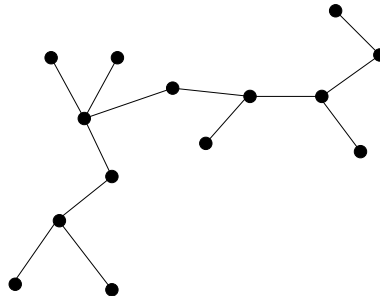


Arbres

1 Généralités

Définition graphe non orienté $\begin{cases} \text{connexe} \\ \text{sans cycle} \end{cases}$ arbre T (*tree*)



$\begin{cases} \text{non connexe} \\ \text{sans cycle} \end{cases}$... plusieurs arbres ... forêt

$\begin{cases} \text{connexe} \\ \text{avec cycle(s)} \end{cases}$ pas arbre

$\begin{cases} \text{non connexe} \\ \text{avec cycle(s)} \end{cases}$ pas arbre

sommet de degré 1 ... au bout d'une branche ... feuille

Construction

- Un graphe avec un seul sommet et zéro arête est un arbre
- À partir d'un arbre avec au moins 1 sommet
 - ★ On accroche une feuille n'importe où
 - ★ On obtient un arbre avec un sommet et une arête de plus

Mise en oeuvre

Exercice 1.

1. Trouvez tous les arbres à 4 sommets à isomorphisme près.
2. Trouvez tous les arbres à 5 sommets à isomorphisme près.

Remarque.

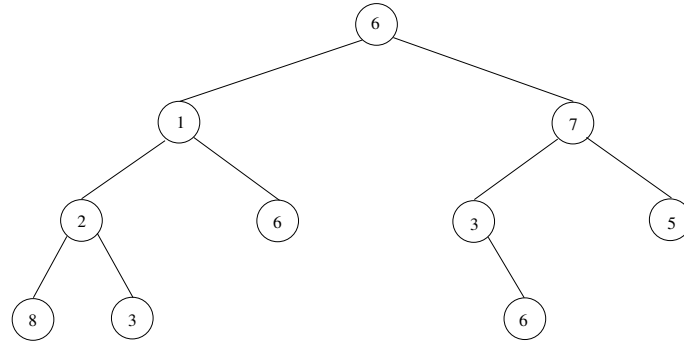
1. Un arbre à n sommets possède exactement $n - 1$ arêtes.
2. Un arbre avec au moins deux sommets a forcément au moins une feuille.

Propriétés caractéristiques

1. deux sommets quelconques d'un arbre sont reliés par une unique chaîne
2. enlever une arête quelconque à un arbre le déconnecte
3. ajouter une arête quelconque à un arbre crée un cycle

Remarque. Un graphe ayant ces propriétés est forcément un arbre.

Arbre enraciné

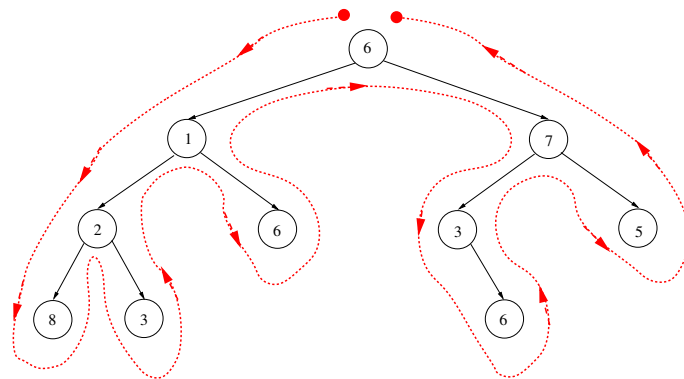


- Racine
- Orientation
- Père et fils
- Fils droit / Fils gauche
- Ancêtre et descendant
- Sous-arbre
- Structure « naturellement » récursive

On se limite au cas des arbres binaires Chaque nœud a au plus deux fils

2 Parcours d'arbres en profondeur

Parcourir les nœuds en suivant les arcs



Principe

- Départ : Racine
- Descendre, descendre...
- Remonter, redescendre...
- Etc.
- Remonter, remonter...
- Arrivée : Racine

Trois variantes

- Parcours préfixe
- Parcours postfixe
- Parcours infixe

Parcours préfixe

Principe

On traite d'abord la racine, puis on parcourt récursivement le sous-arbre gauche et le sous-arbre droit

Illustration

$6 \rightarrow 1 \rightarrow 2 \rightarrow 8 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 6 \rightarrow 5$

Algorithme

Fonction $\text{Pref}(A : n\text{œud})$
début si $A \neq \text{NULL}$ alors Traiter(A) Pref (FilsGauche(A)) Pref (FilsDroit(A)) fin fin

Parcours postfixe

Principe

On parcourt récursivement le sous-arbre gauche et le sous-arbre droit, puis on traite la racine

Illustration

$8 \rightarrow 3 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 7 \rightarrow 6$

Algorithme

Fonction $\text{Postf}(A : n\text{œud})$
début si $A \neq \text{NULL}$ alors Postf (FilsGauche(A)) Postf (FilsDroit(A)) Traiter(A) fin fin

Parcours infixe

Principe

On parcourt récursivement le sous-arbre gauche, puis on traite la racine, et ensuite on parcourt récursivement le sous-arbre droit

Illustration

$8 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 6 \rightarrow 6 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 5$

Algorithme

Fonction $\text{Inf}(A : \text{nœud})$

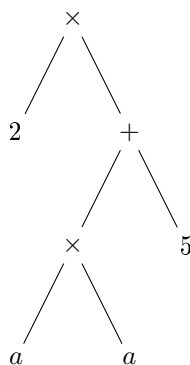
```

début
  | si  $A \neq \text{NULL}$  alors
  |   |  $\text{Inf}(\text{FilsGauche}(A))$ 
  |   |  $\text{Traiter}(A)$ 
  |   |  $\text{Inf}(\text{FilsDroit}(A))$ 
  |   fin
fin

```

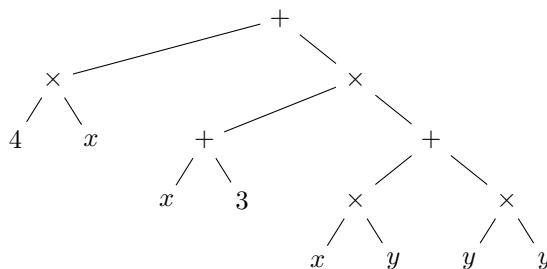
Arbres syntaxiques

- On associe à l'expression $2(a^2 + 5)$ l'arbre ci-dessous :
- À toute expression mathématique, on associe un arbre enraciné qui traduit la construction récursive de l'expression.
- Les feuilles représentent les nombres ou les variables et les nœuds intérieurs représentent les opérations.



Exercice 2.

1. Construire l'arbre associé à l'expression : $(x + y)^2 + (2x + 1)(y + 3)$
2. Déterminer l'expression associée à l'arbre :



Parcours préfixe

$\times 2 + \times aa5$

- Notation polonaise
- Pas besoin de parenthèses

Parcours postfixe

$2aa \times 5 + \times$

- Notation polonaise inversée
- Pas besoin de parenthèses
- Permet une évaluation récursive de l'arbre

Parcours infixe

$2 \times a \times a + 5$

- Problème de parenthèses...
- Première visite $\rightarrow$ (
- Dernière visite $\rightarrow$)
- Sauf pour les feuilles
- $(2 \times ((a \times a) + 5))$

Exercice 3.

1. Déterminer la notation polonaise inversée correspondant à l'expression :

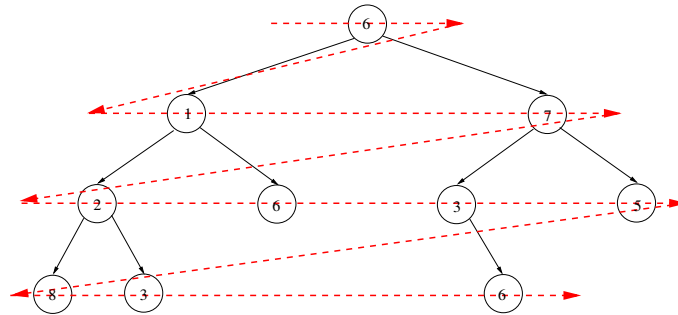
$$(x + y)^2 + (2x + 1)(y + 3)$$

2. Déterminer l'expression associée à la notation polonaise :

$$\times + \times 3xy + 5 \times 2y$$

3 Parcours d'arbres en largeur

Parcourir les nœuds niveau par niveau



Principe

- Départ : Racine
- Parcourir les fils de la racine
- Puis les fils des fils de la racine
- Etc.
- Arrivée : dernier nœud du dernier niveau

Illustration

$6 \rightarrow 1 \rightarrow 7 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 3 \rightarrow 6$

Remarque

Pas de récursivité naturelle dans ce cas

Algorithme

Fonction Larg(*i* : niveau)

début

 Donner à *i* la valeur 0

tant que *i* ≤ *max* **faire**

pour *A* de niveau *i* **faire**

 | Traiter(*A*)

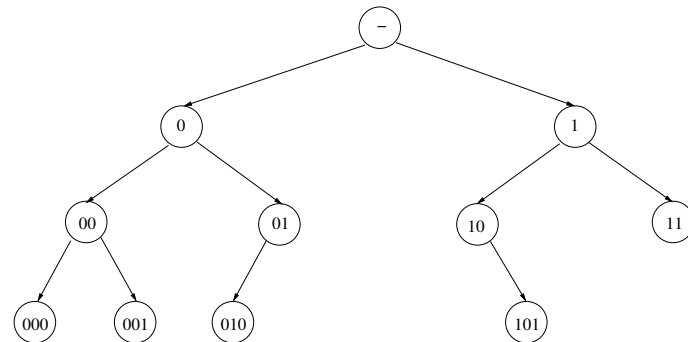
fin

 Donner à *i* la valeur *i* + 1

fin

fin

Étiqueter les nœuds



Principe

- Racine $\rightsquigarrow -$
- Fils gauche $\rightsquigarrow 0$
- Fils droit $\rightsquigarrow 1$
- Etc.
- Fils gauche de $w \rightsquigarrow w0$
- Fils droit de $w \rightsquigarrow w1$

Parcours en largeur

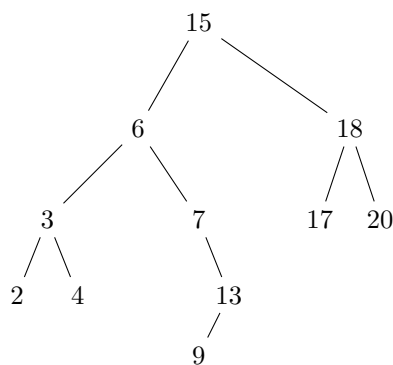
$- \rightarrow 0 \rightarrow 1 \rightarrow 00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \rightarrow 000 \rightarrow 001 \rightarrow 010 \rightarrow 101$

Observation

On visite les nœuds dans l'ordre lexicographique de leurs étiquettes.

4 Arbres binaires de recherche

4.1 Définition



Organisation des étiquettes

Pour tout nœud x :

- Les étiquettes de tous les nœuds du sous-arbre gauche de x sont inférieures à l'étiquette de x
- Les étiquettes de tous les nœuds du sous-arbre droit de x sont supérieures à l'étiquette de x

Remarque. Pour simplifier, on suppose que les étiquettes des nœuds sont toutes différentes.

4.2 Recherche d'une valeur

Y-a-t-il un nœud étiqueté 13 ?

- Départ : Racine 15 $\rightarrow$ trop grand $\rightarrow$ sous-arbre gauche
- Racine 6 $\rightarrow$ trop petit $\rightarrow$ sous-arbre droit
- Racine 7 $\rightarrow$ trop petit $\rightarrow$ sous arbre droit
- Racine 13 $\rightarrow$ trouvé!

Y-a-t-il un nœud étiqueté 16 ?

- Départ : Racine 15 $\rightarrow$ trop petit $\rightarrow$ sous-arbre droit
- Racine 18 $\rightarrow$ trop grand $\rightarrow$ sous-arbre gauche
- Racine 17 $\rightarrow$ trop grand $\rightarrow$ sous-arbre gauche $\rightarrow$ n'existe pas $\rightarrow$ il n'y en a pas!

Algorithme

Fonction Rech($x : label, A : nœud$)

```

début
    si  $x = label(A)$  ou  $A = NULL$  alors
        | Traiter( $A$ )
    sinon
        si  $x < label(A)$  alors
            | Rech( $x$ , FilsGauche( $A$ ))
        sinon
            | Rech( $x$ , FilsDroit( $A$ ))
        fin
    fin
fin

```

Pour aller plus loin

Questions

- Comment trouver le maximum dans un ABR ?
- Comment trouver le minimum dans un ABR ?

Recherche du maximum

- On va toujours à droite...

Algorithme

Fonction Max($A : nœud$)

```

début
     $B$  prend la valeur FilsDroit( $A$ )
    si  $B = NULL$  alors
        | Traiter( $A$ )
    sinon
        | Max( $B$ )
    fin
fin

```

4.3 Parcours infixe d'un ABR

Algorithme

Fonction $\text{Inf}(A : \text{nœud})$

```
début
  si  $A \neq \text{NULL}$  alors
    Inf(FilsGauche( $A$ ))
    Traiter( $A$ )
    Inf(FilsDroit( $A$ ))
  fin
fin
```

Résultat

- Départ $\rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 9 \rightarrow 13 \rightarrow 15 \rightarrow 17 \rightarrow 18 \rightarrow 20 \rightarrow \text{Fin}$

Observation

- Le parcours infixe d'un ABR traite les nœuds en suivant l'ordre de leur étiquette

Pour aller plus loin**Questions**

- Comment trouver le successeur d'un nœud dans un ABR ?
- Comment trouver le prédécesseur d'un nœud dans un ABR ?

Recherche du successeur de A

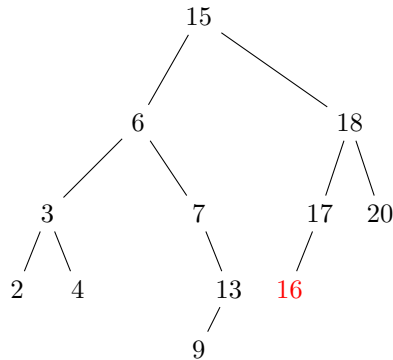
- Si A possède un fils droit, le successeur de A est le minimum du sous-arbre droit de A
- Sinon, le successeur de A est le premier ancêtre de A en remontant dans l'arbre (s'il existe) dont A se trouve dans le sous-arbre gauche, car aucun nœud ne peut s'intercaler entre A et cet ancêtre
- Sinon, A est le maximum de l'arbre

Algorithme

Fonction $\text{Succ}(A : \text{nœud})$

```
début
  si  $\text{FilsDroit}(A) \neq \text{NULL}$  alors
    Max(FilsDroit( $A$ ))
  sinon
     $B$  prend la valeur  $\text{Pere}(A)$ 
    tant que  $B \neq \text{NULL}$  et
       $A = \text{FilsDroit}(B)$  faire
       $A$  prend la valeur  $B$ 
       $B$  prend la valeur  $\text{Pere}(B)$ 
    fin
  fin
  Traiter( $B$ )
fin
```

4.4 Ajouter un élément à un ABR



Question

- Ajouter un nœud portant l'étiquette 16?

Recherche de la position

- On utilise la fonction de recherche d'une valeur dans un ABR
- Comme l'étiquette 16 n'apparaît pas, on obtient la valeur NULL
- Correspondant à la position que devrait occuper le nœud
- On ajoute une feuille d'étiquette 16 à cet endroit

Insertion (provisoire)

Fonction $\text{Insert}(x : \text{label}, A : \text{nœud})$

```
début
  si  $x = \text{label}(A)$  ou  $A = \text{NULL}$  alors
    | Donner à  $\text{label}(A)$  la valeur  $x$ 
  sinon
    | si  $x < \text{label}(A)$  alors
    |   |  $\text{Insert}(x, \text{FilsGauche}(A))$ 
    |   |
    |   | sinon
    |   |   |  $\text{Insert}(x, \text{FilsDroit}(A))$ 
    |   |   |
    |   |   | fin
    |   fin
  fin
fin
```

Problème

- Traditionnellement NULL n'est pas un nœud
- On aura une fonction à trois paramètres
 - ★ x : valeur à insérer
 - ★ A : nœud courant (peut-être NULL)
 - ★ B : père du nœud courant
- On appellera $\text{Insert}(x, \text{Racine}, \text{NULL})$
- On doit (?) aussi traiter le cas où l'arbre est vide au départ
- Bref, c'est un peu plus compliqué

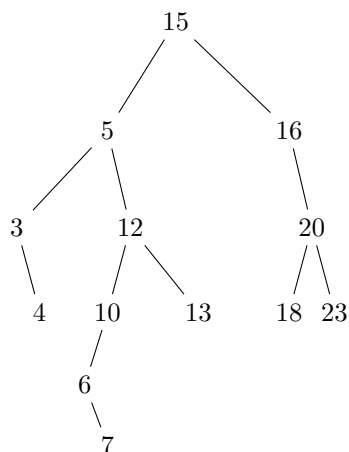
Insertion

Fonction Insert(x, A, B)	
début	
si $A = NULL$ alors	
si $B = NULL$ alors	
CréerABR($x, NULL, NULL$)	
sinon	
si $x < label(B)$ alors	
Donner à FG(B) la valeur x	
sinon	
Donner à FD(B) la valeur x	
fin	
fin	
sinon	
si $x < label(A)$ alors	
Insert($x, FG(A), A$)	
sinon	
Insert($x, FD(A), A$)	
fin	
fin	
fin	

Pour aller plus loin

5cm

<1-2>



Question

- Comment supprimer un nœud dans un ABR ?

Trois cas

- Si le nœud à supprimer est une feuille (par exemple 13) $\rightsquigarrow$ facile
- Si le nœud à supprimer n'a qu'un seul fils (par exemple 16 ou 10) $\rightsquigarrow$ facile
- Si le nœud à supprimer a deux fils (par exemple 5) $\rightsquigarrow$ moins facile...

4.5 ABR vs. dichotomie

Recherche dichotomique dans un tableau trié

Valeur	2	3	4	6	7	9	13	15	17	18	20
Indice	0	1	2	3	4	5	6	7	8	9	10

- Ce tableau T contient-il la valeur 13 entre les indices 0 et 10?
- Je me place au milieu du tableau : indice $0 + \lfloor \frac{10-0}{2} \rfloor = 5$ et valeur $T[5] = 9$
- On a $9 < 13 \rightsquigarrow$ chercher la valeur 13 entre les indices $5 + 1 = 6$ et 10 du tableau
- Etc.

Ajouter un élément dans un tableau trié

<i>Valeur</i>	2	3	4	6	7	9	13	16	15	17	18	20
<i>Indice</i>	0	1	2	3	4	5	6	7	8	9	10	11

- Ajouter la valeur 16 aux valeurs du tableau T ?
- Je crée un tableau T' avec une case de plus
- Je commence à recopier le tableau T dans le tableau T' à partir de l'indice 0
- Lorsque j'atteins la position où doit se trouver 16, je l'insère
- Je finis de recopier T dans T'

Questions de structures de données

- Tableau : structure de données statique
- Arbre : structure de données dynamique

Questions de complexité

- Tableau trié
 - ★ Le nombre d'opérations élémentaires requis par la plupart des algorithmes est proportionnel au nombre d'éléments du tableau
- Arbre binaire de recherche
 - ★ Le nombre d'opérations élémentaires requis par la plupart des algorithmes est proportionnel à la hauteur de l'arbre