

Éléments d'algorithmique : les arbres généraux

Cours 10

décembre 2020

Arbres généraux étiquetés (ou valués)

Soit T un type.

Opérations :

- ▶ $\text{Nœud} : T \times (\text{AV}(T))^* \rightarrow \text{AV}(T)$
- ▶ $\text{Forêt} : \text{AV}(T) \rightarrow (\text{AV}(T))^*$
- ▶ $\text{Val} : \text{AV}(T) \rightarrow T$


Axiomes :


- ▶ $\text{Forêt}(\text{Nœud}(v, [t_1, \dots, t_n])) = [t_1, \dots, t_n]$
- ▶ $\text{Val}(\text{Nœud}(v, [t_1, \dots, t_n])) = v$
- ▶ $\text{Nœud}(\text{Val}(t), \text{Forêt}(t)) = t$.

- ▶ $(\text{AV}(T))^*$ représente l'ensemble des listes d'arbres (qu'on appelle aussi **forêts**).
- ▶ On remarque qu'avec un tel formalisme, il n'y a pas d'arbre vide.
- ▶ Si $\text{Forêt}(t) = []$, on dit que t est une **feuille**.
- ▶ Les parcours préfixe, et postfixe (ou suffixe) sont définis de la même manière que pour les arbres binaires, en revanche, le parcours infixé n'est plus défini.

Arbres généraux : exemples

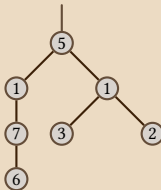
- Exemples -

► $\text{Nœud}(5, []) =$ 

► $\text{Nœud}(6, [\text{Nœud}(5, [])]) =$ 

► $\text{Val}\left(\begin{array}{c} \text{ } \\ | \\ \textcircled{1} \\ / \quad \backslash \\ \textcircled{3} \quad \textcircled{2} \end{array}\right) = 1$

► Forêt $\left(\begin{array}{c} \text{ } \\ | \\ \textcircled{5} \\ / \quad \backslash \\ \textcircled{1} \quad \textcircled{1} \\ | \quad \backslash \quad / \quad \backslash \\ \textcircled{7} \quad \textcircled{3} \quad \textcircled{2} \end{array} \right) = \left[\begin{array}{c} \text{ } \\ | \\ \textcircled{1} \\ | \\ \textcircled{7} \\ | \\ \textcircled{6} \end{array}, \begin{array}{c} \text{ } \\ | \\ \textcircled{1} \\ / \quad \backslash \\ \textcircled{3} \quad \textcircled{2} \end{array} \right]$

► $\text{Nœud}\left(5, \left[\begin{array}{c} \text{ } \\ | \\ \textcircled{1} \\ | \\ \textcircled{7} \\ | \\ \textcircled{6} \end{array}, \begin{array}{c} \text{ } \\ | \\ \textcircled{1} \\ / \quad \backslash \\ \textcircled{3} \quad \textcircled{2} \end{array} \right] \right) =$ 

Codage de Prüfer

Problème : le codage d'un arbre général par pointeurs (comme pour les listes, les piles, les files et les arbres binaires) est faisable mais complexe. Un arbre général serait en effet composée d'une valeur et d'une liste d'arbres binaires généraux (la forêt).

⇒ On cherche un codage plus simple et compact.

Algorithme du codage de Prüfer

- ▶ Entrée : un arbre général t à n nœuds valués bijectivement.
- ▶ Sortie : une liste d'entier de longueur $n - 1$.

```
res <- []
```

```
Tant que  $t$  n'est pas réduit à un seul nœud ( $\text{Forêt}(t) \neq []$ )
```

```
   $x$  <- étiquette minimum parmi les feuilles de  $t$ 
```

```
   $y$  <- étiquette du père de  $x$  dans  $t$ 
```

```
  res <- res .  $y$ 
```

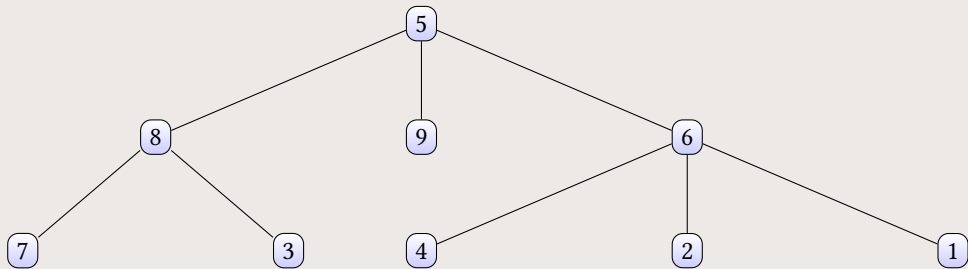
```
   $t$  <-  $t.\text{suppression-feuille}(x)$ 
```

```
Fin
```

```
renvoyer res
```

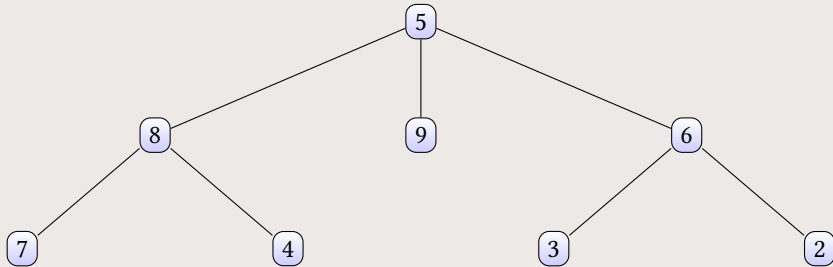
Exemple d'un codage de Prüfer

[]



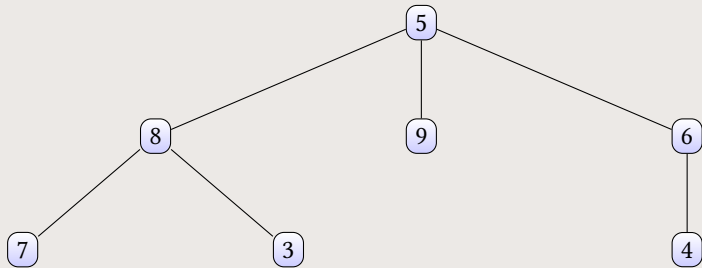
Exemple d'un codage de Prüfer

[6]



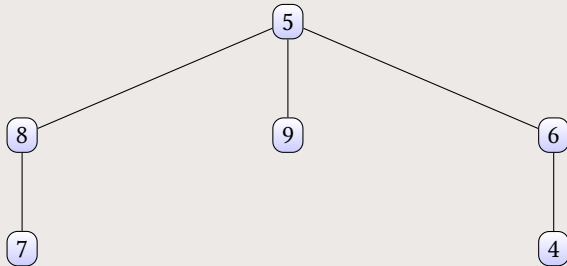
Exemple d'un codage de Prüfer

[6, 6]



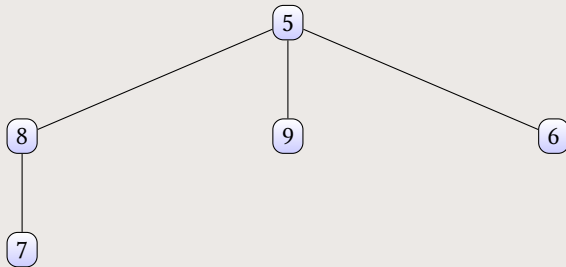
Exemple d'un codage de Prüfer

[6, 6, 8]



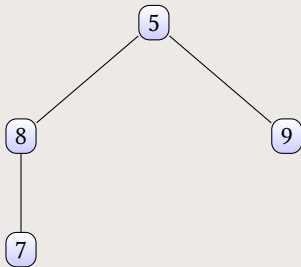
Exemple d'un codage de Prüfer

[6, 6, 8, 6]



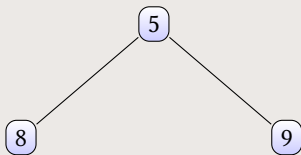
Exemple d'un codage de Prüfer

[6, 6, 8, 6, 5]



Exemple d'un codage de Prüfer

[6, 6, 8, 6, 5, 8]



Exemple d'un codage de Prüfer

[6, 6, 8, 6, 5, 8, 5]



Exemple d'un codage de Prüfer

[6, 6, 8, 6, 5, 8, 5, 5]

5

Question : Peut-on reconstruire l'arbre général à partir d'une liste ?

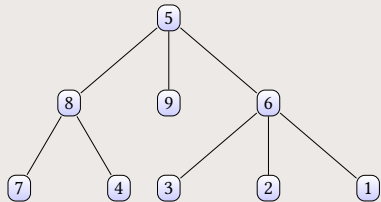
Transformation d'un arbre général en arbre binaire

Objectif : on veut construire une bijection f entre l'ensemble des arbres généraux et l'ensemble des arbres binaires.

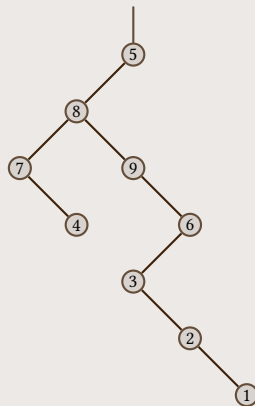
Soit t un arbre général, et soient x et y deux nœuds de t . On construit l'arbre binaire s correspondant de la façon suivante :

1. si y est **fil**s de x dans t alors y est le **fil**s **gauche** de x dans s ;
 2. si y est **frère droit** de x (frère immédiat) dans t alors y est le **fil**s **droit** de x dans s .
- Il s'agit d'une **bijection** entre l'ensemble des arbres généraux et l'ensemble des arbres binaires sans sous-arbre droit ;
 - la **racine** des deux arbres est la même.

Transformation d'un arbre général en arbre binaire : exemple



\xrightarrow{f}



Avantages de la bijection f

⇒ Soit s un arbre binaire sans sous-arbre droit. On peut reconstruire l'arbre général correspondant en changeant les implications de la construction arbres généraux \rightarrow arbres binaires sans sous-arbre droit.

⇒ L'arbre binaire obtenu est plus compact dans le sens où on n'a pas besoin de liste pour représenter les forêts.

⇒ Pour tous arbres généraux t , on a préservation par f du parcours préfixe :

$$\text{parcours-préfixe}(t) = \text{parcours-préfixe}(f(t)).$$

⇒ Pour tous arbres généraux t , le résultat de parcours suffixe (postfixe) de t par f est le même que le résultat du parcours infixe de $f(t)$:

$$\text{parcours-suffixe}(t) = \text{parcours-infixe}(f(t)).$$