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

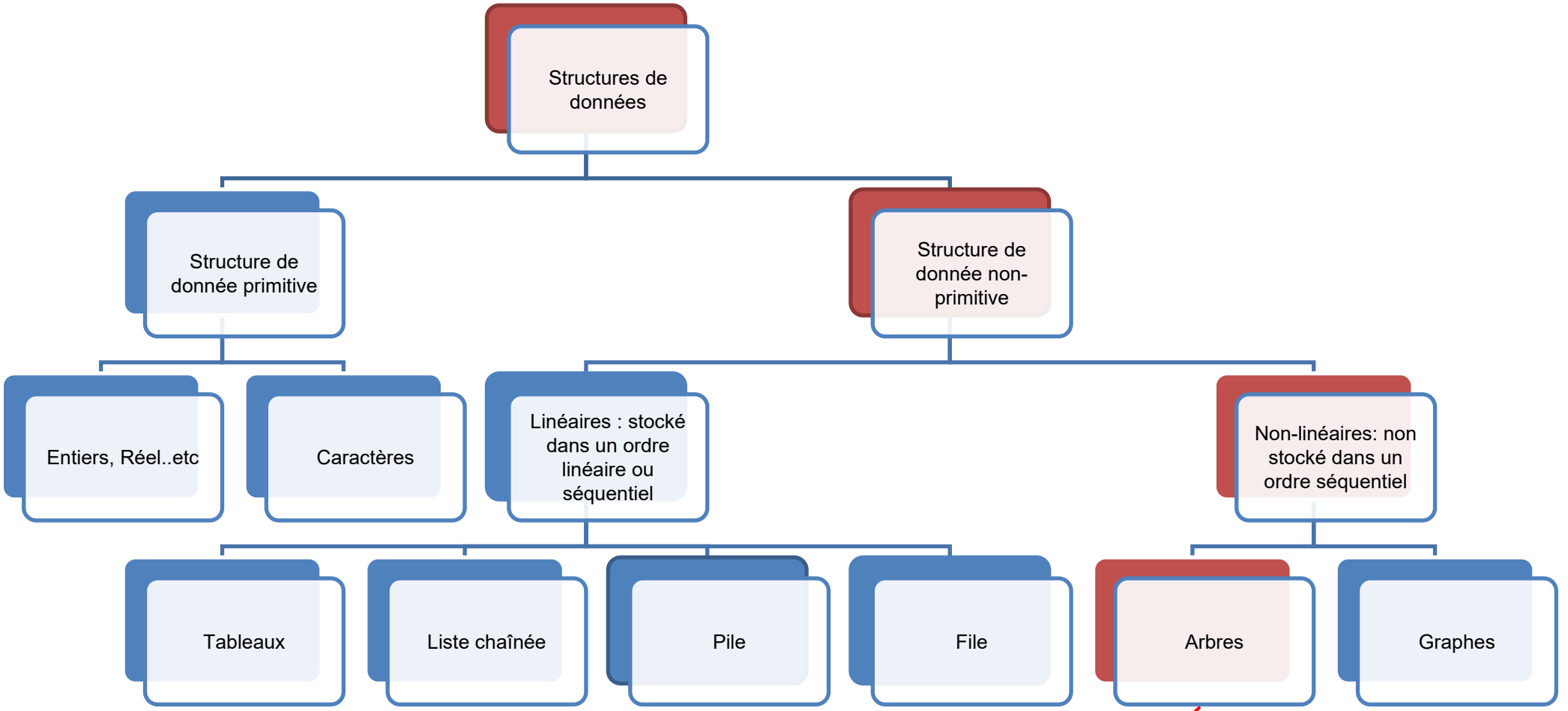
Programmation & Algorithmique II

CM15: Arbres Binaires de Recherche (ABR)

Binary search tree

CLASSIFICATION DES STRUCTURES DE DONNÉES

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



PLAN

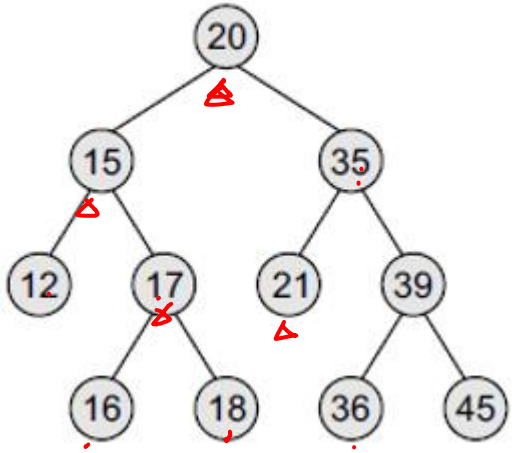
1. Introduction
2. Définition
3. Types des arbres
    1. Arbres généraux
    2. Forêts
    3. Arbres binaires
    4. Créer un arbre binaire à partir d'un arbre général
    5. Parcours d'un arbre binaire
    6. Opérations sur les arbres binaires
    7. Codage (arbre) Huffman
    8. Arbres Binaires de Recherche
        1. Définition
        2. Parcours
        3. Opérations
            1. Recherche
            2. Insertion
            3. Suppression
        4. Applications : Tri par ABR

DÉFINITION

- **QU'EST-CE QU'UN ARBRE BINAIRE DE RECHERCHE (Binary search tree en anglais)?**

Un Arbre Binaire de Recherche (ABR) est un arbre binaire ordonné tel que pour tout noeud **n** :

- Toutes les valeurs du sous-arbre gauche de **n** sont strictement inférieures à la valeur de n, et
- Toutes les valeurs du sous-arbre droit de **n** sont supérieures ou égales à la valeur de **n**.

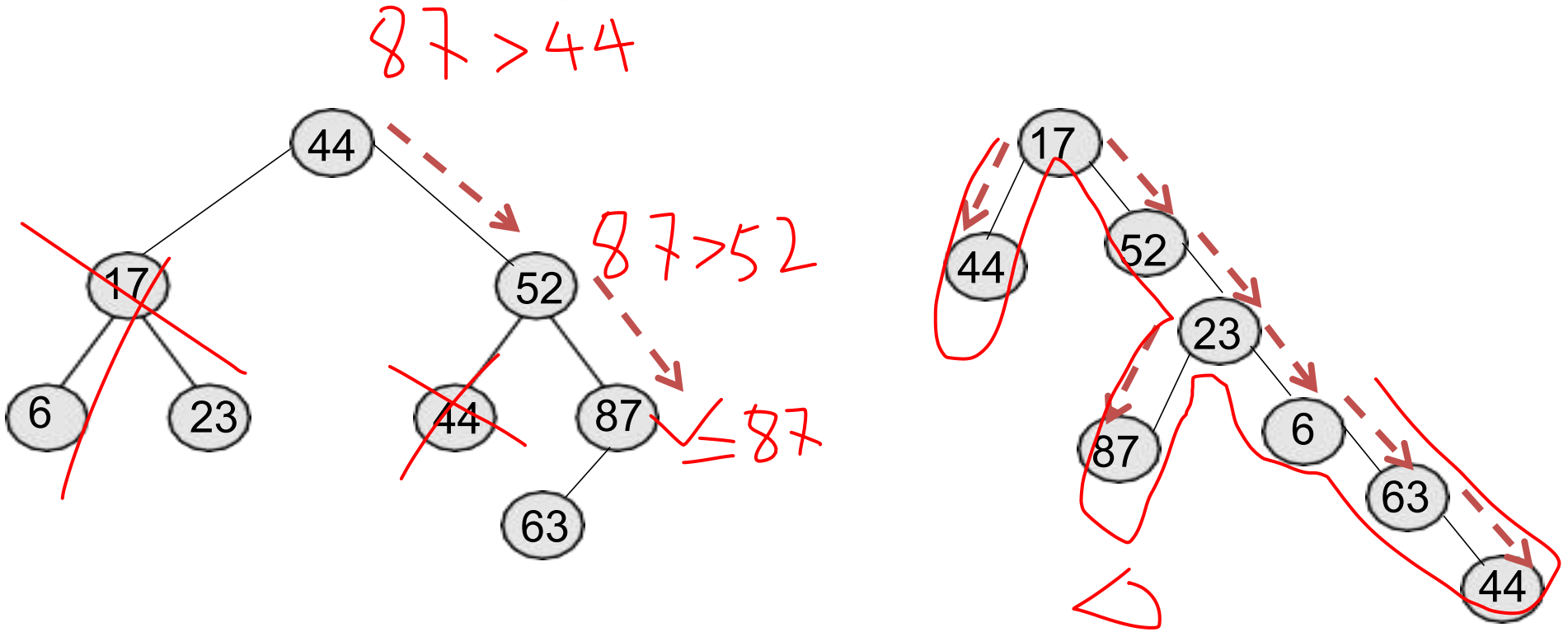



```
typedef struct node
{
    unsigned int key;
    struct node *left;
    struct node *right;
} node ;
```

DÉFINITION (2)

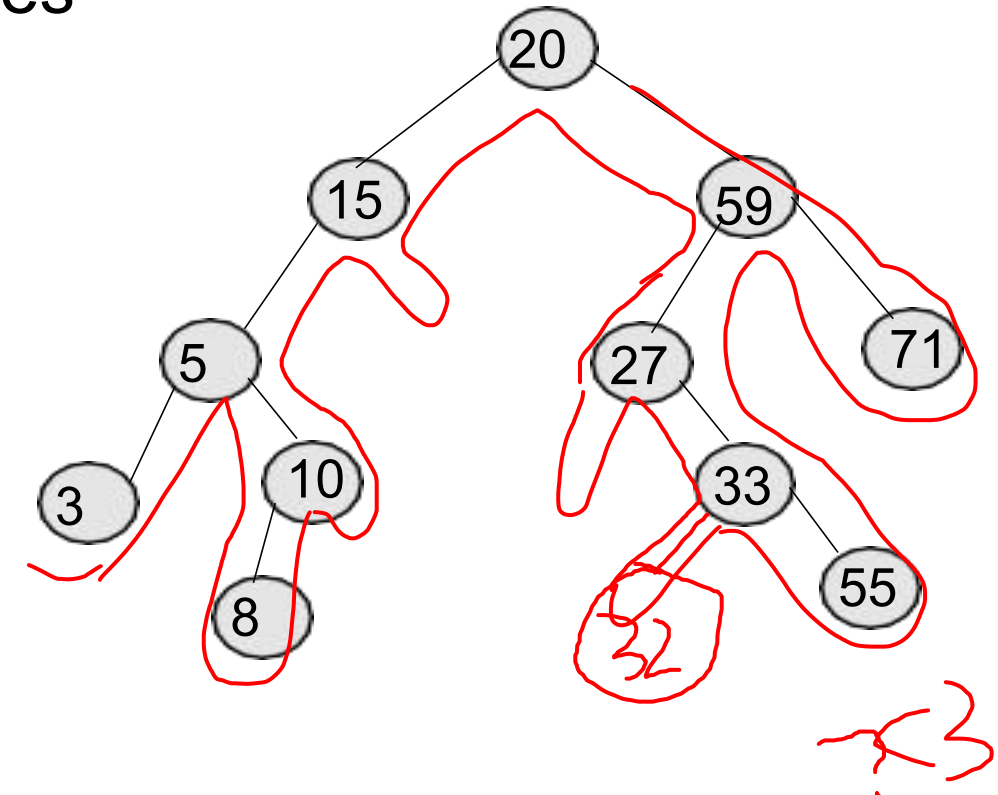
Intérêt de cette propriété : diminuer la complexité temporel de recherche, d'insertion et de suppression dans l'arbre.

- Par exemple, pour l'opération de recherche du nœud 87 : Dans un ABR on visite 2 nœuds seulement, alors que dans un arbre binaire classique, on doit visiter tous les nœuds



PARCOURS D'UN ABR

- Voici un exemple d'un ABR contenant des valeurs entières, appliquer les différents parcours vus sur les arbres binaires

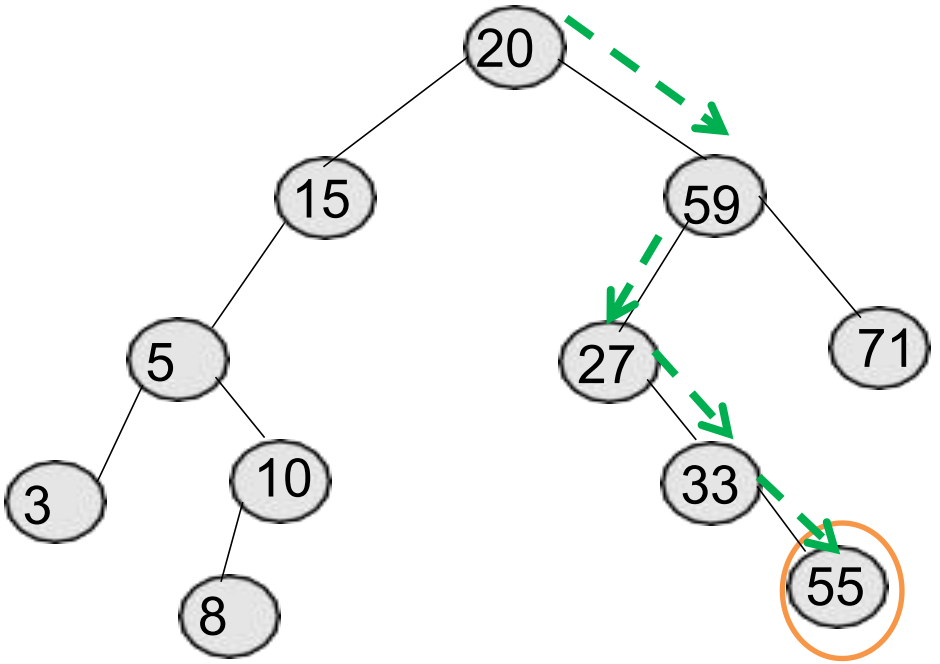


parcour g
traitement
parcour d

- Le parcours infixe (inordre) de cet arbre donne la liste ordonnée suivante : **3, 5, 8, 10, 15, 20, 27, 33, 52, 55, 59, 71**

OPERATIONS DE RECHERCHE

- La recherche est **dichotomique**, à chaque étape, un sous arbre est éliminé:
  - Rechercher (55)
  - Rechercher (Right(20))
  - Rechercher (Left(59))
  - Rechercher (Right(27))
  - Rechercher (Right(33))
  - Élément trouvé



OPERATIONS DE RECHERCHE

- La recherche est dichotomique : implémentation itérative

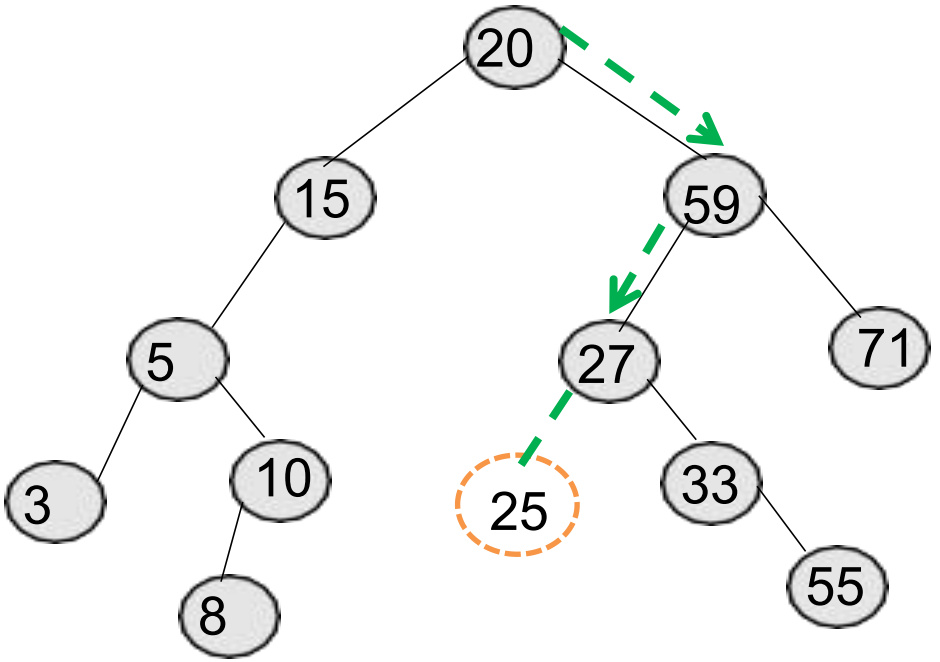
```
//itératif
int searchNode(node *tree, int key)
{
    while(tree)
    {
        if(key == tree->key) return 1;

        if(key > tree->key ) tree = tree->right;
        else tree = tree->left;
    }
    return 0;
}
```

OPÉRATION D'INSERTION

- L'insertion d'un élément se fait toujours au niveau d'une feuille. Cette insertion dans un ABR doit maintenir la propriété des arbres de recherche, ainsi :
  - Rechercher la position d'insertion
  - Raccorder le nouveau nœud à son parent

Example : Insérer(25)



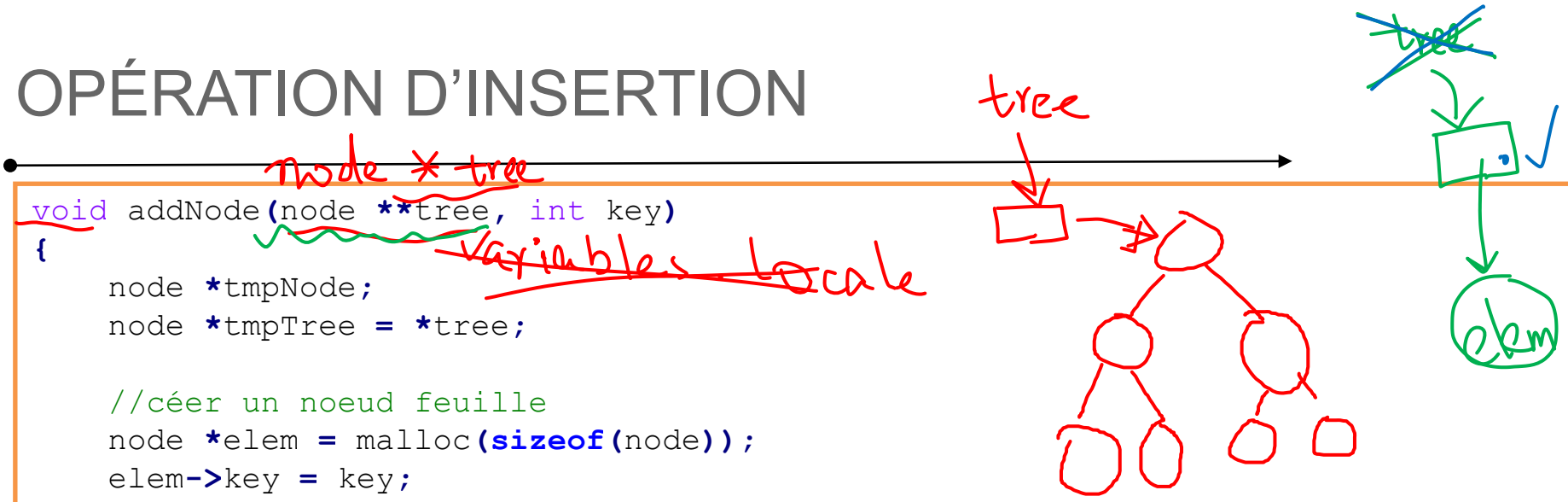
- RechercherPosition (25)
- Rechercher (Right(20))
- Rechercher (Left(59))
- Position trouvé pour l'insertion, le père est le nœud 27
- Insérer 25 au niveau de la feuille dont le père est 27

OPÉRATION D'INSERTION

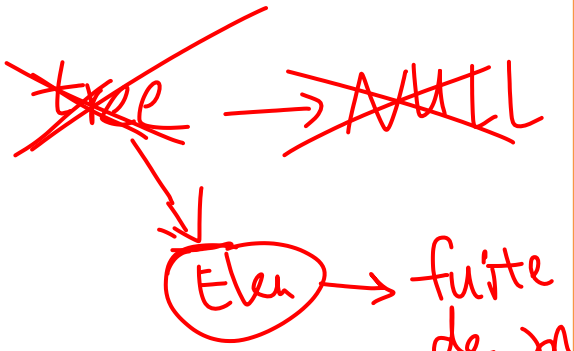
```
void addNode(node **tree, int key)
{
     node *tmpNode;
     node *tmpTree = *tree;

     //céer un noeud feuille
     node *elem = malloc(sizeof(node));
     elem->key = key;
     elem->left = NULL;
     elem->right = NULL;

     if(tmpTree)
       do
       {
            tmpNode = tmpTree;
            if(key > tmpTree->key )
            {
                 tmpTree = tmpTree->right;
                 if(!tmpTree) tmpNode->right = elem;
            }
            else
            {
                 tmpTree = tmpTree->left;
                 if(!tmpTree) tmpNode->left = elem;
            }
       }
       while(tmpTree);
     else  *tree = elem;
}
```



main {
node * root = NULL;
addNode (&root, 5);
Cas Arbre vide



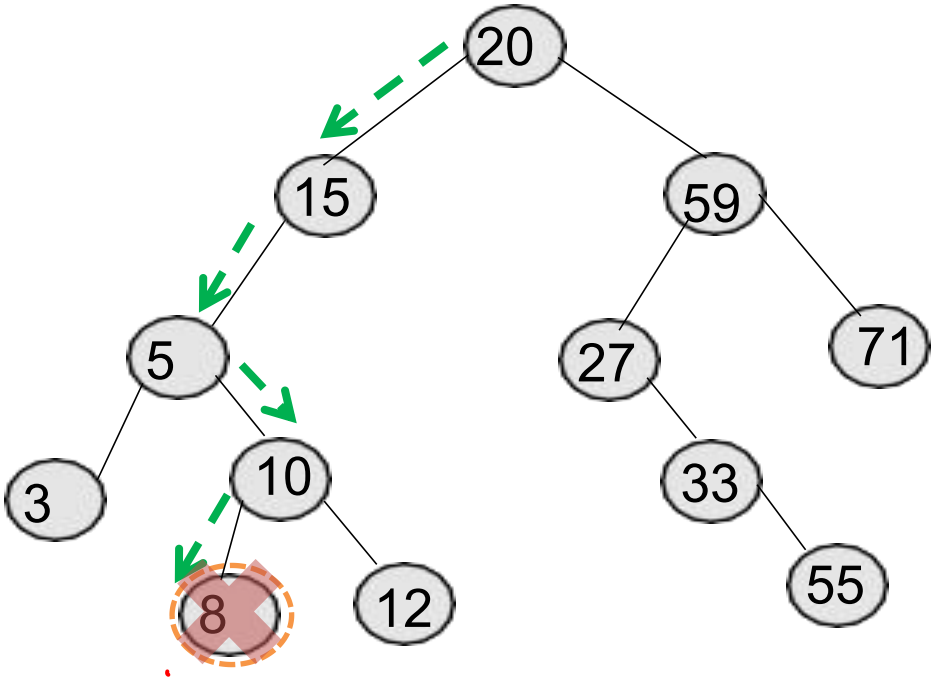
tree = elem

OPÉRATION DE SUPPRESSION

- Pour supprimer le noeud « i » d'un ABR, il faudra le rechercher. Une fois le noeud « i » trouvé, on se trouve dans une des situations suivantes

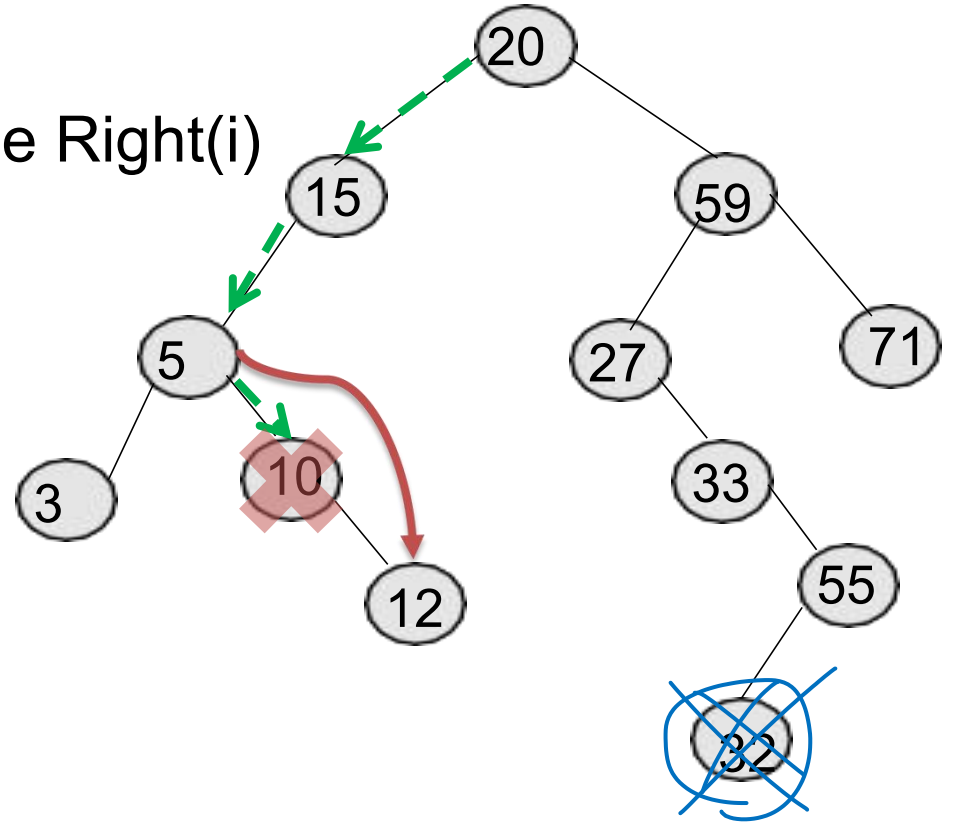
Cas 1 : Suppression d'une feuille

- Il suffit de l'enlever de l'arbre vu qu'elle n'a pas de fils.
- Exemple: supprimer le nœud i qui contient la valeur 8
  1. Rechercher(8)
  2. Libérer le nœud « i »



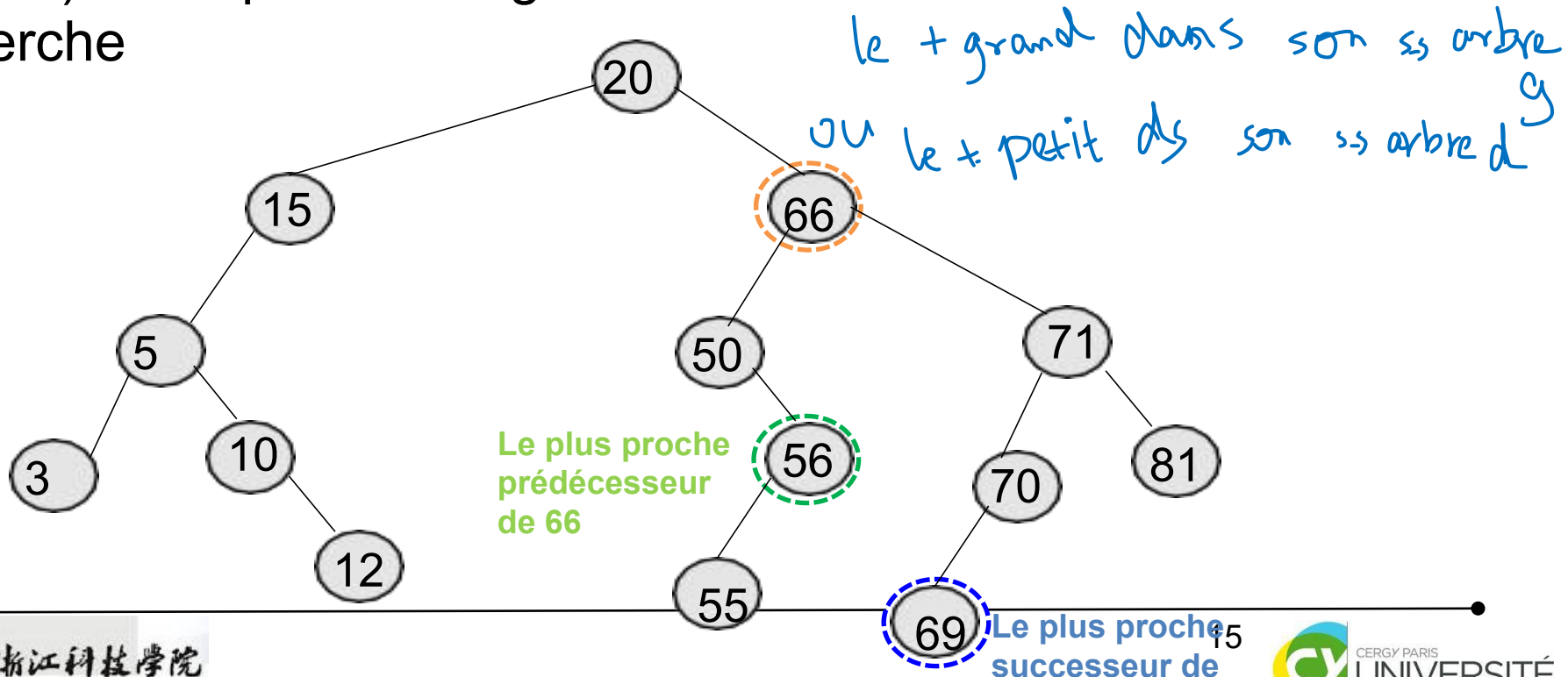
Cas 2 : Suppression d'un nœud avec un fils

- Il faut l'enlever de l'arbre en le remplaçant par son fils.
- Exemple: supprimer le noeud i qui contient la valeur 10
  1. Rechercher(10)
  2. Chainer le père de i avec le Right(i)
  3. Libérer le nœud « i »



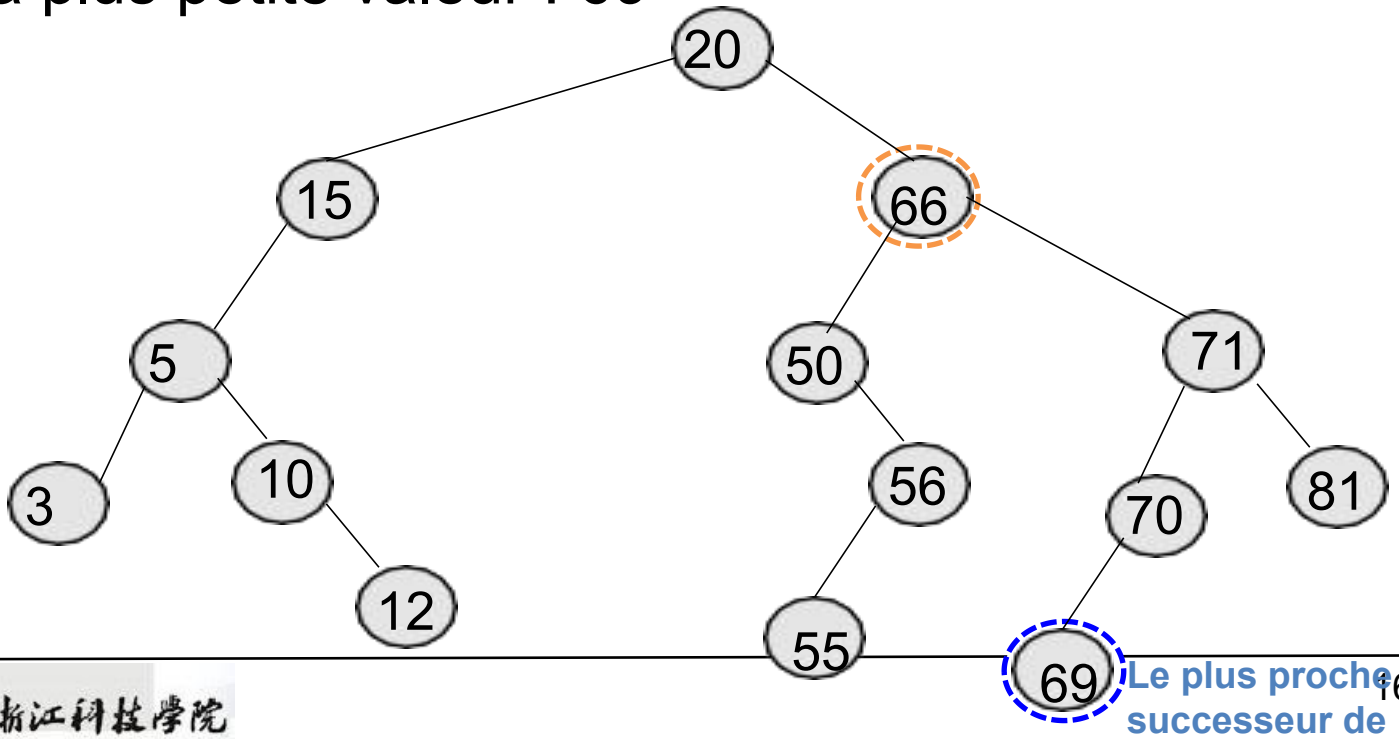
Cas 3: Suppression d'un nœud avec deux fils

- Etape 1: On échange le nœud à supprimer avec ***« CAS A »*** **Le plus grand dans son sous-arbre gauche** (le nœud le plus à gauche du sous-arbre droit) **OU** ***« CAS B »*** **le plus petit dans son sous-arbre droit** (le nœud le plus à droite du sous-arbre gauche). Cela permet de garder une structure d'arbre binaire de recherche



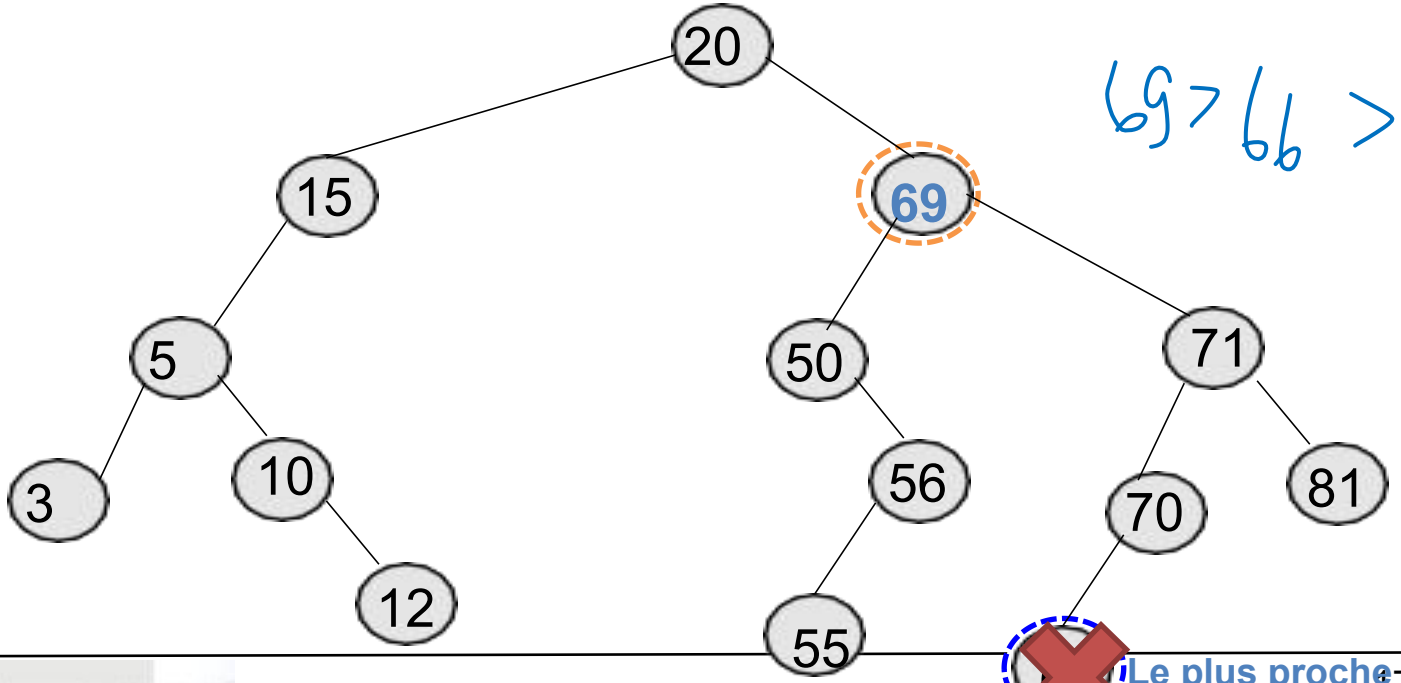
Cas 3: Suppression d'un nœud avec deux fils

- Etape 1 → Cas A: On échange le nœud à supprimer avec **son successeur le plus proche** (le nœud le plus grand à gauche ou le plus petit du sous-arbre droit)
  - Racine: 71
  - La plus petite valeur : 69



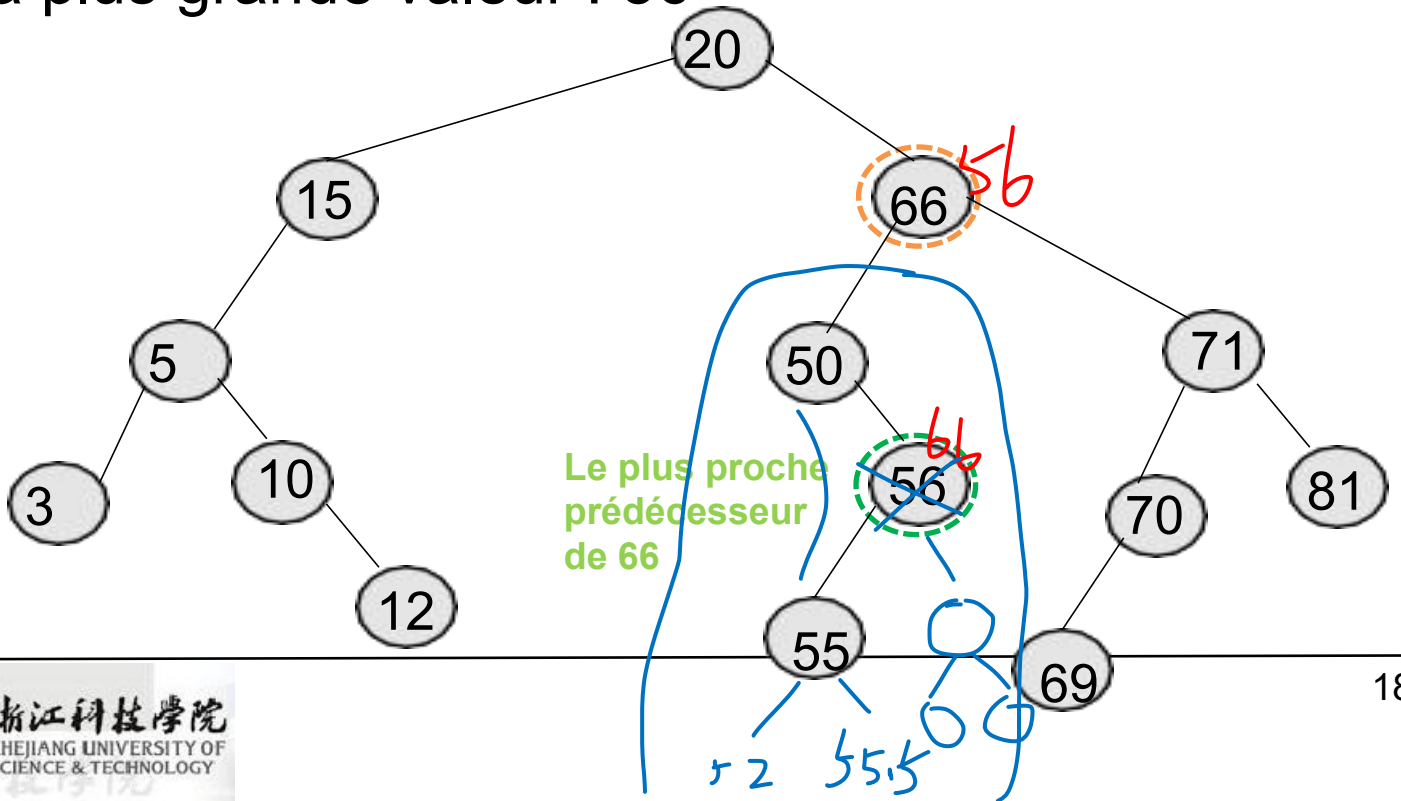
Cas 3: Suppression d'un nœud avec deux fils

- Etape 2 → Cas A : on applique à nouveau la procédure de suppression qui est maintenant une feuille ou un nœud avec un seul fils.
- Ainsi, si on choisit d'échanger le nœud « 66 » avec son plus proche successeur « 69 », on obtient



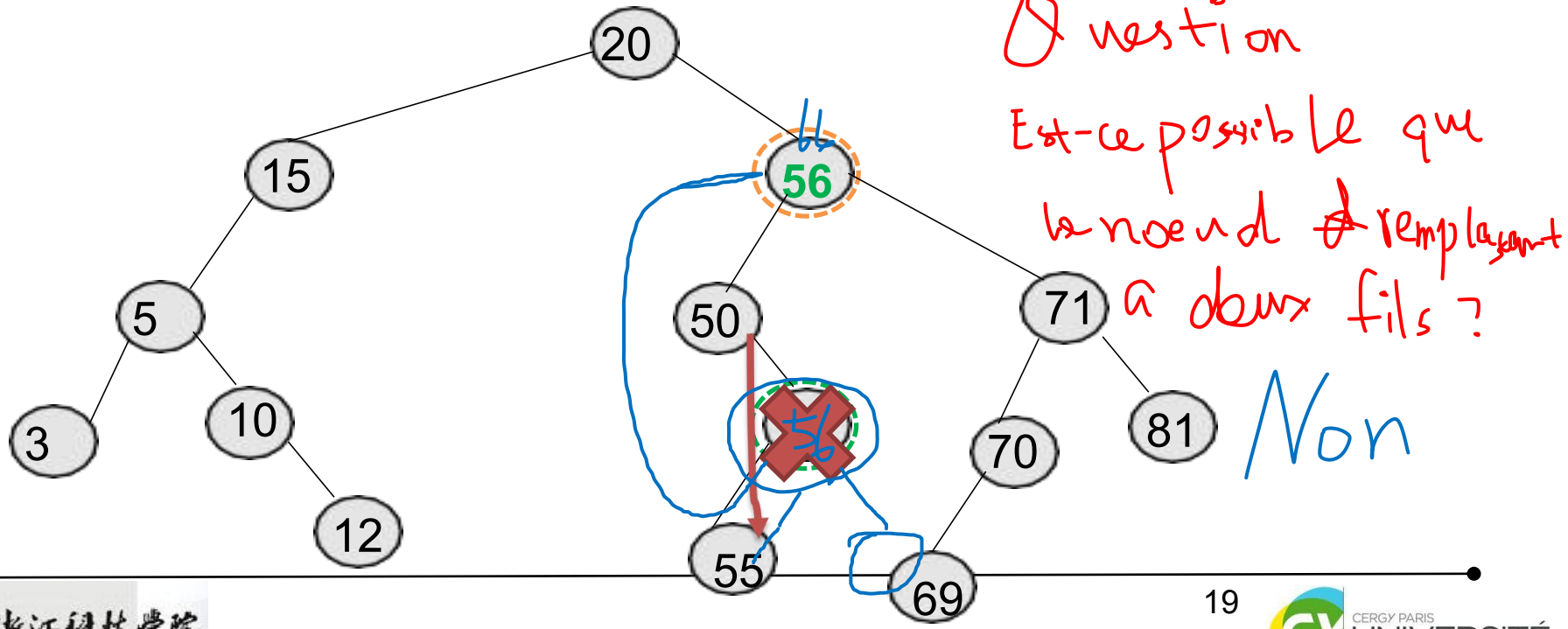
Cas 3: Suppression d'un nœud avec deux fils

- Etape 1 → Cas B: On échange le nœud à supprimer avec **son prédécesseur le plus proche** (le nœud le plus à droite ou le plus grand du sous-arbre gauche)
  - Racine: 50
  - La plus grande valeur : 56



Cas 3: Suppression d'un nœud avec deux fils

- Etape 2 → Cas B : Puis on applique à nouveau la procédure de suppression qui est maintenant une feuille ou un nœud avec un seul fils.
- Ainsi, si on choisit d' échanger le nœud « 66 » avec son plus proche prédécesseur « 56 », on obtient



OPÉRATION DE SUPPRESSION

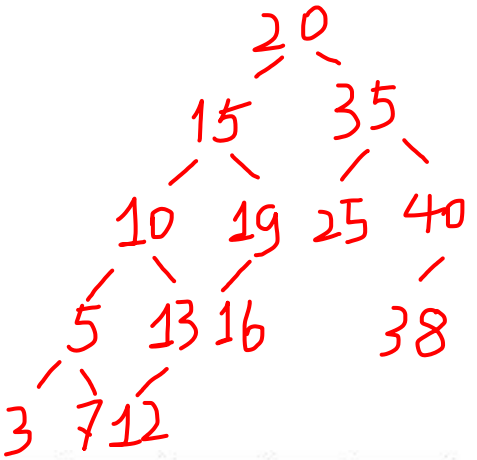
En conclusion, pour supprimer le nœud « i » d'un ARB, il faudra le rechercher. Une fois le nœud «i» trouvé, on se trouve dans une des situations suivantes :

Cas	« i »		Action
	Fils Gauche	Fils Droit	
Feuille	NULL	NULL	Remplacer « i » par NULL
Avec un Fils	NULL	≠NULL	Remplacer « i » par fils droit de « i »
	≠NULL	NULL	Remplacer « i » par fils gauche de « i »
Avec deux fils	≠NULL	≠NULL	1- Rechercher le plus proche prédécesseur ou successeur de « i », soit P. 2. Remplacer valeur de (i) par valeur de (P) 3. Remplacer P par Right(P) ou Left(P)

EXEMPLE D'APPLICATION: TRI PAR ABR

- Étant donné un tableau d'entiers T (n: sa taille), dire comment peut on trier ce tableau en utilisant un Arbre Binaire de Recherche (ABR)?

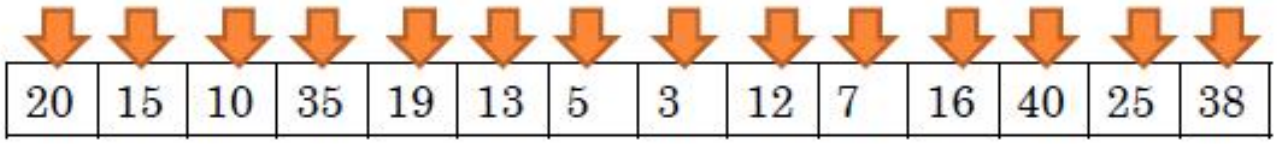
- Exemple:

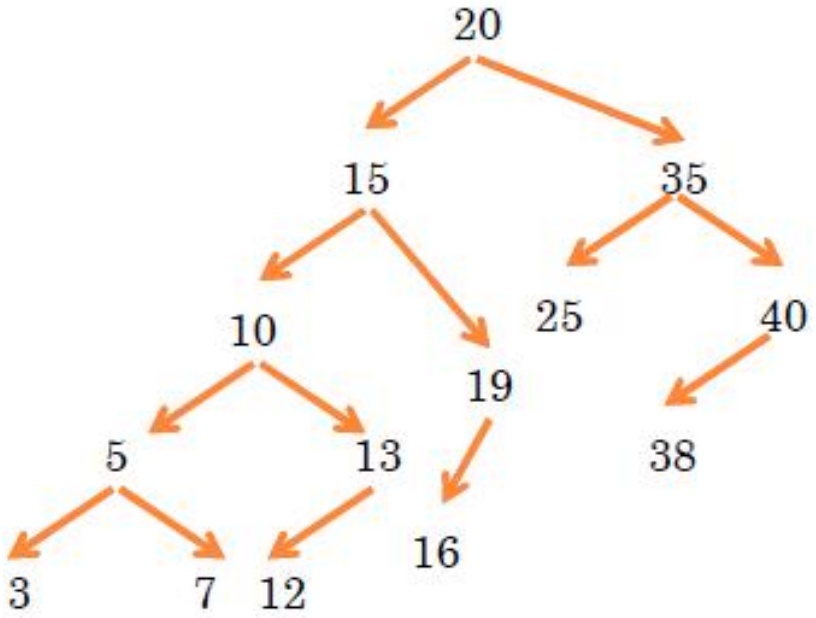


20	15	10	35	19	13	5	3	12	7	16	40	25	38

EXEMPLE D'APPLICATION: TRI PAR ABR

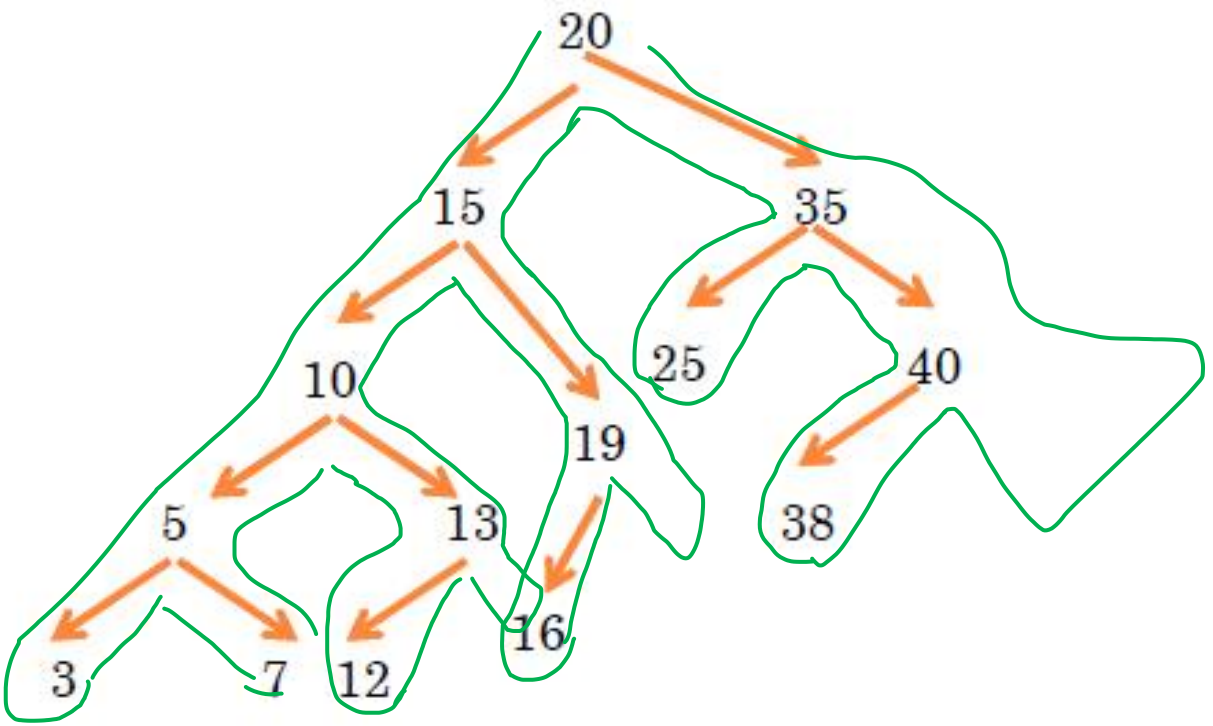
- Insérer toutes les éléments du tableau dans un ABR





EXEMPLE D'APPLICATION: TRI PAR ABR

- Parcourir l'ABR en infixe (in ordrer)



3	5	7	10	12	13	15	16	19	20	25	35	38	40