Fin des tables de hachages et arbres

Algorithmique et structures de données, 2024-2025

P. Albuquerque (B410) et O. Malaspinas (A401), ISC, HEPIA

2025-02-28

En partie inspirés des supports de cours de P. Albuquerque

Rappel

- Qu'est-ce qu'une table de hachage?

Rappel

- Qu'est-ce qu'une table de hachage?
- Structure de données abstraite où chaque *valeur* (ou élément) est associée à une *clé* (ou argument).
- Quelles sont les fonctions typiques définies sur les tables?

Rappel

- Qu'est-ce qu'une table de hachage?
- Structure de données abstraite où chaque *valeur* (ou élément) est associée à une *clé* (ou argument).
- Quelles sont les fonctions typiques définies sur les tables?
- Insertion, consultation, suppression.

```
void insert(table, key, value)
value get(table, key)
value remove(table, key)
```

- Comment fait-on le lien entre une clé et une valeur dans le tableau?

Rappel

- Qu'est-ce qu'une table de hachage?
- Structure de données abstraite où chaque *valeur* (ou élément) est associée à une *clé* (ou argument).
- Quelles sont les fonctions typiques définies sur les tables?
- Insertion, consultation, suppression.

```
void insert(table, key, value)
value get(table, key)
value remove(table, key)
```

- Comment fait-on le lien entre une clé et une valeur dans le tableau?
- On hache!

Exercice 2

- Reprendre l'exercice 1 et utiliser la technique de double hachage pour traiter les collisions avec

$$h_1(k) = k \mod 13,$$
$$h_2(k) = 1 + (k \mod 11).$$

* En cas de collision, on fait un saut de $h_2(k)$, c.-à-d.

$$index = (index + h_2(k)) \mod 13.$$

Exercice 3

- Stocker les numéros de téléphones internes d'une entreprise dans un tableau de 10 positions.
- Les numéros sont compris entre 100 et 299.
- Soit N le numéro de téléphone, la fonction de hachage est

$$h(N) = N \mod 10.$$

- La fonction de gestion des collisions est

$$C_1(N, i) = (h(N) + 3 \cdot i) \mod 10$$

où i compte les collisions.

- Placer 145, 167, 110, 175, 210, 215 (mettre son état à occupé).
- Supprimer 175 (rechercher 175, et mettre son état à supprimé).
- Rechercher 35.
- Les cases ni supprimées, ni occupées sont vides.
- Expliquer se qui se passe si on utilise?

$$C_1(N, i) = (h(N) + 5 \cdot i) \mod 10.$$

Préambule

- Ici, on ne considère pas le cas du chaînage en cas de collisions.
- L'insertion est construite avec une forme du type

```
index = h(key);
while (table[index].state == OCCUPIED
       && table[index].key != key) {
   index = (index + k) % table_size; // attention à pas dépasser
}
table[index].key = key;
table[index].state = OCCUPIED;
```

- Gestion de l'état d'une case *explicite*

```
typedef enum {EMPTY, OCCUPIED, DELETED} state;
```

L'insertion

Pseudocode?

L'insertion

Pseudocode?

```
rien insertion(table, clé, valeur) {
    index = hash(clé)
    index =
        tant que état(table[index]) == occupé
                et clé(table[index]) != clé:
            index = rehash(clé)

    état(table[index]) = occupé
    table[index] = valeur
}
```

La suppression

Pseudocode?

La suppression

Pseudocode?

```
valeur suppression(table, clé):
    index = hash(clé)
    tant que état(table[index]) != vide:
        si état(table[index]) == occupé
                et clé(table[index]) == clé:
            état(table[index]) = supprimé
        sinon
            index = rehash(clé)
}
```

La recherche

Pseudocode?

La recherche

Pseudocode?

```
booléen recherche(table, clé) {
    index = hash(clé)
    tant que état(table[index]) != vide:
        si état(table[index]) == occupé
                et clé(table[index]) == clé:
            retourner vrai
        sinon
            index = rehash
    retourner faux
}
```

Écrivons le code!

- Mais avant:
  - Quelles sont les structures de données dont nous avons besoin?
  - Y a-t-il des fonctions auxiliaires à écrire?
  - Écrire les signatures des fonctions.

Écrivons le code!

- Mais avant:
  - Quelles sont les structures de données dont nous avons besoin?
  - Y a-t-il des fonctions auxiliaires à écrire?
  - Écrire les signatures des fonctions.

Structures de données

Écrivons le code!

- Mais avant:
  - Quelles sont les structures de données dont nous avons besoin?
  - Y a-t-il des fonctions auxiliaires à écrire?
  - Écrire les signatures des fonctions.

Structures de données

```
typedef enum {empty, deleted, occupied};
typedef ... key_t;
typedef ... value_t;
typedef struct _cell_t {
    key_t key;
    value_t value;
    state_t state;
} cell_t;
typedef struct _hm {
    cell_t *table;
    int capacity;
    int size;
} hm;
```

Écrivons le code!

Fonctions auxiliaires

Écrivons le code!

Fonctions auxiliaires

```
static int hash(key_t key);
static int rehash(int index, key_t key);
static int find_index(hm h, key_t key);
```

Signature de l'API

Écrivons le code!

Fonctions auxiliaires

```
static int hash(key_t key);
static int rehash(int index, key_t key);
static int find_index(hm h, key_t key);
```

Signature de l'API

```
void hm_init(hm *h, int capacity);
void hm_destroy(hm *h);
bool hm_set(hm *h, key_t key, value_t *value);
bool hm_get(hm h, key_t key, value_t *value);
bool hm_remove(hm *h, key_t key, value_t *value);
bool hm_search(hm h, key_t key);
void hm_print(hm h);
```

Live code session!

0. Offered to you by ProtonVPN[1]!

[1] The fastest way to connect to BBB!

Live code session!

0. Offered to you by ProtonVPN[1]!

1. Like the video.
2. Subscribe to the channel.
3. Use our one time voucher for ProtonVPN: `PAULISAWESOME`.
4. Consider donating on our patreon.

[1] The fastest way to connect to BBB!

Les arbres

Les arbres: définition

"Un arbre est un graphe acyclique orienté possédant une unique racine, et tel que tous les nœuds sauf la racine ont un unique parent."

Les arbres: définition

"Un arbre est un graphe acyclique orienté possédant une unique racine, et tel que tous les nœuds sauf la racine ont un unique parent."

Santé!

Plus sérieusement

- Ensemble de **nœuds** et d'**arêtes** (graphe).
- Les arêtes relient les nœuds entre eux, mais pas n'importe comment: chaque nœud a au plus un **parent**.
- Le seul nœud sans parent est la **racine**.
- Chaque nœud a un nombre fini d'**enfants**.
- La hiérarchie des nœuds rend les arêtes **orientées** (parent -> enfants), et empêche les **cycles** (acyclique, orienté).
- La **feuille** ou **nœud terminal** est un nœud sans enfants.
- Le **niveau** est 1 à la racine et **niveau+1** pour les enfants.
- Le **degré** d'un nœud est le nombre de enfants du nœud.

Les arbres: définition

"Un arbre est un graphe acyclique orienté possédant une unique racine, et tel que tous les nœuds sauf la racine ont un unique parent."

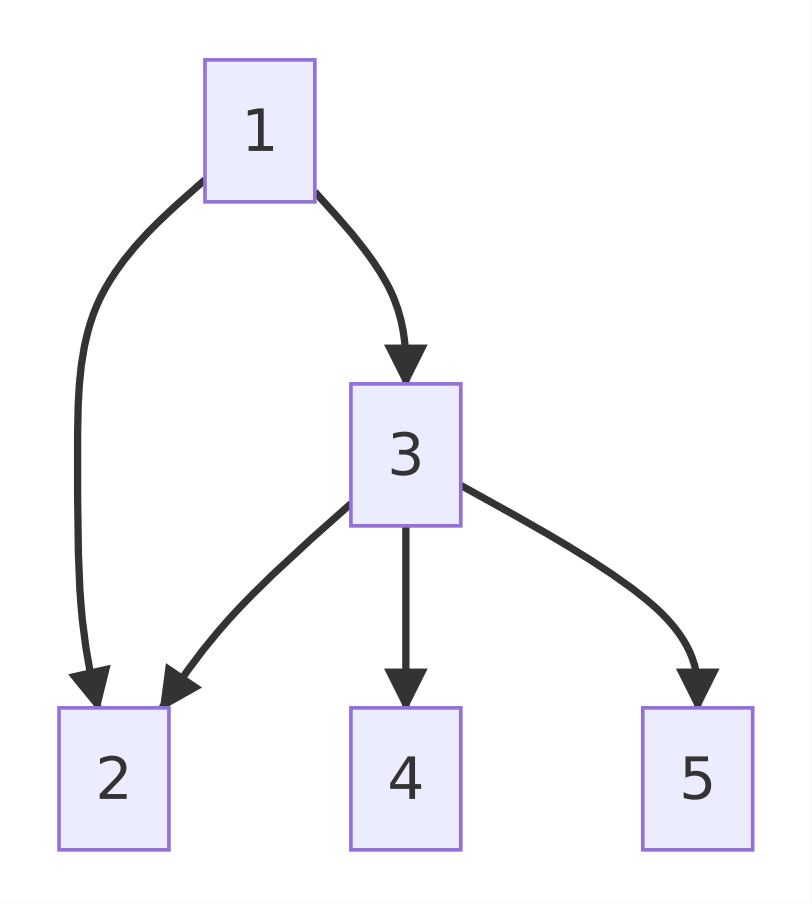
Santé!

Plus sérieusement

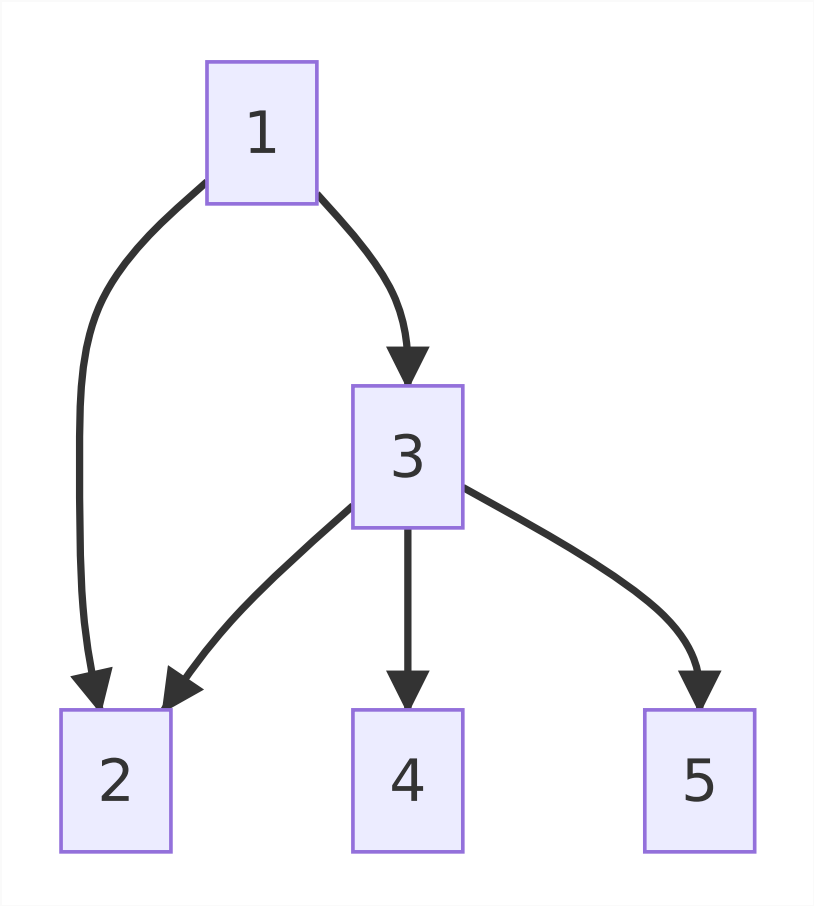
- Ensemble de **nœuds** et d'**arêtes** (graphe).
- Les arêtes relient les nœuds entre eux, mais pas n'importe comment: chaque nœud a au plus un **parent**.
- Le seul nœud sans parent est la **racine**.
- Chaque nœud a un nombre fini d'**enfants**.
- La hiérarchie des nœuds rend les arêtes **orientées** (parent -> enfants), et empêche les **cycles** (acyclique, orienté).
- La **feuille** ou **nœud terminal** est un nœud sans enfants.
- Le **niveau** est 1 à la racine et **niveau+1** pour les enfants.
- Le **degré** d'un nœud est le nombre de enfants du nœud.

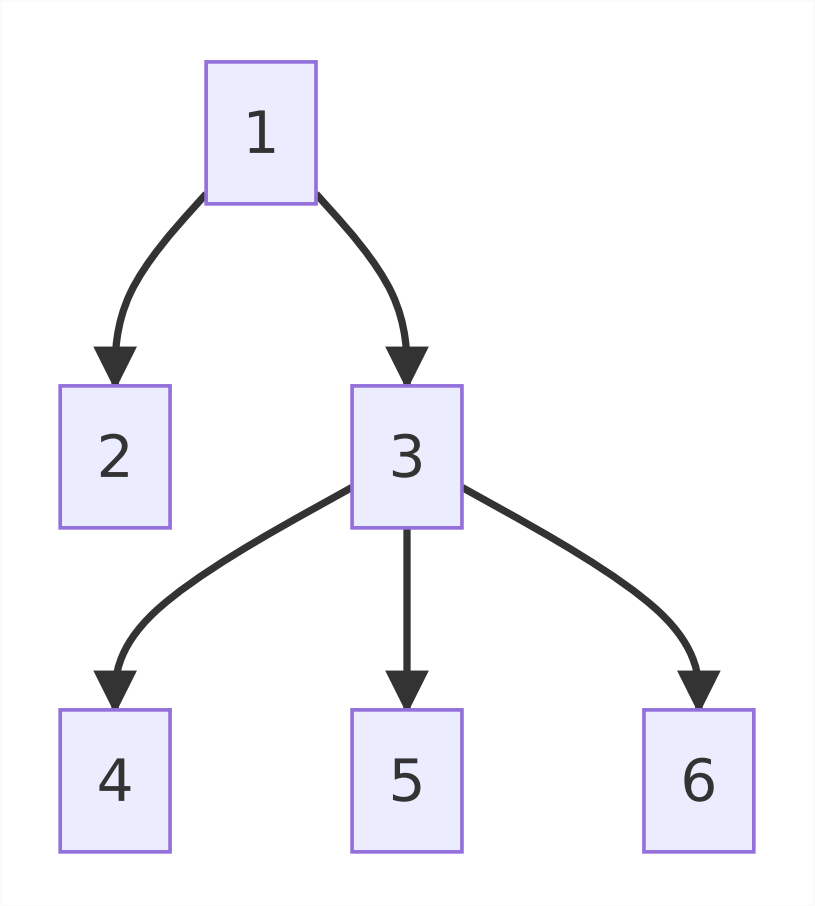
- Chaque nœud est un arbre en lui même.
- La **récursivité** sera très utile!

Arbre ou pas arbre?

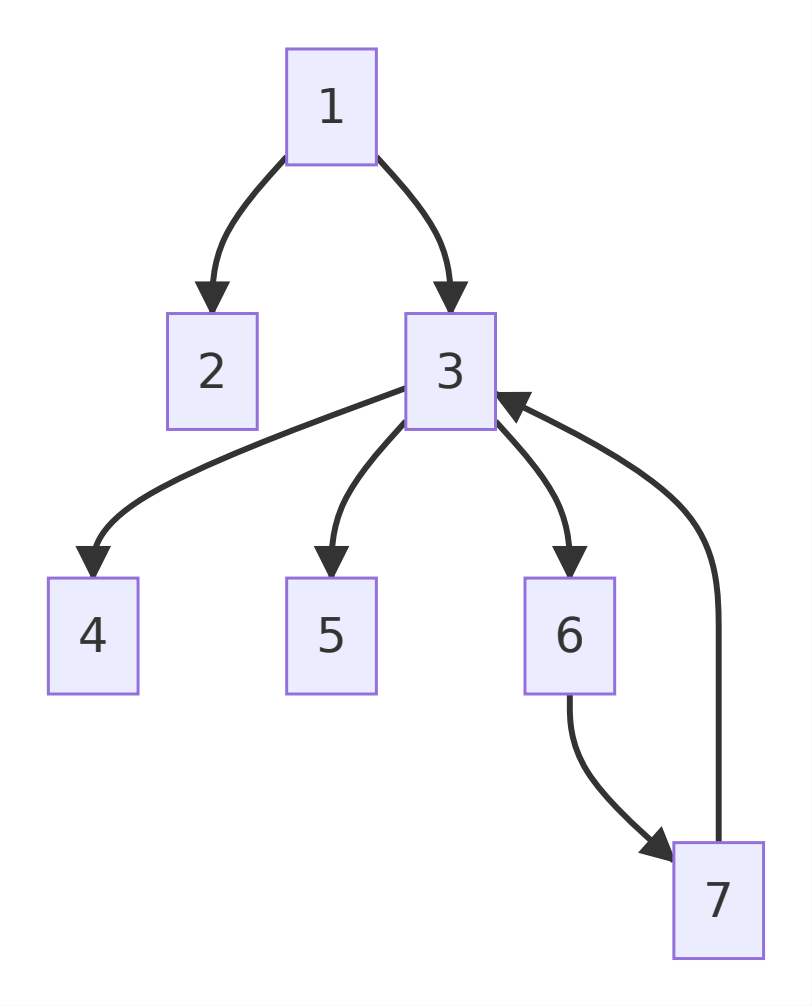


Arbre ou pas arbre?

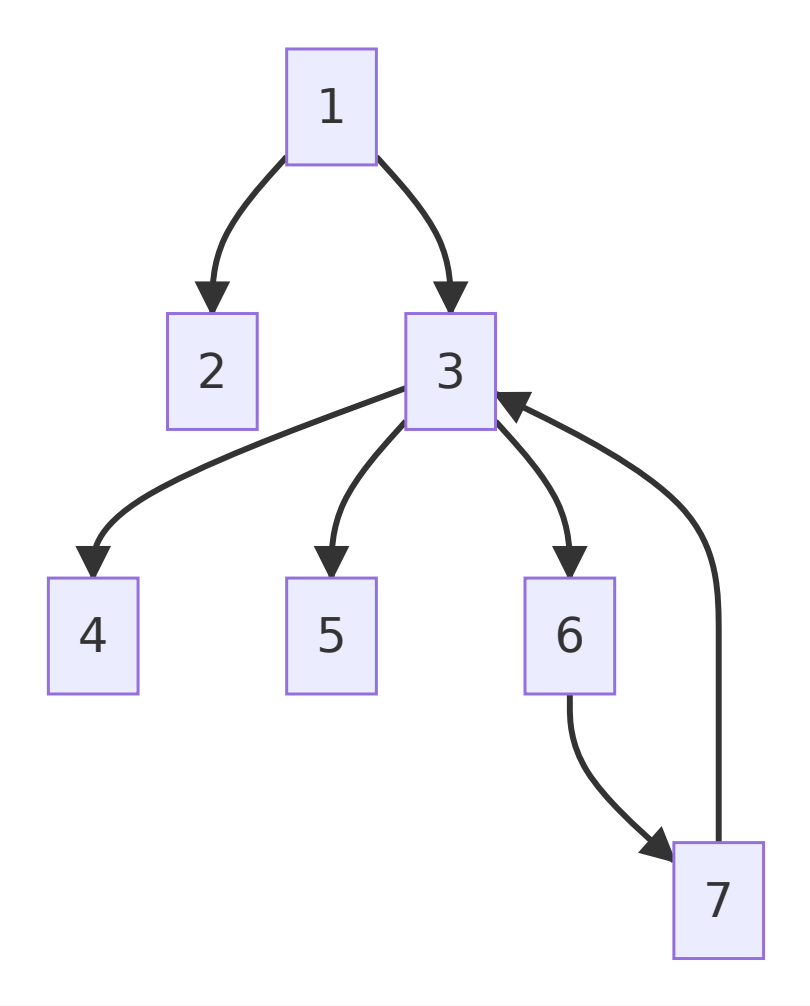


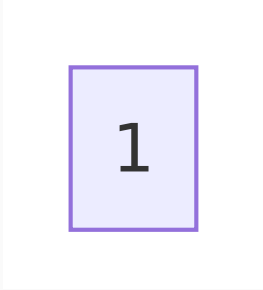


Arbre ou pas arbre?

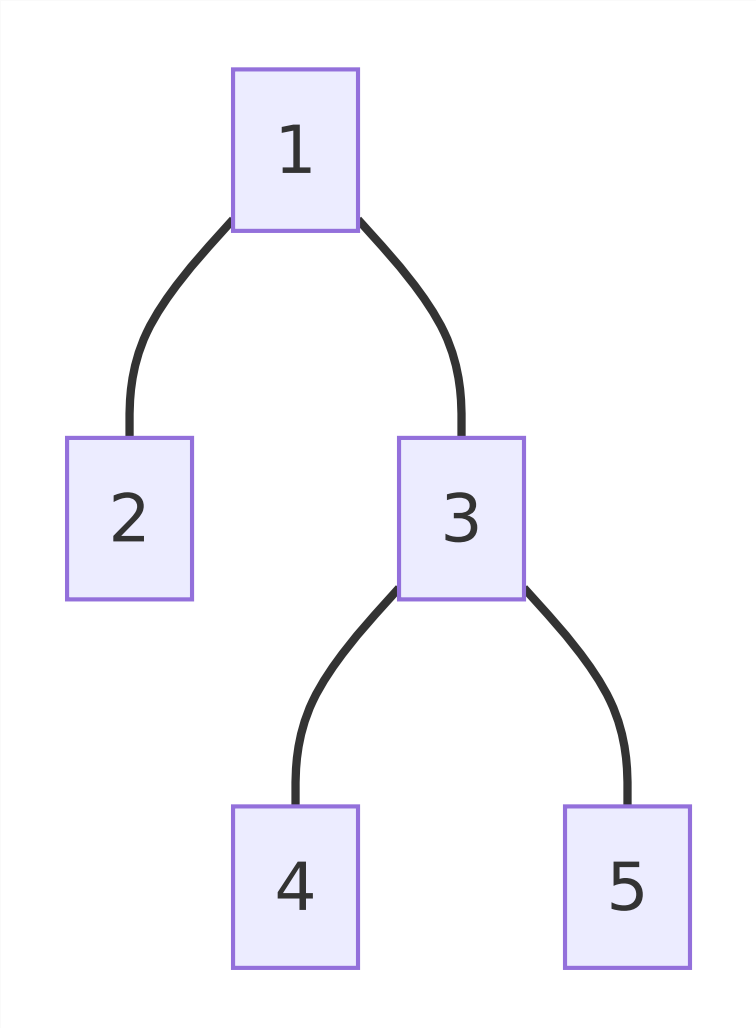


Arbre ou pas arbre?

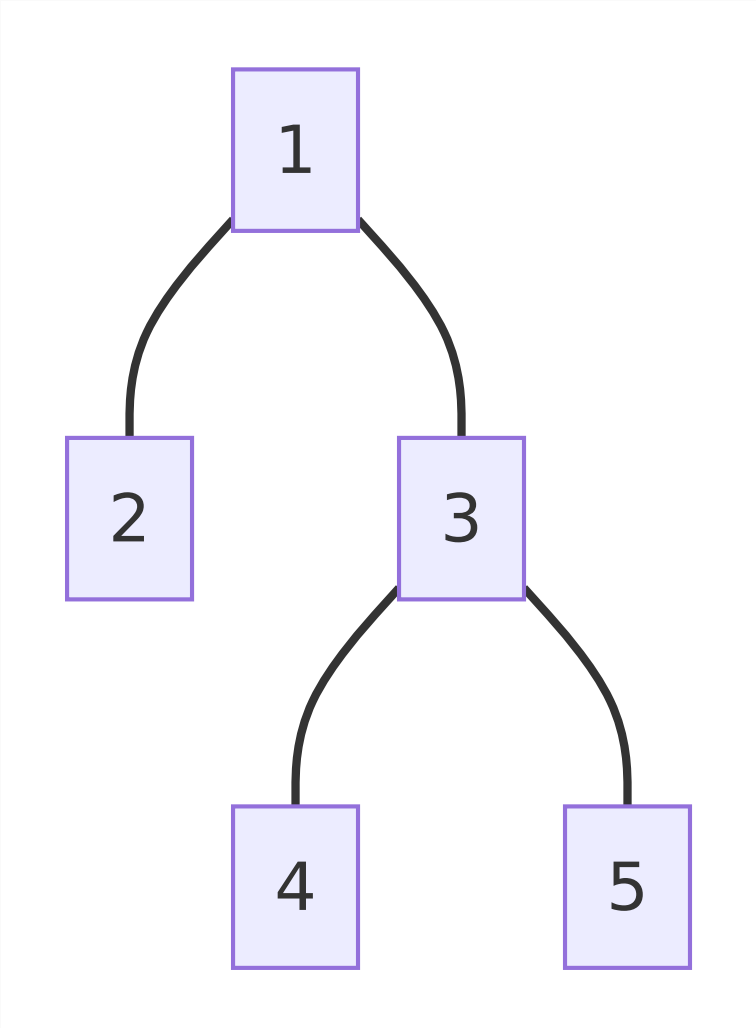


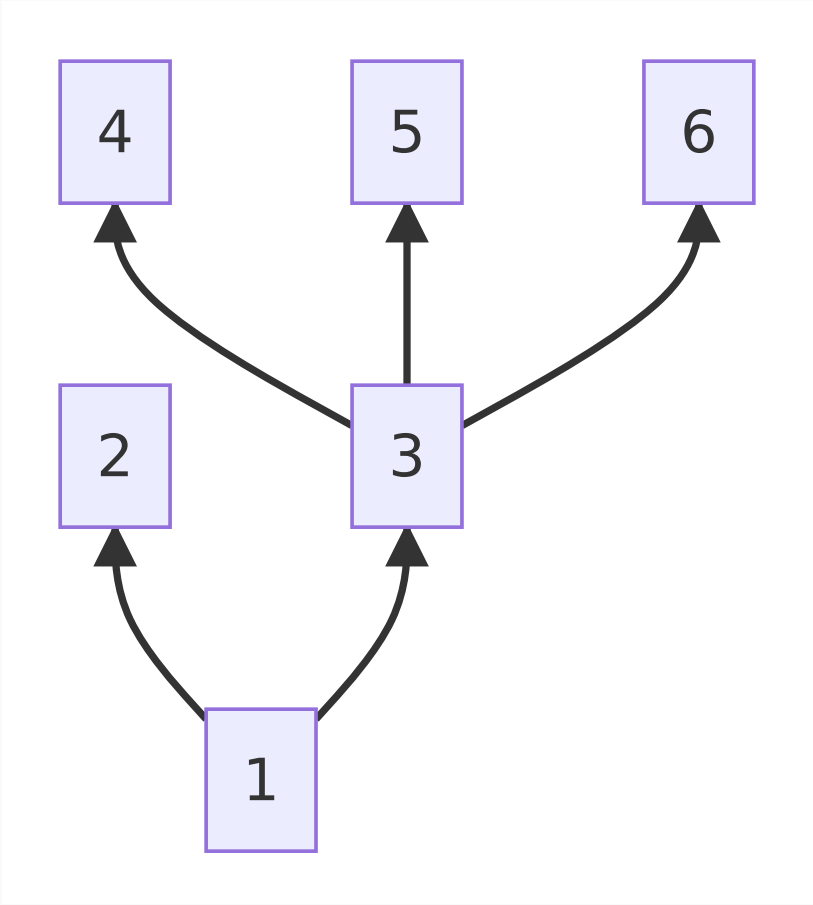


Arbre ou pas arbre?



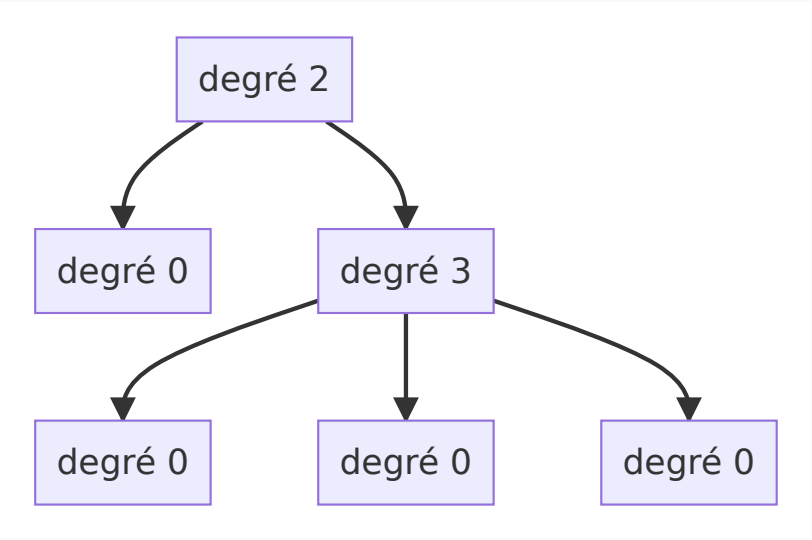
Arbre ou pas arbre?





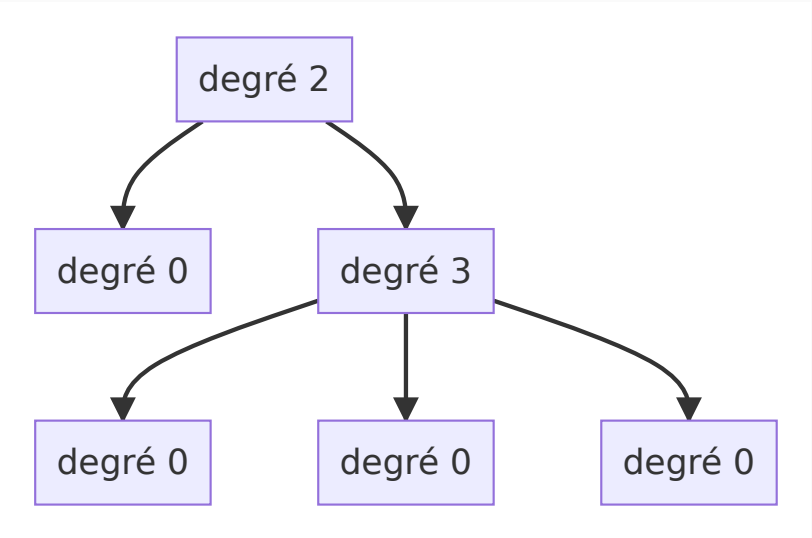
Degré et niveau

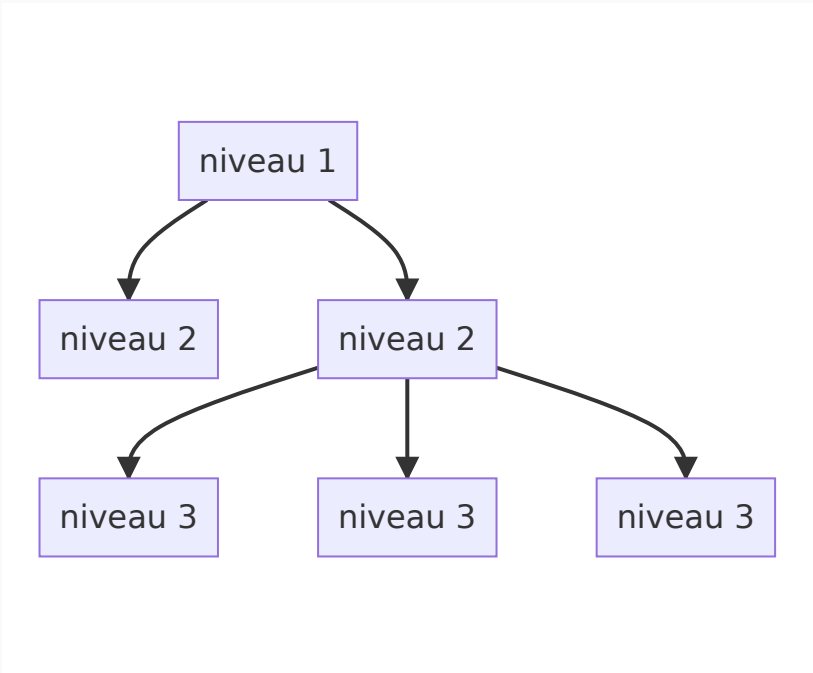
- Illustration du degré (nombre d'enfants) et du niveau (profondeur)



Degré et niveau

- Illustration du degré (nombre d'enfants) et du niveau (profondeur)





- Les nœuds de degré 0 sont des feuilles.

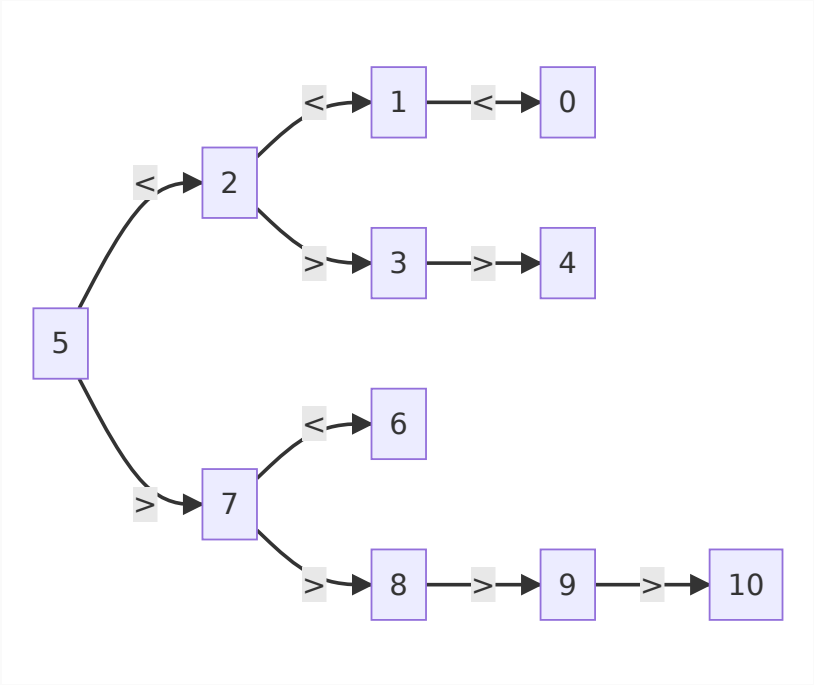
Application: recherche rapide

Pouvez-vous construire un arbre pour résoudre le nombre secret?

Application: recherche rapide

Pouvez-vous construire un arbre pour résoudre le nombre secret?

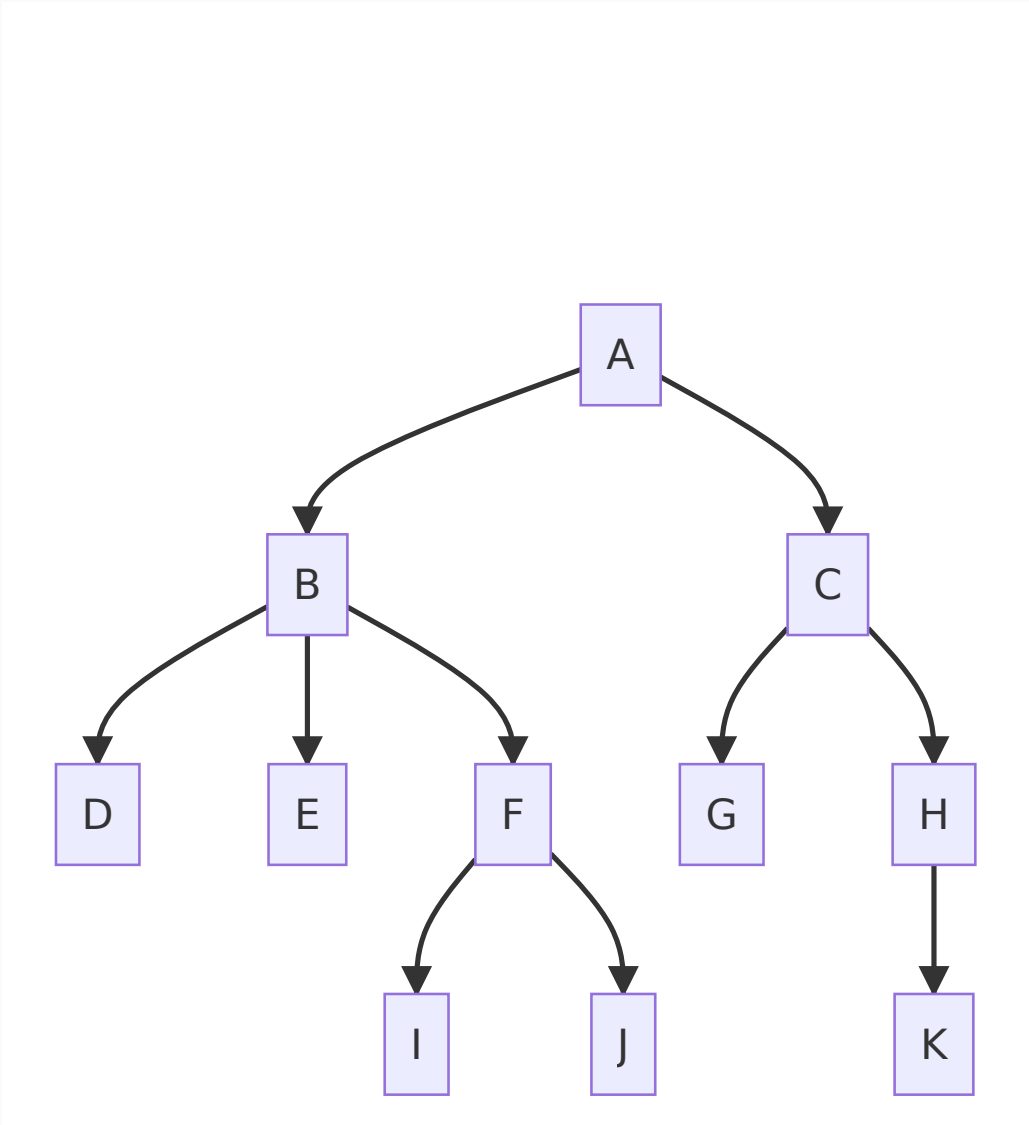
- Le nombre secret ou la recherche dichotomique (nombre entre 0 et 10).



Question: Quelle est la complexité pour trouver un nombre?

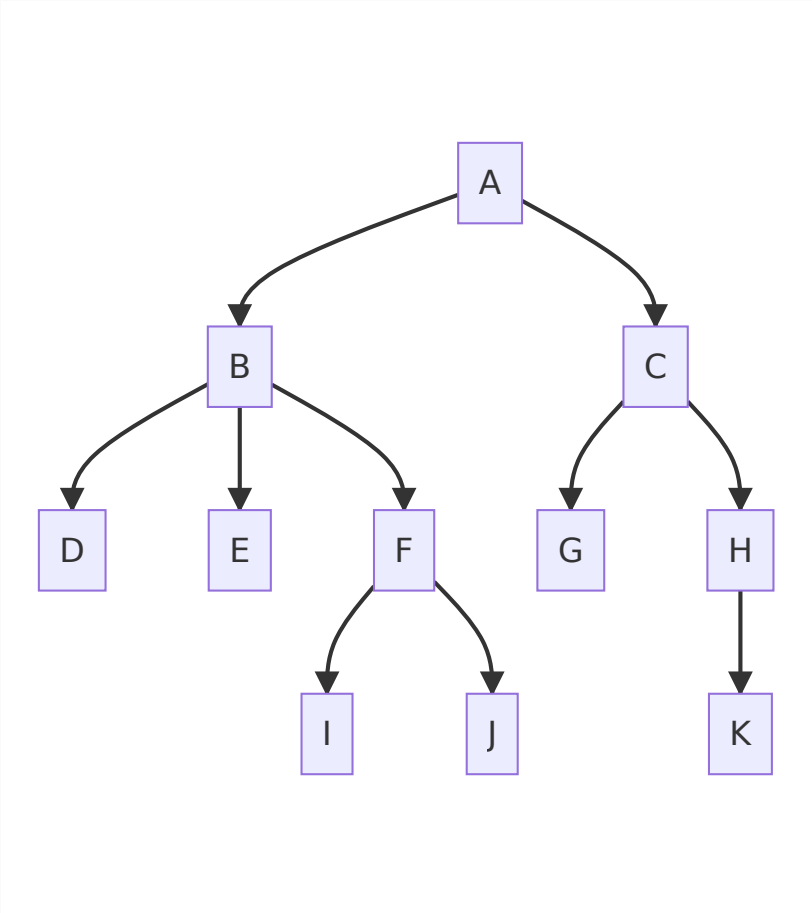
Autres représentations

- Botanique
- **Exercice:** Ajouter les degrés/niveaux et feuilles



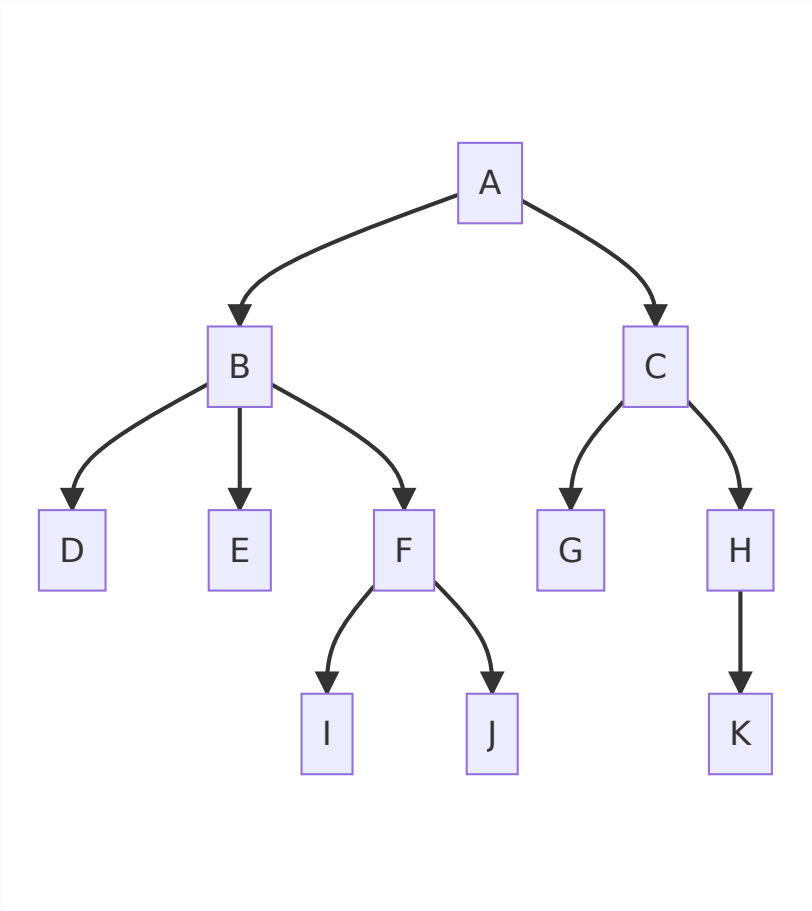
Autres représentations

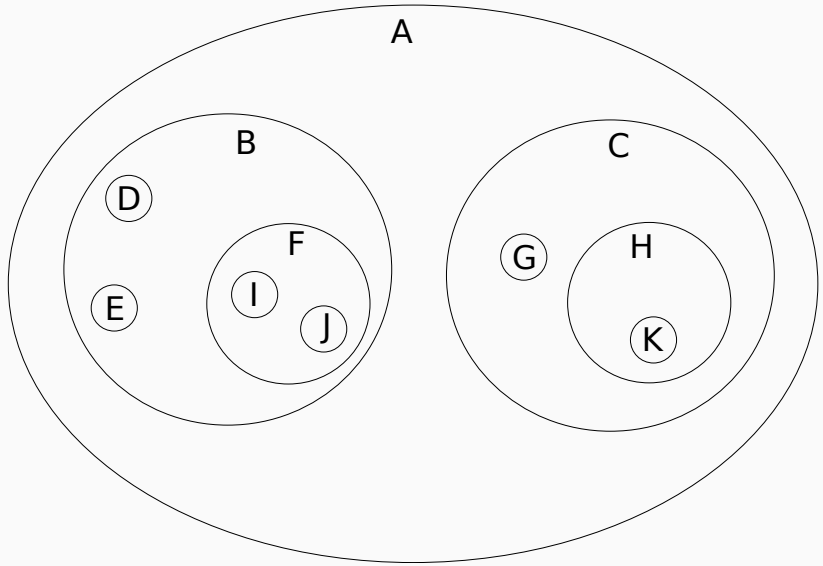
- Ensembliste



Autres représentations

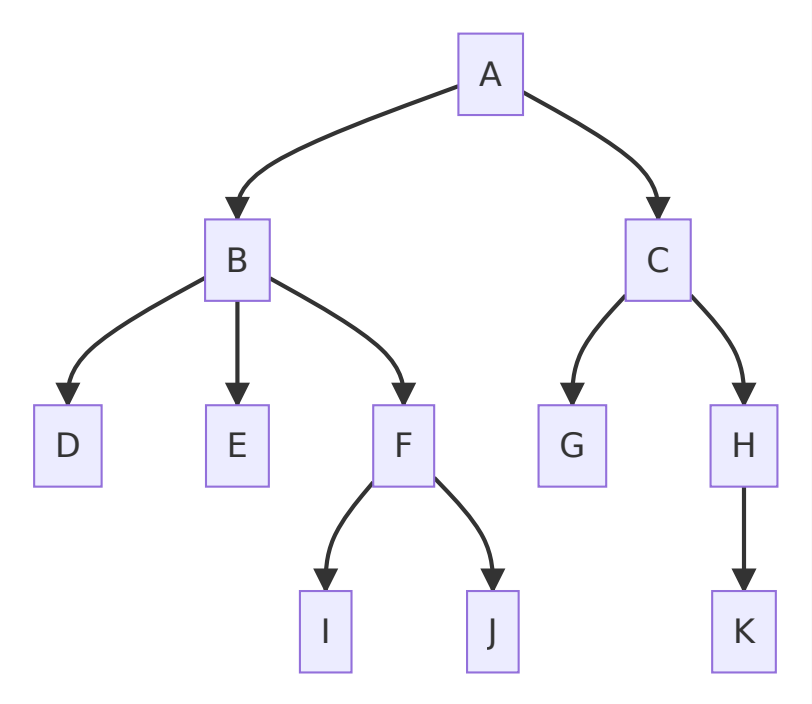
- Ensembliste





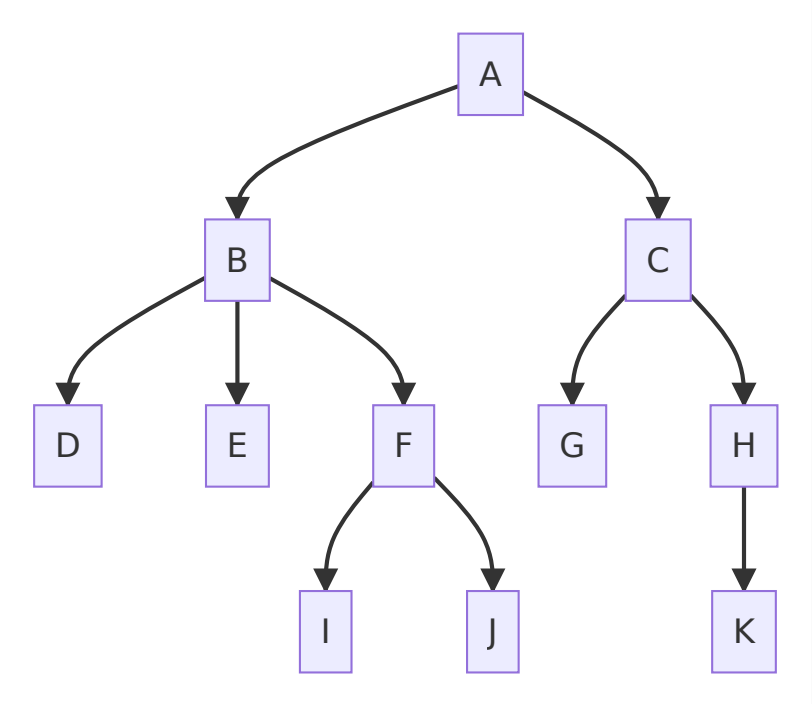
Autres représentations

- Liste



Autres représentations

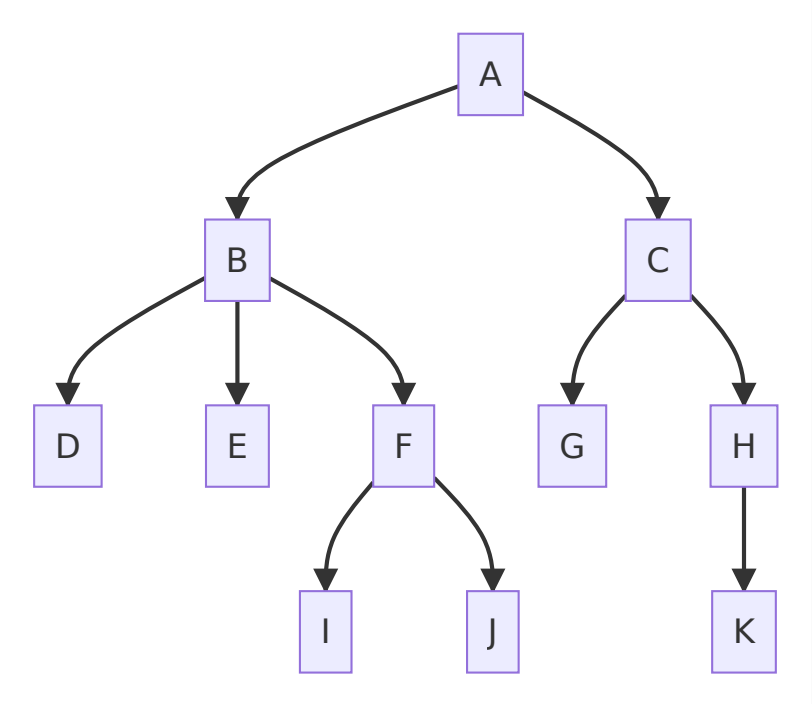
- Liste




```
(A
    (B
        (D)
        (E)
        (F
            (I)
            (J)
        )
    )
    (C
        (G)
        (H
            (K)
        )
    )
)
```

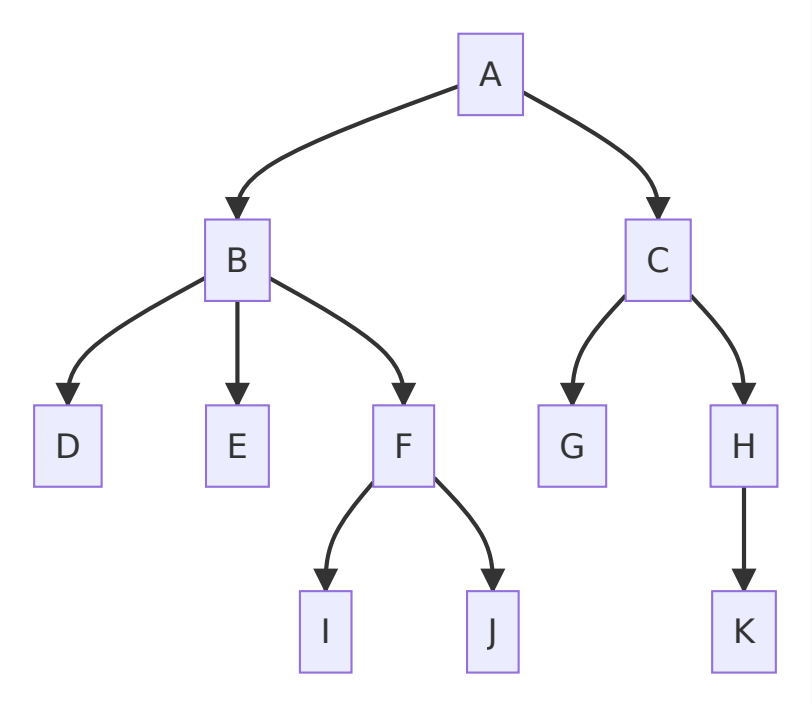
Autres représentation

- Par niveau



Autres représentation

- Par niveau




```
1          2          3          4
-----------------------------------
A
           B
                      D
                      E
                      F
                                 I
                                 J
           C
                      G
                      H
                                 K
```