Arbres Binaires

Alain Camanes

alain.camanes@free.fr

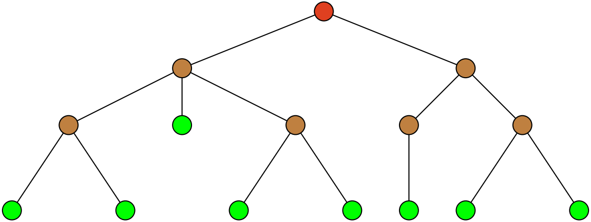
Stanislas

Option Informatique
2021-2022

Plan

1. Arbres
   - Définitions
   - Types
2. Fonctions
3. Parcours
4. Plus d'arbres

Un arbre



Des arbres

- Appels des fonctions récursives (voir les chapitres précédents).
- Arbres de décisions (voir la fin du chapitre).
- Expressions arithmétiques (voir le chapitre sur les Structures de données).
- Arbres pour trier (voir TP).
- ...

Graphe - Idée formelle

$\hookrightarrow$ Graphe non orienté. $G = (V, E)$ où

- V : ensemble des sommets.
- $E \subset \mathscr{P}_2(V)$ ensemble des arêtes.

$\hookrightarrow$ Chemin de u à v. $(u_0, \ldots, u_k) \in V^{k+1}$ telles que

$$u = u_0,\ v = u_k,\ \{u_i, u_{i+1}\} \in E.$$

$\hookrightarrow$ Arbre. Graphe *acyclique* et *connexe* : Deux sommets sont reliés par exactement un chemin.
Sommets = *Nœuds*.

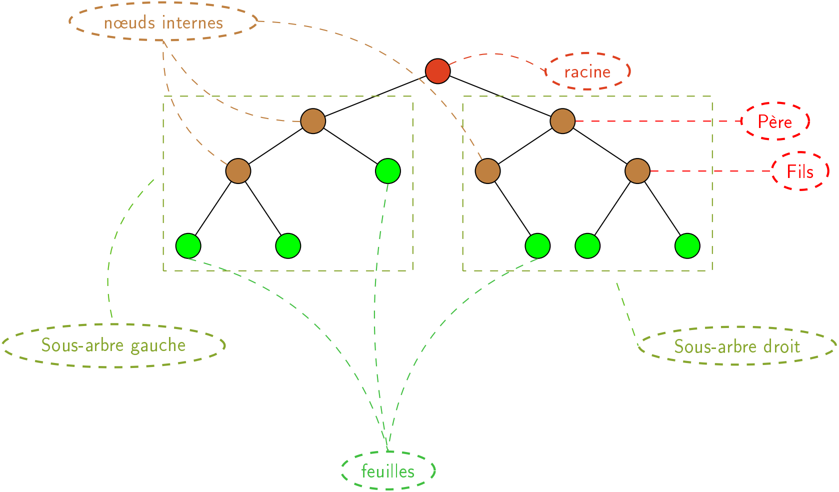
$\hookrightarrow$ Arbre enraciné. Arbre dont un nœud (la *racine*) est distingué.
Degré d'un noœud = Nombre de fils.

$\hookrightarrow$ Feuille. Nœud ne possédant aucun fils.

$\hookrightarrow$ Arbres binaires enracinés ($\mathbb{T}_b$). Chaque nœud est de degré au plus 2.
Structure finie contenant

- aucun nœud (arbre vide),
- un triplet :
  - une racine,
  - un arbre binaire (*sous-arbre gauche*),
  - un arbre binaire (*sous-arbre droit*).

Vocabulaire illustré



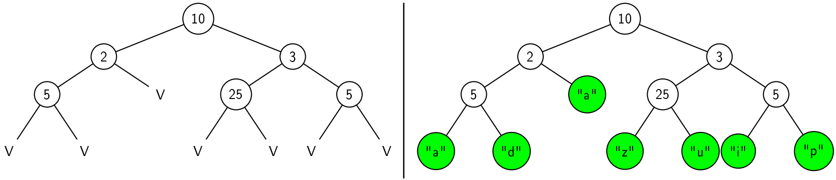
Types

↪ Arbres binaires étiquetés.

```
type 'a arbre_binaire = Vide
         | Noeud of 'a * 'a arbre_binaire
                       * 'a arbre_binaire
```

↪ Arbres binaires stricts (tout nœud a 0 ou 2 fils).

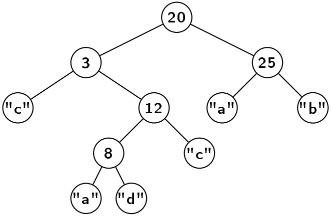
```
type ('a, 'b) abs =
     | Feuille of 'a
     | Noeud_interne of 'b * ('a, 'b) abs
                           * ('a, 'b) abs
```



Arbre binaire strict

```
type ('a, 'b) abs =
    | Feuille of 'a
    | Noeud_interne of 'b * ('a, 'b) abs
                          * ('a, 'b) abs
```

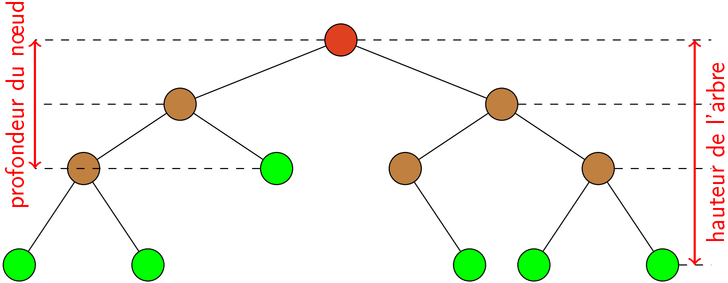
```
let arbre =
 Noeud_interne (20,
   Noeud_interne (3,
     Feuille "c",
     Noeud_interne(12,
       Noeud_interne(8,
         Feuille "a",
         Feuille "d"),
       Feuille "c")),
   Noeud_interne(25,
     Feuille "a",
     Feuille "b")) ;;
```



Plan

1. Arbres
2. Fonctions
   - Fonctions usuelles
   - Propriétés
3. Parcours
4. Plus d'arbres

Grandeurs caractéristiques



Fonctions élémentaires

↪ Profondeur d'un nœud dans T.

$$\begin{cases} p_T(x) & = 0 \text{ si } x \text{ est racine,} \\ p_T(x) & = 1 + p_T(y) \text{ si } y \text{ est le père de } x. \end{cases}$$

↪ Hauteur d'un arbre.

$$\begin{cases} h(\texttt{Vide}) & = -1, \\ h((r, T_1, T_2)) & = 1 + \max\{h(T_1), h(T_2)\}. \end{cases}$$

```
let rec hauteur tree =
  match tree with
    | Feuille n -> 0
    | Noeud_interne (r, t1, t2) ->
        1 + max (hauteur t1) (hauteur t2) ;;
```

$\mathscr{P}(\cdot)$ prédicat sur l'ensemble des arbres binaires.

$\hookrightarrow$ Hauteur. Récurrence classique.
$\mathscr{P}(\texttt{Vide})$ ou $\mathscr{P}(\texttt{Feuille a})$.
$(\forall\ T \in \mathbb{T}_b\ ;\ h(T) = n,\ \mathscr{P}(T)) \ \Rightarrow\ (\forall\ T \in \mathbb{T}_b\ ;\ h(T) = n+1,\ \mathscr{P}(T))$.

$\hookrightarrow$ Nombre de nœuds. Noté $n(T)$. Récurrence classique.
$\mathscr{P}(\texttt{Vide})$ ou $\mathscr{P}(\texttt{Feuille a})$.
$(\forall\ T \in \mathbb{T}_b\ ;\ n(T) = n,\ \mathscr{P}(T)) \ \Rightarrow\ (\forall\ T \in \mathbb{T}_b\ ;\ n(T) = n+1,\ \mathscr{P}(T))$.

$\hookrightarrow$ Sous-arbre Induction structurelle.
$\mathscr{P}(\texttt{Vide})$ ou $\mathscr{P}(\texttt{Feuille a})$.
$\forall\ T_1, T_2 \in \mathbb{T}_b, \forall\ r \in \texttt{'a}, (\mathscr{P}(T_1) \text{ et } \mathscr{P}(T_2)) \ \Rightarrow\ \mathscr{P}((r, T_1, T_2))$.

Propriétés

$\hookrightarrow$ Hauteur d'un arbre vs. Profondeur des nœuds.

$$h(T) = \max\left\{p_T(x),\ x \text{ nœud de } T\right\}.$$

$\hookrightarrow$ Nombre de feuilles.

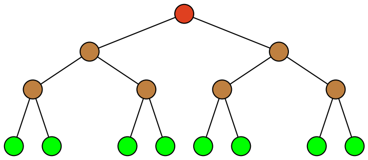
$$\ell(\texttt{Feuille a}) = 1,$$
$$\ell((r, T_1, T_2)) = \ell(T_1) + \ell(T_2).$$

Théorème

Soit $T \in \mathbb{T}_b$ de hauteur h et $|T|$ le nombre de nœuds de T. Alors,

$$h(A) + 1 \leqslant |A| \leqslant 2^{h(A)+1} - 1.$$

$\hookrightarrow$ Cas d'égalité à droite. Arbre binaire *complet*.
Toutes les *feuilles* ont la *même profondeur*.



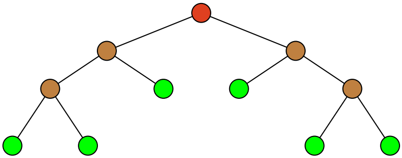
T un arbre binaire strict

$\hookrightarrow$ Relation entre les nombres de nœuds.

- $\ell(T)$ nombre de feuilles,
- $n_2(T)$ nombre de nœuds internes,

Théorème

$$\ell(T) = n_2(T) + 1.$$



Plan

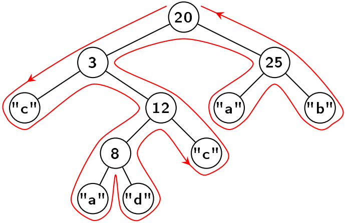
Parcours

↪ Préfixe. Démarre à la racine. Parcourt le sous-arbre de gauche puis le sous-arbre de droite.
praefixus = fixé devant

↪ Infixe. Parcourt le sous-arbre de gauche, puis la racine, puis le sous-arbre de droite.
infixus = inséré

↪ Postfixe. Parcourt le sous-arbre de gauche, puis le sous-arbre de droite, puis la racine.

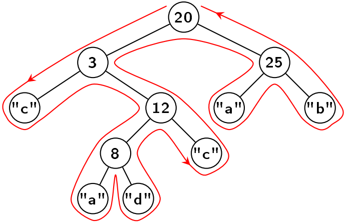
Parcours Préfixe




```
let rec prefixe tree =
  match tree with
    | Feuille n -> print_string n; print_string " "
    | Noeud_interne (r, t1, t2) ->
        print_int r; print_string " ";
        prefixe t1;
        prefixe t2 ;;

prefixe arbre ;;
20 3 c 12 8 a d c 25 a b - : unit = ()
```

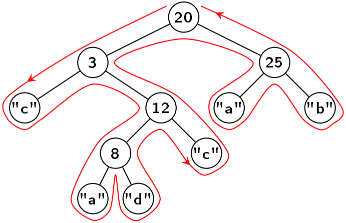
Parcours Infixe




```
let rec infixe tree =
  match tree with
    | Feuille n -> print_string n; print_string " "
    | Noeud_interne (r, t1, t2) ->
      infixe t1;
      print_int r; print_string " ";
      infixe t2 ;;

infixe arbre ;;
c 3 a 8 d 12 c 20 a 25 b - : unit = ()
```

Parcours Postfixe




```
let rec postfixe tree =
  match tree with
    | Feuille n -> print_string n; print_string " "
    | Noeud_interne (r, t1, t2) ->
        postfixe t1;
        postfixe t2;
        print_int r; print_string " " ;;

postfixe arbre ;;
c a d 8 c 12 3 a b 25 20 - : unit = ()
```

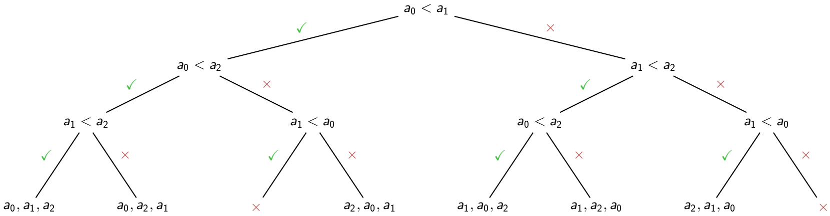
Plan

Arbres de décision & Tris

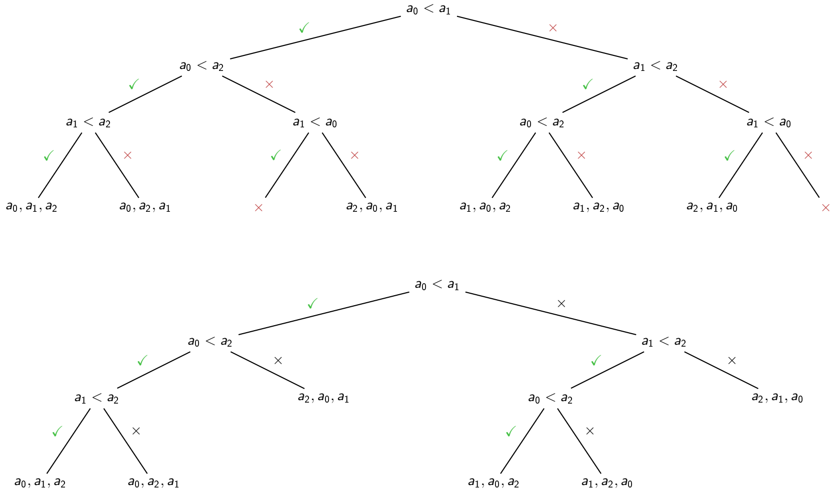
↪ Tris par comparaisons = tableaux à entrées distinctes.

↪ Nœuds = comparaisons des éléments du tableau.

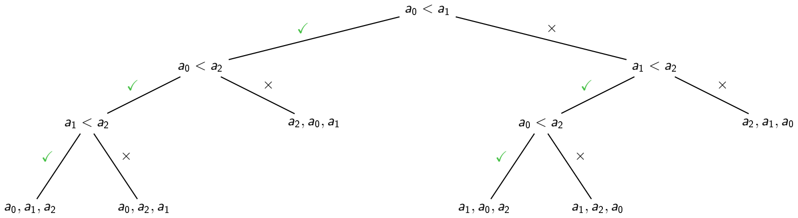
↪ Arbre de décision. (tri par sélection, $n = 3$).



Arbres de décision & Élagage



Complexité maximale



$\hookrightarrow$ Feuille = Permutation des entrées. $\ell(T) \geqslant n!$.

$\hookrightarrow$ Pire des cas = Hauteur h de l'arbre.

$\hookrightarrow$ Arbre binaire. $\ell(T) \leqslant 2^h$.

$\hookrightarrow$ Complexité dans le pire des cas.

$$\begin{aligned} C_{max} &\geqslant \log_2 \ell(T) \geqslant \log_2(n!) \\ &\geqslant \sum_{k=1}^{n} \log_2 k \geqslant \int_1^{n+1} \log_2(t)\, \mathrm{d}t \\ &\geqslant C\, n \log_2 n. \end{aligned}$$

Arbres complets

$\hookrightarrow$ Arbre binaire localement complet. $\mathbb{T}_{lc}$.

- soit l'arbre vide,
- soit tous les nœuds sont d'arité 0 ou 2.

$\hookrightarrow$ Bijection. $\mathbb{T}_{lc} \simeq \mathbb{T}_b$.

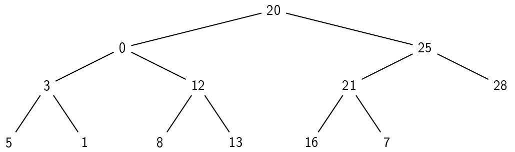
$$\varphi \ : \ \begin{cases} \mathbb{T}_b & \to & \mathbb{T}_{lc} \\ \texttt{Vide} & \mapsto & (f, \texttt{Vide}, \texttt{Vide}) \\ (r, T_1, T_2) & \mapsto & (r, \varphi(T_1), \varphi(T_2)) \end{cases}$$

$\hookrightarrow$ Arbre Parfait. Tous les niveaux hiérarchiques sont remplis sauf éventuellement le dernier, partiellement rempli de la gauche vers la droite.

$\hookrightarrow$ *Représentation* sous forme de tableau. Nœud d'indice i.

- fils d'indices $2i+1$ et $2i+2$.
- père d'indice $(i-1)/2$.

$\hookrightarrow$ Exemple.



`[|20;0;25;3;12;21;28;5;1;8;13;16;7|]`