot")
  ```

  Other options for initializing a graph are using a string,

  ```
  >>> G = pgv.AGraph("graph {1 - 2}")
  ```

  using a dict of dicts,

  ```
  >>> d = {"1": {"2": None}, "2": {"1": None, "3": None}, "3": {"2": None}}
  >>> A = pgv.AGraph(d)
  ```

  or using a SWIG pointer to the AGraph datastructure,

  ```
  >>> h = A.handle
  >>> C = pgv.AGraph(h)
  ```

- Nodes, and edges¶

  Nodes and edges can be added one at a time

  ```
  >>> G.add_node("a")  # adds node 'a'
  >>> G.add_edge("b", "c")  # adds edge 'b'-'c' (and also nodes 'b', 'c')
  >>> nodelist = ["f", "g", "h"]
  ```

  or from lists or containers.

  ```
  >>> nodelist = ["f", "g", "h"]
  >>> G.add_nodes_from(nodelist)
  ```

  If the node is not a string an attempt will be made to convert it to a string

  ```
  >>> G.add_node(1)  # adds node '1'
  ```

- Attributes¶

  To set the default attributes for graphs, nodes, and edges use the graph_attr, node_attr, and edge_attr dictionaries

  ```
  >>> G.graph_attr["label"] = "Name of graph"
  >>> G.node_attr["shape"] = "circle"
  >>> G.edge_attr["color"] = "red"
  ```

  Graph attributes can be set when initializing the graph

  ```
  >>> G = pgv.AGraph(ranksep="0.1")
  ```

  Attributes can be added when adding nodes or edges,

  ```
  >>> G.add_node(1, color="red")
  >>> G.add_edge("b", "c", color="blue")
  ```

  or through the node or edge attr dictionaries,

  ```
  >>> n = G.get_node(1)
  >>> n.attr["shape"] = "box"
  >>> e = G.get_edge("b", "c")
  >>> e.attr["color"] = "green"
  ```

- Layout and Drawing¶

  Pygraphviz provides several methods for layout and drawing of graphs.

  To store and print the graph in dot format as a Python string use

  ```
  >>> s = G.string()
  ```

  To write to a file use

  ```
  >>> G.write("file.dot")
  ```

  To add positions to the nodes with a Graphviz layout algorithm

  ```
  >>> G.layout()  # default to neato
  >>> G.layout(prog="dot")  # use dot
  ```

  To render the graph to an image

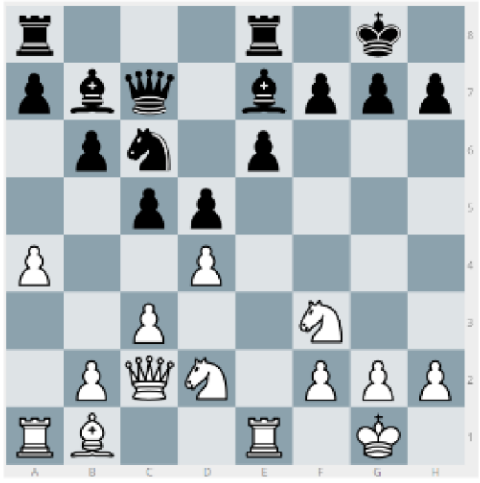
  ```
  >>> G.draw("file.png")  # write previously positioned graph to PNG file
  >>> G.draw("file.ps", prog="circo")  # use circo to position, write PS file
  ```

A faire vous même 5.

- Modélisez les 3 exemples d' arbres ci-dessus avec pygraphviz

A faire vous même 6.

- Lire P. 126 – Arbres de jeu
- Un arbre de jeu permet de représenter toutes les positions et tous les coups possibles d' un jeu à l' aide d' un arbre
- Sur une feuille ou avec pygraphviz, représentez l' arbre de jeu de la situation suivante (c' est aux blancs de jouer) :



3.2 Programmation objet

A faire vous même 7.

- Programmez une classe d' objet `Noeud` avec deux attributs :
  - `etiquette` : de type `str`
  - `enfants` : de type `list`, dans lequel on va enregistrer tous les nœuds enfants
- Et avec des méthodes
  - `ajoute_un_fils` : Ajoute un Noeud dans les fils
  - `__str__` : Méthode récursive permettant d' afficher tous les nœuds enfants, petits-enfants, …
- Avec cette base, faire P. 132 ex 7.

3.3 Représentation à l'aide de structures linéaires

Un arbre est une structure non-linéaire mais on peut modéliser à l' aide de structures linéaires :

- A l'aide d' un dictionnaire : clé ↔ dictionnaire ou clé ↔ liste ou clé ↔ tuple
  Exemples basés sur exercice 2.5.2 :

- A l’aide de listes de listes (ou à l’ aide de tuples de tuples)
  Exemples basés sur exercice 2.5.2

- A l’ aide du XML
  Exemples basés sur exercice 2.5.2