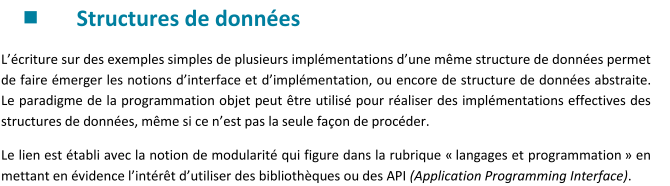
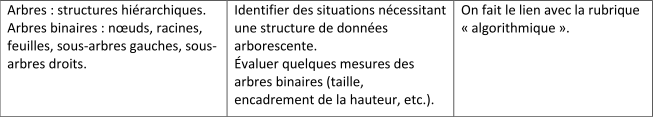
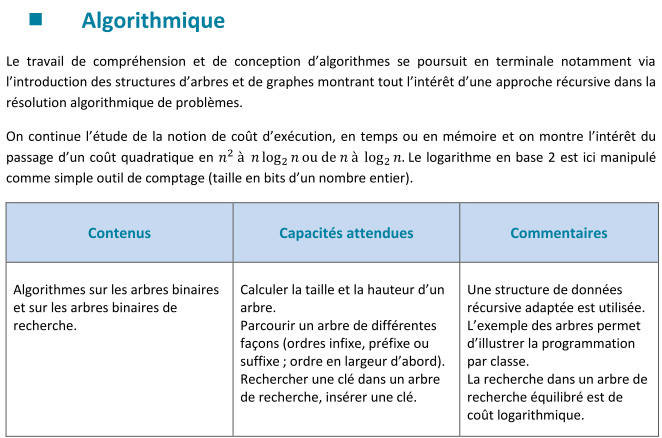
|  |  |
| --- | --- |
| Term NSI | Arbres structures hiérarchiques et binaires |
| COURS/AD : Arbres structures hiérarchiques et binaires [Extrait de pixees](https://pixees.fr/informatiquelycee/n_site/nsi_term.html), Ouvrage Ellipses, Nathan et DUI Toulouse | |





**1) A R B R E S : structure de données hiérarchique**

Un arbre permet, contrairement aux listes, piles, ﬁles, de représenter des relations non séquentielles.

Exemple 1 : répertoire d’un ordinateur Windows (arbre non binaire):

**C :**

**Programs**

**Users**

**Windows**

**Firefox**

**Python 3.7**

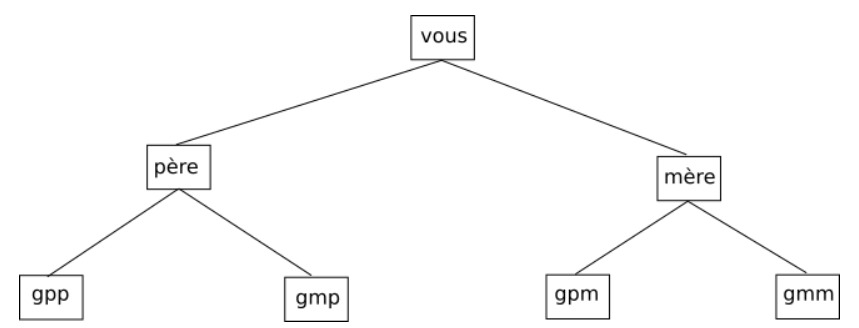
**System32**

**Explorer.exe**

Exemple 2 : L’expression arithmétique (arbre binaire)

|  |  |
| --- | --- |
| Peut-être représentée par : | y  z  x  2  3  y  x |

Exemple 3 : arbre généalogique (arbre binaire)



**2) Déﬁnitions, vocabulaire :** voir <https://fr.wikipedia.org/wiki/Arbre_binaire>

|  |  |
| --- | --- |
| **Déﬁnition 1 :**  – Un arbre est un ensemble de nœuds,  reliés par des arcs.  – Les nœuds contiennent les données,  aussi appelées les valeurs.  – La racine est le nœud le plus haut dans  la hiérarchie, elle est unique. |  |

– Les feuilles sont des nœuds terminaux. (nœud ayant aucun fils)

**Remarque :** Dans un arbre, chaque nœud n’a qu’un seul père ; sauf la racine qui n’en a pas.

|  |  |
| --- | --- |
| **Déﬁnition 2**  – Le chemin d’un nœud est la suite de nœuds qu’il faut traverser jusqu’à la racine.  – La hauteur (ou profondeur) d’un nœud est le nombre de nœuds de son chemin. La   hauteur de la racine est 1. |  |

– La hauteur de l’arbre est la hauteur du nœud le plus haut, c’est à dire, le nombre de   
  
nœuds du ***chemin le plus long*** entre une feuille et la racine.

|  |  |
| --- | --- |
| **Déﬁnition 3 :** Un arbre binaire est un arbre dans lequel un père a au plus deux ﬁls.   Dans un arbre binaire, chaque élément possède au plus deux éléments [fils](https://fr.wikipedia.org/wiki/Fils_(famille)) au niveau inférieur.  Du point de vue de ces éléments fils, l'élément dont ils sont issus au niveau supérieur est appelé *père*. |  |

|  |  |
| --- | --- |
| **Déﬁnition 4**  Soit **A un noeud** d’un arbre binaire, et **G, D** ses deux **ﬁls**.   Le **S**ous-**A**rbre **G**auche (resp.droit) de A est   le **S**ous-**A**rbre **D**roit dont G (resp.D) est la racine. | G  D  A |
|  | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  | **Noeud** |  |  | |  |  | *Valeur* | **A** |  |  | |  |  | *Gauche* | **SAG** |  |  | |  |  | *Droite* | **SAD** |  |  | |  |  |  |  |  |  | |  | **Noeud** |  |  |  | **Noeud** | | Valeur | **G** |  |  | Valeur | **D** | | Gauche | **SAG** |  |  | Gauche | **SAG** | | Droite | **SAD** |  |  | Droite | **SAD** | |

**Remarque :** Comme tout nœud d’un arbre binaire n’a pas nécessairement deux ﬁls, le sous-arbre gauche (resp. droit) peut être un arbre vide.

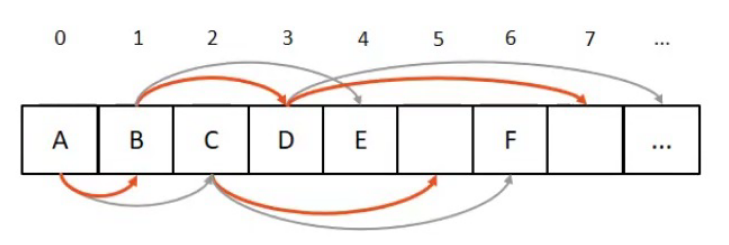
|  |  |  |
| --- | --- | --- |
| Spéciﬁcation du **T**ype **A**bstrait de **D**onnées Arbre Binaire | | |
| **Opérations**  – creerArbreVide : → ArbreBinaire[E]  – estVide : ArbreBinaire[E] → Booléen  – racine : ArbreBinaire[E] → E  – sag, sad : ArbreBinaire[E] → ArbreBinaire[E]  – assembler : E × ArbreBinaire[E] × ArbreBinaire[E] → ArbreBinaire[E] | **Préconditions**  Soit a un ArbreBinaire.  racine(a), sag(a) et sad(a) si et seulement si non estVide(a) | **Axiomes**  Soient a, g,d des ArbreBinaire[E], r un E.  – estVide(arbreVide)= Vrai  – estVide(assembler(r, g,d)) = Faux  – racine(assembler(r, g,d)) = r – sag(assembler(r, g,d)) = g  – sad(assembler(r, g,d)) = d |

###### Faire : Ex1, ex2, ex3 AD1 Arbres définitions et représentations

**3) Représentations d’un arbre par un tableau**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| La représentation d’un arbre binaire par un tableau repose sur la relation suivante entre les indices d’un père et de ses ﬁls : | | | | Indice 2  Indice 1  Indice 0 | | | |
|  |  |  |  |  |  |  | **…..** |
| **Niv0** | **Niv1** | | **Niv2** | | | | **Niv…** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** |  | **F** |  | **…..** |
| **Niv0** | **Niv1** | | **Niv2** | | | |  | **Niv…** |



Nous voyons assez facilement l’inconvénient d’une telle représentation : si l’arbre a beaucoup de "trous", nous nous retrouvons avec un tableau comportant beaucoup de   
  
cases vides. Nous lui privilégierons la seconde représentation.

###### Faire : Ex3 : AD1 Arbres définitions et représentations

**4) Représentation par une structure récursive : le nœud**

**Déﬁnition 6 :** Le nœud est une structure récursive déﬁnie par :

– Une valeur (la donnée du nœud)

– Les références vers le sous-arbre gauche et le sous-arbre droit (qui sont dont eux-mêmes représenté par un nœud, d’où la déﬁnition récursive)

|  |  |
| --- | --- |
|  | class Noeud :  # Constructeur  def \_\_init\_\_(self, valeur) :  self.donnée = valeur  self.gauche = None  self.droit = None  # Affichage  def \_\_str\_\_(self) :  return str(self.donnée) |

N\_A=nœud(A) soit :

Nœud = N\_A, self.donnée=A, self.gauche = None, self.droit = None

print(N\_A) affiche A

|  |  |
| --- | --- |
| **Exemple avec l’opération assembler :** Élément × ArbreBinaire × ArbreBinaire → ArbreBinaire  Avec cette représentation, l’opération assembler est particulièrement facile à comprendre, et à écrire. | class Noeud :  [...]  # Assembler  def assembler(self,ag,ad) :  self.gauche = ag  self.droit = ad |
| Self ag ad | |

###### Faire : Ex4 : AD1 Arbres définitions et représentations