Structures de données

Listes chaînées - Piles et Files - Arbres - Graphes

M. Tellene

Structures de données

Une liste chaînée est une structure de données parfaite pour structure un ensemble d'éléments destiné à être destiné à être énuméré séquentiellement

Structures de données

Une liste chaînée est une structure de données parfaite pour structure un ensemble d'éléments destiné à être destiné à être énuméré séquentiellement

Les piles et les files sont des structures de données optimisant l'accès au premier et au dernier élément de la séquence

Structures de données

Une liste chaînée est une structure de données parfaite pour structure un ensemble d'éléments destiné à être destiné à être énuméré séquentiellement

Les piles et les files sont des structures de données optimisant l'accès au premier et au dernier élément de la séquence

Ces structures de données ne sont pas adaptées aux accès ponctuels à des positions arbitraires dans la séquence

De manière générale, le temps de recherche est proportionnel au nombre d'éléments stockés dans la séquence

Structures de données

Afin de fournir une structure de données optimisant cet aspect, nous allons nous intéresser aux **structures arborescentes**

Dans ce genre de structure, le nombre de sauts à effectuer pour aller, depuis le point de départ, jusqu'à une position souhaitée est potentiellement bien moindre

Structures de données

Afin de fournir une structure de données optimisant cet aspect, nous allons nous intéresser aux **structures arborescentes**

Dans ce genre de structure, le nombre de sauts à effectuer pour aller, depuis le point de départ, jusqu'à une position souhaitée est potentiellement bien moindre

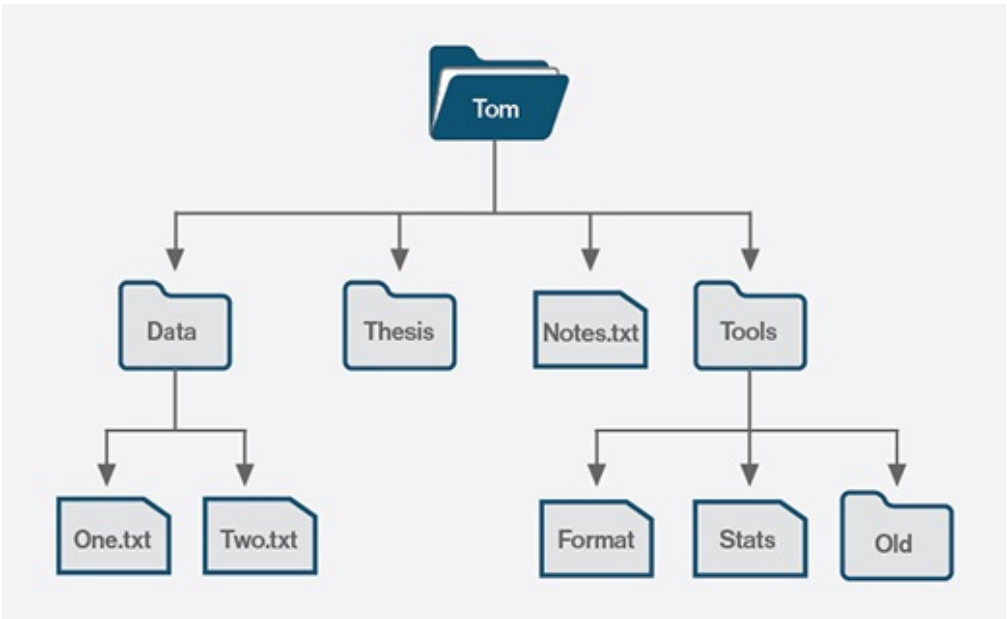
La notion de structure arborescente a déjà été vu, mais quand ?

Structures de données

Afin de fournir une structure de données optimisant cet aspect, nous allons nous intéresser aux **structures arborescentes**

Dans ce genre de structure, le nombre de sauts à effectuer pour aller, depuis le point de départ, jusqu'à une position souhaitée est potentiellement bien moindre

La notion de structure arborescente a déjà été vu, mais quand ? Les système de fichiers (système d'exploitation)



Structures de données

Le principe d'un point de départ **unique** à partir duquel une structure chaînée se scinde à chaque étape en plusieurs branches donne l'idée générale de la structure **d'arbre**

Structures de données

Le principe d'un point de départ **unique** à partir duquel une structure chaînée se scinde à chaque étape en plusieurs branches donne l'idée générale de la structure **d'arbre**

Un arbre permet la structure hiérarchique de l'information, ce qui rend utile la représentation de programmes, formules logiques, contenu de pages web...

Structures de données - Arbre

Il existe plusieurs types d'arbres :

- arbre binaire
- arbre binaire de recherche
- arbre AVL
- ...

Nous ne nous intéresserons qu'au arbre binaire et arbre binaire de recherche

Structures de données - Arbre binaire

Un arbre binaire est un cas particulier de structures arborescentes, où chaque position ouvre **sur exactement** deux branches

Structures de données - Arbre binaire

Un arbre binaire est un cas particulier de structures arborescentes, où chaque position ouvre **sur exactement** deux branches

Un arbre binaire est un ensemble fini de noeuds correspondant à l'un des cas suivants :

- arbre vide : l'arbre ne contient aucun noeuds

Structures de données - Arbre binaire

Un arbre binaire est un cas particulier de structures arborescentes, où chaque position ouvre **sur exactement** deux branches

Un arbre binaire est un ensemble fini de noeuds correspondant à l'un des cas suivants :

- arbre vide : l'arbre ne contient aucun noeuds
- arbre non vide : ses noeuds sont structurés de la forme suivante

Structures de données - Arbre binaire

Un arbre binaire est un cas particulier de structures arborescentes, où chaque position ouvre **sur exactement** deux branches

Un arbre binaire est un ensemble fini de noeuds correspondant à l'un des cas suivants :

- arbre vide : l'arbre ne contient aucun noeuds
- arbre non vide : ses noeuds sont structurés de la forme suivante
  - un noeud est appelé racine
  - les noeuds restants sont séparés en deux sous-ensemble, qui **forment récursivement** deux sous-arbres (sous-arbre gauche et sous-arbre droit)
  - la racine est reliée à ses deux sous-arbres gauche et droit
  - les noeuds n'ayant aucun sous-arbre gauche et droit sont appelés feuilles

Structures de données - Arbre binaire

On peut rapprocher la notion de noeud d'un arbre binaire à la notion de cellule d'une liste chaînée

La racine d'un arbre non vide est l'équivalent de la tête d'une liste non vide et les liens vers les deux sous arbres correspondant à deux chaînages suivants, menant à deux suites

Structures de données - Arbre binaire

On peut rapprocher la notion de noeud d'un arbre binaire à la notion de cellule d'une liste chaînée

La racine d'un arbre non vide est l'équivalent de la tête d'une liste non vide et les liens vers les deux sous arbres correspondant à deux chaînages suivants, menant à deux suites

De même que la **taille d'une liste** était définie par le nombre de cellules, la **taille d'un arbre binaire** est définie comme son nombre de noeuds

Structures de données - Arbre binaire

Les arbres binaires possèdent **une taille** mais aussi **une hauteur**, ces notions sont différentes

Structures de données - Arbre binaire

Les arbres binaires possèdent **une taille** mais aussi **une hauteur**, ces notions sont différentes

La hauteur d'un arbre binaire est définie comme le plus grand nombre de noeuds rencontrés en descendant de la racine jusqu'à une feuille

Tous les noeuds sont comptés, y compris la racine et la feuille

Structures de données - Arbre binaire

On peut définir la hauteur récursivement sur la structure de l'arbre de la manière suivante :

- l'arbre vide a pour hauteur 0
- un arbre non vide a pour hauteur la maximum des hauteurs de ses deux sous-arbres, auquel on ajoute 1

Structures de données - Arbre binaire

On peut définir la hauteur récursivement sur la structure de l'arbre de la manière suivante :

- l'arbre vide a pour hauteur 0
- un arbre non vide a pour hauteur la maximum des hauteurs de ses deux sous-arbres, auquel on ajoute 1

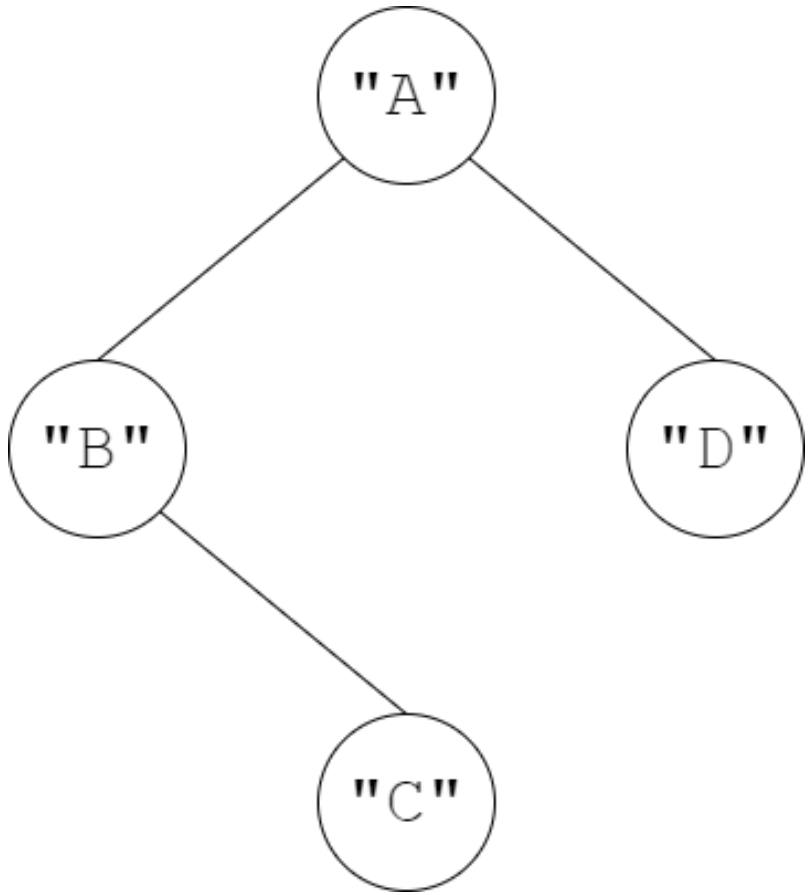
Propriété très très utile : Si N désigne la taille d'un arbre binaire et si h désigne sa hauteur, alors on a :

$$h \leq N \leq 2^h - 1$$

Structures de données - Arbre binaire

L'intérêt d'un arbre binaire est d'y stocker de l'information

Pour cela, on attache une information à chaque noeud



Structures de données - Arbre binaire

Il y a de nombreuses façons de représenter un arbre binaire

Une façon traditionnelle consiste à représenter chaque noeud par un objet d'une classe `Noeud`

Comment écrire cette classe ?

Structures de données - Arbre binaire

Il y a de nombreuses façons de représenter un arbre binaire

Une façon traditionnelle consiste à représenter chaque noeud par un objet d'une classe `Noeud`

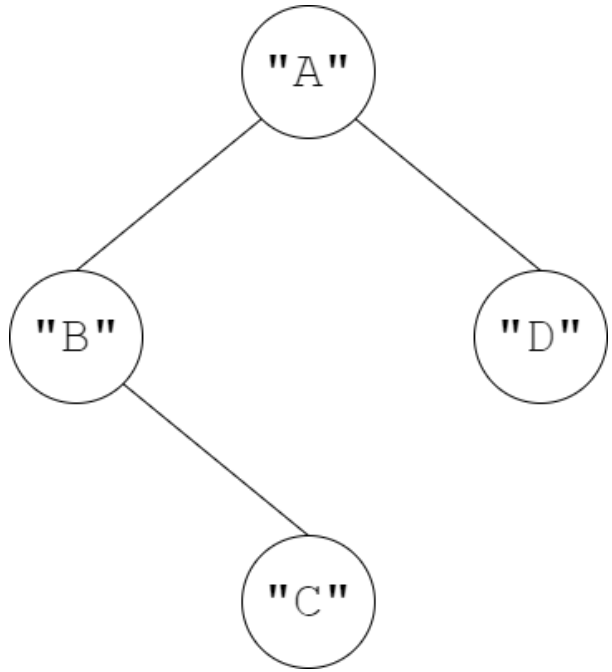
Comment écrire cette classe ?

```
1 class Noeud:
2     def __init__(self, v, g, d):
3         self.valeur = v
4         self.fg = g
5         self.fd = d
```

Dans les TPs, nous mettrons en lumière l'aspect récursif des arbres, nous n'écrirons pas de classe `Noeud`

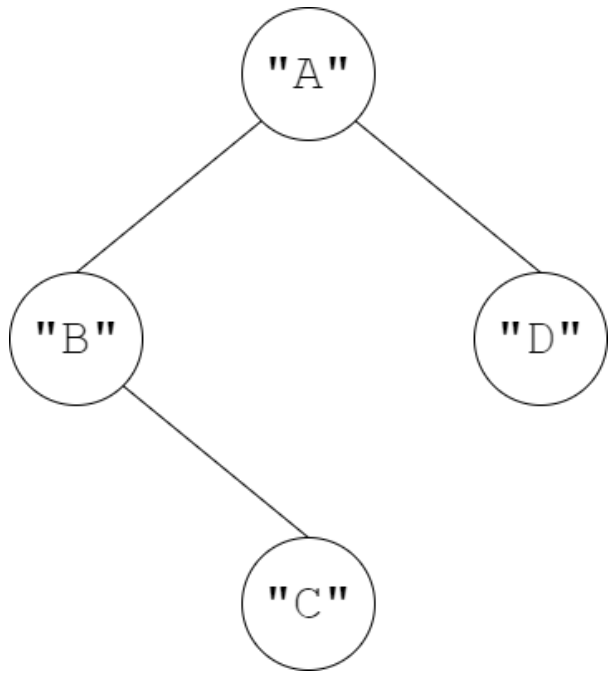
Structures de données - Arbre binaire

Avec la classe fourni, comment créer l'arbre suivant :



Structures de données - Arbre binaire

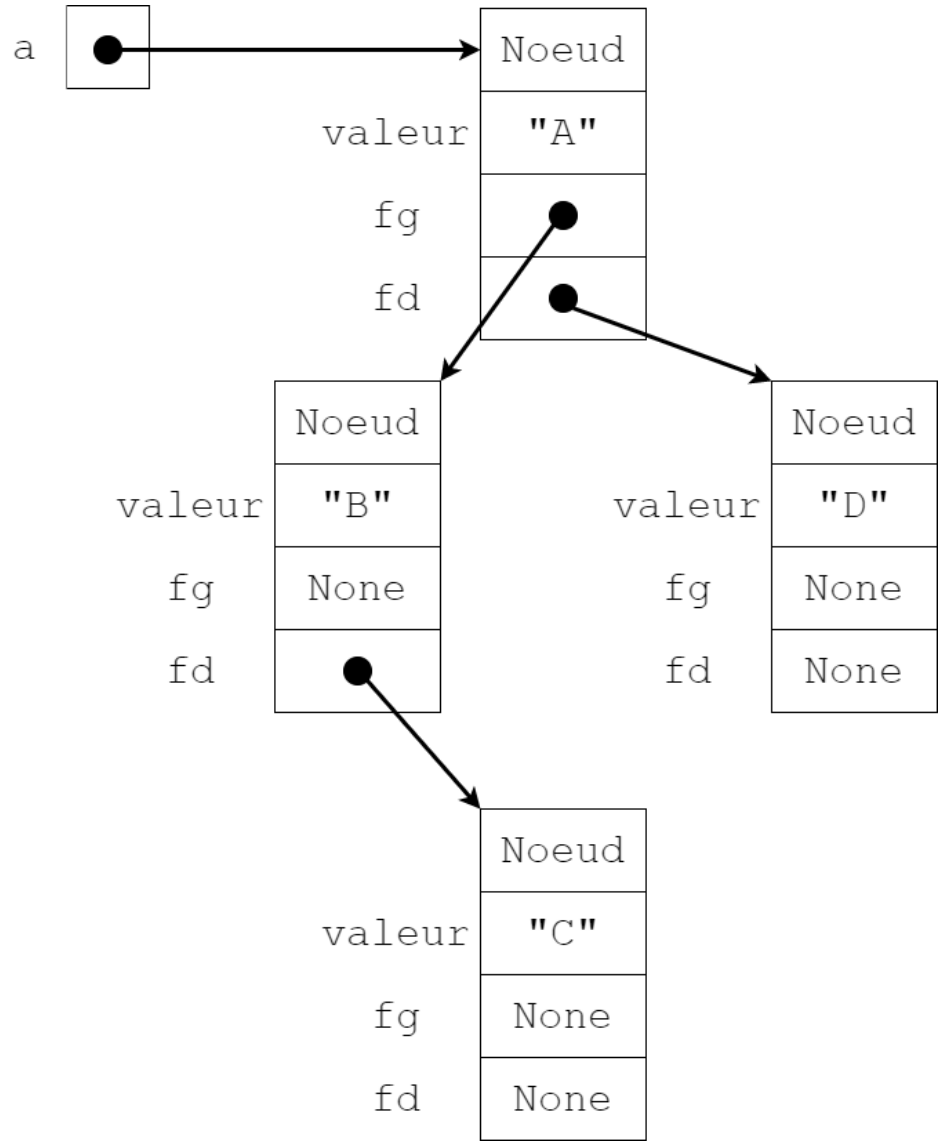
Avec la classe fourni, comment créer l'arbre suivant :




```
a = Noeud("A",
        Noeud("B", None, Noeud("C", None, None)),
        Noeud("D", None, None))
```

Structures de données - Arbre binaire

En machine, l'arbre précédent donne



Structures de données - Arbre binaire

Les parcours d'arbre sont des manières d'accéder aux valeurs d'un arbre

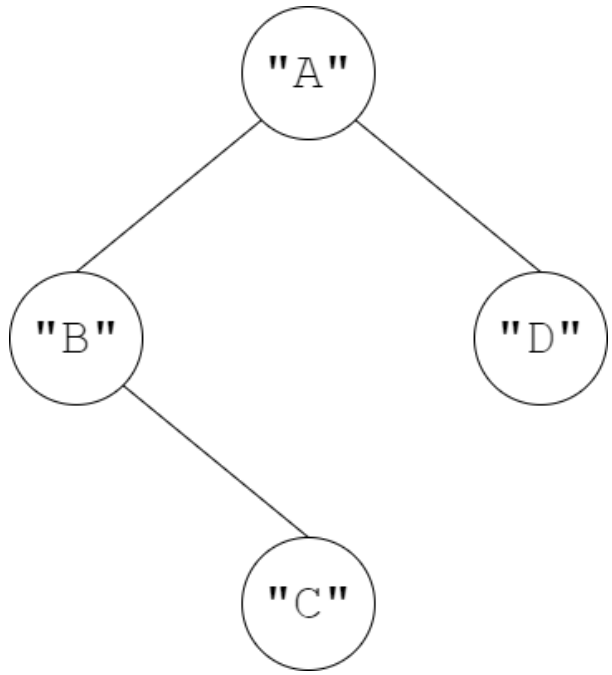
Nous verrons trois parcours différents :

- parcours infixe
- parcours préfixe
- parcours postfixe

Ces parcours sont notamment utilisés pour afficher les valeurs contenues dans un arbre

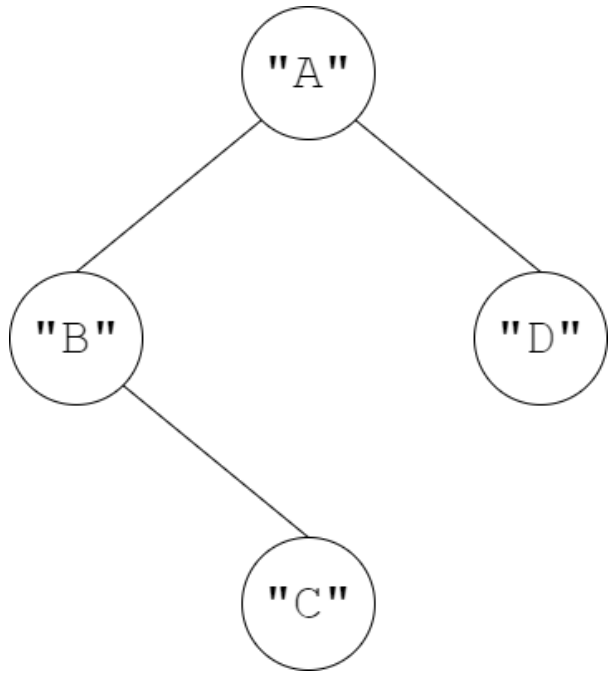
Structures de données - Arbre binaire

Soit l'arbre suivant :



Structures de données - Arbre binaire

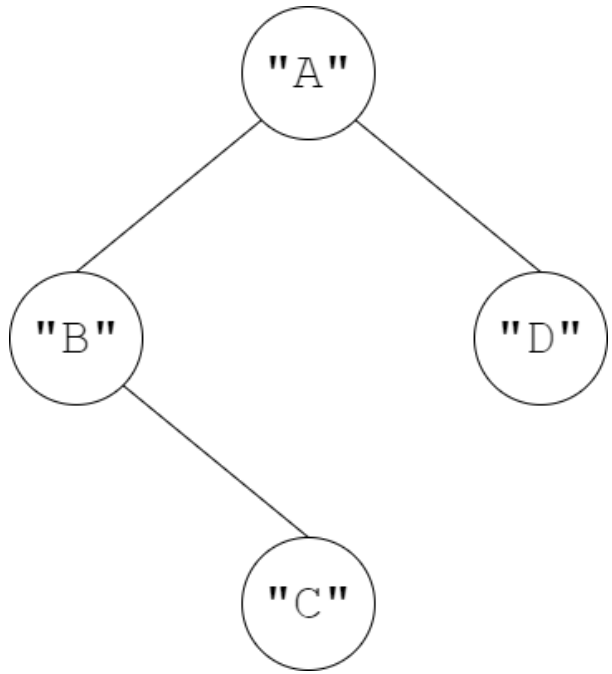
Soit l'arbre suivant :



- parcours infixe : `"B"` ; `"C"` ; `"A"` ; `"D"`

Structures de données - Arbre binaire

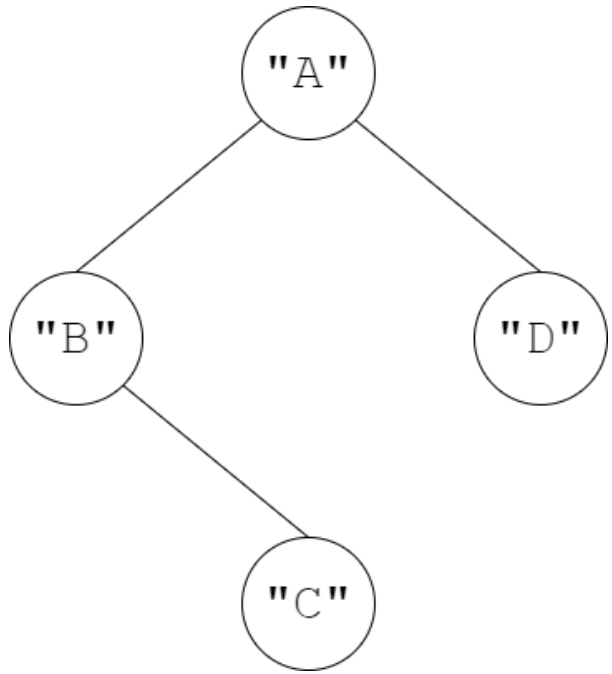
Soit l'arbre suivant :



- parcours infixe : `"B"` ; `"C"` ; `"A"` ; `"D"`
- parcours préfixe : `"A"` ; `"B"` ; `"C"` ; `"D"`

Structures de données - Arbre binaire

Soit l'arbre suivant :



- parcours infixe : `"B"` ; `"C"` ; `"A"` ; `"D"`
- parcours préfixe : `"A"` ; `"B"` ; `"C"` ; `"D"`
- parcours postfixe : `"C"` ; `"B"` ; `"D"` ; `"A"`

Structures de données - Arbre binaire

A retenir sur les arbres :

- un arbre binaire est un ensemble fini de noeuds
- un arbre binaire est soit vide, soit structuré à partir d'un noeud particulier, **racine**, et de deux **sous-arbre gauche** et **sous-arbre droite**

Structures de données - Arbre

Une « utilisation » particulière des arbres binaires sont les arbres binaires de recherche [1]

1. Que j'appellerai ABR dans toute la suite

Structures de données - ABR

Soit une bibliothèque contenant beaucoup de livres, répartis dans 17576 salles reliées les unes aux autres par des portes

Structures de données - ABR

Soit une bibliothèque contenant beaucoup de livres, répartis dans 17576 salles reliées les unes aux autres par des portes

Chaque salle contient une porte d'entrée et, possiblement, deux portes de sorties vers deux autres salles

Structures de données - ABR

Soit une bibliothèque contenant beaucoup de livres, répartis dans 17576 salles reliées les unes aux autres par des portes

Chaque salle contient une porte d'entrée et, possiblement, deux portes de sorties vers deux autres salles

Comment optimiser la recherche d'un livre précis ?

Structures de données - ABR

Soit une bibliothèque contenant beaucoup de livres, répartis dans 17576 salles reliées les unes aux autres par des portes

Chaque salle contient une porte d'entrée et, possiblement, deux portes de sorties vers deux autres salles

Comment optimiser la recherche d'un livre précis ?

En faisant une répartition « intelligente »

Structures de données - ABR

Dans la première salle, on place les ouvrages dont le titre commence par `MON`

Mais problème, le titre du livre recherché commence par `FIR`

Structures de données - ABR

Dans la première salle, on place les ouvrages dont le titre commence par `MON`

Mais problème, le titre du livre recherché commence par `FIR`

Ce n'est pas un soucis, il suffit d'aller dans une autre salle, mais où aller ? A gauche ou à droite ?

Structures de données - ABR

Dans la première salle, on place les ouvrages dont le titre commence par `MON`

Mais problème, le titre du livre recherché commence par `FIR`

Ce n'est pas un soucis, il suffit d'aller dans une autre salle, mais où aller ? A gauche ou à droite ?

C'est là qu'intervient la répartition « intelligente » :

- les livres dont le titre commence par trois lettres **avant** `MDB` sont placés derrière la porte de gauche

Structures de données - ABR

Dans la première salle, on place les ouvrages dont le titre commence par `MON`

Mais problème, le titre du livre recherché commence par `FIR`

Ce n'est pas un soucis, il suffit d'aller dans une autre salle, mais où aller ? A gauche ou à droite ?

C'est là qu'intervient la répartition « intelligente » :

- les livres dont le titre commence par trois lettres **avant** `MDB` sont placés derrière la porte de gauche
- les livres dont le titre commence par trois lettres **après** `MDB` sont placés derrière la porte de droite

Structures de données - ABR

Dans la première salle, on place les ouvrages dont le titre commence par `MON`

Mais problème, le titre du livre recherché commence par `FIR`

Ce n'est pas un soucis, il suffit d'aller dans une autre salle, mais où aller ? A gauche ou à droite ?

C'est là qu'intervient la répartition « intelligente » :

- les livres dont le titre commence par trois lettres **avant** `MDB` sont placés derrière la porte de gauche
- les livres dont le titre commence par trois lettres **après** `MDB` sont placés derrière la porte de droite

Dans notre cas nous allons donc à gauche

Structures de données - ABR

Dans la deuxième salle, on y trouve les livres dont le titre commence par `GAB`

Mais problème, le titre du livre recherché commence toujours par `FIR`

Structures de données - ABR

Dans la deuxième salle, on y trouve les livres dont le titre commence par `GAB`

Mais problème, le titre du livre recherché commence toujours par `FIR`

D'après vous, faudra-t-il aller à gauche ou à droite ?

Structures de données - ABR

Dans la deuxième salle, on y trouve les livres dont le titre commence par `GAB`

Mais problème, le titre du livre recherché commence toujours par `FIR`

D'après vous, faudra-t-il aller à gauche ou à droite ?

Étant donné que `F` est **avant** `G` dans l'alphabet, il faudra aller à gauche

Et on continue jusqu'à trouver notre livre

Structures de données - ABR

En quoi cette méthode est efficace ?

Structures de données - ABR

En quoi cette méthode est efficace ?

La bibliothèque est composée de 17576 salles, à votre avis combien de salles va-t-on traverser **au maximum** avant de trouver son livre ?

Structures de données - ABR

En quoi cette méthode est efficace ?

La bibliothèque est composée de 17576 salles, à votre avis combien de salles va-t-on traverser **au maximum** avant de trouver son livre ?

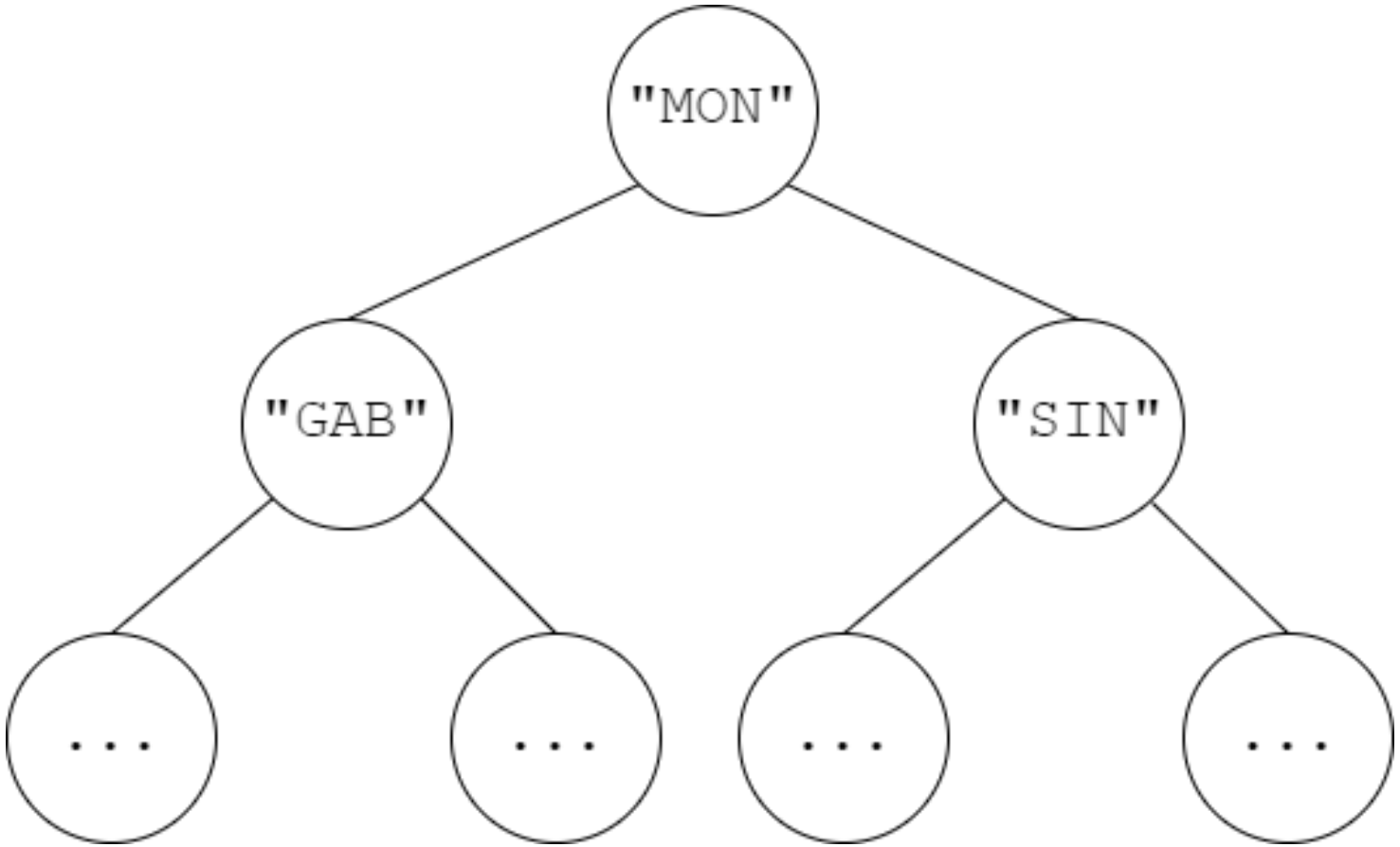
Il ne faut **jamais traverser plus** de 15 salles pour trouver un ouvrage

Structures de données - ABR

Notre bibliothèque peut être représentée sous forme d'arbre :

Structures de données - ABR

Notre bibliothèque peut être représentée sous forme d'arbre :



Structures de données - ABR

Un `ABR` est un arbre binaire dont les noeuds contiennent des valeurs qui peuvent être comparées entre elles, comme des entiers ou des chaînes de caractères, et tel que, pour tout les noeuds de l'arbre, toutes les valeurs situés dans le sous-arbre gauche (resp. droit) sont plus petites (resp. plus grandes) que la valeur située dans le noeud

Structures de données - ABR

La représentation d'un `ABR` en Python **est la même** que la représentation d'un arbre binaire [2]

2. Comme nous avons pu faire en TP

Structures de données - ABR

La représentation d'un `ABR` en Python **est la même** que la représentation d'un arbre binaire [2]

On ne fait qu'ajouter deux contraintes :

- les valeurs des noeuds peuvent être comparées avec les opérateurs $<$, $>=$, ...

2. Comme nous avons pu faire en TP

Structures de données - ABR

La représentation d'un `ABR` en Python **est la même** que la représentation d'un arbre binaire [2]

On ne fait qu'ajouter deux contraintes :

- les valeurs des noeuds peuvent être comparées avec les opérateurs $<$, $>=$, ...
- les arbres vérifient la propriété d'`ABR`

2. Comme nous avons pu faire en TP

Structures de données - ABR

Étant donné que la représentation d'un `ABR` et d'un arbre binaire est la même, ces deux structures ont des méthodes en commun :

- `taille()`
- `hauteur()`
- `parcours_infixe()`

Les parcours infixe est intéressant sur un `ABR`, pourquoi ?

Structures de données - ABR

Étant donné que la représentation d'un `ABR` et d'un arbre binaire est la même, ces deux structures ont des méthodes en commun :

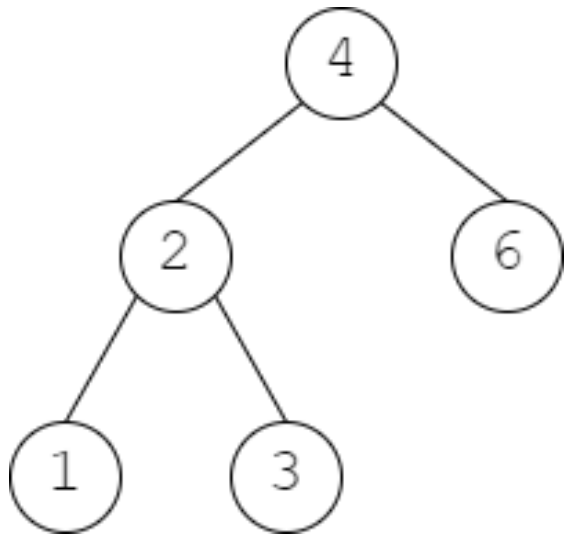
- `taille()`
- `hauteur()`
- `parcours_infixe()`

Les parcours infixe est intéressant sur un `ABR`, pourquoi ?

Ce parcours affiche les valeurs de l'`ABR` dans l'ordre croissant

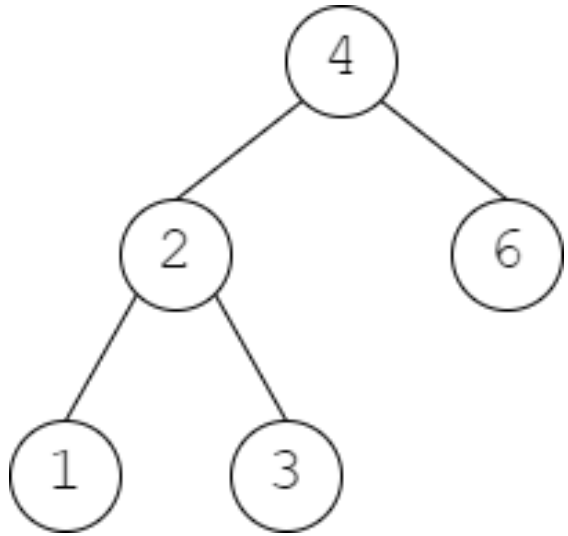
Structures de données - ABR

Présentation d'une opération propre aux `ABR` : l'insertion
Prenons l'arbre :

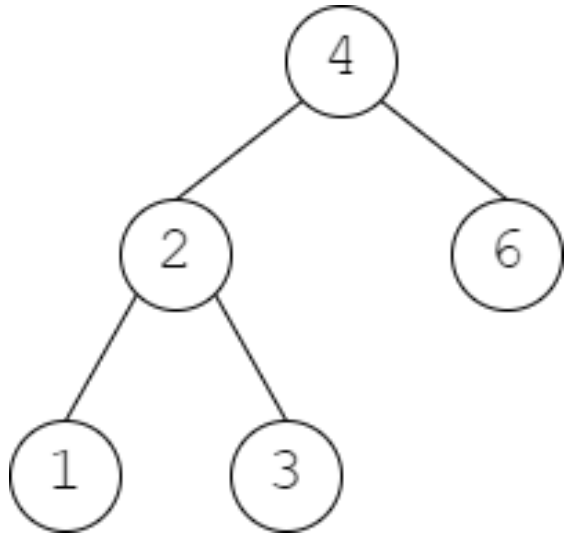


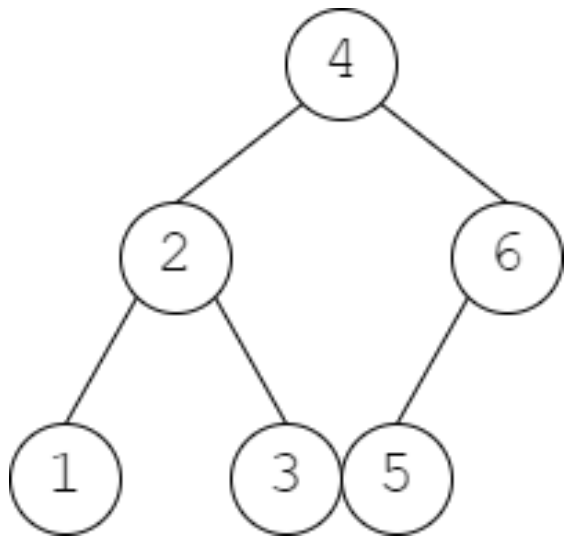
Nous voulons ajouter la valeur 5

Structures de données - ABR



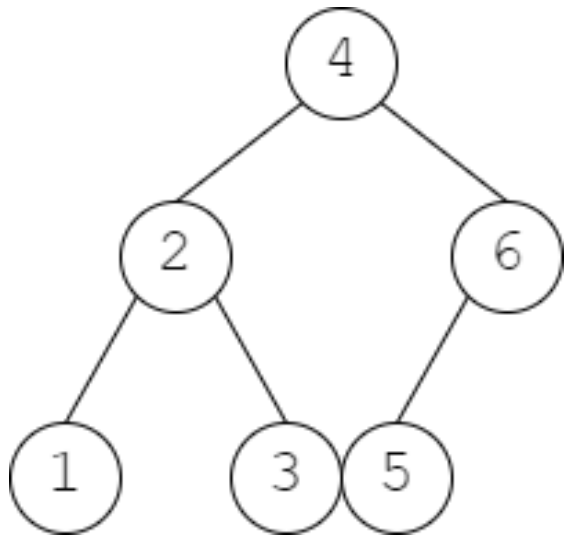
Structures de données - ABR





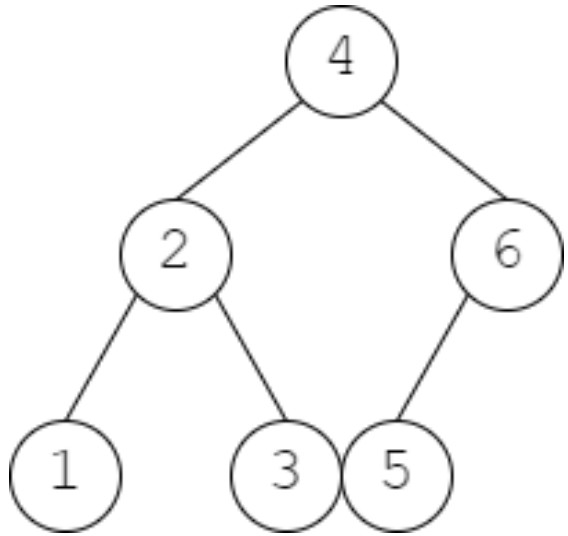
Structures de données - ABR

Soit l'ABR :

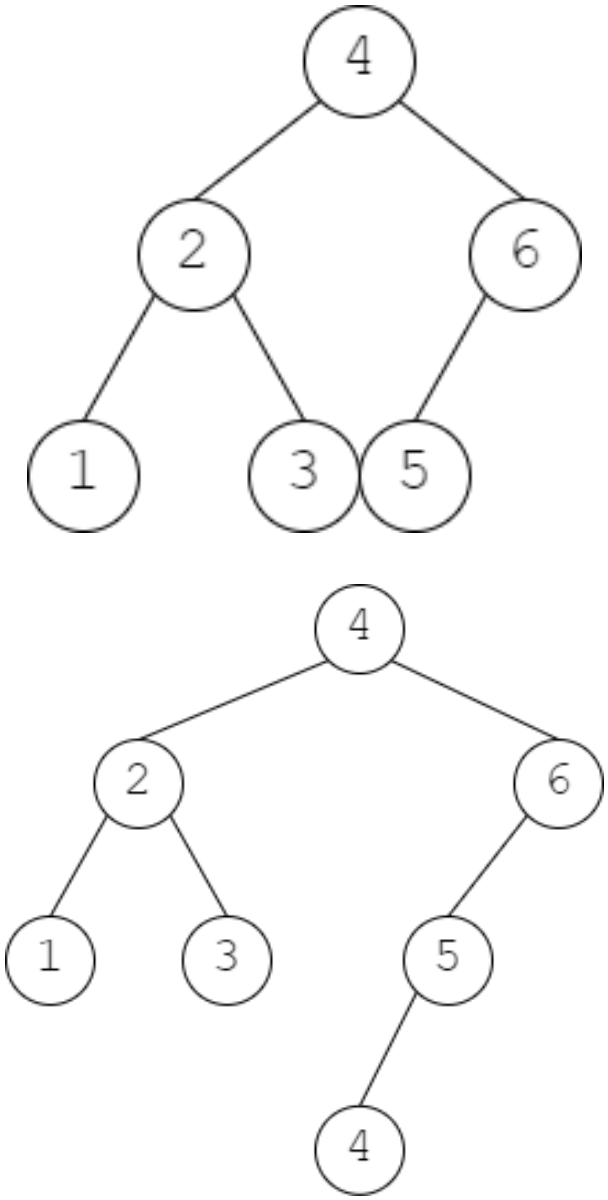


Comment insérer la valeur 4 ?

Structures de données - ABR



Structures de données - ABR



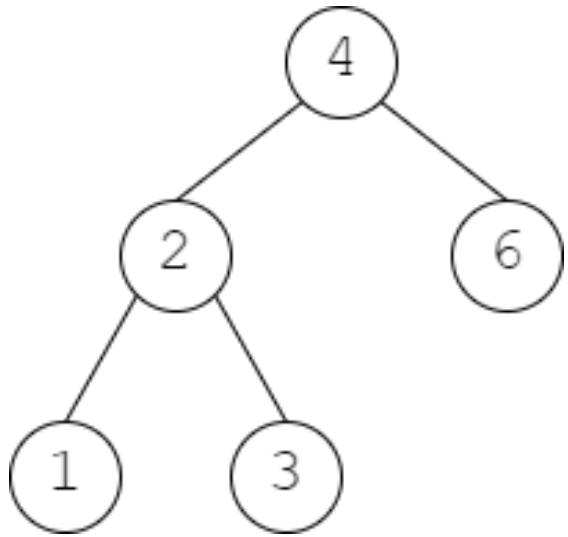
Structures de données - ABR

Si l'élément `x` apparaît déjà dans l'`ABR`, un nouveau noeud contenant une nouvelle occurence de `x` va être ajouté

Lorsque `x` est égal à la valeur d'un autre noeud, on poursuit notre insertion dans le sous-arbre droit

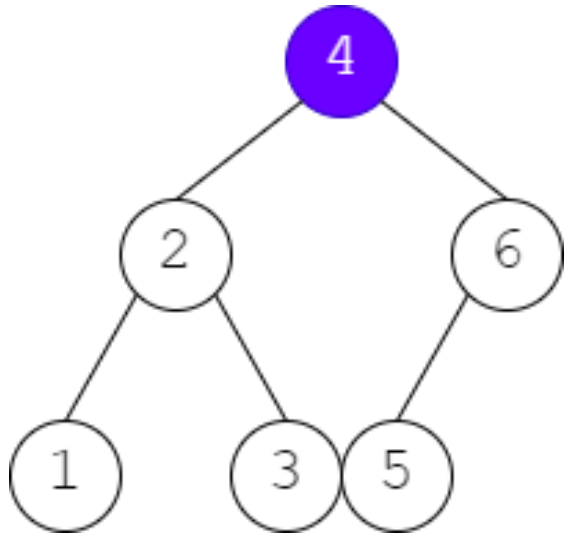
Structures de données - ABR

Présentation d'une opération propre aux `ABR` : la recherche
Prenons l'arbre :

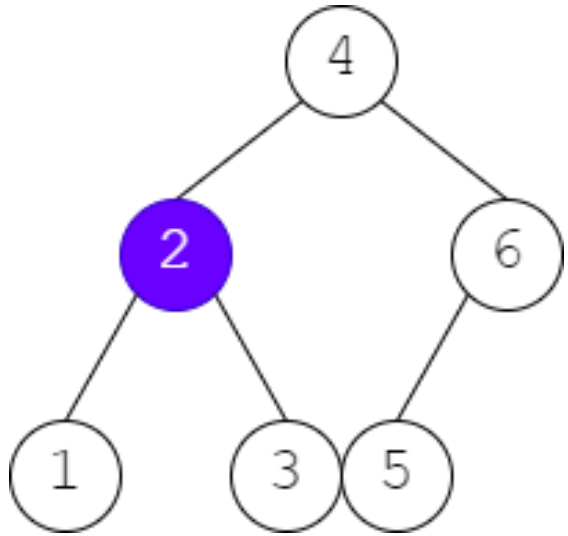


Nous voulons rechercher la valeur 3

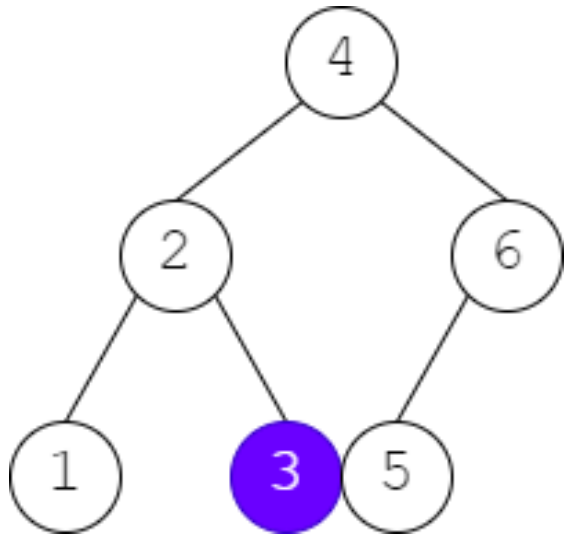
Structures de données - ABR



Structures de données - ABR



Structures de données - ABR



Structures de données - ABR

La recherche d'un élément dans un `ABR` ressemble à un algorithme déjà vu :

Structures de données - ABR

La recherche d'un élément dans un `ABR` ressemble à un algorithme déjà vu : la recherche dichotomique

A chaque étape de la recherche, on élimine un sous-arbre, soit la moitié des éléments restants

Structures de données - ABR

A retenir sur les `ABR` :

- un `ABR` est un arbre binaire particulier
- soit un noeud d'un `ABR`, les valeurs des noeuds de son sous-arbre gauche (resp. droit) sont strictement plus petites (resp. égales ou plus grandes) que la valeur du noeud considéré

Structures de données - ABR

Comparaison de complexité (dans le pire des cas) entre les tableaux, les listes chaînées, les piles, les files et les `ABR`

Structure de données	Accès	Recherche	Insertion	Suppression
Tableaux	$O(1)$	$O(n)$	$O(n)$	$O(n)$
Listes chaînées	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Piles	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Files	$O(n)$	$O(n)$	$O(1)$	$O(1)$
`ABR`	$O(n)$	$O(n)$	$O(n)$	$O(n)$