INF6 ALGORITHMIQUE AVANCÉE

Carine Souveyet
Manuele Kirsch Pinheiro

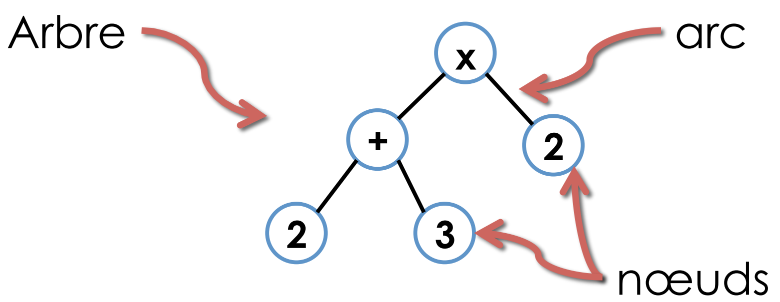
Contenu prévisionnel

- Piles et files
- Listes
- Récursivité
  - Récursivité dans le calcul
  - Récursivité structurelle
- **Arbres binaires**
  - **Parcours en profondeur et en largeur**
- Généralisation de la notion d'arbre
  - Insertion et suppression de nœuds
- Arbre de recherche
  - Rééquilibrage
- Graphes

ARBRES

Arbres
Définitions et terminologie

- Un arbre est une structure de données caractérisée par son structure **hiérarchique**.
- Un **arbre** est formé par un ensemble de **sommets** (ou **nœuds**), reliés par des **arcs** et organisés de manière **hiérarchique**.



Définitions et terminologie

- Dans un arbre, il existe un nœud particulier, appelé **racine**, qui est à l'origine de l'arborescence
- Chaque nœud possède 0 ou plusieurs nœuds **fils** directement connectés à lui par un **arc**
- Chaque nœud, à l'exception de la racine, possède un **parent** (ou **nœud père**) **unique**

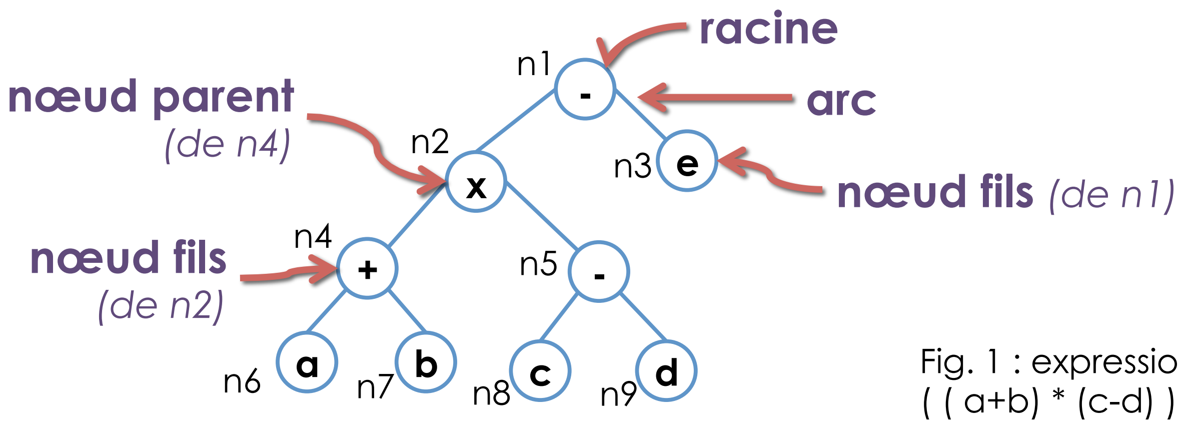
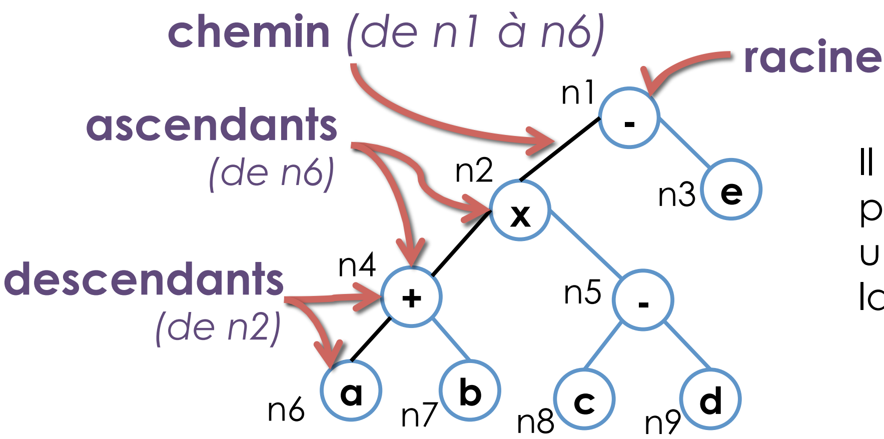


Fig. 1 : expression
((a+b) * (c-d)) - e

Définitions et terminologie

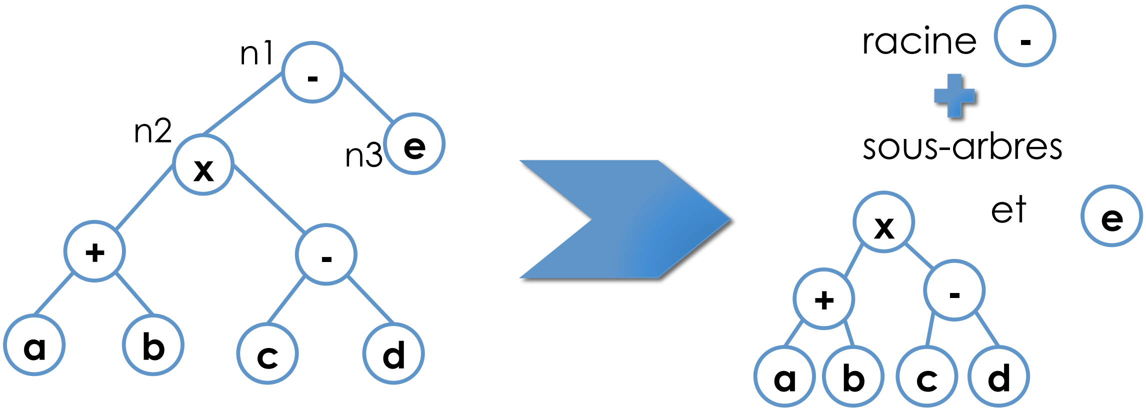
- Tout nœud *n* est accessible par un **chemin** unique qui part de la **racine** et passe par un ensemble de nœuds appelés **ascendants** de *n*
- Tous les nœuds accessibles par un **chemin** à partir de *n* sont des **descendants** de ce nœud



Il n'y a qu'**un chemin** possible pour atteindre un **nœud *n*** à partir de la **racine**.

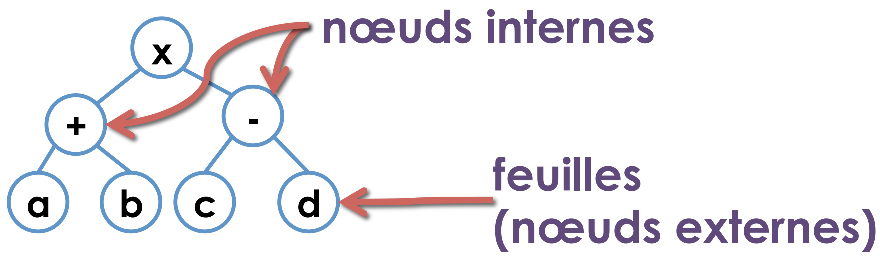
Définitions et terminologie

- Un **arbre** peut être ainsi défini comme l'ensemble formé d'un nœud **racine** et d'une suite éventuellement vide de **sous-arbres** $S_1, \ldots, S_m$ $m \geq 0$
- L'arbre ci-dessous peut être définie comme la **racine n1** et les **sous-arbres n2** et **n3**



Définitions et terminologie

- Chaque nœud dans un arbre contient
  - une information spécifique à l'application (on parle alors d'**arbre étiquetée**)
  - des pointeurs vers d'autres **sous-arbres**
- Le **degré** d'un nœud est le nombre de fils de ce nœud
- Les nœuds qui ne possèdent pas de sous-arbre (c.a.d. pas de fils ou de descendant) sont appelés **nœuds externes** ou **feuilles**
- Les nœuds possédant au moins un sous-arbre (c.a.d. *degré ≥ 1*) sont appelés **nœuds internes**



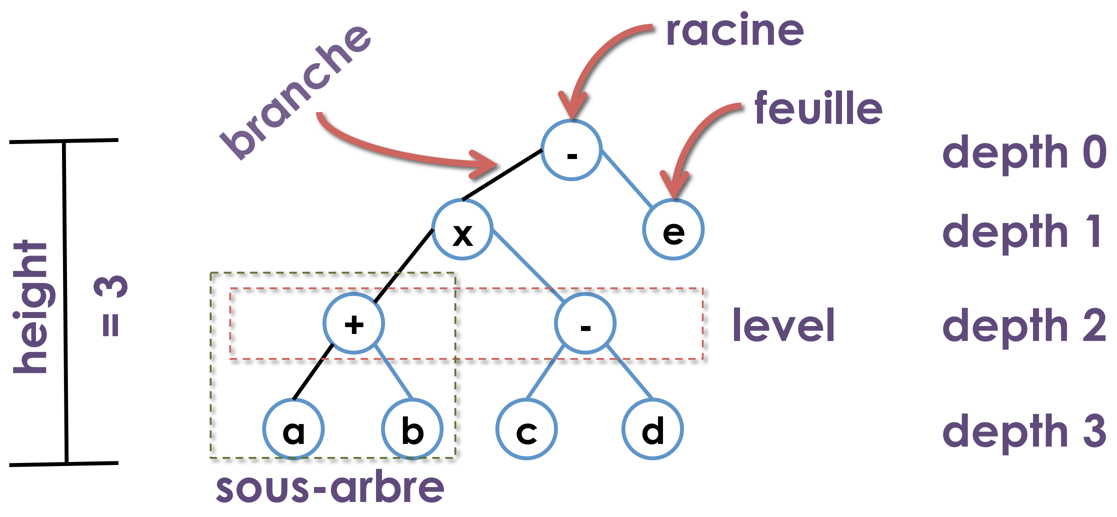
UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Définitions et terminologie

- Une **branche** est un **chemin** entre la **racine** et une **feuille**. Un arbre a autant de branches que de feuilles.
- La **longueur** d'un chemin entre deux nœuds appartenant à une même branche est égale au **nombre d'arcs** qui les séparent
- L'**hauteur** (***depth***) d'un nœud *n* est la **longueur** du **chemin** entre la **racine** et **lui**
- Un **niveau** (***level***) de l'arbre est formé de tous les nœuds placés à une **même hauteur**
- La **profondeur** (***height***) de l'arbre correspond à son **hauteur maximale**, ou le plus long chemin entre la racine et une feuille

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Définitions et terminologie



UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Définitions et terminologie

- On appelle une **forêt** une suite disjointe d'arbres
- **Opérations abstraites**
  - cons : Nœud x Forêt → Arbre construction d'un arbre
  - racine : Arbre → Nœud racine de l'arbre
  - forêt : Arbre → Forêt fils (sous-arbres)
  - valeur : Nœud → e valeur d'un nœud

  - ièmeArbre : Forêt x entier → Arbre ième sous-arbre
  - ajouterArbre : Forêt x entier x Arbre → Forêt
  - supprimerArbre : Forêt x entier → Forêt
  ajouter / supprimer un fils

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

ARBRE BINAIRE

Arbre binaire

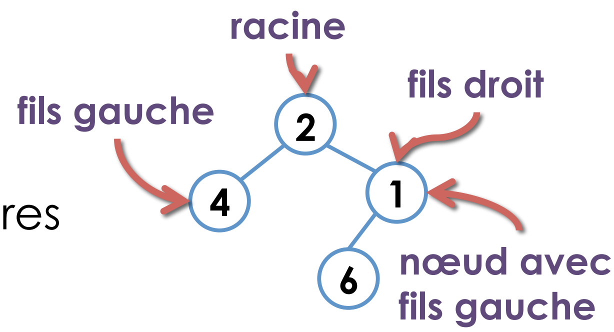
- Un arbre binaire est un arbre **ordonné** dont le **degré** des nœuds **ne dépasse pas 2**

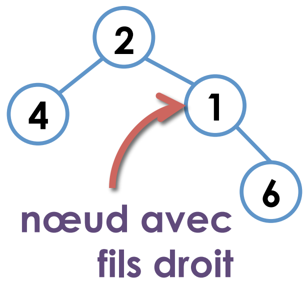
Chaque nœud possède au plus 2 fils

Ordonné car l'ordre des sous-arbres est significatif

On distingue alors le fils gauche du fils droit

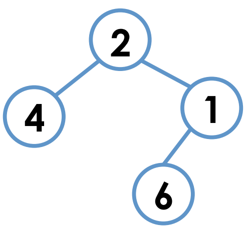
Deux arbres binaires **distincts**



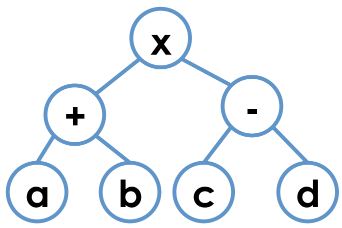


Arbres binaires

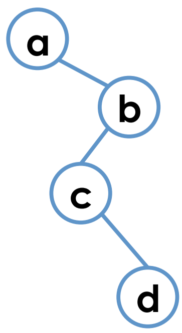
- Quelques formes caractéristiques



arbre binaire
forme quelconque



arbre binaire
complet
↓
les nœuds internes (pas les feuilles) possèdent toujours 2 fils (degré 2)

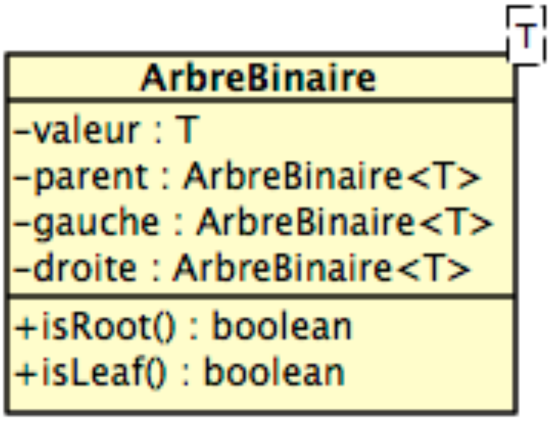


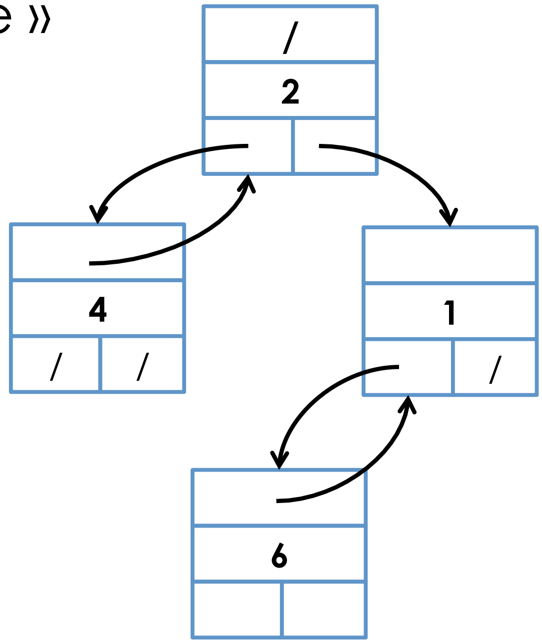
arbre binaire
dégénéré
↓
chaque niveau ne possède qu'un nœud. Tous les nœuds appartiennent à une même branche

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Arbre binaire

- Représentation « chaînée »





UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

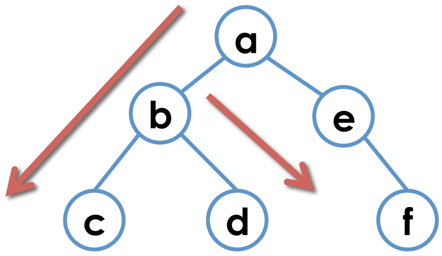
Parcours

- **Parcours arbre binaire**
  - Un algorithme de parcours d'arbre est un algorithme permettant d'accéder à chaque nœud de l'arbre afin d'y effectuer un traitement (test, affichage, modification, comptage…)
  - Les algorithmes de parcours sont indépendants de ce traitement (qu'on appelle communément « visite »)
  - On distingue deux catégories de parcours
    - parcours en profondeur
    - parcours en largeur

Parcours

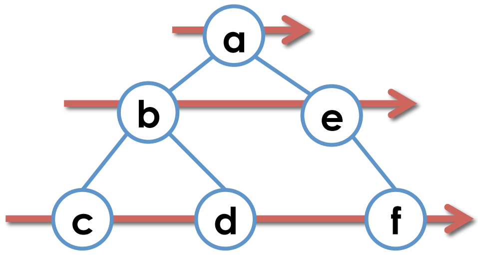
Parcours en profondeur

on explore l'arbre
branche par branche



Parcours en largeur

on explore l'arbre
niveau par niveau



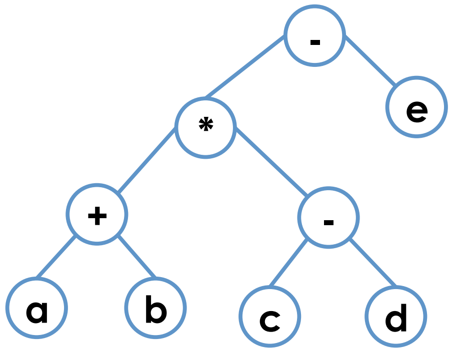
Parcours en profondeur

- **Parcours en profondeur**
  - Selon le moment où le nœud courant est traité, on distingue trois type (ou méthodes) : **préfixe**, **infixe** et **postfixe**
  - **Parcours Préfixe**
    - On va **d'abord traiter** le nœud courant, puis explorer les sous-arbres **gauche** et **droite**
  - **Parcours infixe**
    - On va **visiter** le nœud courant ***après*** avoir exploré la sous-arbre gauche et ***avant*** d'explorer la sous-arbre droite
  - **Parcours Préfixe**
    - On va d'abord explorer les sous-arbres **gauche** et **droite** ***avant*** de **visiter** le nœud

UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Parcours en profondeur

- **Parcours préfixe**
  1. visiter le nœud
  2. explorer sous-arbre gauche
  3. explorer sous-arbre droit



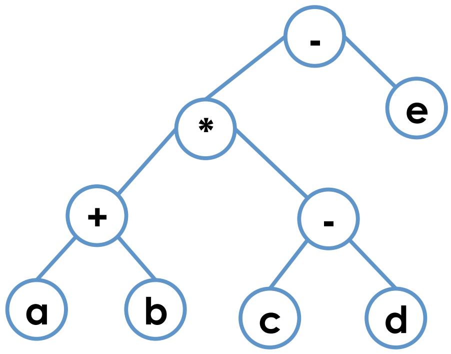
\- * + a b - c d e

```
prefixe (racine)
   Entrée :
         Nœud racine
   Si racine != null
   alors
         visite (racine)
         prefixe ( gauche (racine) )
         prefixe ( droit (racine) )
   fin si
fin
```


UNIVERSITÉ PARIS 1 PANTHÉON SORBONNE

Parcours en profondeur

- **Parcours infixe**
  1. explorer sous-arbre gauche
  2. visiter le nœud
  3. explorer sous-arbre droit

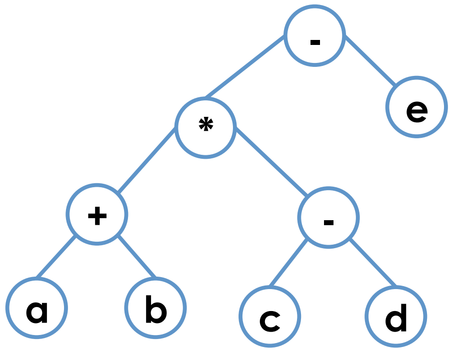


a + b * c - d - e

```
infixe (racine)
   Entrée :
         Nœud racine
   Si racine != null
   alors
         infixe ( gauche (racine) )
         visite (racine)
         infixe ( droit (racine) )
   fin si
fin
```

Parcours en profondeur

- **Parcours postfixe**
  ① explorer sous-arbre gauche
  ② explorer sous-arbre droit
  ③ visiter le nœud

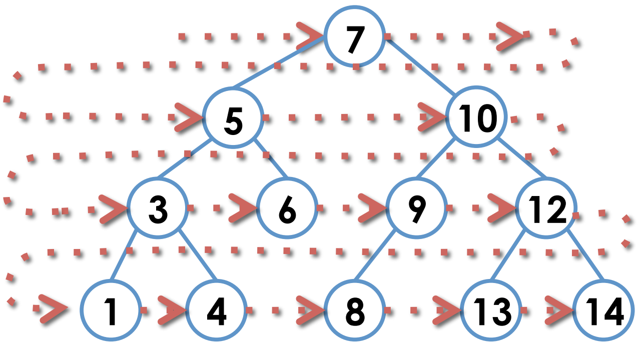


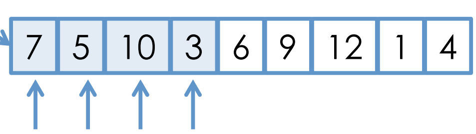
a b + c d - * e -

```
postfixe (racine)
   Entrée :
         Nœud racine
   Si racine != null
   alors
         postfixe ( gauche (racine) )
         postfixe ( droit (racine) )
         visite (racine)
   fin si
fin
```


Parcours en largeur

- **Parcours en largeur**
  - On explore l'arbre **niveau** par **niveau**
  - On utilise une **file d'attente** pour conserver les nœuds à traiter dans chaque niveau





A chaque nœud visité, ses fils sont mis dans la file d'attente

7 5 10 3

Parcours en largeur

```
Largeur (racine)
   Entrée :
       Nœud racine
   Si racine != null
   alors
       File f = nouvelle File
       enfiler ( f , racine )
       Tant que ! EstVide ( f )
       faire
           Nœud n = defiler ( f )
           visite ( n )
           Si gauche ( n ) != null
           alors   enfiler ( f , gauche ( n ) )
           fin si
           Si droit ( n ) != null
           alors   enfiler ( f , droit( n ) )
           fin si
       fin tant
   fin si
fin
```