Course descriptions : semester 2, Data Science and Big Data..

Programmation & Algorithmique II

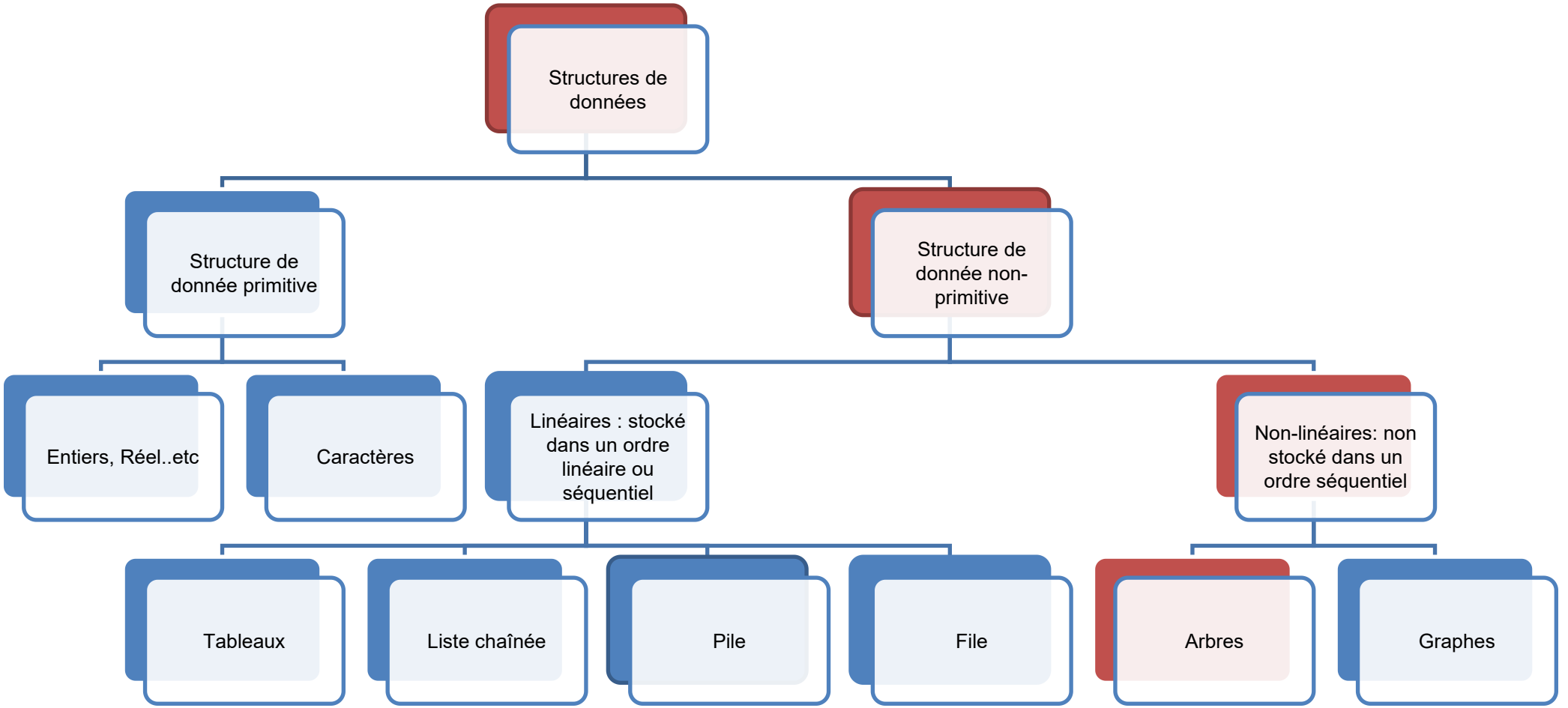
CM13: Les arbres (2)

浙江科技學院 ZHEJIANG UNIVERSITY OF SCIENCE & TECHNOLOGY

CERGY PARIS UNIVERSITÉ

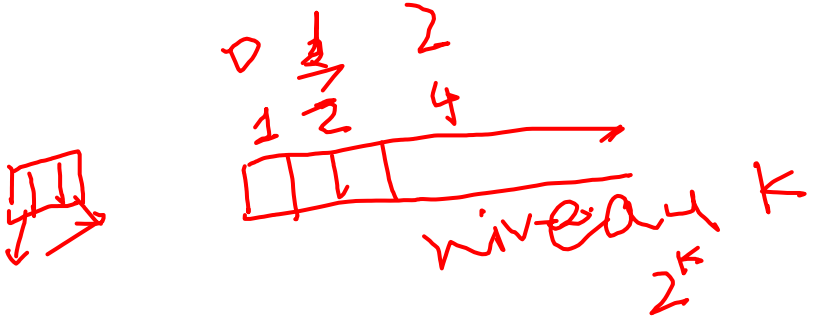
CLASSIFICATION DES STRUCTURES DE DONNÉES

- **Structures de données primitives et non primitives**



PLAN

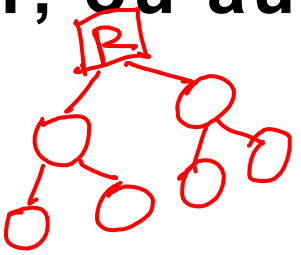
1. Introduction
2. Définition
3. Types des arbres
   1. Arbres généraux
   2. Forêts
   3. Arbres binaires
      1. Arbres binaires complets
      2. Arbres de recherche binaire
      3. Arbres d'expression
4. Créer un arbre binaire à partir d'un arbre général
5. Parcours d'un arbre binaire
   1. Parcours préfixe (pre-order)
   2. Parcours infixe (in-order)
   3. Parcours postfixe (post-order)
   4. Parcours en largeur (BFS)



convertir de arb general → arb binaire (binarisation)

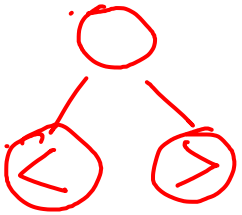
Rappel : Arbre binaire

Un arbre binaire est une structure de données qui est définie comme une collection d'éléments appelés nœuds. Dans un arbre binaire l'élément supérieur est appelé le nœud racine et **chaque nœud a 0, 1, ou au plus 2 nœuds files**.



Arbres de recherche binaire

Un arbre de recherche binaire, également connu sous le nom d'**arbre binaire ordonné**, est une variante de l'arbre binaire dans lequel les nœuds sont disposés dans un **ordre particulier**. (à introduire la semaine prochaine)



CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

(arbres binaires sont facile à représenter dans le mémoire

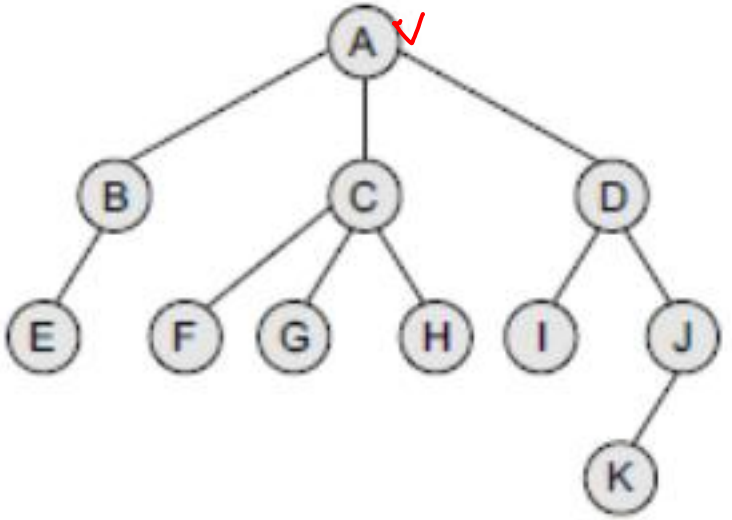
Les règles de conversion d'un arbre général en arbre binaire sont données ci-dessous. Notez qu'un arbre général est converti en arbre binaire et ~~non en arbre de recherche~~ binaire.

convert (ang.)

Règle 1: Racine de l'arbre binaire **=** Racine de l'arbre général
Règle 2: Fils gauche d'un nœud dans l'arbre binaire **=** Fils le plus à gauche du nœud dans l'arbre général
Règle 3: Fils droit d'un nœud dans l'arbre binaire **=** « Frère » droit du nœud dans l'arbre général

eg. fils g de A: B

brother (sibling)



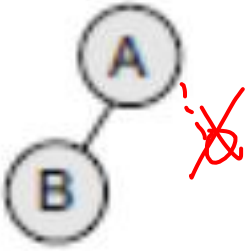
eg. A: pas de frère donc pas de fils d

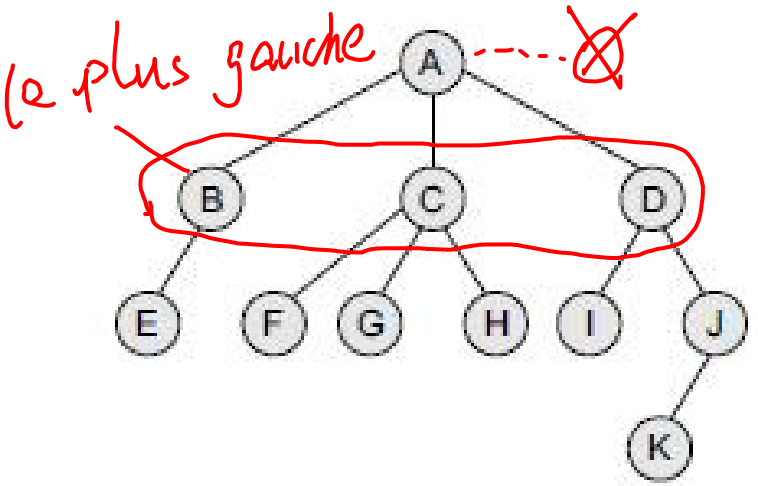
B: frère: C donc C est son fils droit dans l'arbre binaire

CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

Étape 1: Nœud A est la racine de l'arbre général, il sera donc aussi la racine de l'arbre binaire.

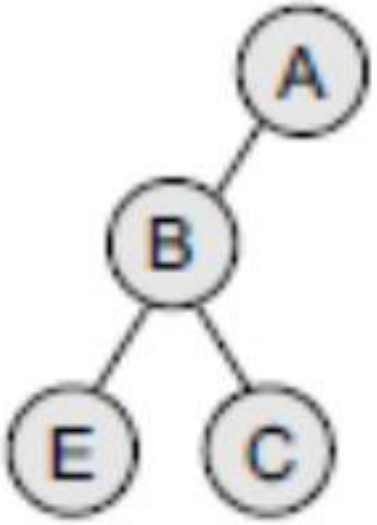
Étape 2: Le nœud fils gauche du nœud A est le nœud fils le plus à gauche du nœud A dans l'arbre général et le fils droit du nœud A est le ''frère'' droit du nœud A dans l'arbre général. Puisque le nœud A n'a pas de frère droit dans l'arbre général, il n'a pas d'enfant droit dans l'arbre binaire.

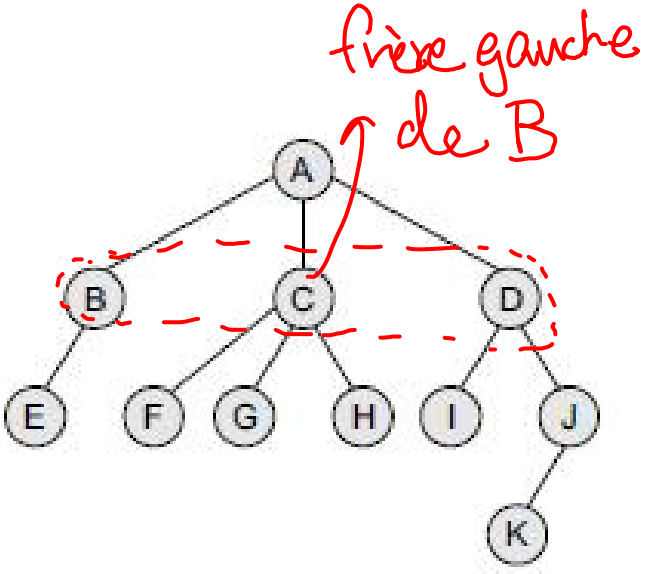




CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

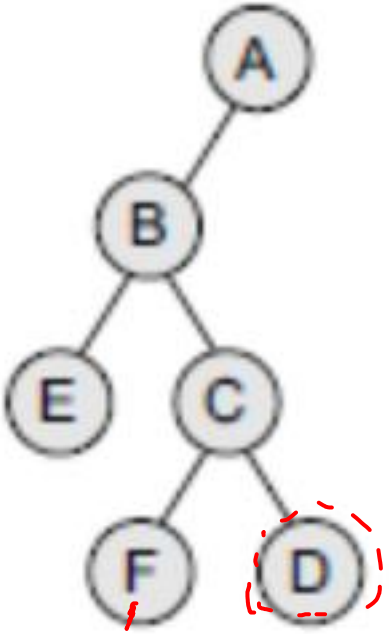
Étape 3: Maintenant traitons le nœud B. Le fils gauche de B est E et son fils droit est C (frère droit dans l'arbre général).

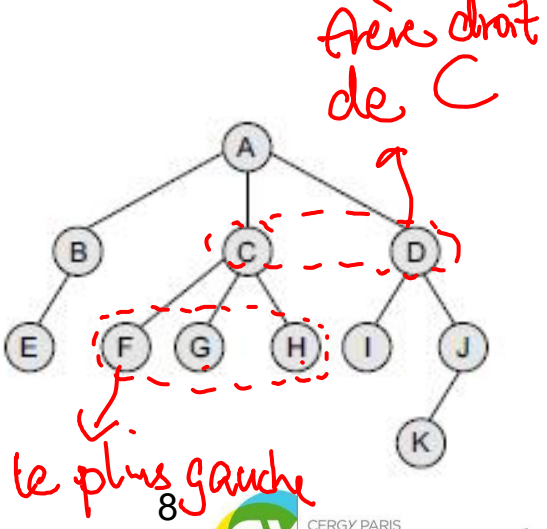




CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

Étape 4: Maintenant traitons le nœud C. Le fils gauche de C est F (le fils plus à gauche) et son fils droit est D (frère droit dans l'arbre général).



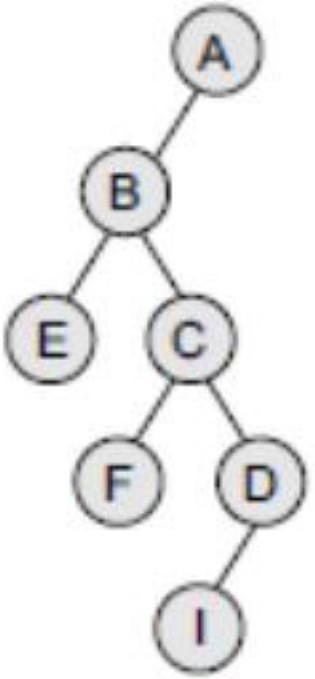


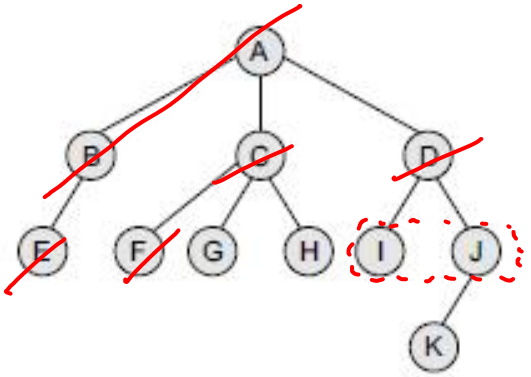
浙江科技學院 ZHEJIANG UNIVERSITY OF SCIENCE & TECHNOLOGY

CERGY PARIS UNIVERSITÉ

CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

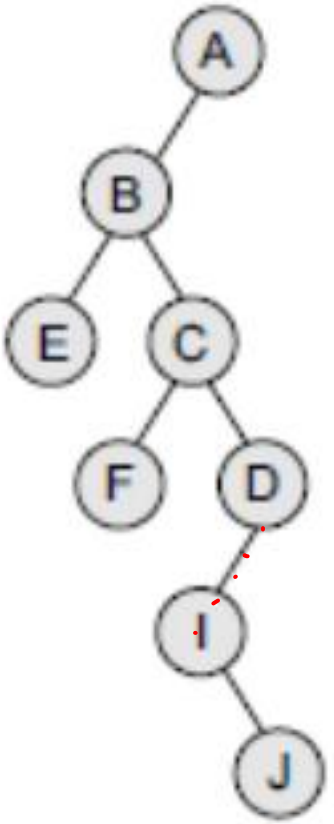
Étape 5: Maintenant traitons le nœud D. Fils gauche de D est I (le fils le plus à gauche). Il n'y aura pas de fils droit de D parce qu'il n'a pas de frère droit dans l'arbre général.

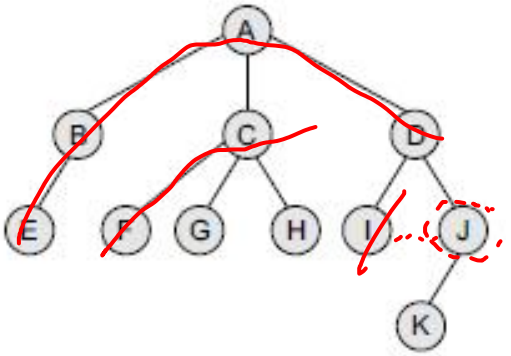




CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

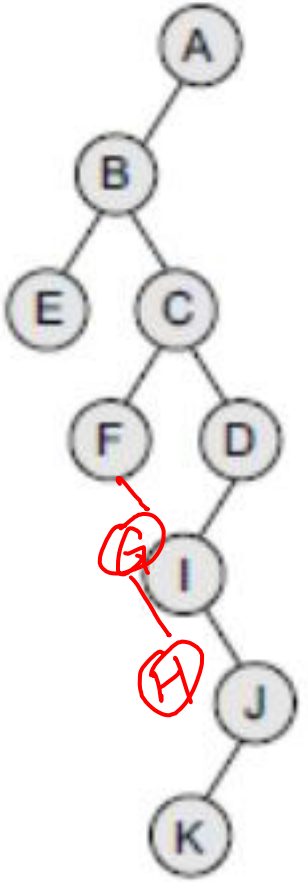
Étape 6: Maintenant, traitons le nœud I. Il n'y aura pas de fils de I dans l'arbre binaire parce que I n'a pas de fils dans l'arbre général. Cependant, I a un frère droit J, alors J sera ajouté comme le fils droit de I.

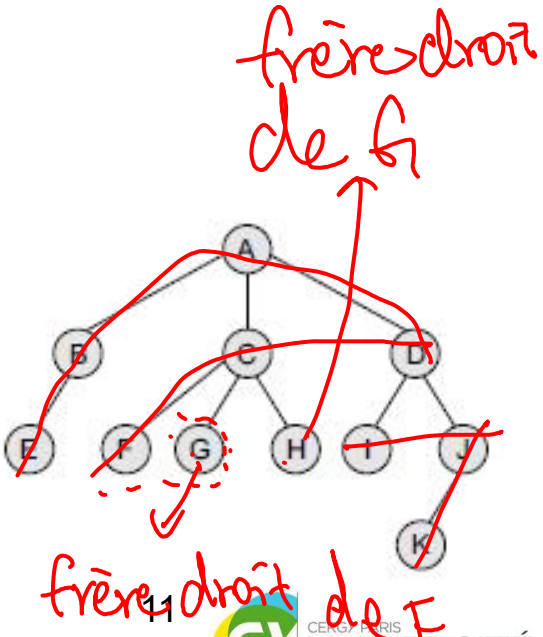




CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

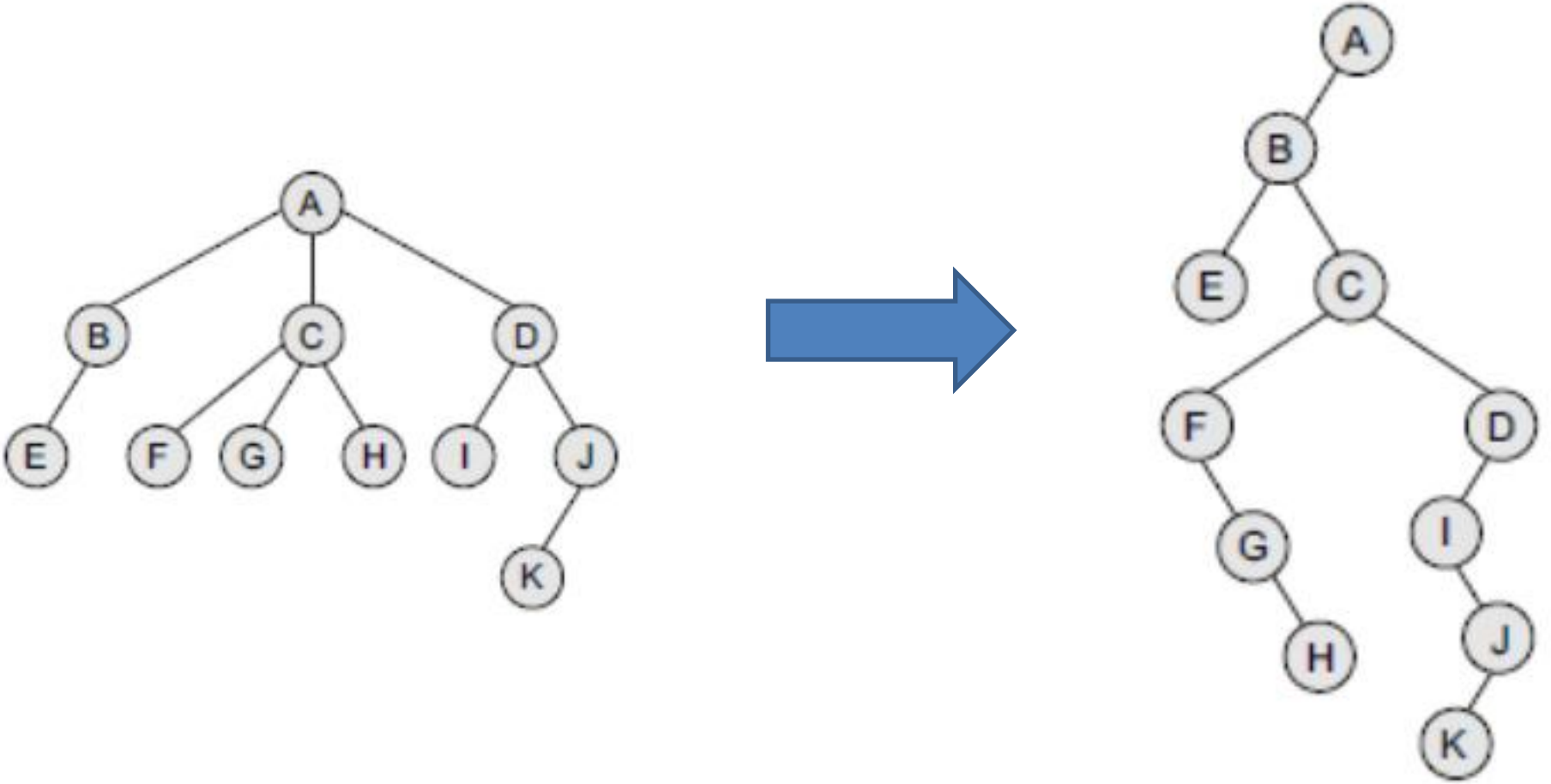
Étape 7: Maintenant, traitons le nœud J. Le fils gauche de J est K (fils le plus à gauche). Il n'y aura pas de fils droit de J parce qu'il n'a pas de frère droit dans l'arbre général.



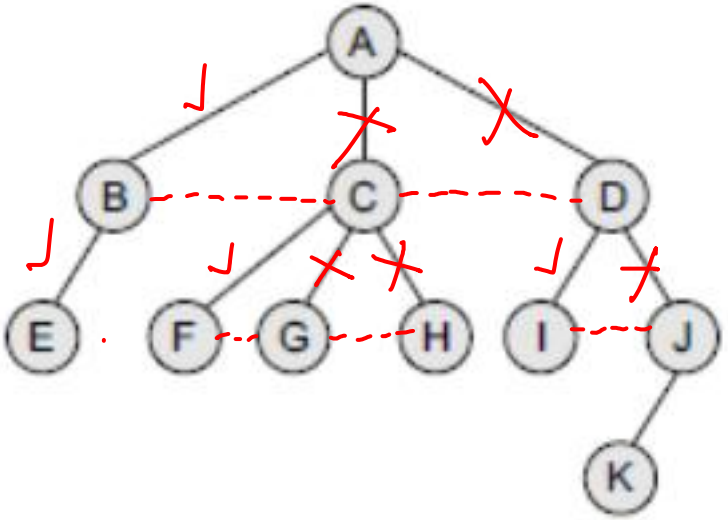


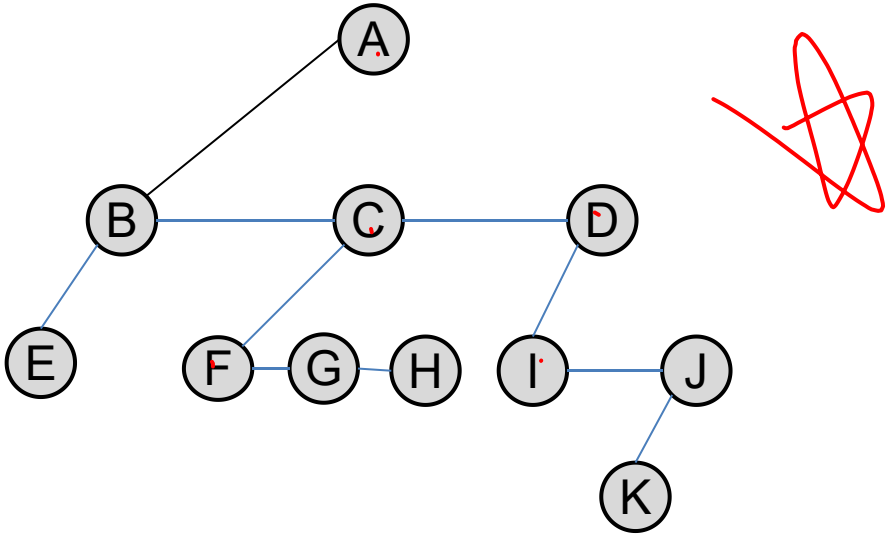
CRÉER UN ARBRE BINAIRE À PARTIR D'UN ARBRE GÉNÉRAL

Étape 8: Maintenant traiter tous les nœuds non transformés (E, F, G, H, K) de la même manière, de sorte que l'arbre binaire qui en résulte peut être donné comme suit.

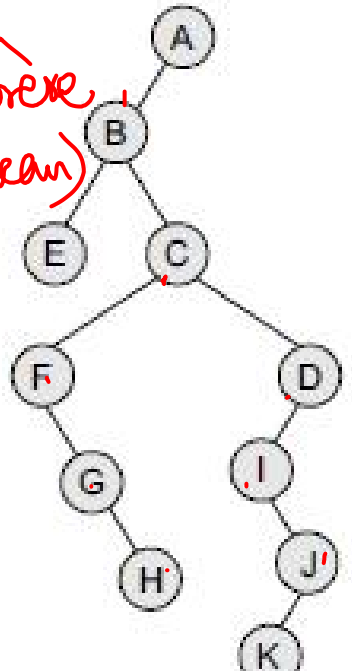


Une demosntration plus simple





1 : lier tous les frères
(nœuds de même père et de même niveau)



2 : effacer les relations père - fils sauf le fils le plus gauche

QUIZ

La représentation contiguë des arbres binaires permet de remplir tout le tableau réservé. Ceci permet d'économiser de l'espace mémoire.

A- ~~Vrai~~

B- Faux

5 minute

PARCOURS D'UN ARBRE BINAIRE

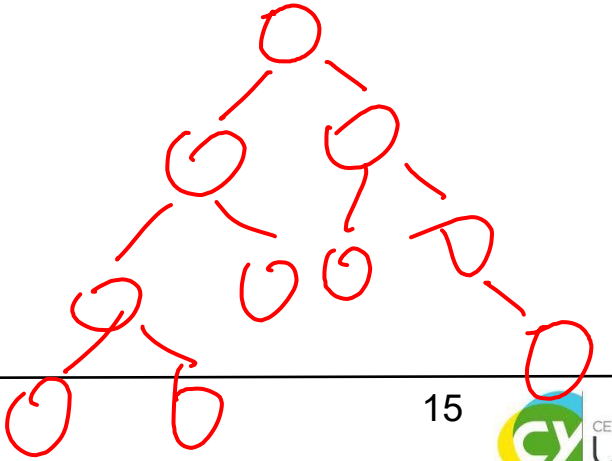
traversal a binary tree

Parcourir (traverser) un arbre binaire est le processus de visite de chaque nœud dans l'arbre exactement une fois d'une manière systématique. Contrairement aux structures de données linéaires dans lesquelles les éléments sont traversés séquentiellement, l'arbre est une structure de données non linéaire dans laquelle les éléments peuvent être traversés de différentes façons. Il existe différents algorithmes pour les traversées d'arbres. Ces algorithmes diffèrent dans l'ordre dans lequel les nœuds sont visités.

structures linéaires

tableau

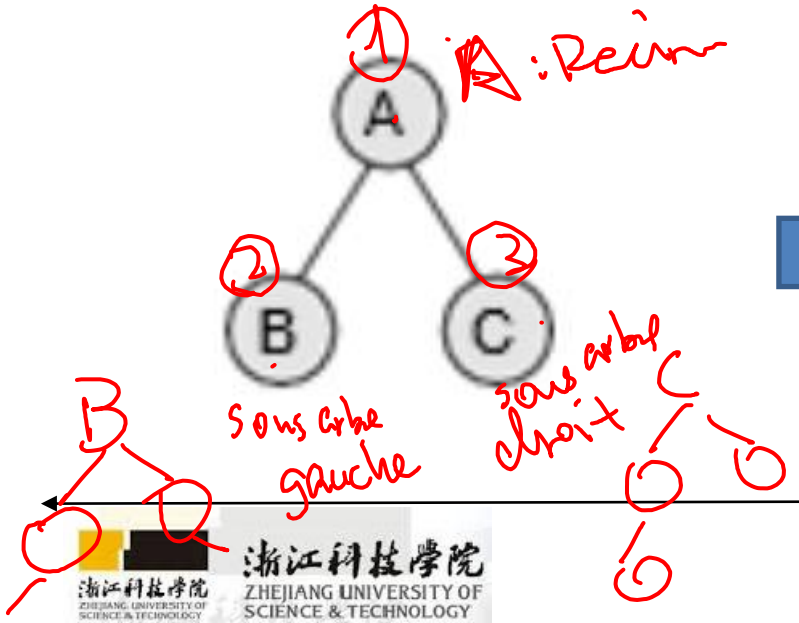
liste chaînée



PARCOURS PRÉFIXE (PRE-ORDER)

Pour parcourir un arbre binaire non-vide en ordre préfixe, les opérations suivantes sont effectuées récursivement à chaque nœud :

1. Visiter le nœud racine,
2. Parcourir le sous-arbre gauche, et enfin
3. Parcourir le sous-arbre droit.



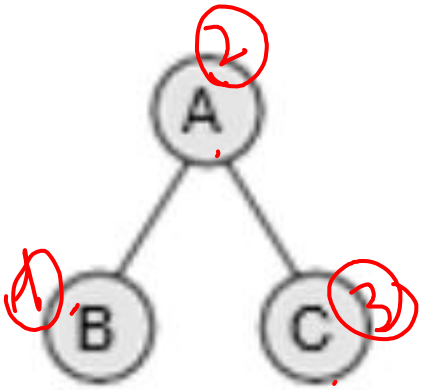
Parcours préfixe de cet arbre binaire donne

A, B, C

PARCOURS INFIXE (IN-ORDER)

Pour parcourir un arbre binaire non-vide en ordre infixe, les opérations suivantes sont effectuées récursivement à chaque nœud :

1. Parcourir le sous-arbre gauche
2. Visiter le nœud racine
3. Parcourir le sous-arbre droit.



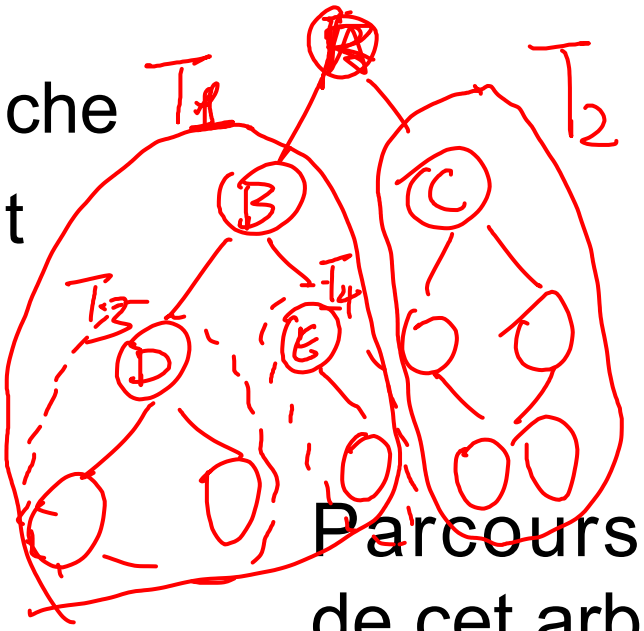
Parcours infixe de cet arbre binaire donne

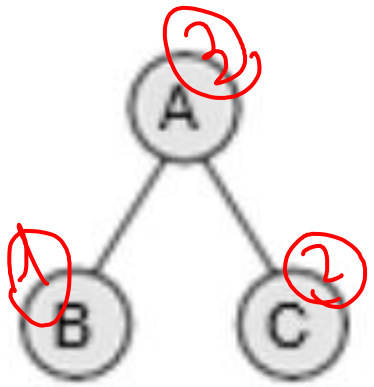
B, A, C

PARCOURS POSTFIXE (POST-ORDER)

Pour parcourir un arbre binaire non-vide en ordre postfixe, les opérations suivantes sont effectuées récursivement à chaque nœud :

1. Parcourir le sous-arbre gauche
2. Parcourir le sous-arbre droit
3. Visiter le nœud racine





Parcours postfixe de cet arbre binaire donne

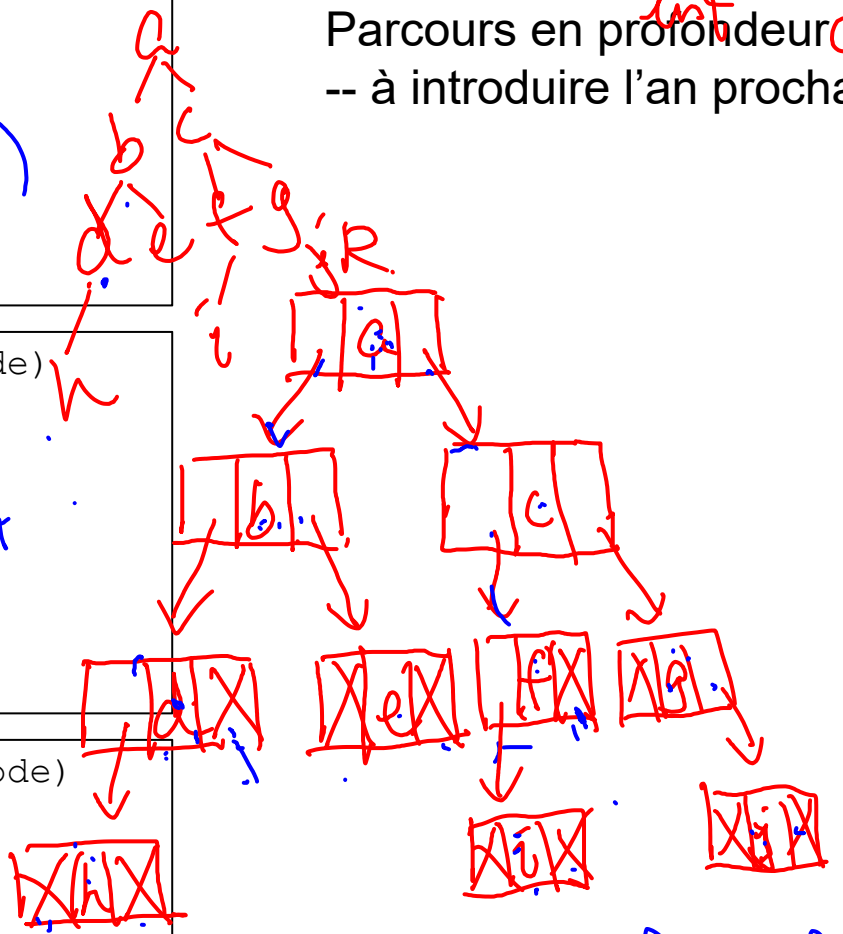
B, C, A

```
void inOrderTraversal(struct node *node)
{
  if (node == NULL)
    return;
  inOrderTraversal(node->leftNode);
  traitement("\t%d\t", node->data);
  inOrderTraversal(node->rightNode);
}
```

```
void preOrderTraversal(struct node *node)
{
  if (node == NULL)
    return;
  traitement("\t%d\t", node->data);
  preOrderTraversal(node->leftNode);
  preOrderTraversal(node->rightNode);
}
```

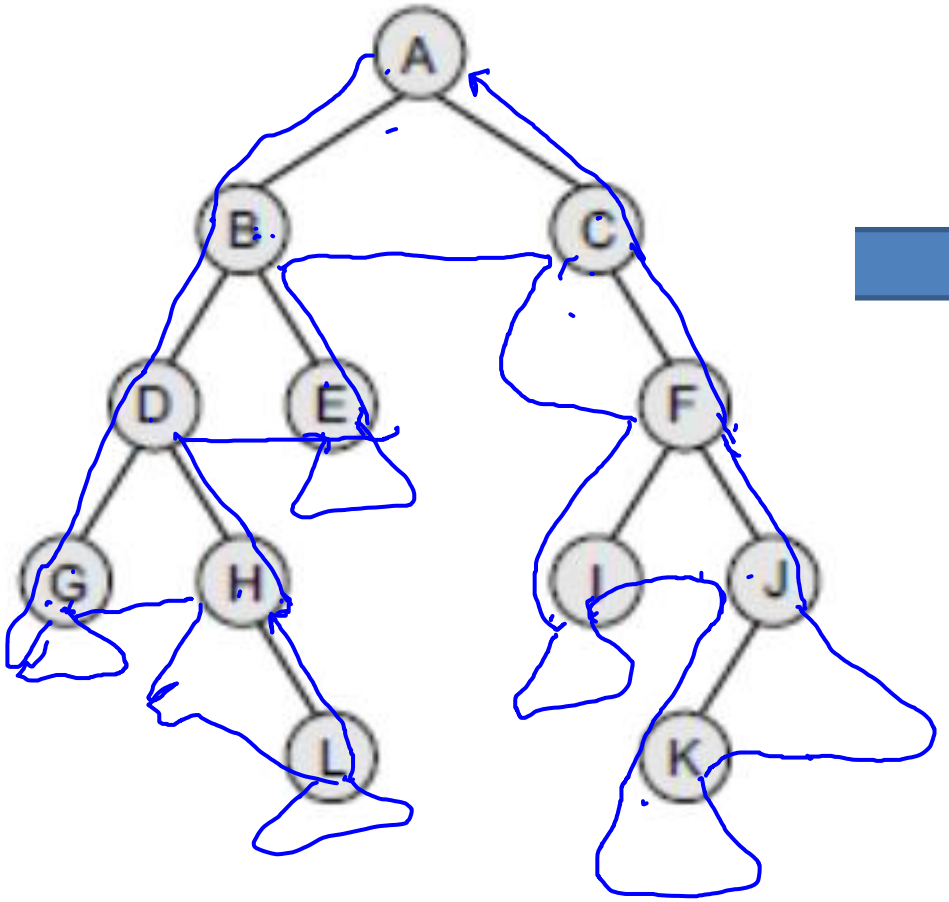
```
void postOrderTraversal(struct node *node)
{
  if (node == NULL)
    return;
  postOrderTraversal(node->leftNode);
  postOrderTraversal(node->rightNode);
  traitement("\t%d\t", node->data);
}
```

Parcours en profondeur itérative?
-- à introduire l'an prochain



PARCOURS PRÉFIXE (PRE-ORDER)

Exemple :



Parcours préfixe

A, B, D, G, H, L, E, C, F, I, J, K

Parcours infixe

G D H L B E A C I F K J

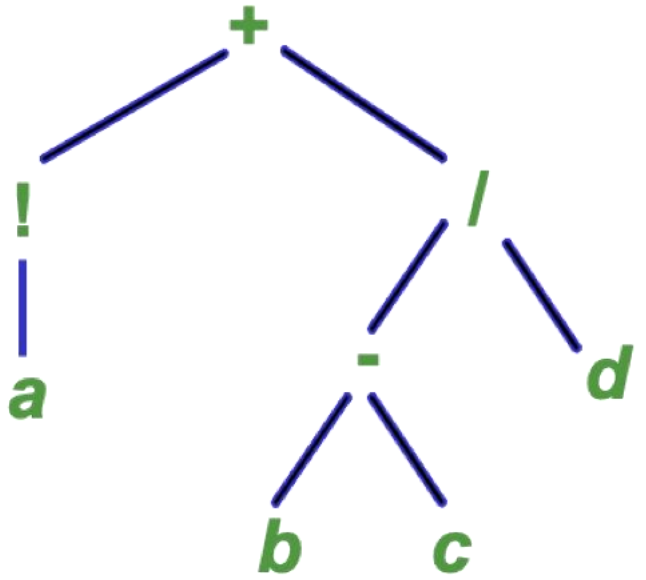
Parcours postfixe

G L H D E B I K J F C A

ARBRES D'EXPRESSION

Les arbres binaires sont largement utilisés pour stocker les expressions algébriques. Par exemple, considérez l'expression algébrique donnée comme :

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



Parcours préfixe

+ ! a / - b c d

Parcours postfixe

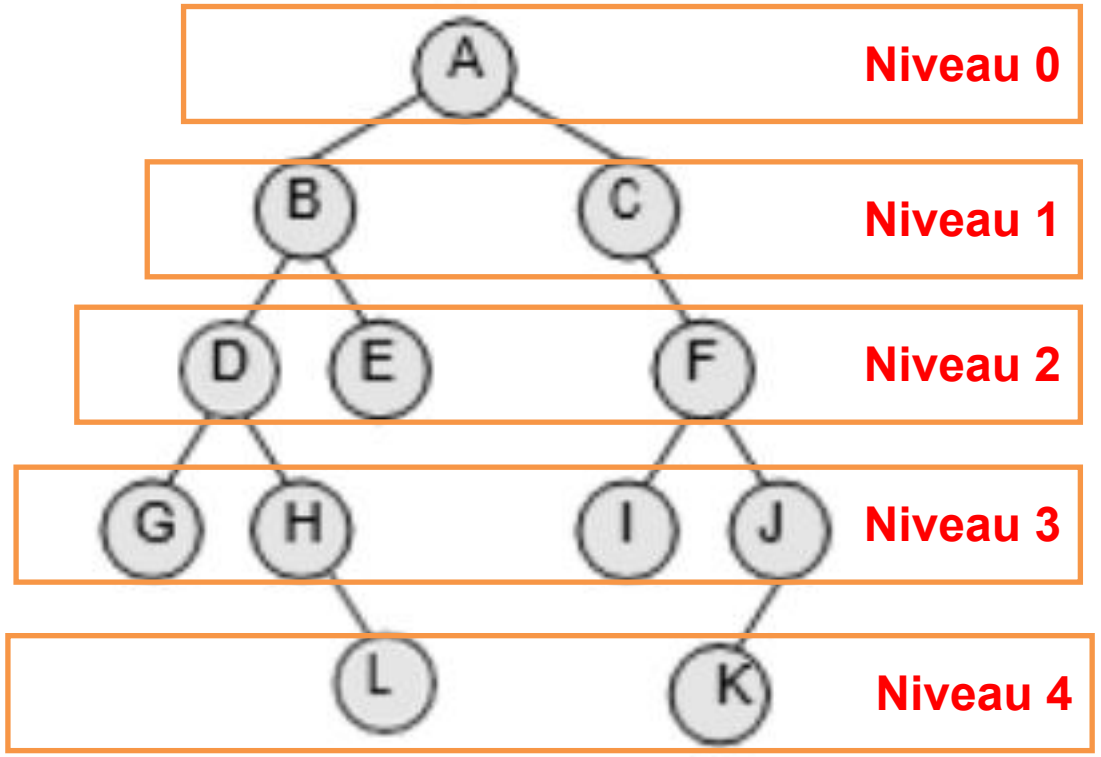
a ! b c - d / +

Parcours infixe (priorités et parenthèses)

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

PARCOURS EN LARGEUR (BFS)

Dans le parcours en largeur, aussi dit Breadth First Search (BFS), tous les nœuds au même niveau sont visités avant d'aller au niveau suivant.

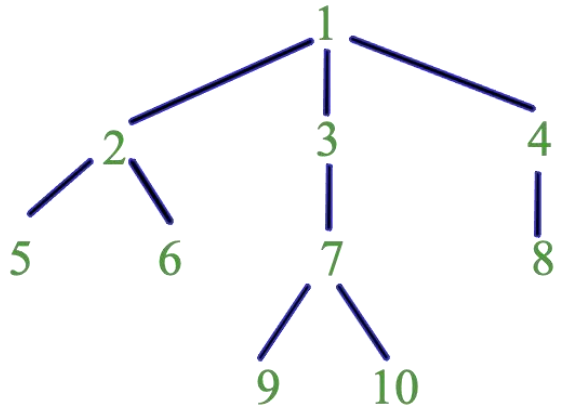


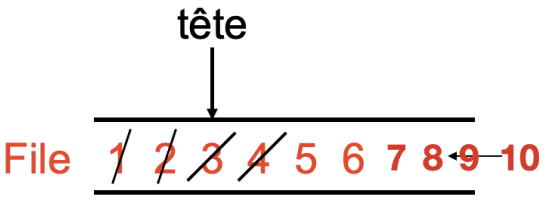
Parcours en largeur de cet arbre binaire donne

A B C D E F G H I J L K

PARCOURS EN LARGEUR (BFS)

```
fonction Largeur (A arbre) : liste de nœuds ;
début
    L <- ( ) ;
    File <- Enfiler( File-vide, A ) ;
    tant que File n'est pas vide faire {
        A' <- tête ( File ) ;
        File <- Défiler ( 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
```





Liste *L* = (1, 2, .

```
BFS(struct node *node) {
  struct fifo* q = createFifo(); // Création d'une file
  fifo.put(node) // Mise de la racine dans la file
  while(!fifo.empty()) {
    Node n = fifo.get(); // Nouveau noeud à traiter en tête de file
    traitement(n); // Traitement du noeud courant
    if (n.ls != nil)
      fifo.put(n.ls); // Ajout du fils gauche s'il existe
    if (n.rs != nil)
      fifo.put(n.rs); // Ajout du fils droit s'il existe
  }
}
```

Exercice :

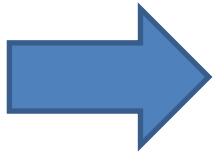
- Avec une représentation chaînée d'un arbre binaire, écrire en langage C les algorithmes qui affichent les parcours Préfixe, Infixe, Post-Fixe, en largeur (BFS),

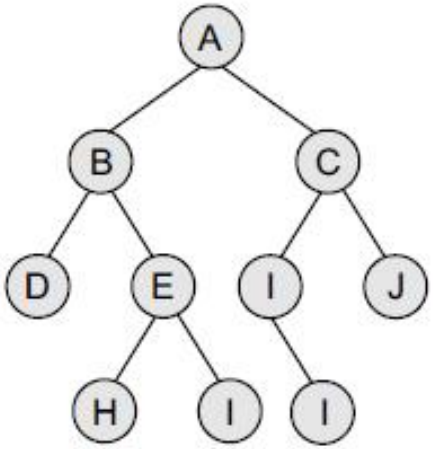
CONSTRUCTION D'UN ARBRE BINAIRE À PARTIR DES RÉSULTATS DE SON PARCOURS

Nous pouvons construire un arbre binaire si on nous donne au moins deux résultats de son parcours. Le premier parcours doit être le parcours infixe et le second peut être soit un le parcours préfixe ou le parcours postfixe. Le résultat du parcours infixe sera utilisé pour déterminer les nœuds fils gauche et fils droit, et la parcours préfixe/postfixe peut être utilisée pour déterminer le nœud racine.

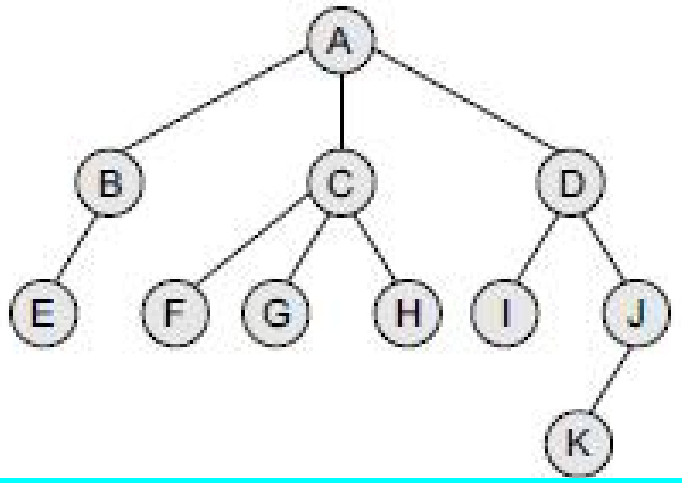
Infixe : D B H E I A F J C G

Prefixe : D H I E B J F G C A



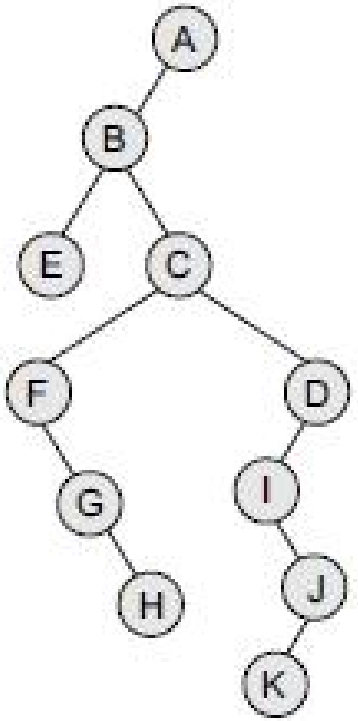


Les résultats de parcours sur un arbre général et sa conversion binaire



Préfixe: A, B, E, C, F, G, H, D, I, J, K

Postfixe: E, B, F, G, H, C, I, K, J, D, A



Préfixe: A, B, E, C, F, G, H, D, I, J, K

Infixe: E, B, F, G, H, C, I, K, J, D, A

Postfixe: E, H, G, F, K, J, I, D, C, B, A

- Le parcours préfixe d'un arbre général est identique au parcours préfixe de sa conversion binaire.
- Le parcours postfixe d'un arbre général est identique au parcours infixe de sa conversion binaire.