

ARBRES

1. Arbres généraux

L'arbre vide est noté Λ .

Soit A un arbre non vide de racine r ayant k sous-arbres A_1, A_2, \dots, A_k .

Parcours préfixe P

$$P(\Lambda) = \Lambda$$

$$P(A) = rP(A_1)P(A_2) \cdots P(A_k)$$

Parcours interne I

$$I(\Lambda) = \Lambda$$

$$I(A) = I(A_1)rI(A_2) \cdots I(A_k)$$

Parcours suffixe S

$$S(\Lambda) = \Lambda$$

$$S(A) = S(A_1)S(A_2) \cdots S(A_k)r$$

Remarque : si l'arbre est réduit à une feuille f , $P(f) = I(f) = S(f) = f$.

Implantation par chaînage FilsGauche - FrereDroit

Un nœud est une cellule contenant un champ *FilsGauche*, un champ *Etiquette* et un champ *FrereDroit*. Un arbre est un pointeur sur une cellule de ce type.

Si n est un nœud, $n \rightarrow \text{FilsGauche}$ pointe sur le fils aîné de n , $n \rightarrow \text{FrereDroit}$ pointe sur le frère de n qui suit immédiatement n , et $n \rightarrow \text{Etiquette}$ contient l'étiquette (ou la valeur) de n .

Algorithme 1 Fonction *ParcoursPrefixe*(n)

```
si  $n \neq \text{NULL}$  alors
  Afficher( $n \rightarrow \text{Etiquette}$ );
   $p := n \rightarrow \text{FilsGauche}$ ;
  tant que  $p \neq \text{NULL}$  faire
    ParcoursPrefixe( $p$ );
     $p := p \rightarrow \text{FrereDroit}$ 
  fin tant que
fin si
```

Appel avec *ParcoursPrefixe*(A).

Parcours par niveau

On utilise une queue de pointeurs Q , et les fonctions définies dans le cours sur les queues.

Algorithme 2 Fonction ParcoursNiveau(A)

```
si  $A \neq \text{NULL}$  alors
  Initialiser( $Q$ );
  AjouterQueue( $A, Q$ )
  tant que non Vide( $Q$ ) faire
     $p := \text{Tete}(Q)$ ;
    EnlQueue( $Q$ );
    Afficher( $p \rightarrow \text{Etiquette}$ );
     $p := p \rightarrow \text{FilsGauche}$ ;
    tant que  $p \neq \text{NULL}$  faire
      AjouterQueue( $p, Q$ );
       $p := p \rightarrow \text{FrereDroit}$ 
    fin tant que
  fin tant que
fin si
```

2. Arbres binaires

Dans tout ce qui suit A est un arbre binaire.

Implantation par chaînage fils gauche - fils droit

Un nœud est une cellule contenant un champ FG, un champ Etiquette et un champ FD. Un arbre est un pointeur sur une cellule de ce type.

Si n est un nœud, $n \rightarrow FG$ pointe sur le fils gauche de n , $n \rightarrow FD$ pointe sur le fils droit, et $n \rightarrow \text{Etiquette}$ contient l'étiquette (ou la valeur) de n .

Algorithme 3 Fonction ParcoursPrefixe(n)

```
si  $n \neq \text{NULL}$  alors
  Afficher( $n \rightarrow \text{Etiquette}$ );
  ParcoursPrefixe( $n \rightarrow FG$ );
  ParcoursPrefixe( $n \rightarrow FD$ )
fin si
```

Appel avec ParcoursPrefixe(A).

Algorithme 4 Fonction ParcoursInterne(n)

```
si  $n \neq \text{NULL}$  alors
  ParcoursInterne( $n \rightarrow FG$ );
  Afficher( $n \rightarrow \text{Etiquette}$ );
  ParcoursInterne( $n \rightarrow FD$ )
fin si
```

Appel avec ParcoursInterne(A).

Algorithme 5 Fonction `ParcoursSuffixe(n)

---`

H

```
si  $n \neq \text{NULL}$  alors
  ParcoursSuffixe( $n \rightarrow FG$ );
  ParcoursSuffixe( $n \rightarrow FD$ );
  Afficher( $n \rightarrow Etiquette$ )
fin si
```

Appel avec `ParcoursSuffixe(A)`.

Parcours par niveau

On utilise une queue de pointeurs Q , et les fonctions définies dans le cours sur les queues.

Algorithme 6 Fonction `ParcoursNiveau(A)

---`

```
si  $A \neq \text{NULL}$  alors
  Initialiser( $Q$ );
  AjouterQueue( $A, Q$ )
  tant que non Vide( $Q$ ) faire
     $p := \text{Tete}(Q)$ ;
    EnlQueue( $Q$ );
    Afficher( $p \rightarrow Etiquette$ );
    si  $p \rightarrow FG \neq \text{NULL}$  alors
      AjouterQueue( $p \rightarrow FG, Q$ );
    fin si
    si  $p \rightarrow FD \neq \text{NULL}$  alors
      AjouterQueue( $p \rightarrow FD, Q$ );
    fin si
  fin tant que
fin si
```

Hauteur d'un arbre binaire

Algorithme 7 Fonction `Hauteur(A)

---`

```
si  $A = \text{NULL}$  alors
  retourner  $(-1)$ 
sinon
  retourner  $(1 + \max(\text{Hauteur}(A \rightarrow FG), \text{Hauteur}(A \rightarrow FD)))$ 
fin si
```
