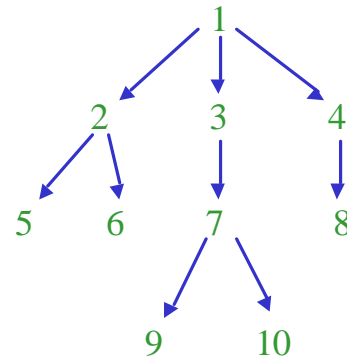


ARBRES

UMLV©

Arbre ordinaire : $A = (N, P)$

- N ensemble des nœuds
- P relation binaire « parent de »
- $r \in N$ la racine



$\forall x \in N \exists$ un seul chemin de r vers x

$$r = y_0 P y_1 P y_2 \dots P y_n = x$$

$\Rightarrow r$ n'a pas de parent

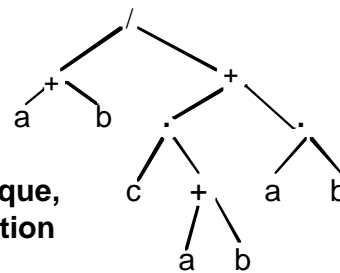
$\Rightarrow \forall x \in N - \{r\}$ x a exactement un parent

151

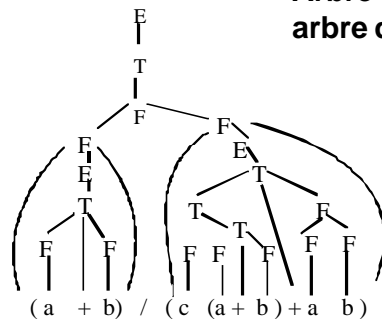
Étiquettes

UMLV©

Étiquette : $N \rightarrow E$



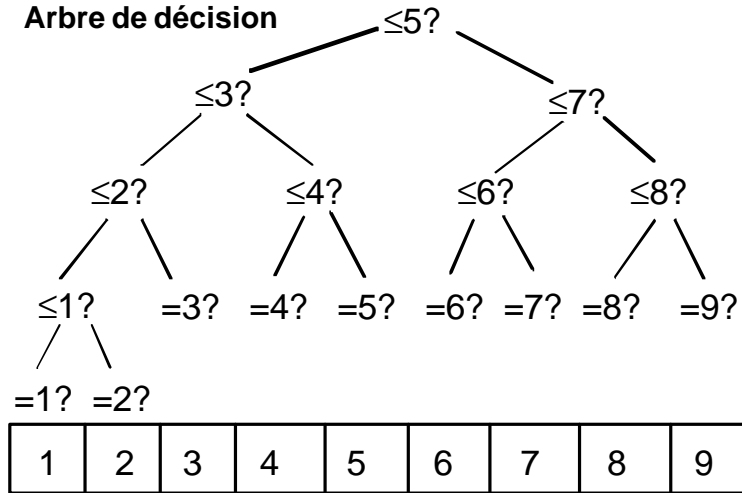
Arbre syntaxique,
arbre d'exécution



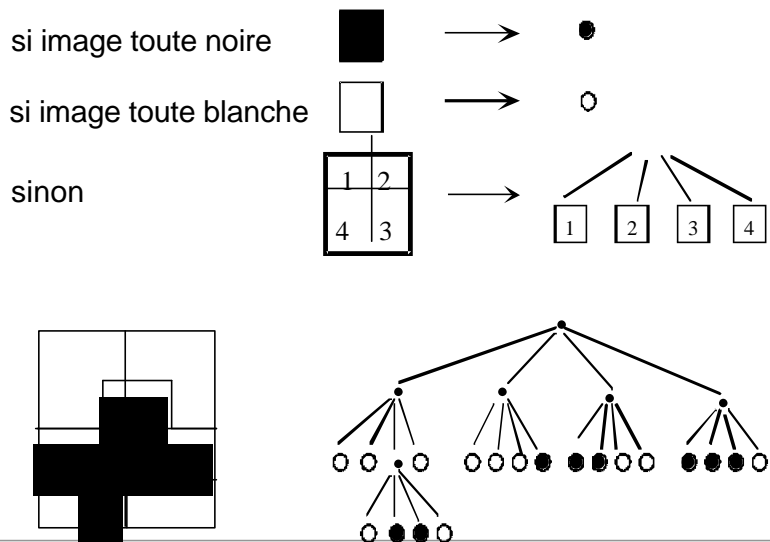
Arbre d'analyse,
pour une grammaire

152

Arbre de décision



Arbre quartique

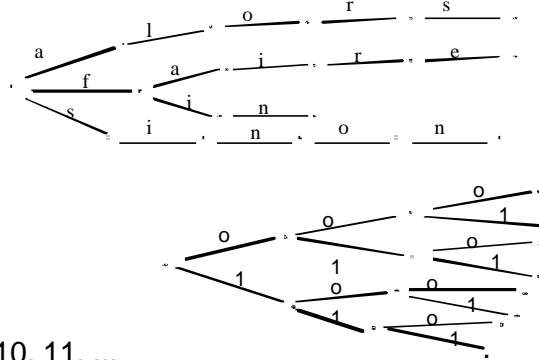


Arbres lexicographiques

UMLV©

Arcs $(A) = \{ (x,y) / x \text{ parent de } y \}$

Étiquette : Arcs $(A) \rightarrow E$

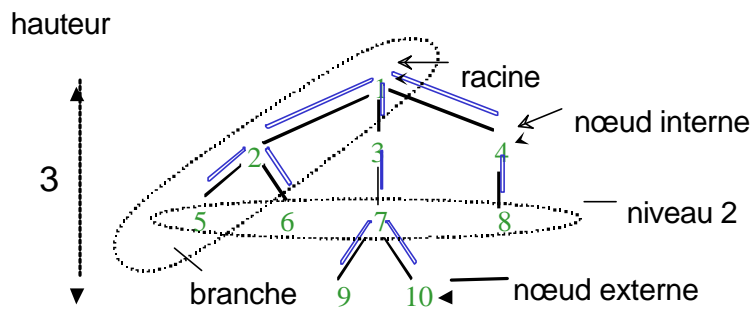


$\epsilon, 0, 1, 00, 01, 10, 11, \dots$
 0 1 2 3 4 5 6 ...

155

Terminologie

UMLV©



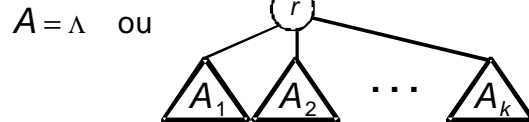
- 2, 3, 4 enfants de 1
- 3, 4 frères de 2
- 1, 3, 7 ancêtres de 7
- 7, 9, 10 descendants de 7

156

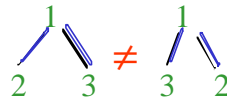
Définitions récursives

UMLV©

Arbre $A = \begin{cases} \Lambda & \text{arbre vide ou} \\ (r, \{A_1, \dots, A_k\}) & r \text{ élément, } A_1, \dots, A_k \text{ arbres} \end{cases}$
 $\text{Nœuds}(A) = \{r\} \cup (\cup \text{Nœuds}(A_i))$
 unions disjointes



Arbre planaire $A = \begin{cases} \Lambda & \text{arbre vide ou} \\ (r, A_1, \dots, A_k) & \end{cases}$ Condition analogue



157

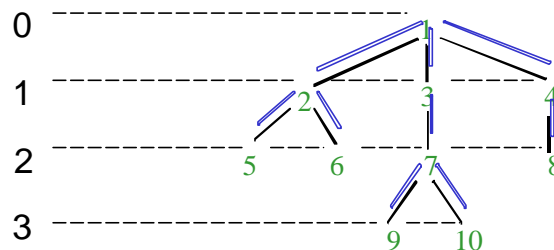
Niveaux

UMLV©

A arbre x nœud de A

$\text{niveau}_A(x) = \text{distance de } x \text{ à la racine}$

$\text{niveau}_A(x) = \begin{cases} 0 & \text{si } x = \text{racine}(A) \\ 1 + \text{niveau}(\text{parent}(x)) & \text{sinon} \end{cases}$



158

Hauteurs

UMLV©

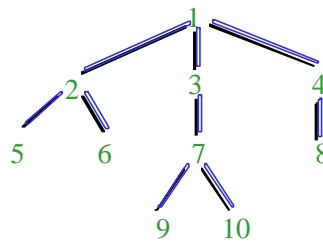
A arbre x nœud de A

$h_A(x)$ = distance de x à son plus lointain descendant
qui est un nœud externe

$$h_A(x) = \begin{cases} 0 & \text{si } x \text{ nœud externe} \\ 1 + \max \{ h_A(e) \mid e \text{ enfant de } x \} & \text{sinon} \end{cases}$$

$$h(A) = h_A(\text{racine}(A))$$

$$\begin{aligned} h_A(8) &= 0 \\ h_A(7) &= 1 \\ h_A(3) &= 2 \\ h(A) = h_A(1) &= 3 \end{aligned}$$



159

Sous-arbres

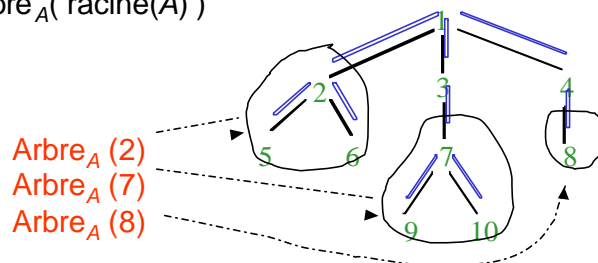
UMLV©

A arbre x nœud de A

$\text{Arbre}_A(x)$ = sous-arbre de A qui a racine x

$$h_A(x) = h(\text{Arbre}_A(x))$$

$$A = \text{Arbre}_A(\text{racine}(A))$$



160

Parcours

UMLV©

Fonction : arbre → liste de ses nœuds
arbre vide → liste vide

Utile pour l'exploration des arbres

Deux types :

parcours en profondeur

préfixe, suffixe, symétrique

parcours branche après branche

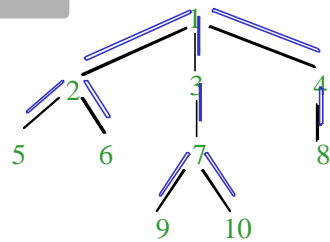
parcours en largeur ou hiérarchique

parcours niveau après niveau

161

Parcours en profondeur

UMLV©



Arbre non vide $A = (r, A_1, A_2, \dots, A_k)$

Parcours préfixe

$P(A) = (r).P(A_1). \dots .P(A_k)$

(1, 2, 5, 6, 3, 7, 9, 10, 4, 8)

Parcours suffixe

$S(A) = S(A_1). \dots .S(A_k).(r)$

(5, 6, 2, 9, 10, 7, 3, 8, 4, 1)

Parcours symétrique (ou interne)

$I(A) = I(A_1).(r).I(A_2). \dots .I(A_k)$

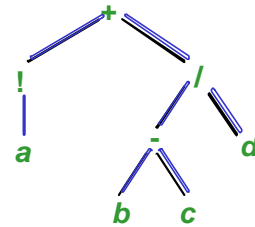
(5, 2, 6, 1, 9, 7, 10, 3, 8, 4)

162

Expressions arithmétiques

UMLV©

Arbre syntaxique de $a! + \frac{b - c}{d}$



Parcours préfixe

$+ ! a / - b c d$

Parcours suffixe

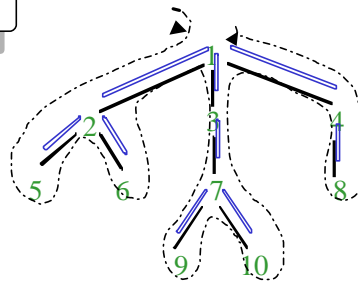
$a ! b c - d / +$

Parcours symétrique (priorités et parenthèses)

$(a !) + ((b - c) / d)$

Rencontres

UMLV©



Parcours préfixe = première rencontre

$(1, 2, 5, 6, 3, 7, 9, 10, 4, 8)$

Parcours Suffixe = dernière rencontre

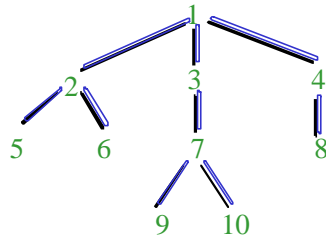
$(5, 6, 2, 9, 10, 7, 3, 8, 4, 1)$

Parcours Symétrique = deuxième rencontre

$(5, 2, 6, 1, 9, 7, 10, 3, 8, 4)$

Parcours en largeur

UMLV©



Arbre non vide $A = (r, A_1, A_2, \dots, A_k)$

Parcours hiérarchique

$$H(A) = (r, \underline{x_1}, \dots, \underline{x_j}, \underline{x_{j+1}}, \dots, \underline{x_j}, \underline{x_{j+1}}, \dots, x_n)$$

nœuds de niveau 0, 1, 2, ...

(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

165

Type arbre

UMLV©

Ensemble

Arbres planaires de nœuds étiquetés

Opérations

Arbre-vide : \mathbb{R} arbre

Racine : arbre \mathbb{R} nœud

Enfants : arbre \mathbb{R} liste d'arbres

Cons : nœud \times liste d'arbres \mathbb{R} arbre

Vide : arbre \mathbb{R} booléen

Elt : nœud \mathbb{R} élément

Axiomes principaux

Racine(A) et Enfants(A) définis ssi A non vide

Racine (Cons(r, L)) = r

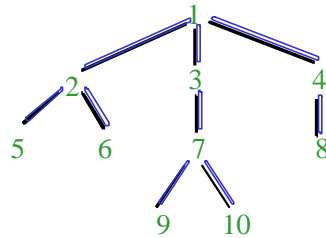
Enfants(Cons(r, L)) = L

166

Implémentation

UMLV©

Représentation de la relation P
 table des parents



Avantages

représentation simple
 parcours faciles vers la racine
 économique en mémoire

Inconvénients

accès difficiles aux nœuds
 depuis la racine

table des parents P

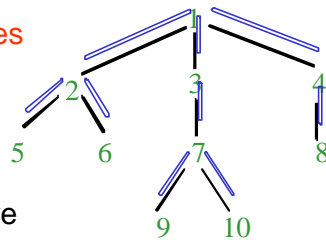
-	1	1	1	2	2	3	4	7	7
1	2	3	4	5	6	7	8	9	10

167

Implémentation (suite)

UMLV©

Représentation des listes de sous-arbres
 par chaînage



Avantages

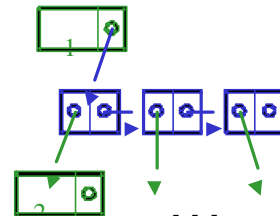
accès faciles depuis la racine
 correspond à la définition récursive

Inconvénients

parcours difficiles vers la racine
 relativement gourmand en mémoire

Deux types de pointeurs

Arbre souvent identifié à l'adresse de
 sa racine (comme pour un tableau en C)



168

Fonction Préfixe

UMLV©

```
fonction Préfixe (A arbre) : liste de nœuds ;  
début  
  si A = arbre vide alors retour (suite vide)  
  sinon {  
    L ← ( Racine (A) ) ;  
    pour B ← premier au dernier élément de Enfants(A) faire  
      L ← L . Préfixe(B) ;  
    retour ( L ) ;  
  }  
fin
```

Temps d'exécution : $O(n)$
sur un arbre à n nœuds représenté par pointeurs

169

Fonction suffixe

UMLV©

```
fonction Suffixe (A arbre) : liste de nœuds ;  
début  
  si A = arbre vide alors retour (suite vide)  
  sinon {  
    L ← ( ) ;  
    pour B ← premier au dernier élément de Enfants(A) faire  
      L ← L . Suffixe(B) ;  
    L ← L . (racine(A) ) ;  
    retour ( L ) ;  
  }  
fin
```

Temps d'exécution : $O(n)$
sur un arbre à n nœuds représenté par pointeurs

170

Fonction Préfixe itérative

UMLV©

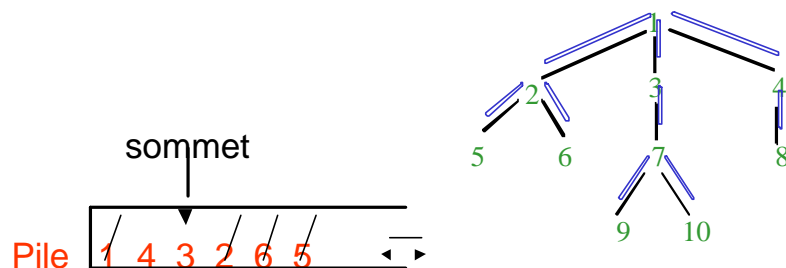
```
fonction Préfixe (A arbre) : liste de nœuds ;
début
  L ← ( ) ;
  Pile ← Empiler( Pile-vide, A ) ;
  tant que non Vide( Pile ) faire {
    A' ← sommet ( Pile ) ; Pile ← Dépiler ( Pile ) ;
    si A' non vide alors {
      L ← L . (racine(A) ) ;
      pour B ← dernier au premier élément de Enfants(A) faire
        Pile ← Empiler ( Pile, B ) ;
    }
  }
  retour ( L ) ;
fin
```

Temps d'exécution $O(n)$ comme version récursive
si, en plus, bonne implémentation de la pile

171

Exemple

UMLV©



Liste $L = (1, 2, 5, 6, .$

172

Fonction Niveaux

UMLV©

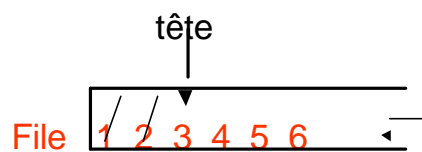
```
fonction Niveaux (A arbre) : liste de nœuds ;  
début  
  L ← ( ) ;  
  File ← Enfiler( File-vide, A ) ;  
  tant que non Vide( File) faire {  
    A' ← tête( File) ; File ← Enlever( File ) ;  
    si A' non vide alors {  
      L ← L . (racine(A)) ;  
      pour B ← premier au dernier élément de Enfants(A) faire  
        File ← Ajouter( File, B ) ;  
    }  
  }  
  retour ( L ) ;  
fin
```

Temps d'exécution $O(n)$ si bonne implémentation de la file
pas de version récursive

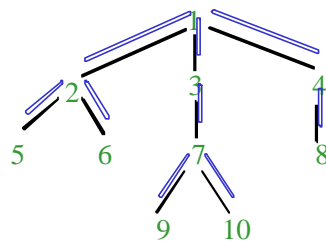
173

Exemple

UMLV©



Liste $L = (1, 2, .$



174

Arbres k-aires

UMLV©

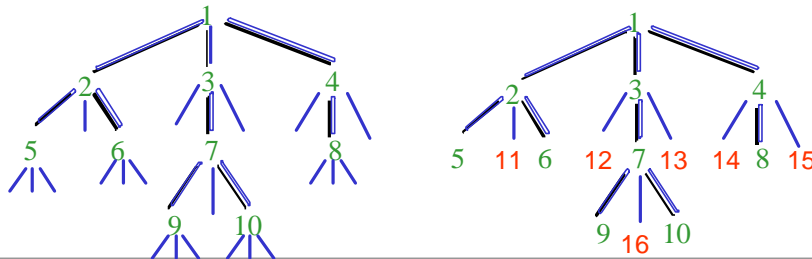
Arbre k-aire : tout nœud possède k sous-arbres (vides ou non), k fixé

$$A = \begin{cases} \Lambda & \text{arbre vide ou} \\ (r, A_1, \dots, A_k) & r \text{ élément, } A_1, \dots, A_k \text{ arbres k-aires} \end{cases}$$

$$\text{Nœuds}(A) = \{r\} \cup (\cup \text{Nœuds}(A_i))$$

unions disjointes

Arbre k-aire complet : tout nœud interne possède k enfants



175

Arbres binaires

UMLV©

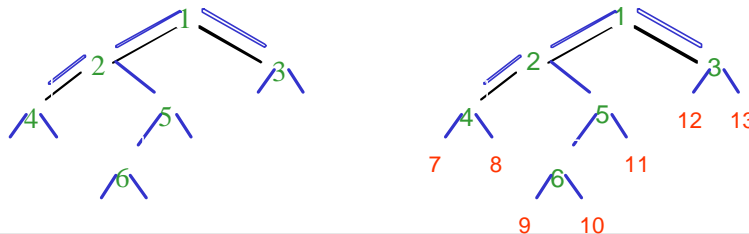
Arbre binaire : tout nœud possède deux sous-arbres (vides ou non)

$$A = \begin{cases} \Lambda & \text{arbre vide ou} \\ (r, G, D) & r \text{ élément, } G, D \text{ arbres binaires} \end{cases}$$

$$\text{Nœuds}(A) = \{r\} \cup \text{Nœuds}(G) \cup \text{Nœuds}(D)$$

unions disjointes

Arbre binaire complet : tout nœud interne possède deux enfants



176

Type arbre binaire

UMLV©

Ensemble

Arbres binaires étiquetés

Opérations

Arbre-vide : $\text{\textcircled{R}}$ arbre

Racine : arbre $\text{\textcircled{R}}$ nœud

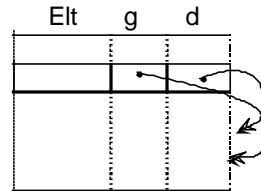
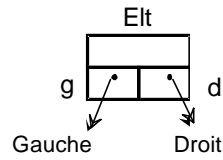
Gauche, Droit : arbre $\text{\textcircled{R}}$ arbre

Cons : nœud \times arbre \times arbre $\text{\textcircled{R}}$ arbre

Elt : nœud $\text{\textcircled{R}}$ élément

Vide : arbre $\text{\textcircled{R}}$ booléen

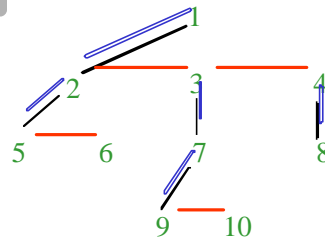
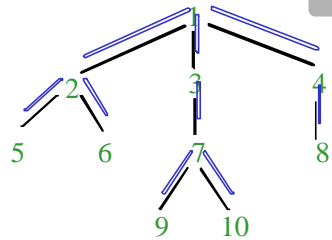
Implémentations par pointeurs ou curseurs



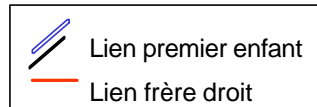
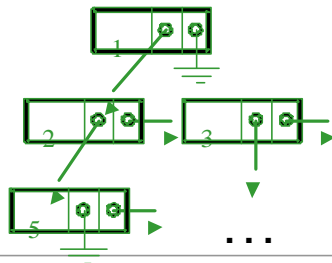
177

Binarisation

UMLV©



Implémentation des arbres planaires



Un seul type de pointeur

178

Arbres feuillus

UMLV©

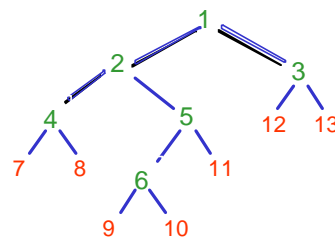
Arbre binaire feuillus (complets) : deux types de nœuds, internes ou feuilles
-- tout nœud interne possède deux enfants ;
-- toute feuille est un nœud externe.

$$A = \begin{cases} (f) & f \text{ de type feuille} \\ (r, G, D) & r \text{ de type interne, } G, D \text{ arbres binaires feuillus} \end{cases}$$

Nœuds (A) = $\{r\} \cup \text{Nœuds}(G) \cup \text{Nœuds}(D)$ unions disjointes

Nœuds internes : 1, 2, 3, 4, 5, 6

Feuilles : 7, 8, 9, 10, 11, 12, 13



179

Taille des arbres feuillus

UMLV©

Arbre feuillu : nombre de feuilles = nombre de nœuds internes + 1

Récurrance sur le nombre de nœuds de l'arbre A :

- si un seul nœud, c'est une feuille ; propriété satisfaite.
- sinon, il existe un nœud interne dont les enfants sont des feuilles, *i.e.* un sous-arbre (x, g, d) où g, d sont des feuilles.
Soit B obtenu de A en remplaçant (x, g, d) par une feuille f .
 B est un arbre feuillu de plus petite taille ; par récurrence l'égalité est satisfaite sur B ; donc aussi sur A qui possède un nœud interne et une feuille de plus. CQFD.

Arbre feuillu : $2.m + 1$ nœuds

180

Mesures des arbres binaires

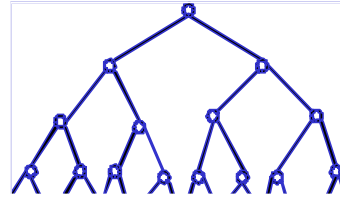
UMLV©

Arbre binaire, hauteur h et n nœuds

Arbre plein

2^i nœuds au niveau i

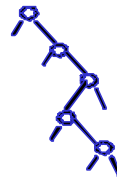
$$n = 2^{h+1} - 1$$



Arbre filiforme

1 nœud par niveau

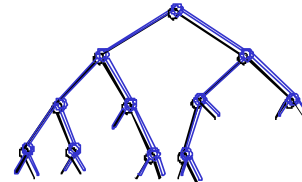
$$n = h + 1$$



Arbre binaire

$$h + 1 \leq n \leq 2^{h+1} - 1$$

$$\log_2(n + 1) - 1 \leq h \leq n - 1$$

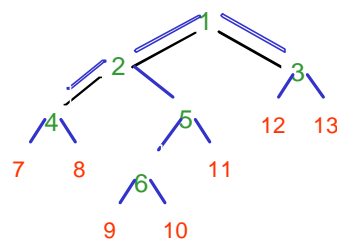
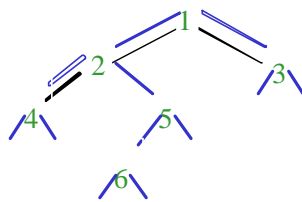


181

Bijection

UMLV©

Arbres binaires à n nœuds \leftrightarrow arbres binaires complets à $2.n+1$ nœuds

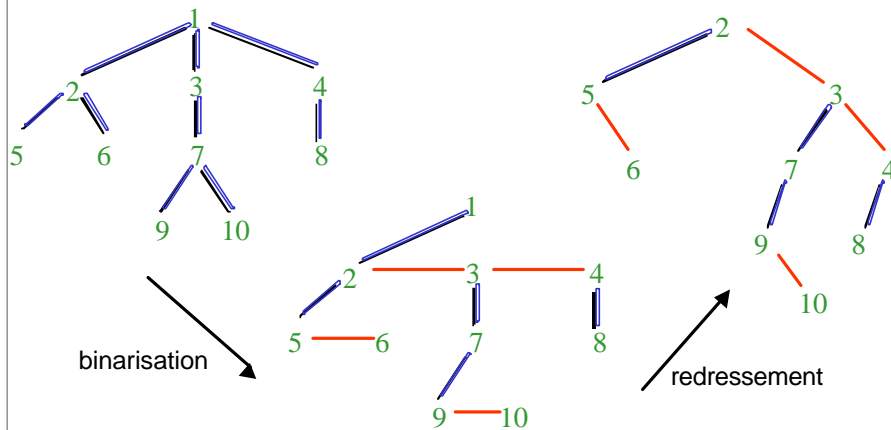


182

Autre bijection

UMLV©

Arbres planaires à $n+1$ nœuds \leftrightarrow arbres binaires à n nœuds



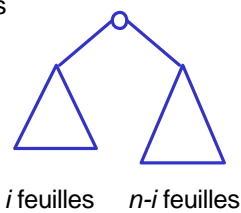
183

Énumération

UMLV©

C_n = nombre d'arbres binaires complets à n feuilles

$$\begin{cases} C_0 = 0, & C_1 = 1 \\ C_n = \sum_{i=1}^{n-1} C_i C_{n-i} & n \geq 2 \end{cases}$$



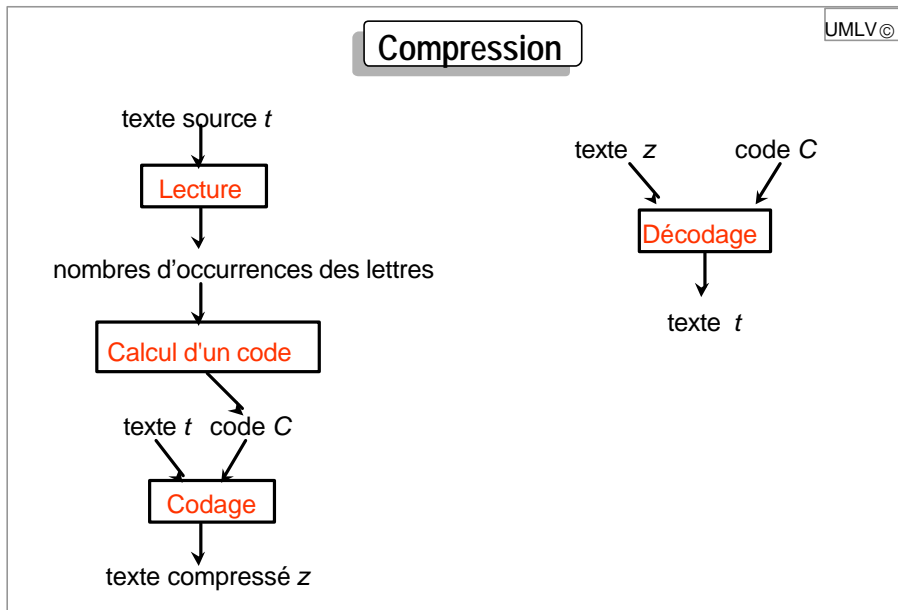
Série génératrice : $s(t) = \sum_{n \geq 1} C_n t^n$

$$s(t) = s(t)^2 + t \Rightarrow s(t) = \frac{1 - \sqrt{1 - 4t}}{2}$$

$$\sqrt{1 - 4t} = \sum_{n \geq 0} (-1)^n 4^n t^n \frac{1/2 (1/2 - 1) \dots (1/2 - n + 1)}{n!}$$

$$\Rightarrow C_n = \frac{1}{2n-1} \binom{2n-1}{n} = \frac{(2n-2)!}{n!(n-1)!} \quad \text{nombres de catalan}$$

184



185

Codes Préfixes

UMLV©

$C \subseteq \{0,1\}^*$
 C code préfixe ssi $u, v \in C$ et u préfixe de $v \Rightarrow u = v$
 aucun mot de C n'est un préfixe d'un autre mot de C

$\{00,01,10,11\}$ $a \ b \ c \ d$ code ASCII	$\{0,10,11\}$ $a \ b \ c$ 0^*1
--	--

décodage unique et séquentiel de $x \in C^*$ (avec $C = 0^*1$)

$0 \ 0 \ 1 \ | \ 1 \ | \ 00001 \ | \ 01 \ | \ 1 \ | \ 1 \ | \ 001 \ | \ \dots$

186

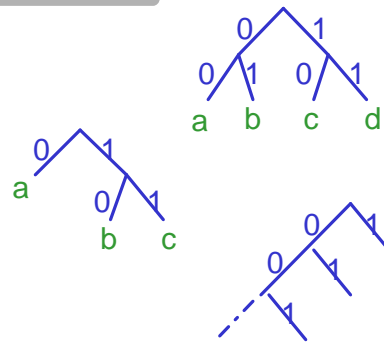
Bijection

UMLV©

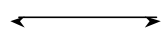
{ 00,01,10,11 }
a b c d

{ 0,10,11 }
a b c

0*1



Code préfixe



Arbre binaire

mot du code

nœud externe

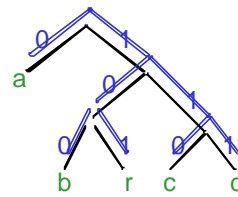
187

Compression statistique

UMLV©

Codage des caractères en fonction de leur fréquence

t = a b r a c a d a b r a
 a 5 fois 0
 b 2 fois 100
 c 1 fois 110
 d 1 fois 111
 r 2 fois 101



Texte codé

z = 0 1 0 0 1 0 1 0 1 1 0 0 1 1 1 0 1 0 0 1 0 1 0
 23 bits (si on ne compte pas le code) contre 11 x 8 en ASCII

Décodage

0 | 100 | 101 | 0 | 110 | 0 | 111 | 0 | 100 | 101 | 0 |
 a | b | r | a | c | a | d | a | b | r | a

188

Arbres pondérés

UMLV©

t texte initial, z texte compressé

$$p(a) = \text{nombre d'occurrences de } a \text{ dans } t \quad |z| = \sum_{a \in A} p(a) \cdot |h(a)|$$

$h(a)$ = mot du code associé à a

Problème :

connaissant $p : A \rightarrow \mathbb{N} - \{0\}$

calculer $h : A \rightarrow \{0,1\}^*$ tel que

- $h(A)$ est un code préfixe et
- $\sum p(a) \cdot |h(a)|$ minimal

Problème équivalent :

déterminer un A arbre binaire feuillu (complet) dont les feuilles sont les lettres de A , avec $p : \text{Feuilles}(A) \rightarrow \mathbb{N} - \{0\}$, tel que

$$\text{Poids}(A) = \sum_{f \text{ feuille de } A} p(f) \cdot \text{niveau}(f) \text{ est minimal}$$

189

Arbres de Huffman

UMLV©

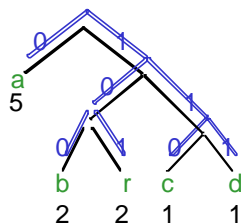
Arbre pondéré :

A arbre binaire feuillu (complet) dont les feuilles sont les lettres de A

$p : \text{Feuilles}(A) \rightarrow \mathbb{N} - \{0\}$

$$\text{Poids}(A) = \sum_{f \text{ feuille de } A} p(f) \cdot \text{niveau}(f)$$

A **arbre de Huffman** (pour p) si $\text{Poids}(A)$ minimal



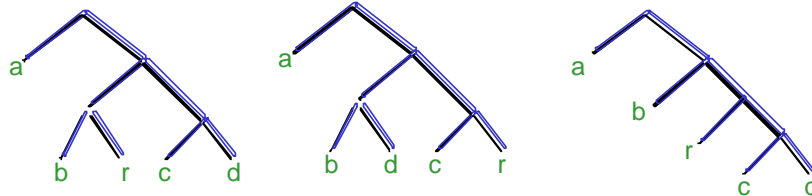
$$\text{Poids}(A) = \text{longueur du texte compressé} = 5 \times 1 + 2 \times 3 + 2 \times 3 + 1 \times 3 + 1 \times 3 = 23$$

190

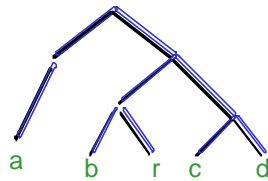
Exemple (suite)

UMLV©

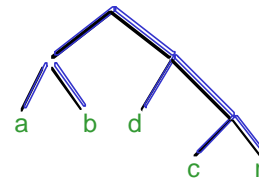
Trois arbres de Huffman (Poids = 23)



Poids = 28



Poids = 25



191

Propriétés

UMLV©

A arbre pondéré pour $p : A \rightarrow \mathbb{N} - \{0\}$; $A = \text{Feuilles}(A)$

1 - A arbre de Huffman $\Leftrightarrow A$ arbre binaire complet

2 - A arbre de Huffman et $\text{card } A > 1$

Alors, à une permutation des feuilles près,
 A possède un sous-arbre (x, f, g) où f, g sont des feuilles
 telles que $p(f), p(g)$ minimaux



3 - Soit B obtenu de A en y remplaçant (x, f, g) par la feuille y ;

soit $q : \text{Feuilles}(B) \rightarrow \mathbb{N} - \{0\}$ définie par

$$\begin{cases} q(y) = p(f) + p(g) \\ q(u) = p(u) \quad \text{pour } u \neq y \end{cases}$$

Alors, A arbre de Huffman pour p ssi B arbre de Huffman pour q



Preuve : car $\text{Poids}(A) = \text{Poids}(B) + p(f) + p(g)$

192

Algorithme de Huffman

UMLV©

$p: A \rightarrow \mathbb{N} - \{0\}$

arbre élémentaire $X_a = (x_a)$, pour chaque lettre $a \in A$

fonction Huffman (alphabet A , fonction p)

début

$L \leftarrow (X_a \mid a \in A)$;

tant que $|L| > 1$ **faire** {

$B, C \leftarrow$ arbres de L avec $p(B), p(C)$ minimaux ;

$Y \leftarrow$ (nouveau-nœud, B, C) ; $p(Y) \leftarrow p(B) + p(C)$;

remplacer B et C dans L par Y ;

}

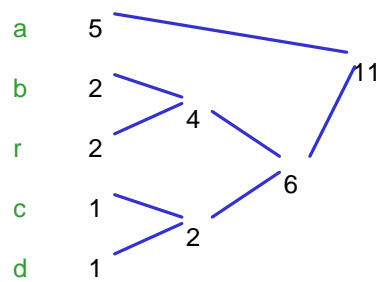
retour l'unique arbre de L ;

fin

193

Exemple

UMLV©



Arbre de Huffman

a 0
b 100
r 101
c 110
d 111

Code de Huffman

194

Implémentation

UMLV©

		<i>p</i>	<i>g</i>	<i>d</i>
1	a	5	-	-
2	b	2	-	-
3	r	2	-	-
4	c	1	-	-
5	d	1	-	-
6	-	2	4	5
7	-	4	2	3
8	-	6	7	6
9	-	11	1	8

Table de taille $2 \cdot \text{card } A - 1$

Calcul : tri en temps $O(\text{card } A \times \log \text{card } A)$
 construction de l'arbre en $O(\text{card } A)$

