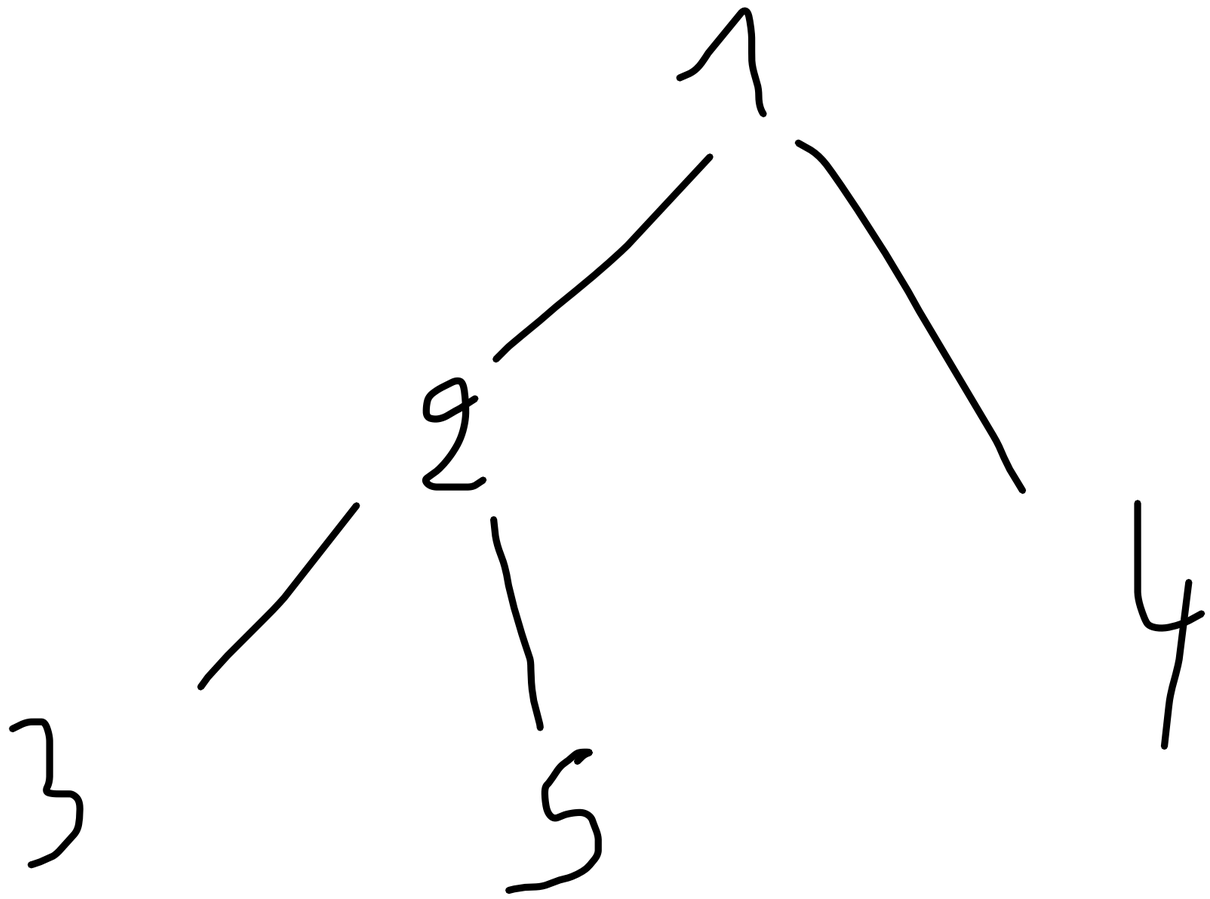
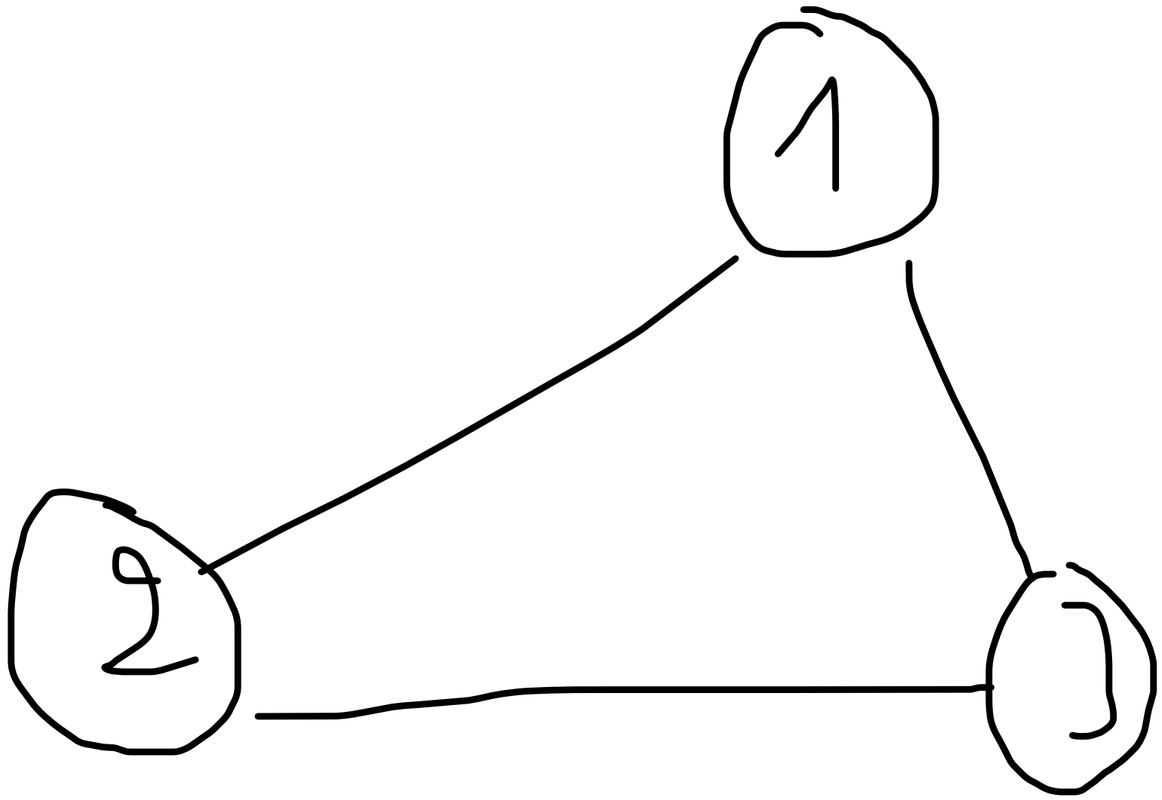
Arbre:Une structure non linéraire

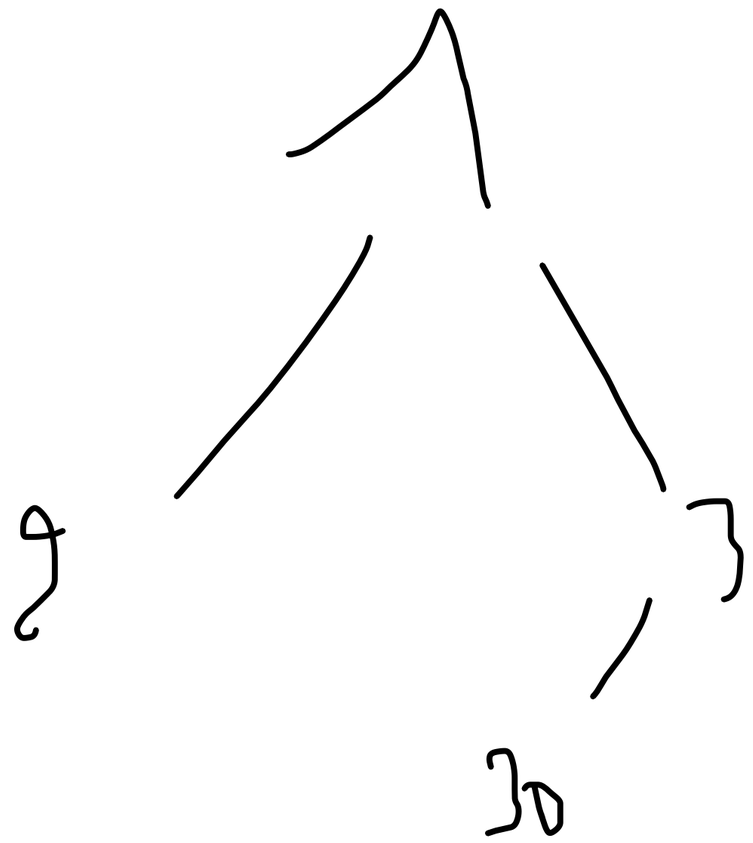


Graphe:Une structure non linéraire



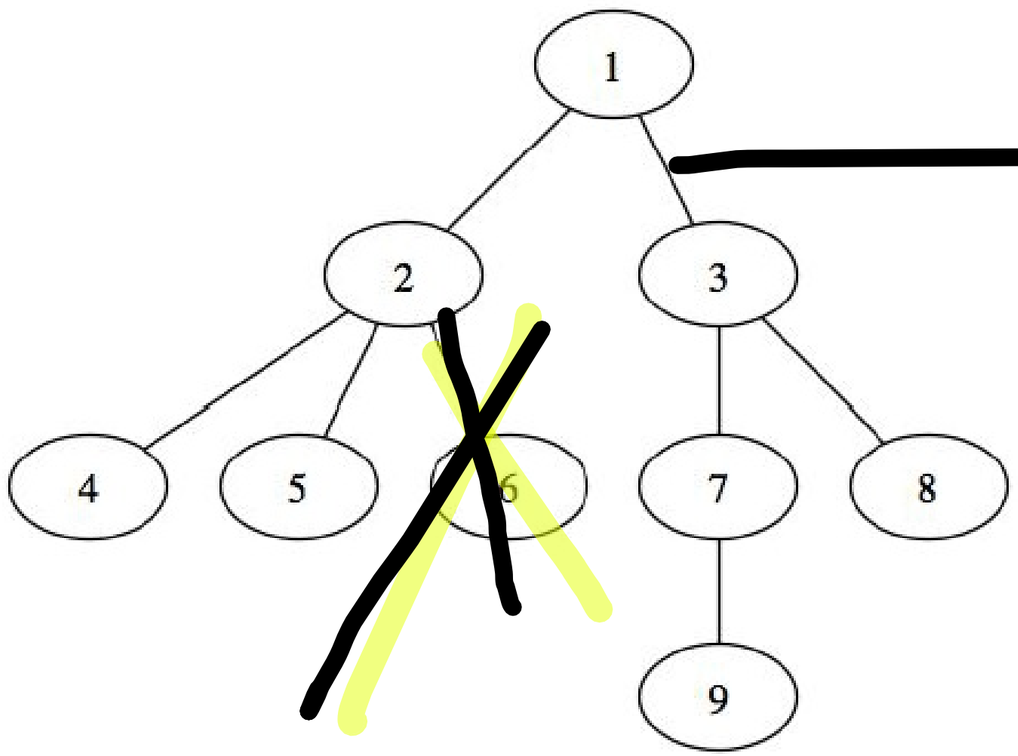

محمد ﷺ
إلياس
هارون
إلياس
يونس
يونس
عمران
مدان
لاوي
يعقوب
يهوذا
داود
سليمان
زكريا
أيوب
ذو الكفل
هارون
إسماعيل
إسحاق
عيص
إبراهيم
آزر
سام
إدريس
حام
نوح
آدم
مدين
شعيب
صالح
حود
يافث

arbre binaire:
est un arbre dont l'arité vaut 2
chaque sommet possède au maximum deux fils



La profondeur d'un sommet

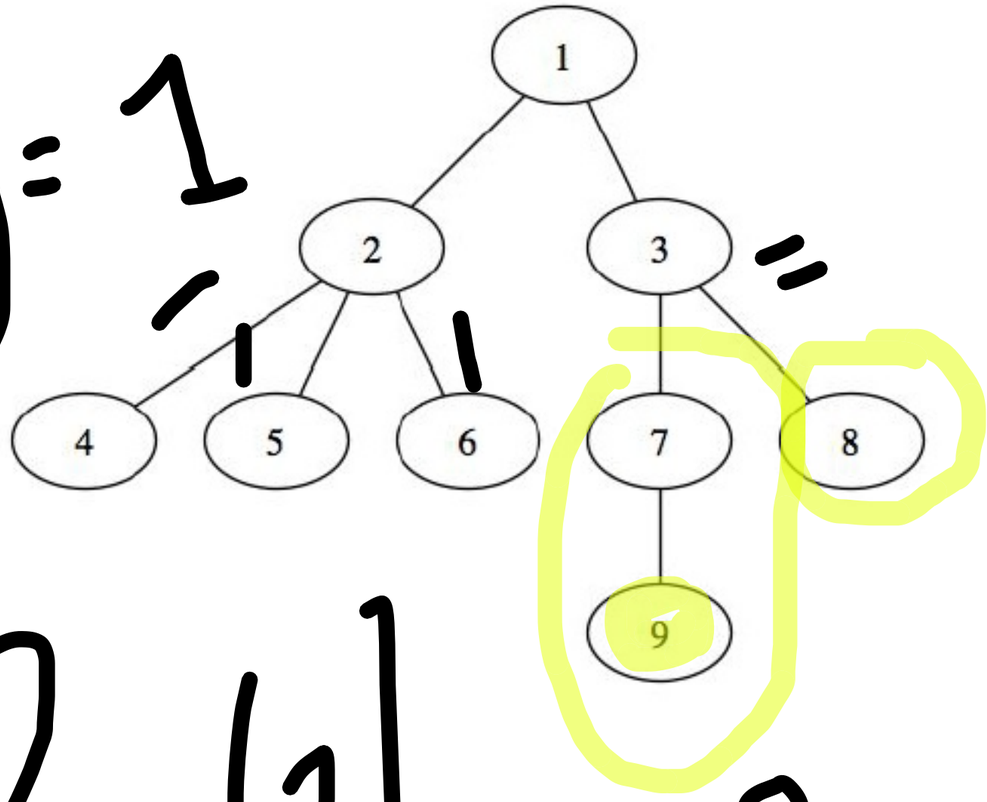
- La profondeur (niveau) d'un noeud est la longueur de la branche depuis la racine
- La profondeur d'un sommet est définie récursivement par :
  - prof(v) = 0 si v est la racine
  - prof(v) = prof(parent(v)) + 1



branche = lien entre deux noeuds

La hauteur d'un sommet

- La hauteur d'un sommet est la plus grande profondeur d'une feuille du sous-arbre dont il est la racine



$h(2) = 1$

$h(8) = 0$

$h(3) = 2$

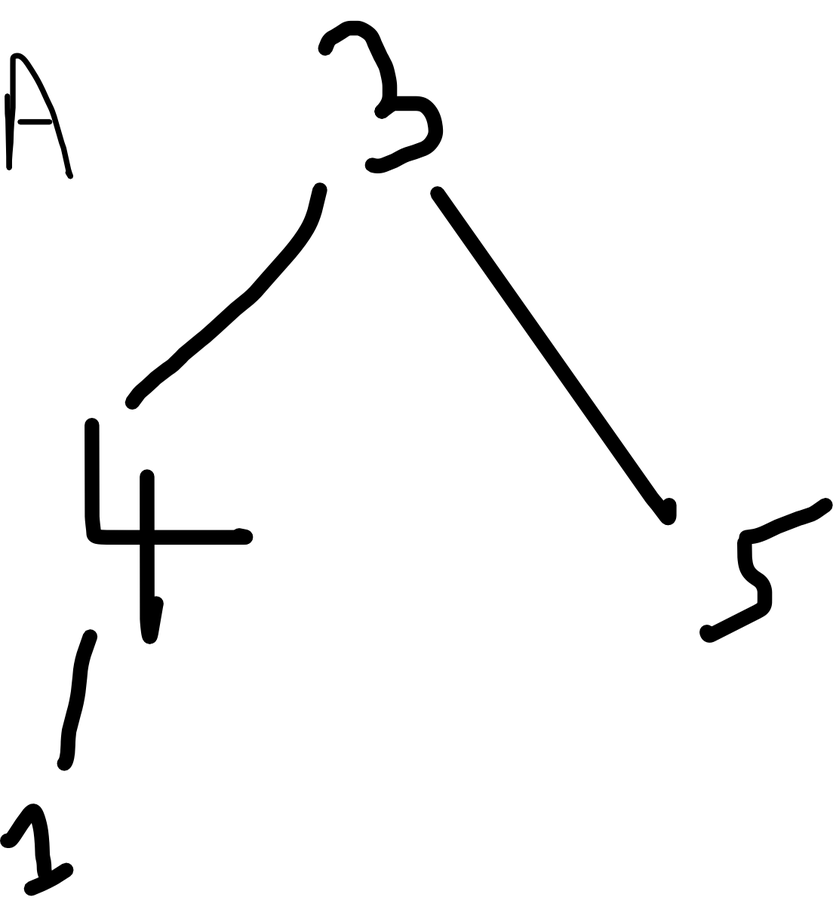
$h(1) = 3$

Qu'est ce qu'un arbre binaire?

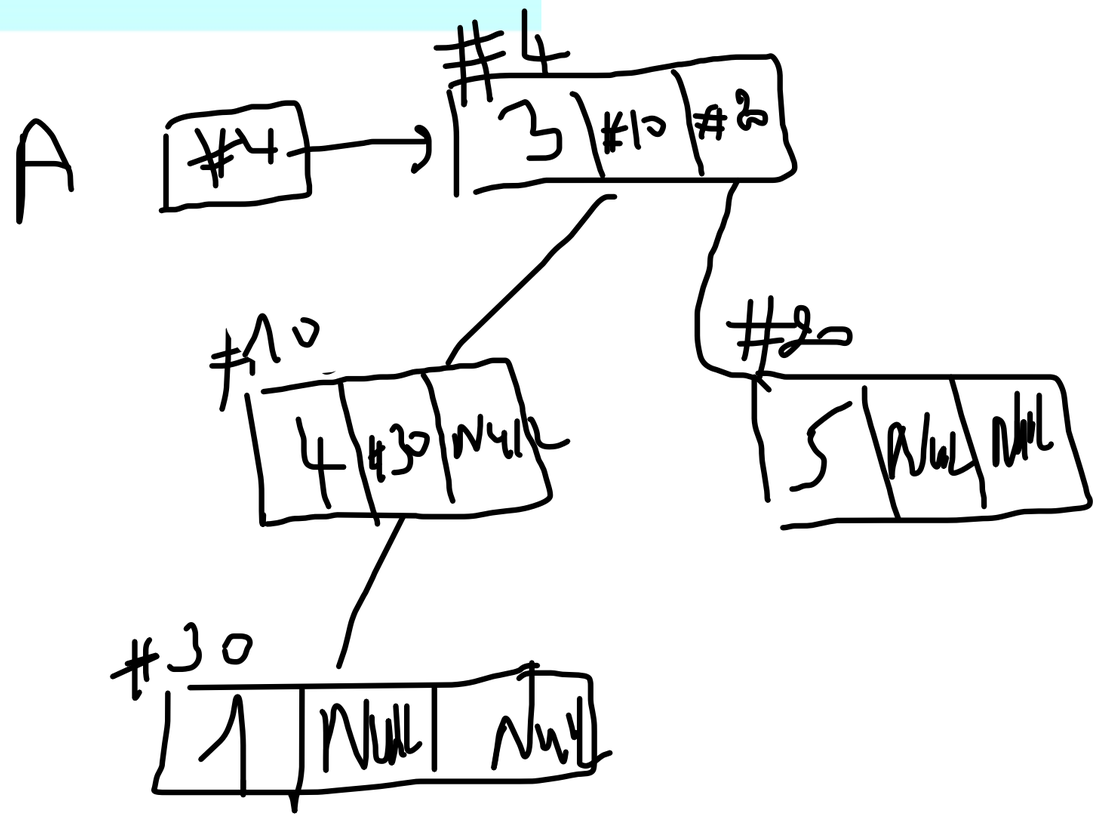
Définition récursive

- Soit vide
- Soit composé
  - d'une racine r
  - de 2 sous arbres binaires ABG et ABD disjoints
    * ABG : sous Arbre Binaire Gauche
    * ABD : sous Arbre Binaire Droit

```c
#include<stdio.h>
#include<stdlib.h>
/* Définition du type noeud d'un arbre binaire */
typedef struct noeud{
int  info; /*le champ etiq peut avoir n'importe quel type*/
struct noeud *fg ; /*pointeur contenant l'adresse du ABG*/
struct noeud *fd ; /*pointeur contenant l'adresse du ABD*/
}Btree ;
```

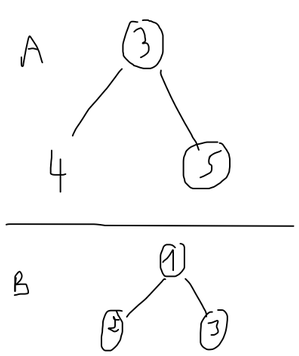


Btree *A;



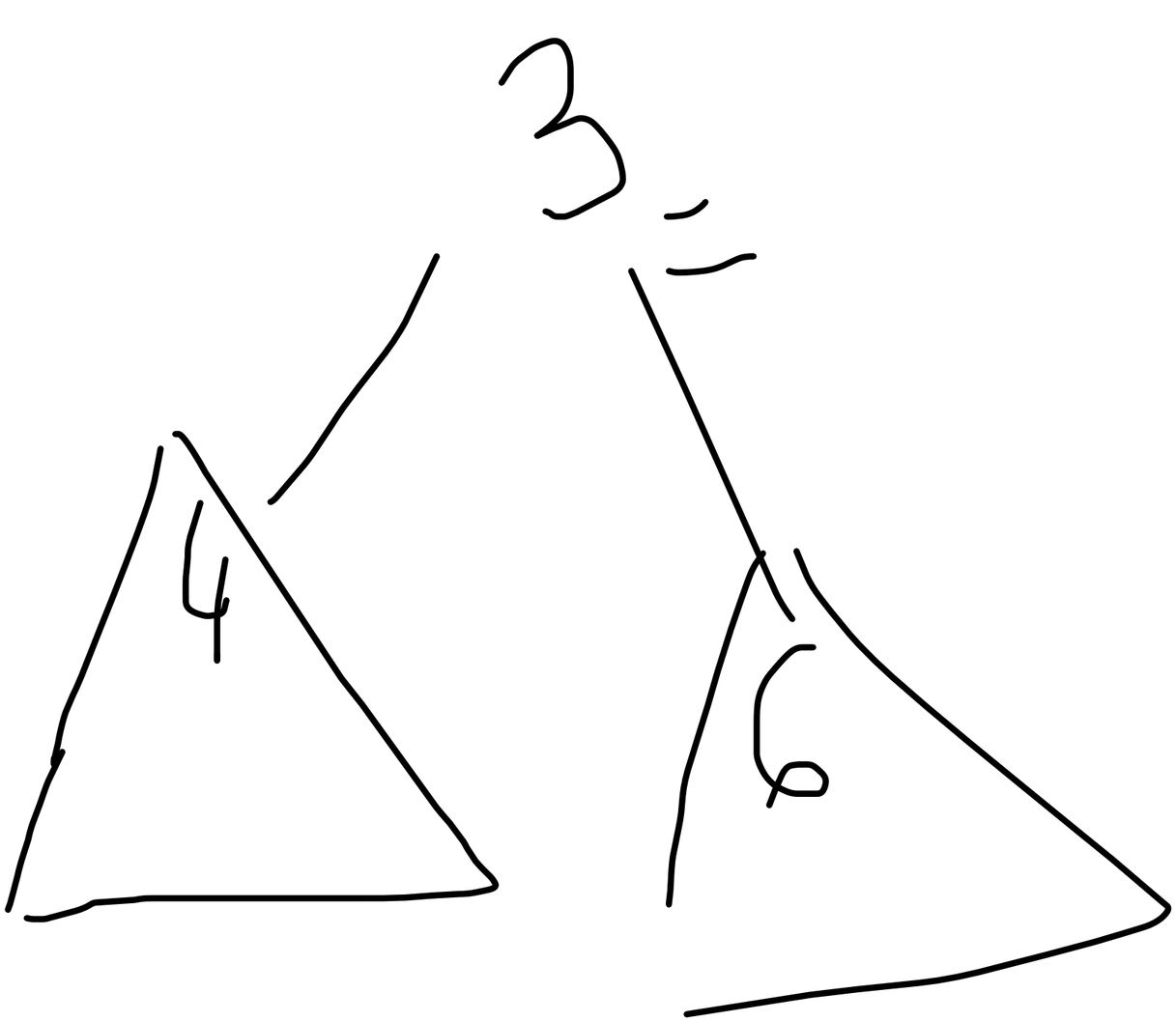
```c
#include<stdio.h>
#include<stdlib.h>
/* Définition du type noeud d'un arbre binaire */
typedef struct noeud{
int  info; /*le champ etiq peut avoir n'importe quel type*/
struct noeud *fg ; /*pointeur contenant l'adresse du ABG*/
struct noeud *fd ; /*pointeur contenant l'adresse du ABD*/
}Btree ;
```

```
//// Primitives
Btree * ArbrVide()
//////////////////////
int EstVide(Btree *A)
//////////////////
Btree *FilsGauche(Btree *A)
//////////////////
Btree*FilsDroit(Btree *A)
////////////////////////
int ValRacine(Btree *A)
//////////////////////
Btree *CreerNoeud(int e,Btree *g,Btree *d)
/////////////////
Btree *CreerFeuille(int e)
/////////////////
Btree *CreerFeuille_Sol2(int e)
////////////////////////////////////////
int EstFeuille(Btree *A)
```



```c
main()
{
    Btree *A,*B,*C;
    A=ArbrVide();
    A=CreerFeuille(1);
    A->fg=CreerFeuille(2);
    A->fd=CreerFeuille(3);
    A->fg->fg=CreerFeuille(4);
    B=ArbrVide();
    B=CreerFeuille(5);
    B->fg=CreerFeuille(6);
    B->fd=CreerFeuille(7);
    //ParcoursPrefixe(A);
    //ParcoursInfixe(A);
    //ParcoursPostfixe(A);
    C=CreerNoeud(0,A,B);
    //ParcoursPrefixe(C);
    printf("%d",RechercheS2(C,5));
}
```

Parcours Prefixe: RGD



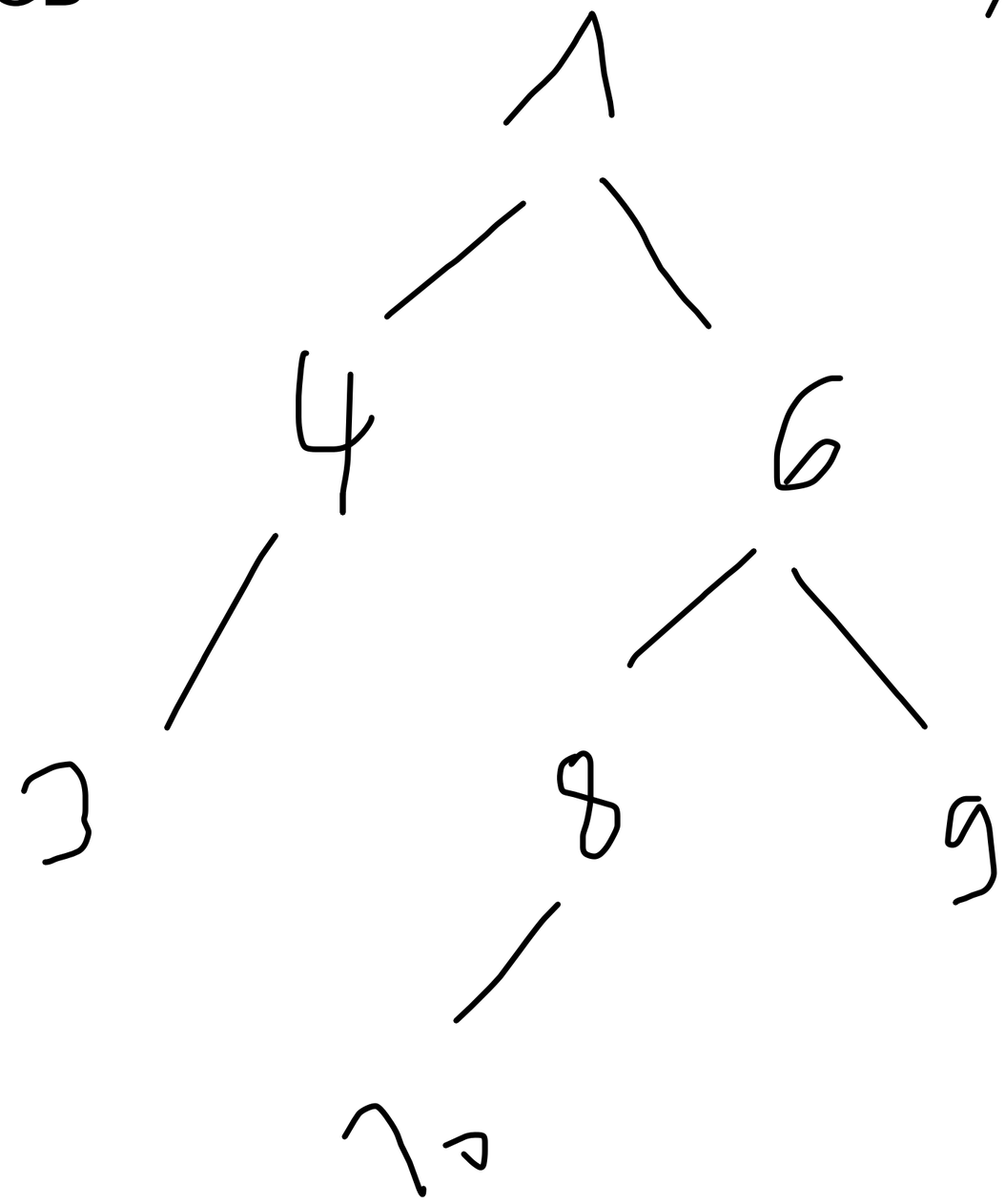
3 4 6

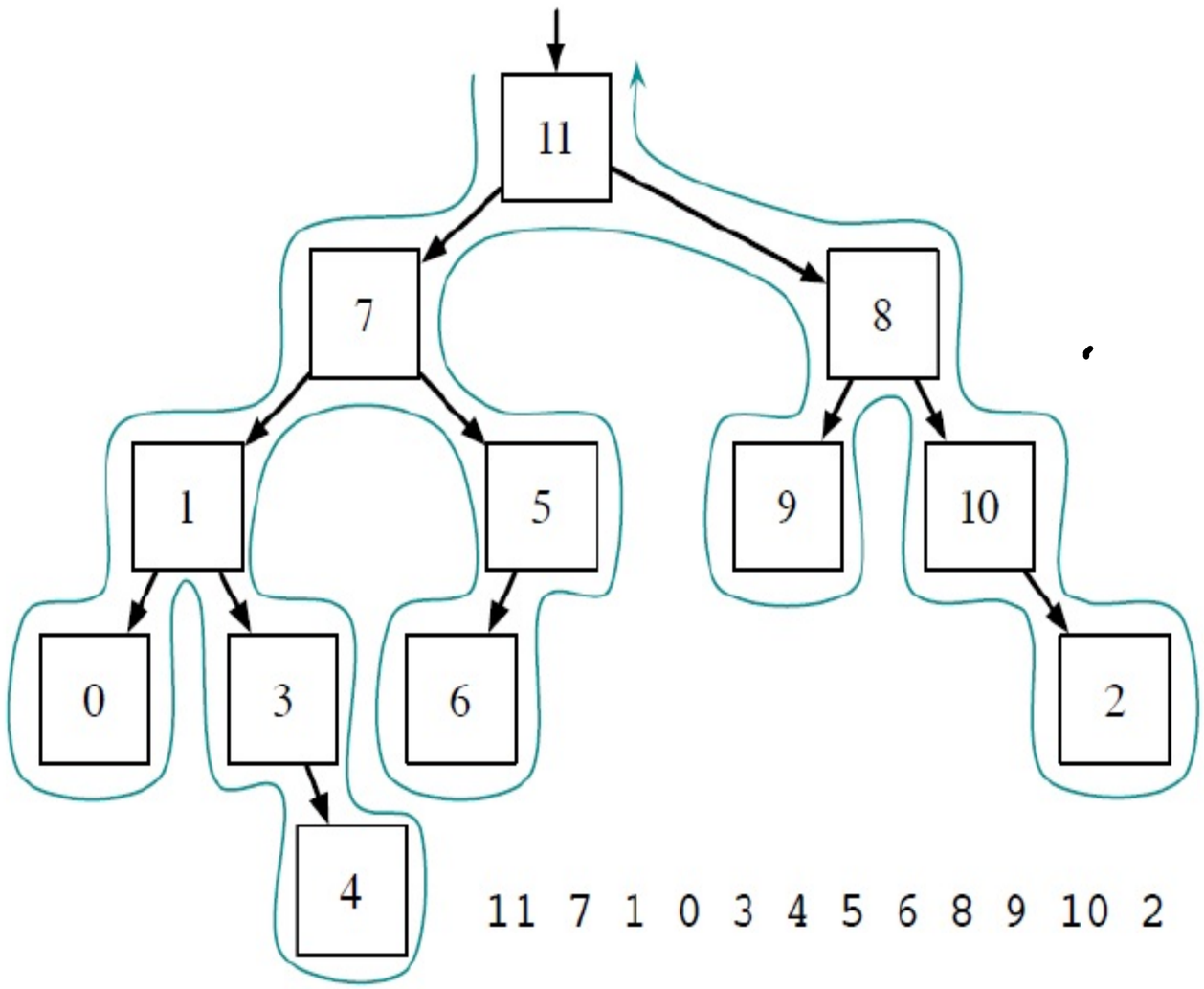
Parcours Prefixe: RGD

1 2 4 5 3

Parcours Prefixe: RGD

1 4 3 6 8 10 9





11 7 1 0 3 4 5 6 8 9 10 2

```
void parcoursPrefixe(Btree *A)
{
if(A!=NULL)
  {
     printf("%d",A->info);
    parcoursPrefixe(A->fg);
    parcoursPrefixe(A->fd)
  }
}
```

C:\Users\Student\Downloads\abrV1.exe

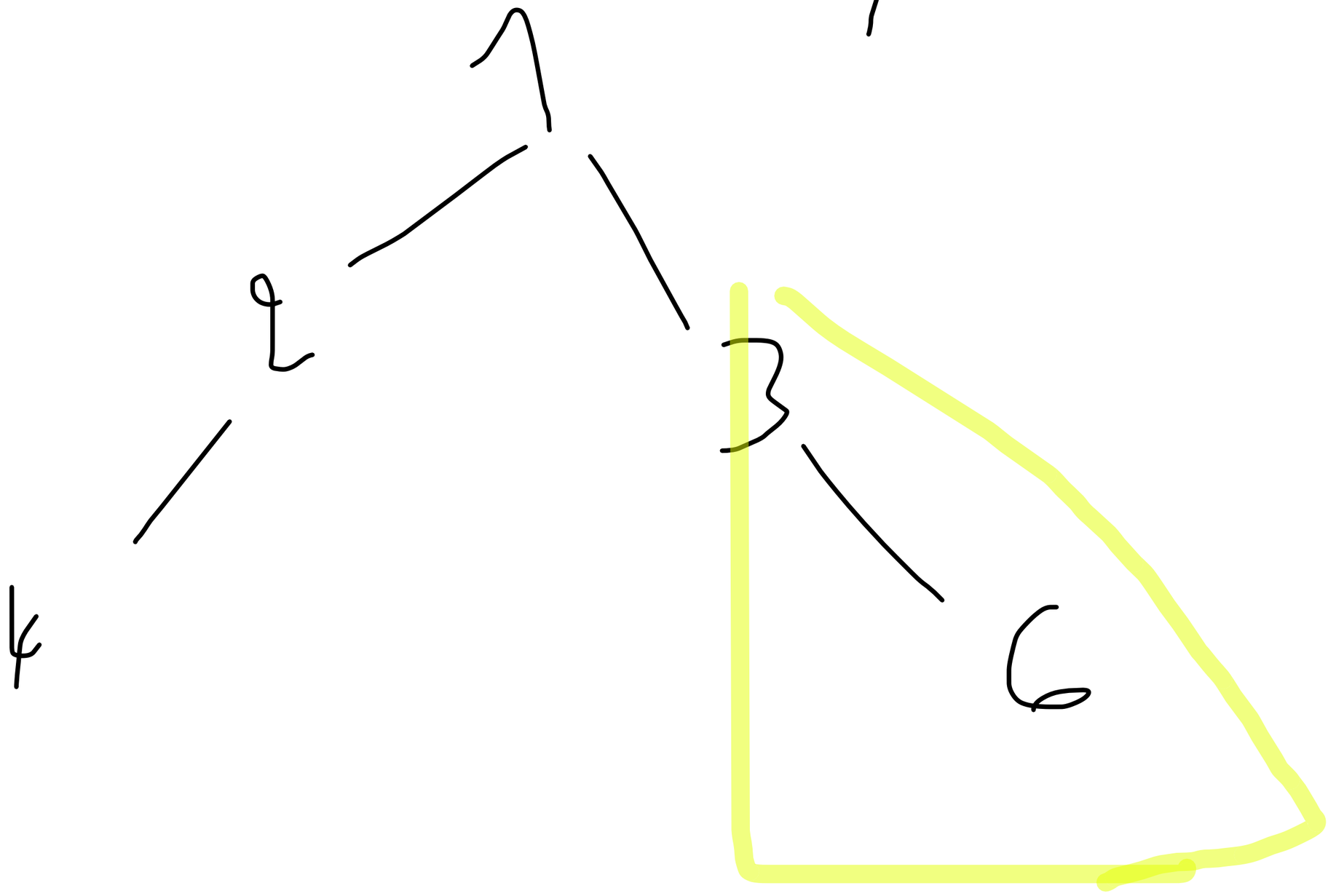
```
0
1
2
3
4
5
6


--------------------------------
Process exited after 1.755 seconds with return value 0
Appuyez sur une touche pour continuer...
```

Parcours Infixe:GRD

2 1 3

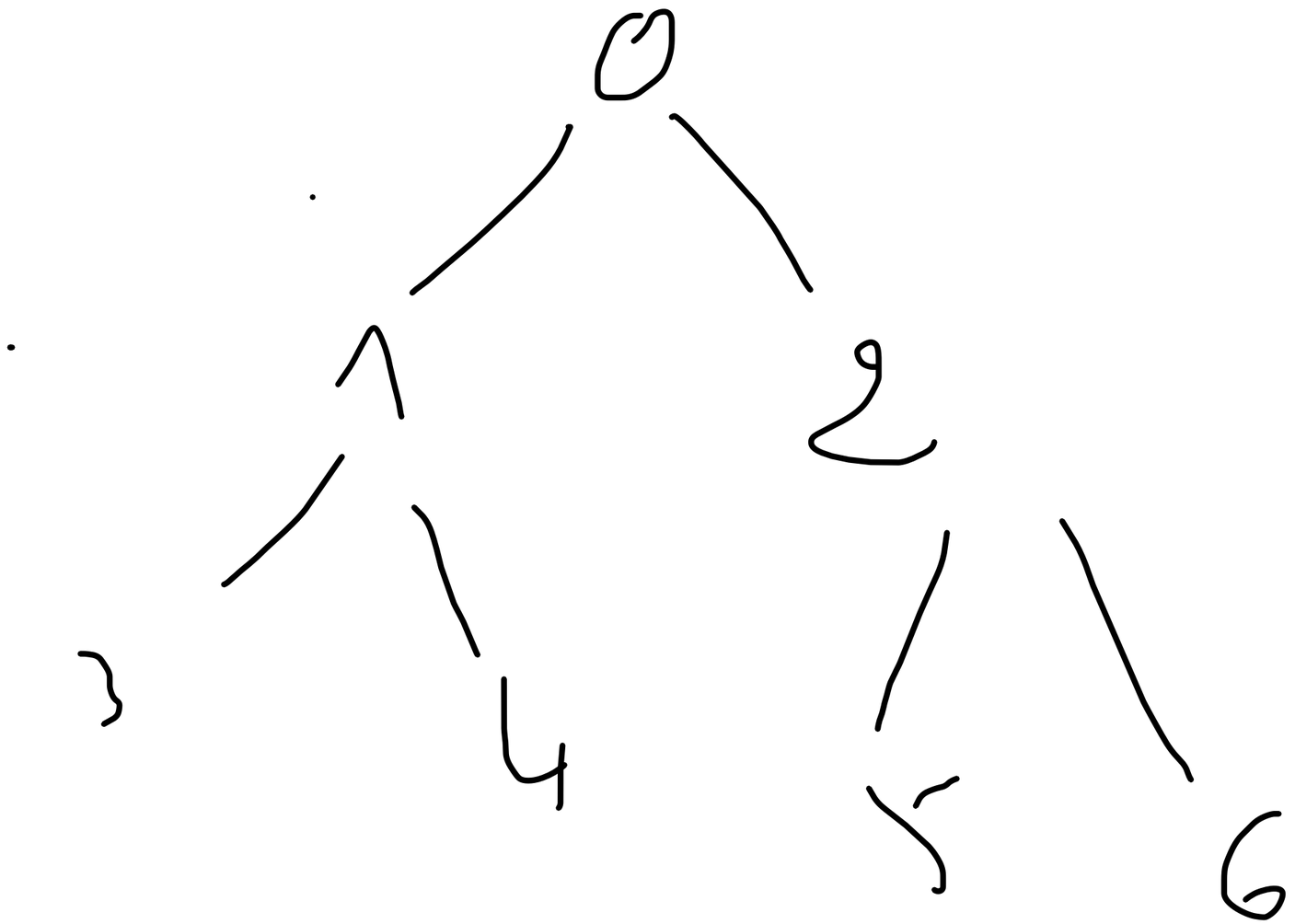
Parcours Infixe:GRD

4 2 1 3 6

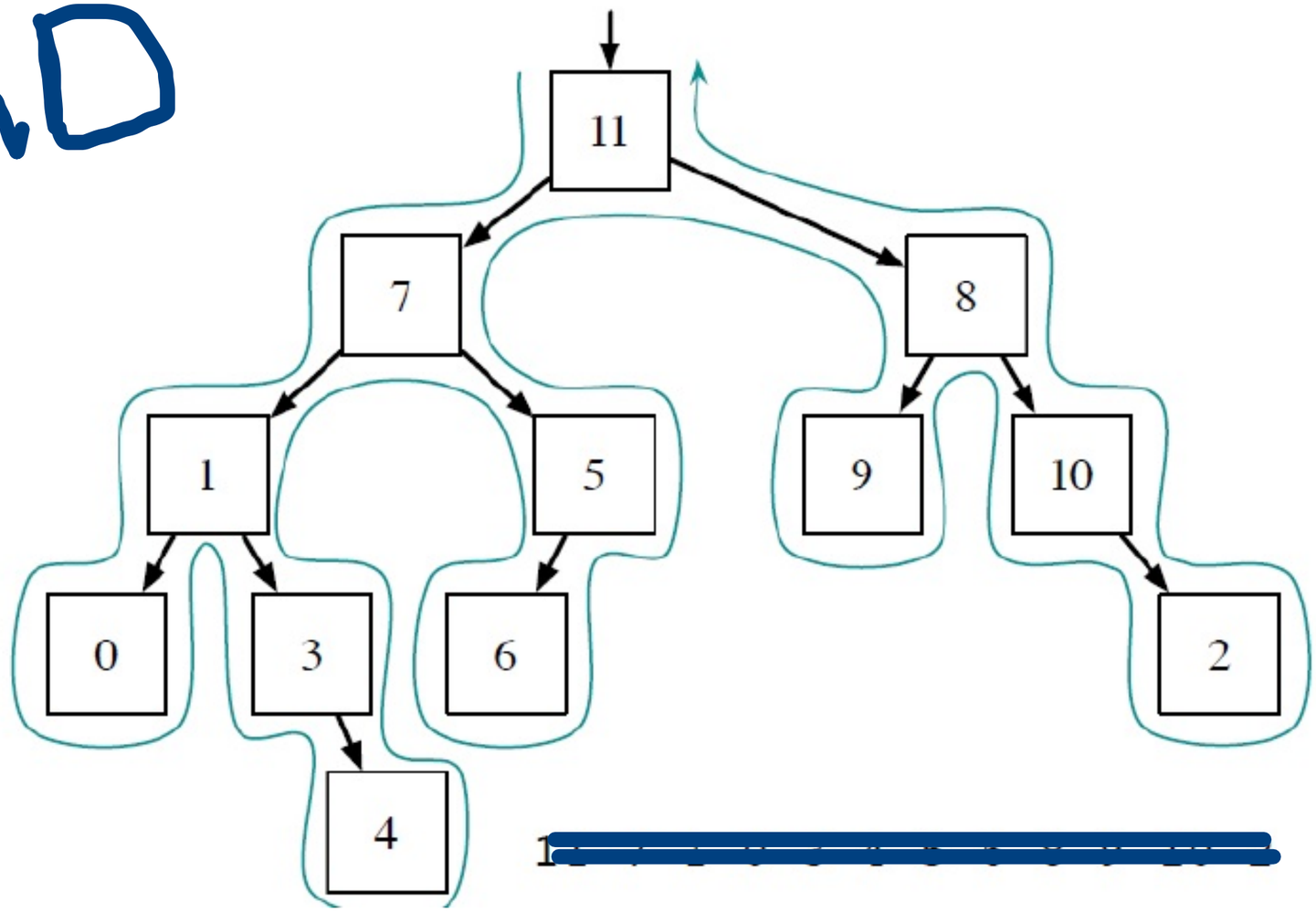


3 1 4 0 5 2 6

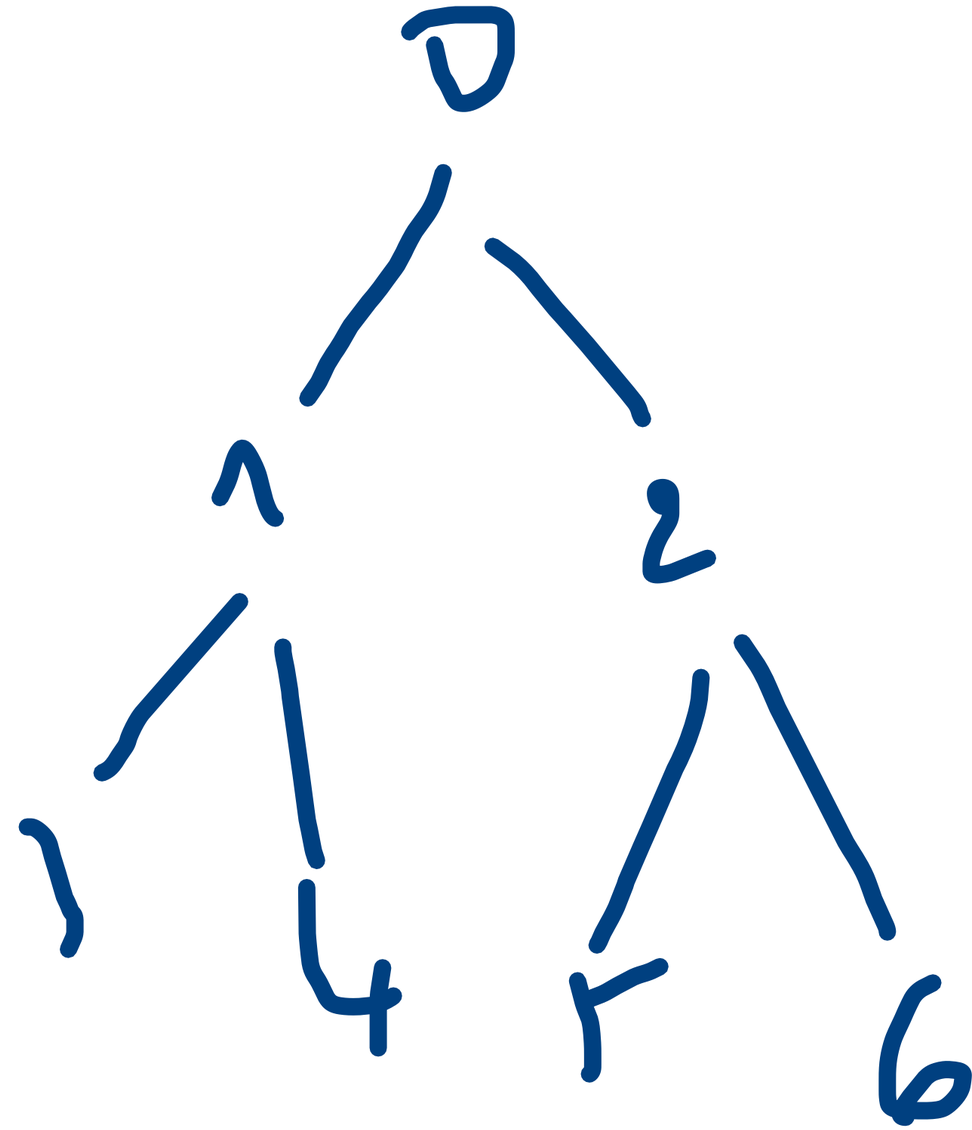
Parcours Infixe:GRD



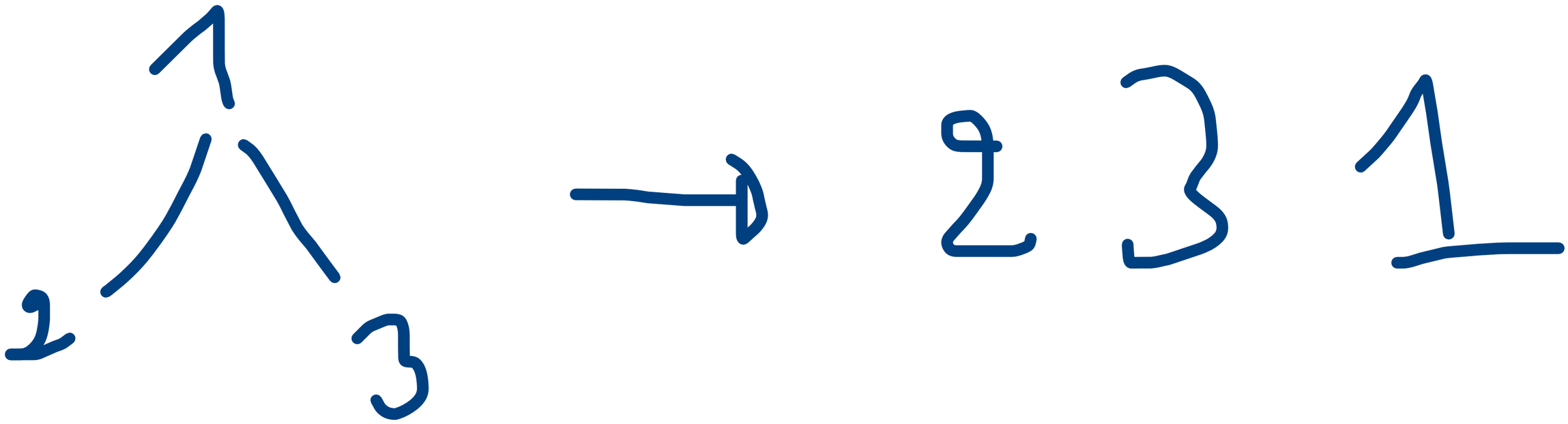
GRD



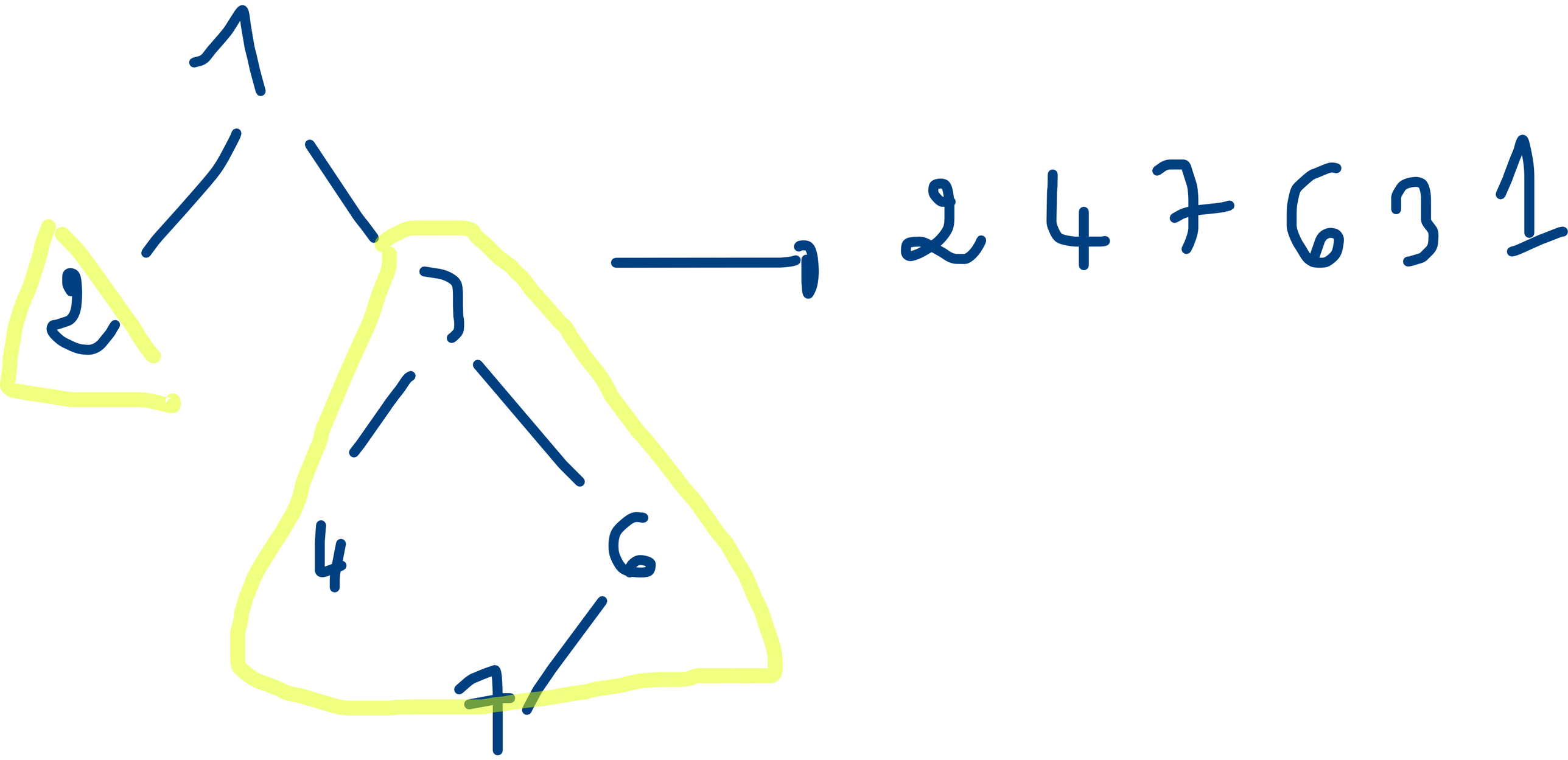
Parcours Infixe



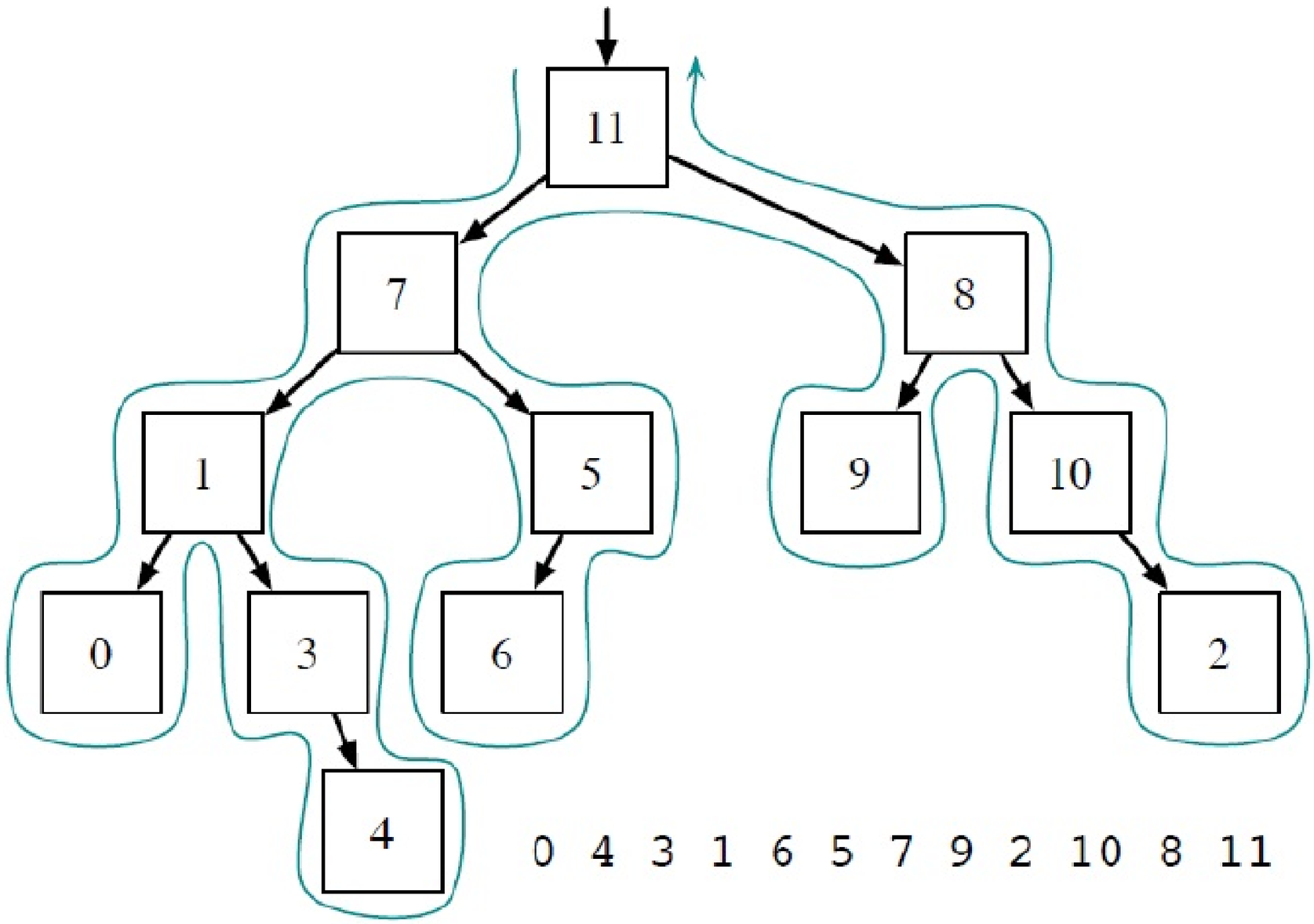
Parcours postfixe: GDR



Parcours postfixe: GDR

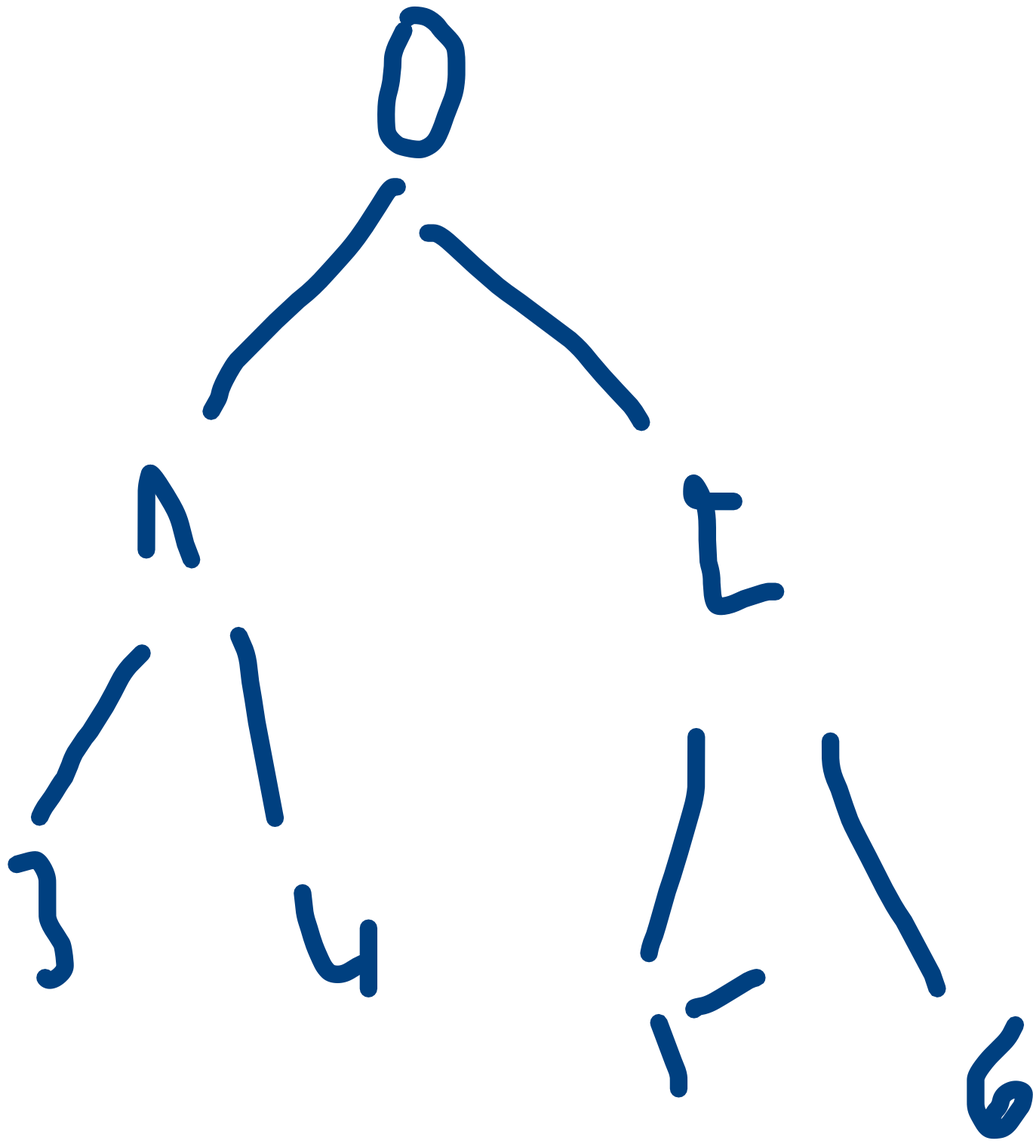


Parcours postfixe: GDR

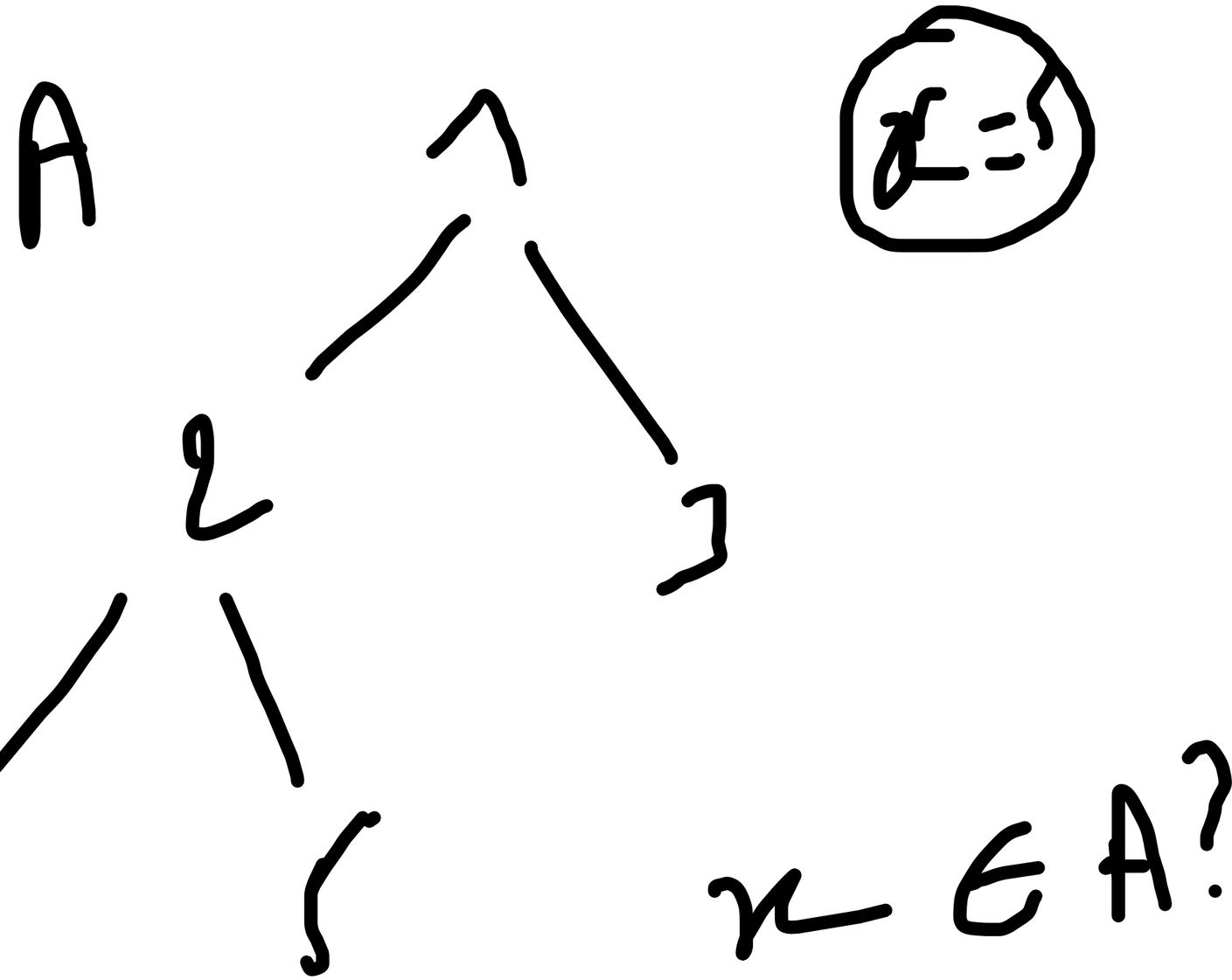


0 4 3 1 6 5 7 9 2 10 8 11

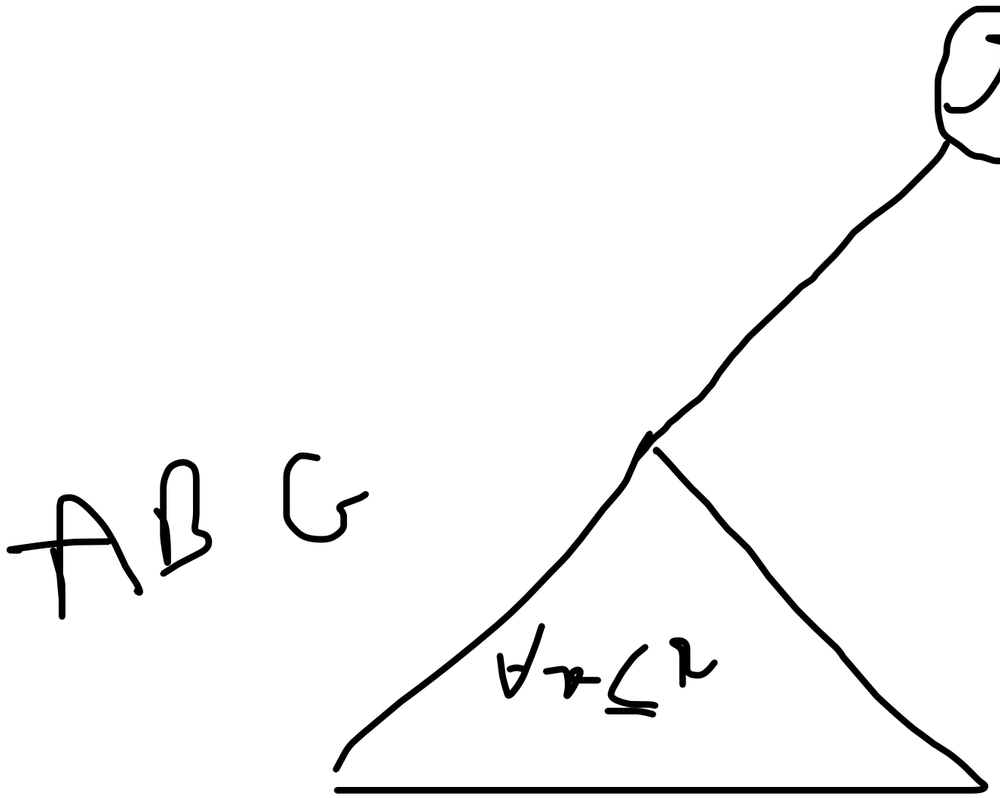
Parcours Postfixe

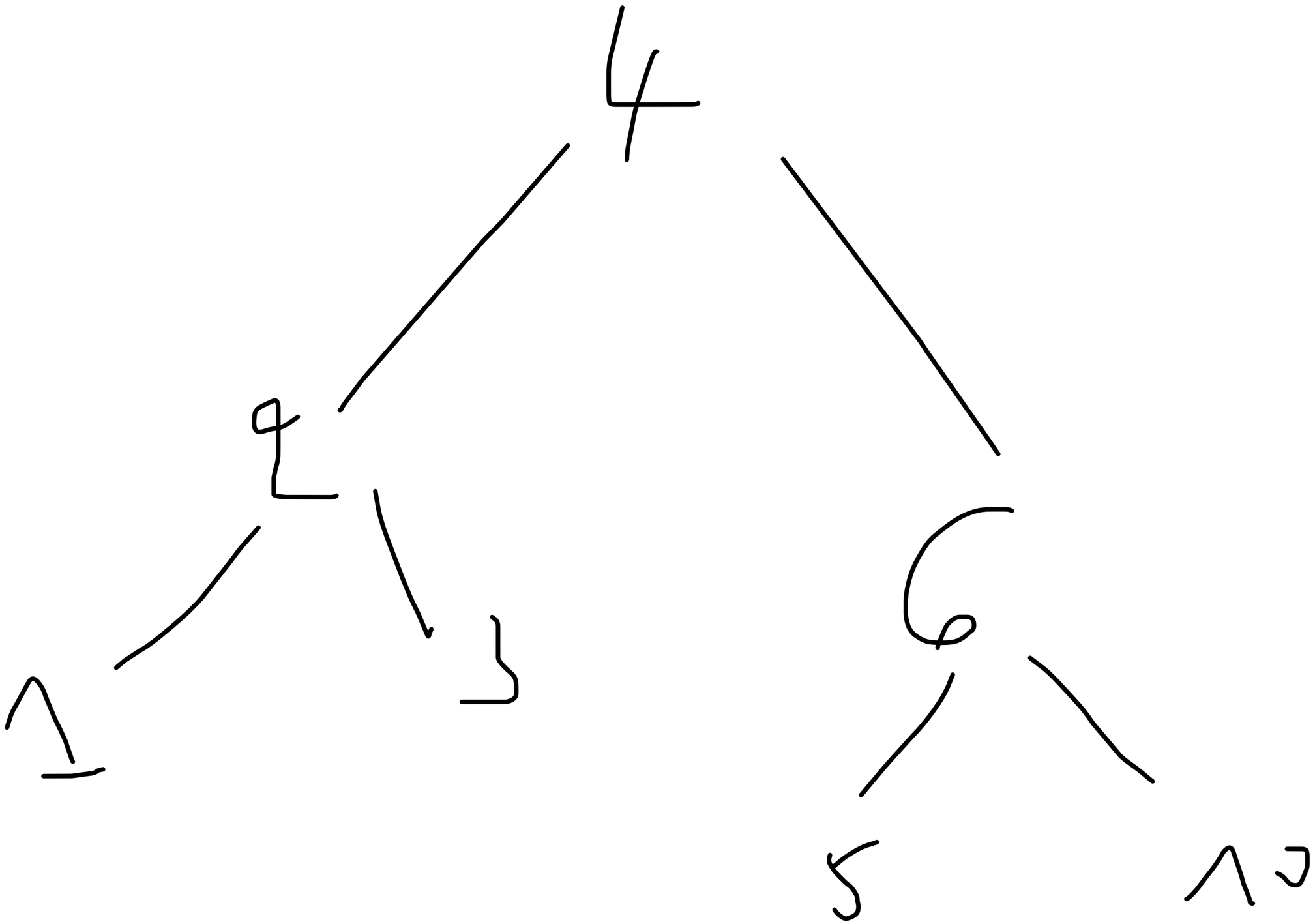


```
int recherche(Btree *A ,int x)
{
if(A==NULL)
  return 0;
if (A->info==x)
   return 1;
if( recherche(A->fg ,x)==1)
   return 1
else
 return recherche(A->fd , x);

}
```



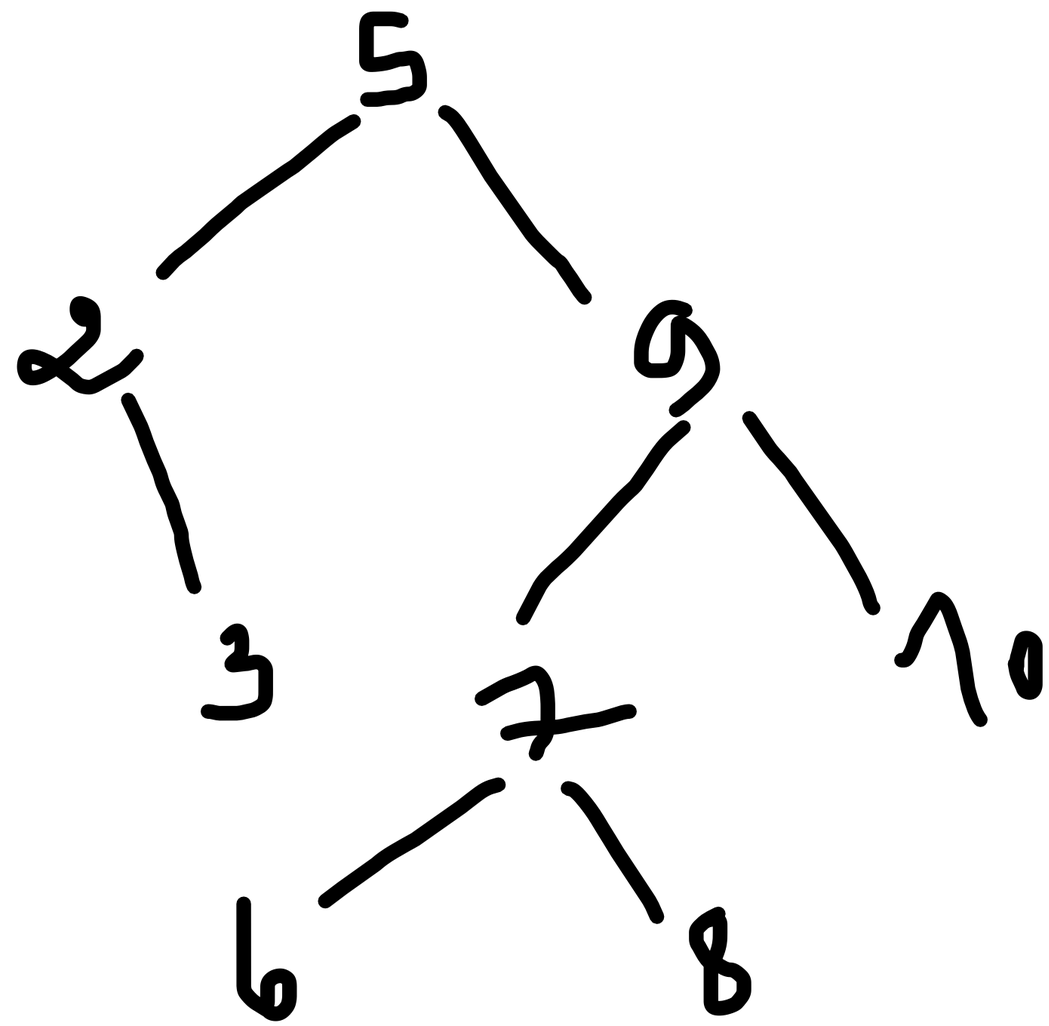
chapitre 8. Arbre binaire de recherche

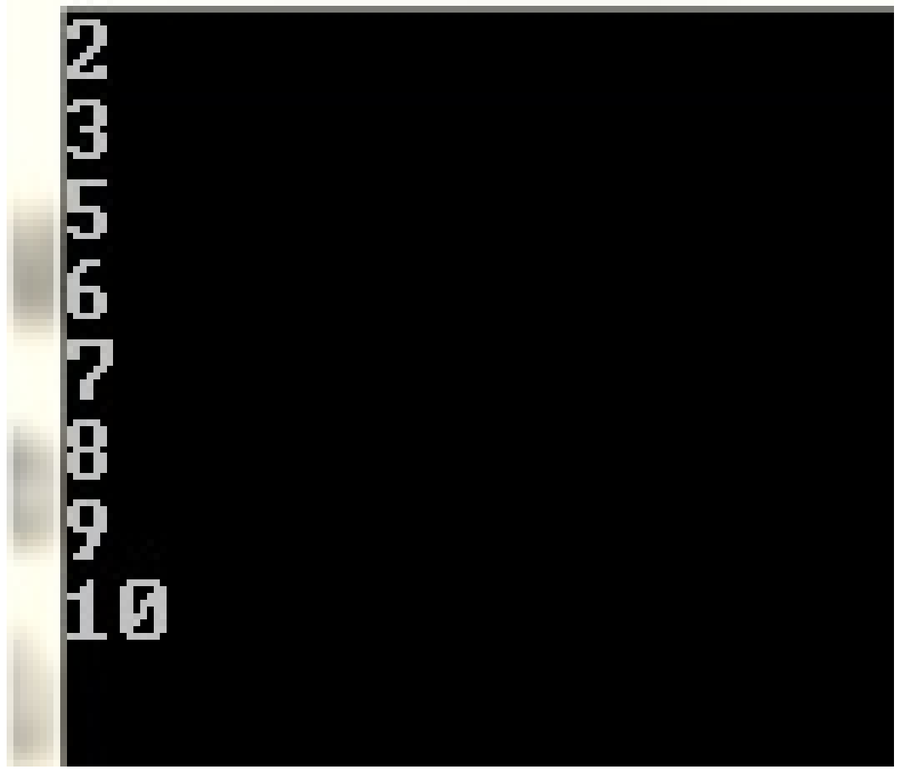




Soit T un tableau contenant T={5,2,3,9,10,7,6,8}

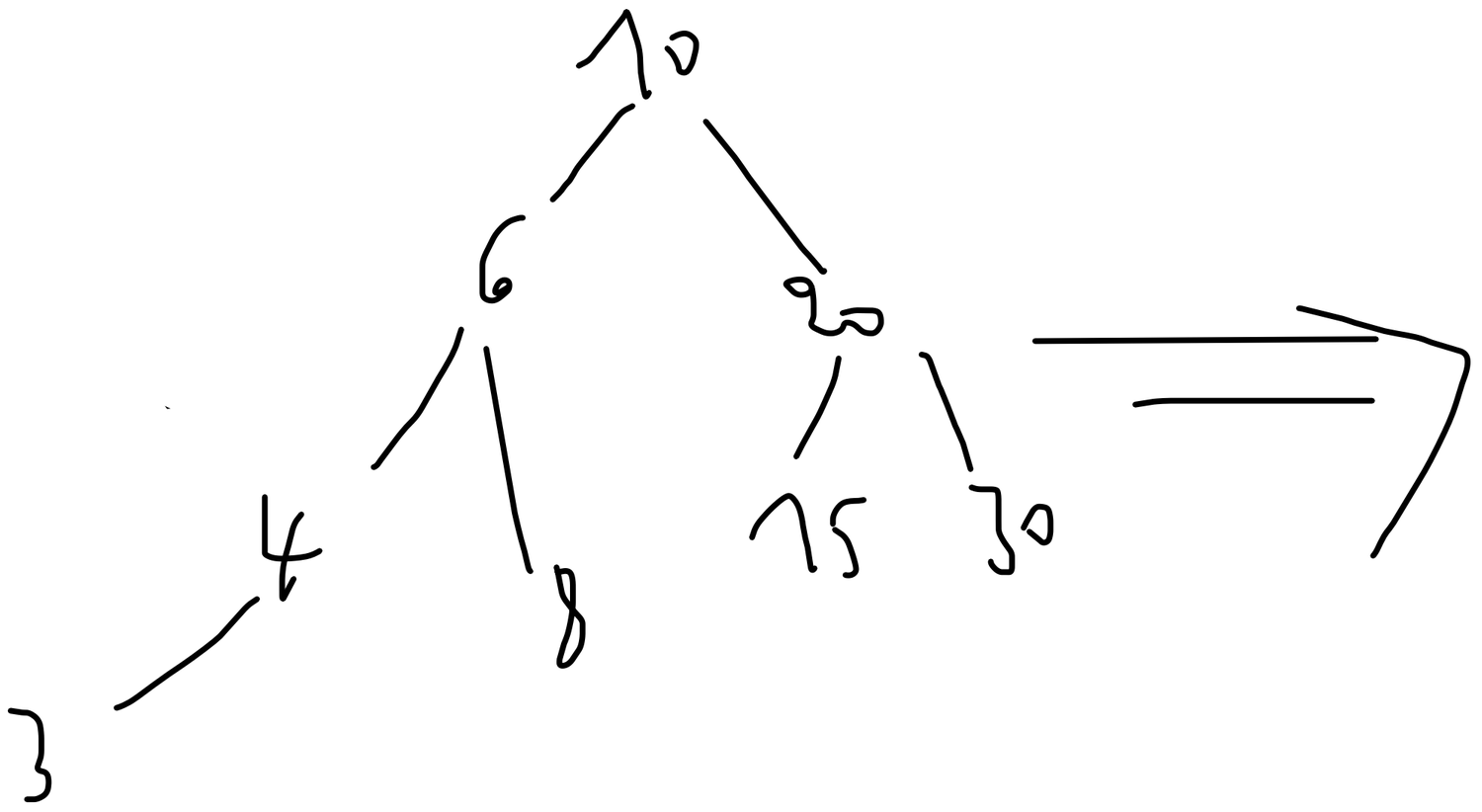
ABC

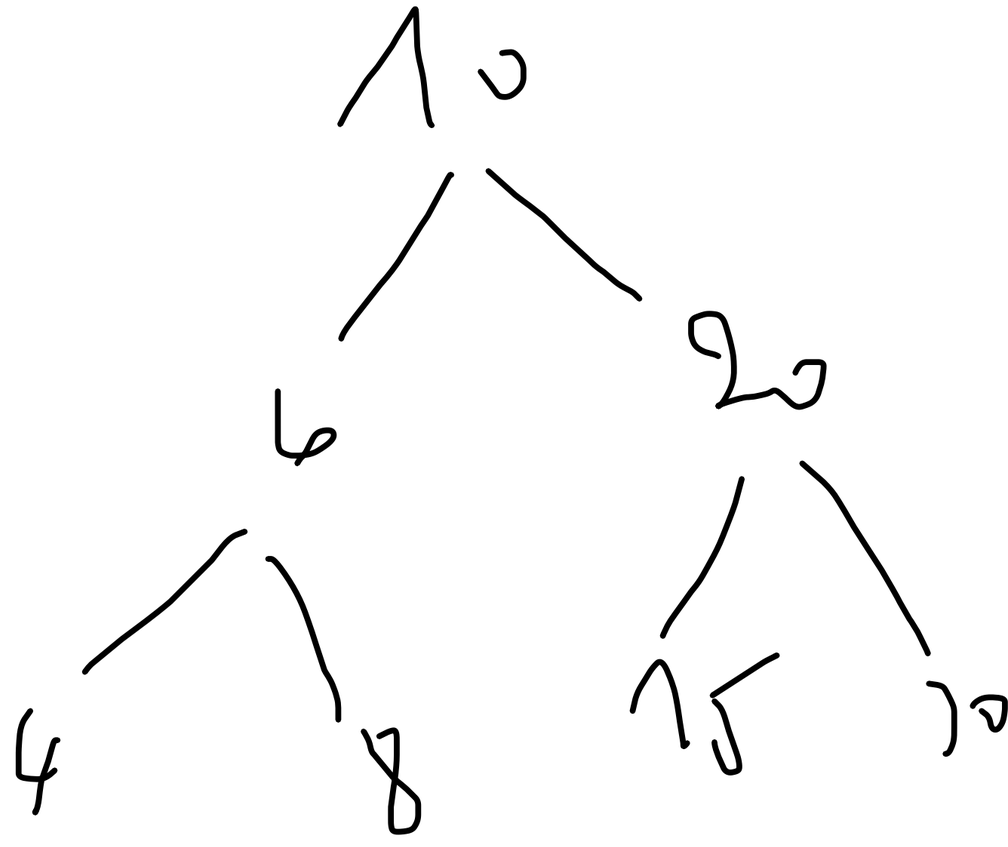




Supprimer un élément d'un arbre binaire de recherche

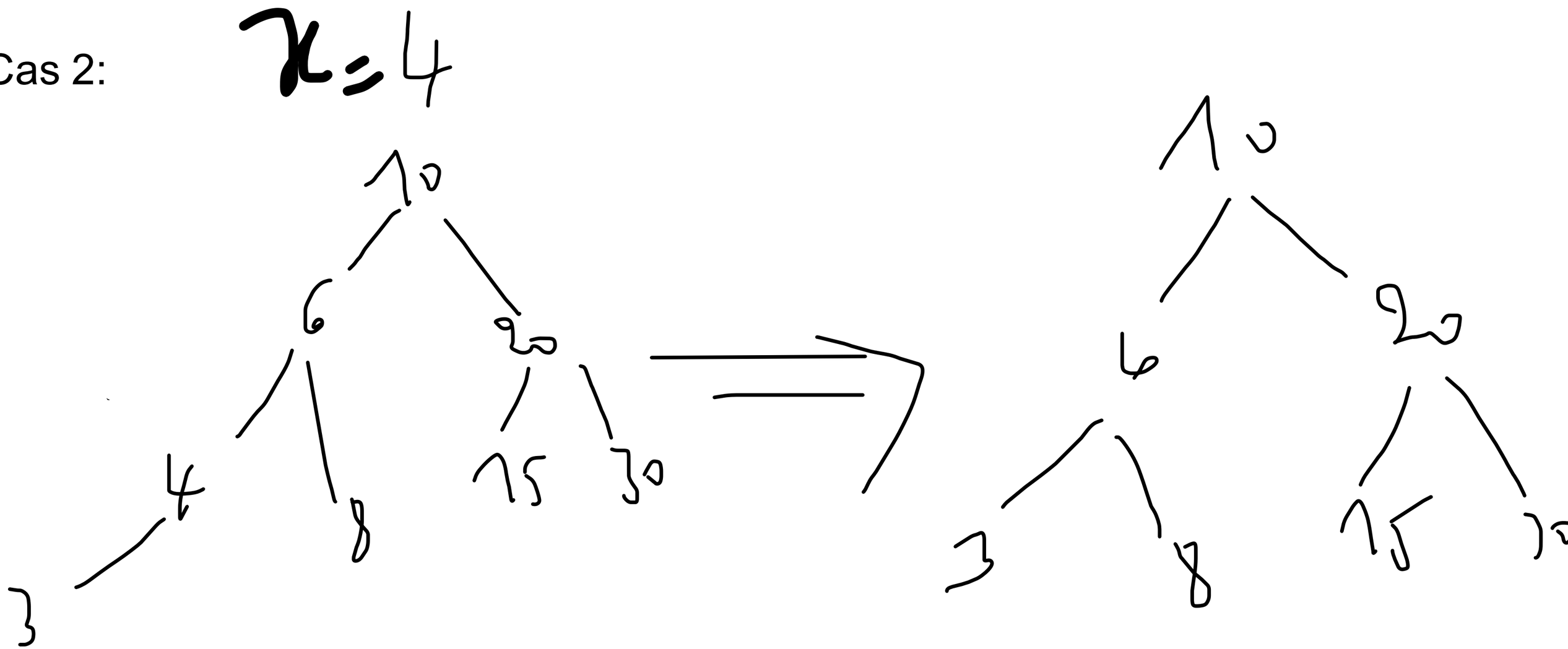
Cas 1: $x = 3$





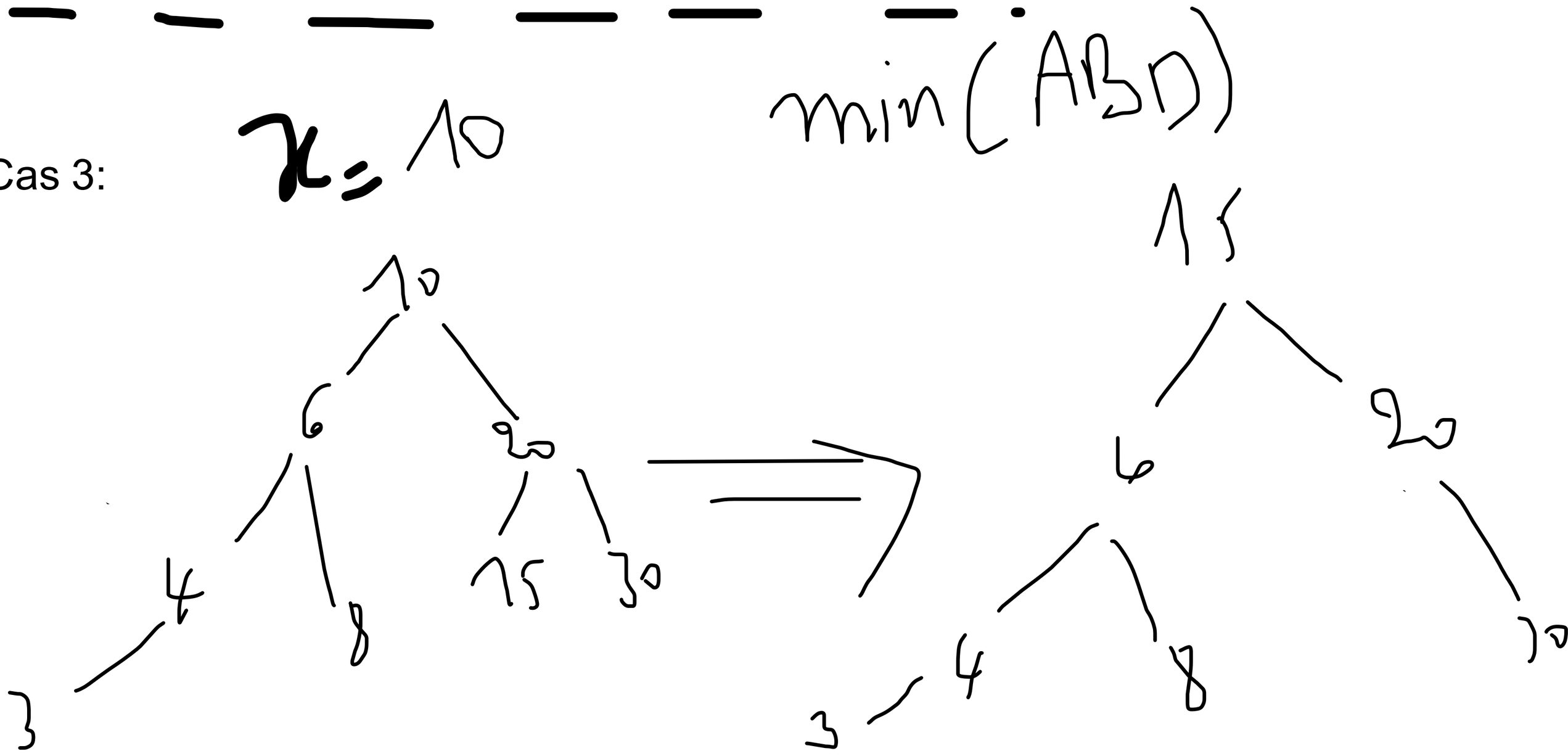
Supprimer un élément d'un arbre binaire de recherche

Cas 2:



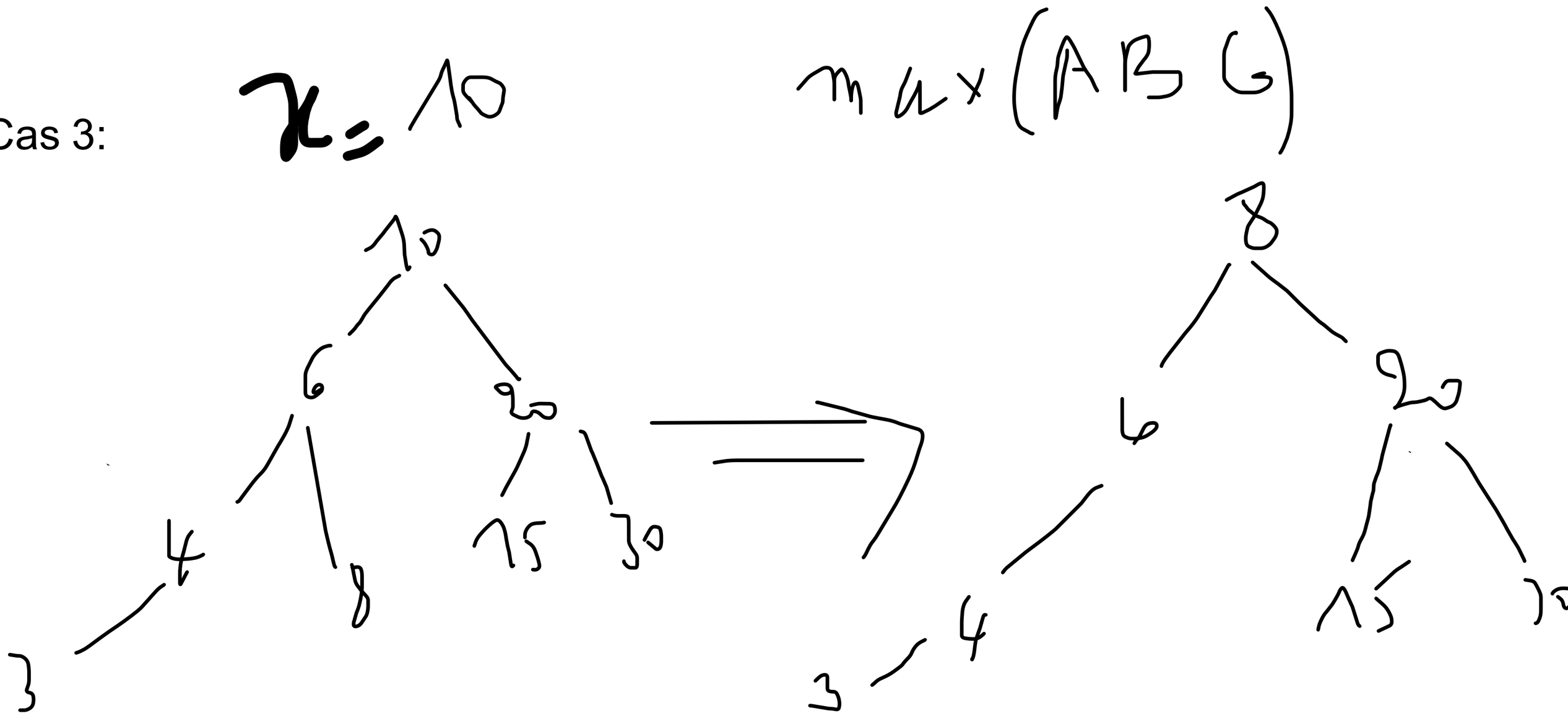
Supprimer un élément d'un arbre binaire de recherche

Cas 3:

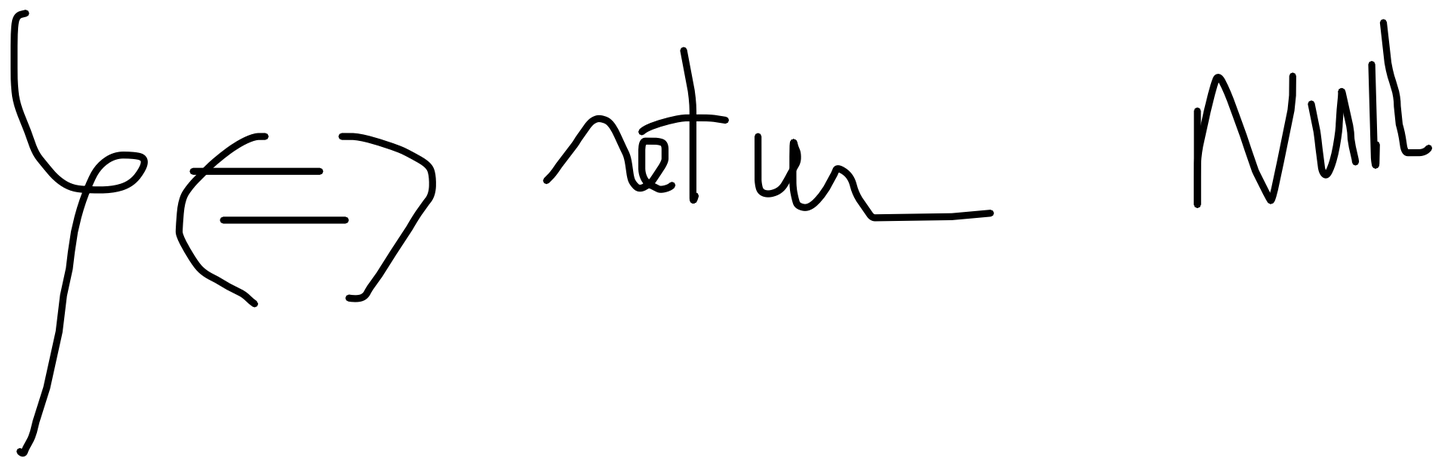


Supprimer un élément d'un arbre binaire de recherche

Cas 3:



```
Btree * ArbrVide()
{
    Btree *A;
    A=NULL;
    return A;
}
```



```
Btree *CreerNoeud(int e,Btree *g,Btree *d)
{
        Btree *c;
        c=(Btree*)malloc(sizeof(Btree));
        if(c!=NULL)
        {
```

c → info = e

c → fg = g ;

c → fd = d ;