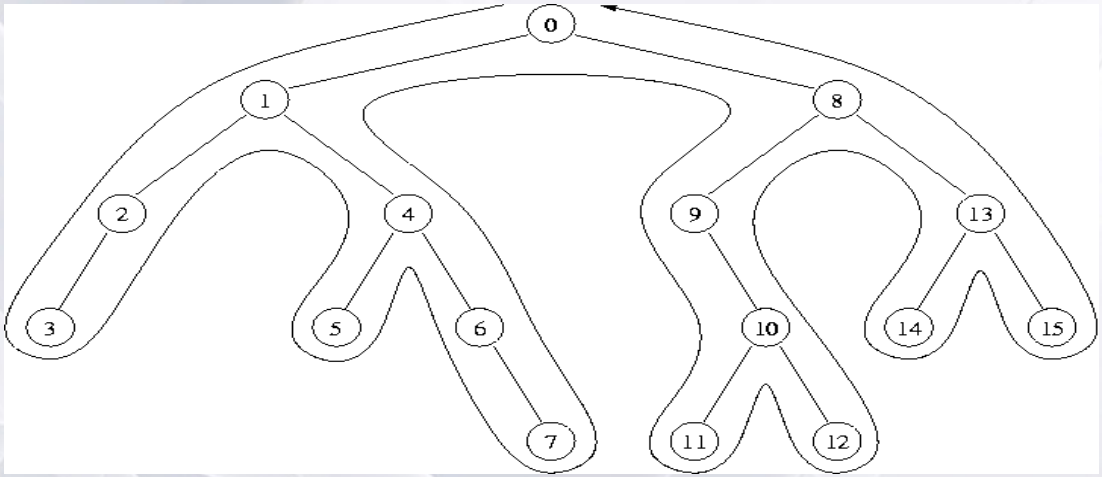
Initiation à la programmation Java
IP2 - Séance No 10

Yan Jurski

Université de Paris

Parcours en largeur

Le résultat attendu de l'affichage du contenu des noeuds de l'arbre



lorsqu'ils sont pris dans l'ordre correspondant à un **parcours en largeur** donne la séquence :

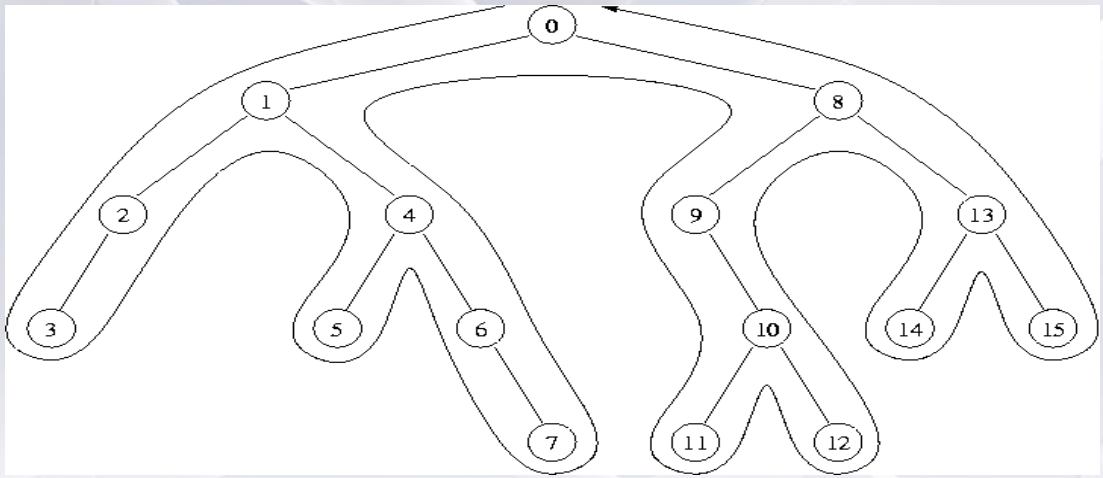
0 1 8 2 4 9 13 3 5 6 10 14 15 7 11 12

Pour écriture le code, on a eu besoin d'une structure auxiliaire : une file. (Remarquez que ce parcours n'est pas récursif)

Université de Paris

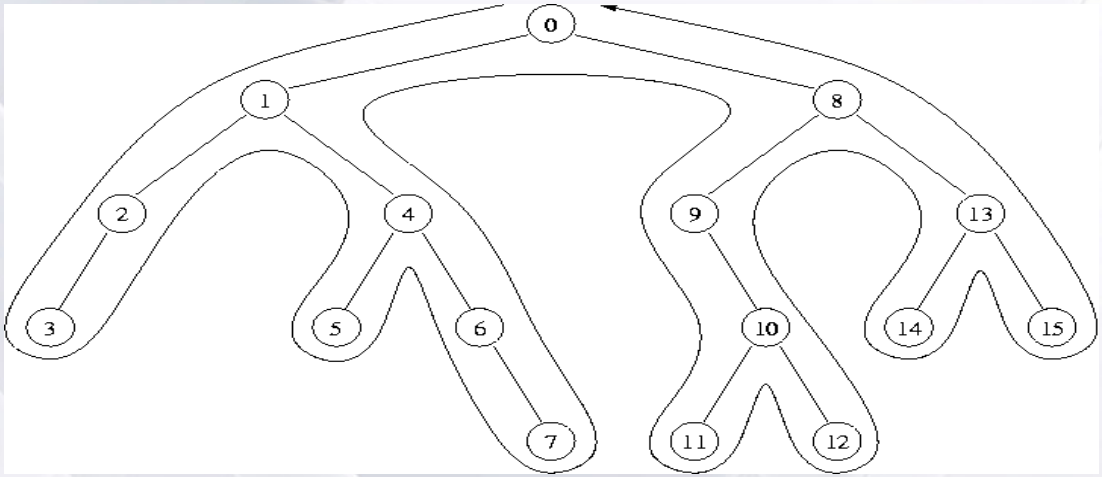
Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // définie pour stocker des Noeuds
 Noeud tmp ;
 p.add (this); // opération d'ajout, this sera le point de départ
 while (! p.isEmpty() ){
  tmp = p.get(); // opération de retrait
  System.out.print( tmp.content.toString()+' ' ); // en premier
  if ( tmp.filsG != null ) p.add( tmp.filsG ); // en second
  if ( tmp.filsD != null ) p.add( tmp.filsD ); // en troisième
 }
}
```



- Donne bien 0 1 8 2 4 9 13 3 5 6 10 14 15 7 11 12

Exercice - Variante



- On souhaite afficher les choses "plus proprement" :
  1 : 0
  2 : 1 8
  3 : 2 4 9 13
  4 : 3 5 6 10 14 15
  5 : 7 11 12
- C'est à dire par lignes de la formes :
  profondeur donnée : les noeuds situés à ce niveau
- Difficulté : il faut faire "apparaître" la profondeur des noeuds

Université de Paris

Reprenons et modifions légèrement le parcours en largeur

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // définie pour stocker des Noeuds
 Noeud tmp ;
 p.add (this);
 while (! p.isEmpty() ){
  tmp = p.get(); // pas d'infos de profondeur lors de la sortie
  System.out.print( tmp.content.toString()+' ' );
  if (tmp.filsG!=null) p.add( tmp.filsG );
  if (tmp.filsD!=null) p.add( tmp.filsD );
 }
}
```

Université de Paris

Reprenons et modifions légèrement le parcours en largeur

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Noeud tmp ;
 p.add (this); // ??
 while (! p.isEmpty() ){
  tmp = p.get(); // ??
  System.out.print( tmp.content.toString()+' ' ); // ??
  if (tmp.filsG!=null) p.add( tmp.filsG ); // ??
  if (tmp.filsD!=null) p.add( tmp.filsD ); // ??
 }
}
```

Université de Paris

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Noeud tmp ;
 p.add (this); // ??
 while (! p.isEmpty() ){
  tmp = p.get(); // ??
  System.out.print( tmp.content.toString()+' ' ); // ??
  if (tmp.filsG!=null) p.add( tmp.filsG ); // ??
  if (tmp.filsD!=null) p.add( tmp.filsD ); // ??
 }
}
```

Fichier Couple.java

```
public class Couple{
 private Noeud n;
 private int p;
 public Couple(Noeud a, int b) { n=a; p=b; }
 public Noeud getN() { return n; }
 public int getP(){ return p; }
}
```

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Noeud tmp ;
 p.add (new Couple(this,1)); // la premiere profondeur est connue
 while (! p.isEmpty() ){
  tmp = p.get(); // ??
  System.out.print( tmp.content.toString()+' ' ); // ??
  if (tmp.filsG!=null) p.add( tmp.filsG ); // ??
  if (tmp.filsD!=null) p.add( tmp.filsD ); // ??
 }
}
```

Fichier Couple.java

```
public class Couple{
 private Noeud n;
 private int p;
 public Couple(Noeud a, int b) { n=a; p=b; }
 public Noeud getN() { return n; }
 public int getP(){ return p; }
}
```

Université de Paris

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Couple tmp; // tmp est de type couple
 p.add (new Couple(this,1)); // la premiere profondeur est connue
 while (! p.isEmpty() ){
  tmp = p.get(); // de type couple
  System.out.print( tmp.content.toString()+' ' ); // ??
  if (tmp.filsG!=null) p.add( tmp.filsG ); // ??
  if (tmp.filsD!=null) p.add( tmp.filsD ); // ??
 }
}
```

Fichier Couple.java

```
public class Couple{
 private Noeud n;
 private int p;
 public Couple(Noeud a, int b) { n=a; p=b; }
 public Noeud getN() { return n; }
 public int getP(){ return p; }
}
```

Université de Paris

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Couple tmp; // tmp est de type couple
 p.add (new Couple(this,1)); // la première profondeur est connue
 while (! p.isEmpty() ){
  tmp = p.get(); // de type couple
  Noeud tmpN = tmp.getN(); // récupération
  System.out.print( tmpN.content.toString()+' ' );
  if (tmp.filsG!=null) p.add( tmp.filsG ); // ??
  if (tmp.filsD!=null) p.add( tmp.filsD ); // ??
 }
}
```

Fichier Couple.java

```
public class Couple{
 private Noeud n;
 private int p;
 public Couple(Noeud a, int b) { n=a; p=b; }
 public Noeud getN() { return n; }
 public int getP(){ return p; }
}
```

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Couple tmp; // tmp est de type couple
 p.add (new Couple(this,1)); // la première profondeur est connue
 while (! p.isEmpty() ){
  tmp = p.get(); // de type couple
  Noeud tmpN = tmp.getN(); // récupération
  System.out.print( tmpN.content.toString()+' ' );
  if (tmpN.filsG!=null) p.add(new Couple(tmpN.filsG,tmp.getP()+1));
  if (tmp.filsD!=null) p.add( tmp.filsD ); // ??
 }
}
```

Fichier Couple.java

```
public class Couple{
 private Noeud n;
 private int p;
 public Couple(Noeud a, int b) { n=a; p=b; }
 public Noeud getN() { return n; }
 public int getP(){ return p; }
}
```

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile(); // construite pour couple(Noeud,profondeur)
 Couple tmp;
 p.add (new Couple(this,1));
 while (! p.isEmpty() ){
  tmp = p.get();
  Noeud tmpN = tmp.getN();
  System.out.print( tmpN.content.toString()+' ' );
  if (tmpN.filsG!=null) p.add(new Couple(tmpN.filsG,tmp.getP()+1));
  if (tmpN.filsD!=null) p.add(new Couple(tmpN.filsD,tmp.getP()+1));
 }
}
```

- Nous avons mis en place l'association du noeud et de sa profondeur
- L'affichage reste le même pour le moment
- Il faut encore détecter le changement de niveau

Université de Paris

(exercice suite)

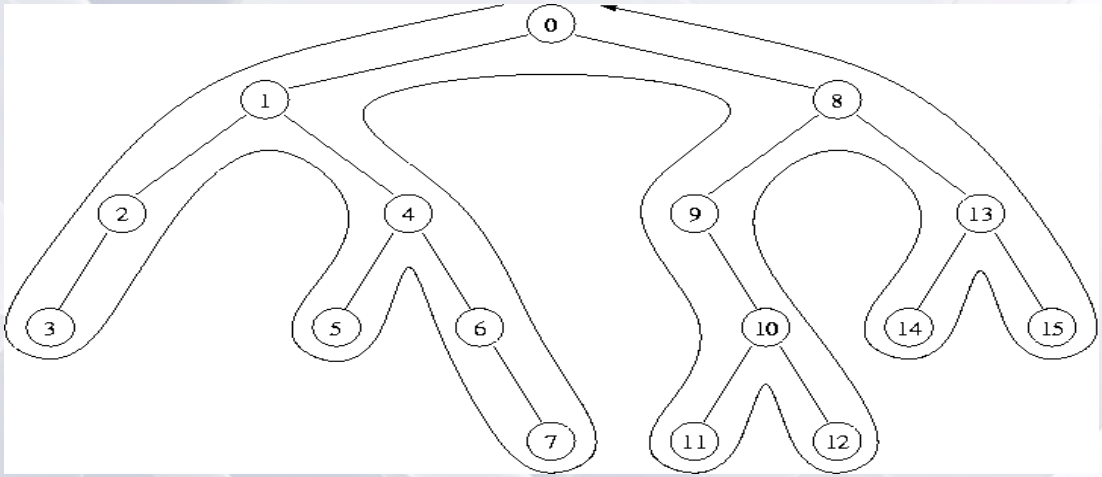
détection du changement de niveau

Fichier Noeud.java

```
public void parcoursLargeur (){
 MaFile p = new MaFile();
 Couple tmp;
 p.add (new Couple(this,1));
 int niveauActuel=0; // aucun affichage n'a eu lieu
 while (! p.isEmpty() ){
  tmp = p.get();
  Noeud tmpN = tmp.getN();
  if (tmp.getP() != niveauActuel) {
   niveauActuel = tmp.getP();
   System.out.print("\n" + niveauActuel+" : "); // remarquez le \n
  }
  System.out.print( tmpN.content.toString()+' ' );
  if (tmpN.filsG!=null) p.add(new Couple(tmpN.filsG,tmp.getP()+1));
  if (tmpN.filsD!=null) p.add(new Couple(tmpN.filsD,tmp.getP()+1));
 }
}
```

Exercice 2 - Retour sur un parcours récursif

- Ecrivons un **parcours préfixe** qui accompagne l'affichage de chaque étiquette des nœuds, du mot composé des lettres 'g' et 'd' décrivant le chemin qui mène à ce noeud. ('g' pour gauche et 'd' pour droite)

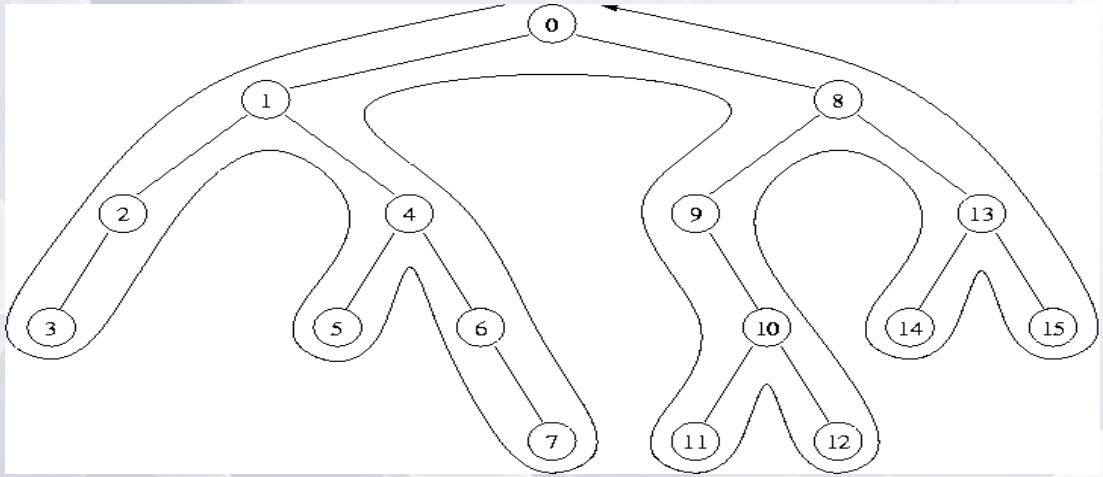


0 : ""
1 : g
2 : gg
3 : ggg
4 : gd
5 : gdg ...

Université de Paris

Exercice 2 - Retour sur un parcours récursif

- Ecrivons un **parcours préfixe** qui accompagne l'affichage de chaque étiquette des nœuds, du mot composé des lettres 'g' et 'd' décrivant le chemin qui mène à ce noeud. ('g' pour gauche et 'd' pour droite)



0 : ""
1 : g
2 : gg
3 : ggg
4 : gd
5 : gdg ...

- Lors de la descente on va transmettre le mot caractérisant le chemin emprunté
- Il sera ainsi disponible pour l'affichage

Université de Paris

Exercice 2

Rappel parcours préfixe

Fichier Arbre.java

```
public void affiche(){
 if (racine==null) System.out.println("Arbre vide");
 else racine.affiche(); // méthode de la classe Noeud
}
```

Fichier Noeud.java

```
public class Noeud{
 private E content;
 private Noeud filsG, filsD;
 public void affiche(){
  System.out.println(content.toString()+", "); // supposé exister
  if (filsG!=null) filsG.affiche();
  if (filsD!=null) filsD.affiche();
 }
}
```

Université de Paris

Exercice 2

parcours préfixe légèrement modifié

Fichier Arbre.java

```
public void affiche(){
 if (racine==null) System.out.println("Arbre vide");
 else racine.affiche(""); // méthode de la classe Noeud
}
```

Fichier Noeud.java

```
public class Noeud{
 private E content;
 private Noeud filsG, filsD;
 public void affiche(String mot){ //mot : le chemin qui atteind ce noeud
  System.out.println(content.toString()+": "+mot);
  if (filsG!=null) filsG.affiche(mot+'g');
  if (filsD!=null) filsD.affiche(mot+'d');
 }
}
```

Université de Paris

Exercice 2

parcours préfixe légèrement modifié

Fichier Arbre.java

```
public void affiche(){
 if (racine==null) System.out.println("Arbre vide");
 else racine.affiche(""); // méthode de la classe Noeud
}
```

Fichier Noeud.java

```
public class Noeud{
 private E content;
 private Noeud filsG, filsD;
 public void affiche(String mot){ //mot : le chemin qui atteind ce noeud
  System.out.println(content.toString()+": "+mot);
  if (filsG!=null) filsG.affiche(mot+'g');
  if (filsD!=null) filsD.affiche(mot+'d');
 }
}
```

Et c'est fait ! Remarquez qu'il ne fallait pas changer grand chose !

Université de Paris

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ...

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ... on utilisera une liste d'arbre

  (On peut à présent utiliser la librairie Java **linkedList<Arbre>**)
- Les étiquettes n'ont pas d'importance dans cet exercice, on utilisera '-'

Université de Paris

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ... on utilisera une liste d'arbre

  (On peut à présent utiliser la librairie Java **linkedList<Arbre>**)
- Les étiquettes n'ont pas d'importance dans cet exercice, on utilisera '-'
- Quel raisonnement serait efficace ?
  - Ajouter une feuille aux arbres de hauteur $h-1$?

Université de Paris

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ... on utilisera une liste d'arbre

  (On peut à présent utiliser la librairie Java **linkedList<Arbre>**)
- Les étiquettes n'ont pas d'importance dans cet exercice, on utilisera '-'
- Quel raisonnement serait efficace ?
  - Ajouter une feuille aux arbres de hauteur $h-1$?
    Pas évident ... ca reste compliqué...

Université de Paris

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ... on utilisera une liste d'arbre

  (On peut à présent utiliser la librairie Java **linkedList<Arbre>**)
- Les étiquettes n'ont pas d'importance dans cet exercice, on utilisera '-'
- Quel raisonnement serait efficace ?
  - Ajouter une feuille aux arbres de hauteur $h-1$?
    Pas évident ... ca reste compliqué...
  - On regarde de l'autre coté, vers la racine :
    - Si on ajoute un noeud au dessus ...

Université de Paris

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ... on utilisera une liste d'arbre

  (On peut à présent utiliser la librairie Java **linkedList<Arbre>**)
- Les étiquettes n'ont pas d'importance dans cet exercice, on utilisera '-'
- Quel raisonnement serait efficace ?
  - Ajouter une feuille aux arbres de hauteur $h-1$?
    Pas évident ... ca reste compliqué...
  - On regarde de l'autre coté, vers la racine :
    - Si on ajoute un noeud au dessus ...
      ... de l'autre coté n'importe quel arbre de hauteur $\leq h-1$ convient
    - Ce raisonnement permet une énumération ! Qqs précautions et allons y !

Université de Paris

Exercice 3

Où on combine listes, arbres et raisonnement récursif

Problème :

Comment générer tous les arbres binaires de hauteur $\leq h$

- Il va falloir les stocker ... on utilisera une liste d'arbre

  (On peut à présent utiliser la librairie Java **linkedList<Arbre>**)
- Les étiquettes n'ont pas d'importance dans cet exercice, on utilisera '-'
- Quel raisonnement serait efficace ?
  - Ajouter une feuille aux arbres de hauteur $h-1$?
    Pas évident ... ca reste compliqué...
  - On regarde de l'autre coté, vers la racine :
    - Si on ajoute un noeud au dessus ...
      ... de l'autre coté n'importe quel arbre de hauteur $\leq h-1$ convient
    - Ce raisonnement permet une énumération ! Qqs précautions et allons y !
- Un même arbre de hauteur $h-1$ pourrait être utilisé en greffe pour construire plusieurs arbres : il sera plus clair de travailler sur des copies

Préalable - copie d'un arbre

Fichier Arbre.java

```
public class Arbre{
 private Noeud racine;
 public Arbre(){ racine=null;}
 public Arbre(Noeud x){racine=x;}
 public Arbre copie(){
  if (racine==null) return new Arbre();
  else return new Arbre (racine.copie());
 }
}
```

Fichier Noeud.java

```
public class Noeud{
 public Noeud(E x, Noeud a, Noeud b){ content=x; filsG=a; filsD=b;}
 public Noeud(E x, Arbre a, Arbre b){ content=x; filsG=a.getRacine();
     filsD=b.getRacine();}
 public Noeud copie(){
  return new Noeud(content,
   (filG!=null)?filsG.copie():null , // notation pratique ici
   (filD!=null)?filsD.copie():null );
 }
```

Exercice 3

générer tous les arbres binaires de hauteur $\leq h$

Fichier Arbre.java

```
public class Arbre{
 public static List<Arbre> generation(int h){ // méthode statique
  List<Arbre> réponse=new LinkedList(); // on utilise l'API
  if (h<0) return réponse;
  if (h==0) { réponse.add(new Arbre()); return réponse;}
  List<Arbre> arbresPP =generation(h-1); // appel récursif
  for(Arbre a: arbresPP){ // notation élégante
   for (Arbre b: arbresPP){
    Noeud nouv=new Noeud('-',a.copie(),b.copie()); // label arbitraire
    réponse.add(new Arbre(nouv));
   }
  }
  réponse.add(new Arbre()); // le seul manquant
  return réponse;
 }
}
```

- Y a t'il des doublons dans notre réponse ? (vérifiez, mais non ...)

Exercice 4

Question :

Deux arbres sont dits **similaires** s'ils ont la même structure et le même contenu. Ecrivez une méthode qui effectue ce test.

Exercice 4

Question :

Deux arbres sont dits **similaires** s'ils ont la même structure et le même contenu. Ecrivez une méthode qui effectue ce test.

Fichier Arbre.java

```
public boolean similaire(Arbre a){
 if (estVide() && a.estVide()) return true;
 if (estVide() || a.estVide()) return false;
 return (racine.similaire(a.racine));
}
```

Université de Paris

Exercice 4

Question :

Deux arbres sont dits **similaires** s'ils ont la même structure et le même contenu. Ecrivez une méthode qui effectue ce test.

Fichier Arbre.java

```
public boolean similaire(Arbre a){
 if (estVide() && a.estVide()) return true;
 if (estVide() || a.estVide()) return false;
 return (racine.similaire(a.racine));
}
```

Fichier Noeud.java

```
public boolean similaire(Noeud x){
 if (x==null) return false;
 if (content!=x.content) return false;
 if (estFeuille() && x.estFeuille()) return true;
 if (fg==null && x.fg!=null) || (fd==null && x.fd!=null) return false;
 boolean testg=true, testd=true;
 if (fg!=null) testg=fg.similaire(x.fg);
 if (fd!=null) testd=fd.similaire(x.fd);
 return (testg && testd);
}
```

Exercice 5

Question :

Il est fréquent que des sous arbres soient similaires au sein d'un arbre. On considère le nombre de nœuds comme mesure de la taille d'un arbre. Quelle est la taille du plus grand sous arbre apparaissant plusieurs fois (au sens de la similarité) ?

Université de Paris

Exercice 5

Question :

Il est fréquent que des sous arbres soient similaires au sein d'un arbre. On considère le nombre de nœuds comme mesure de la taille d'un arbre. Quelle est la taille du plus grand sous arbre apparaissant plusieurs fois (au sens de la similarité) ?

- Comment aborder cette question ?

Université de Paris

Exercice 5

Question :

Il est fréquent que des sous arbres soient similaires au sein d'un arbre. On considère le nombre de nœuds comme mesure de la taille d'un arbre. Quelle est la taille du plus grand sous arbre apparaissant plusieurs fois (au sens de la similarité) ?

- Comment aborder cette question ?
- Lors d'un parcours on peut énumérer les racines des sous arbres, mais il faut encore les comparer entre eux.

Université de Paris

Exercice 5

Question :

Il est fréquent que des sous arbres soient similaires au sein d'un arbre.
On considère le nombre de nœuds comme mesure de la taille d'un arbre.
Quelle est la taille du plus grand sous arbre apparaissant plusieurs fois (au sens de la similarité) ?

- Comment aborder cette question ?
- Lors d'un parcours on peut énumérer les racines des sous arbres, mais il faut encore les comparer entre eux.
- On va rencontrer une seconde fois un sous arbre similaire à un autre ... Si on dispose alors de ceux déjà rencontrés on peut effectuer un calcul.

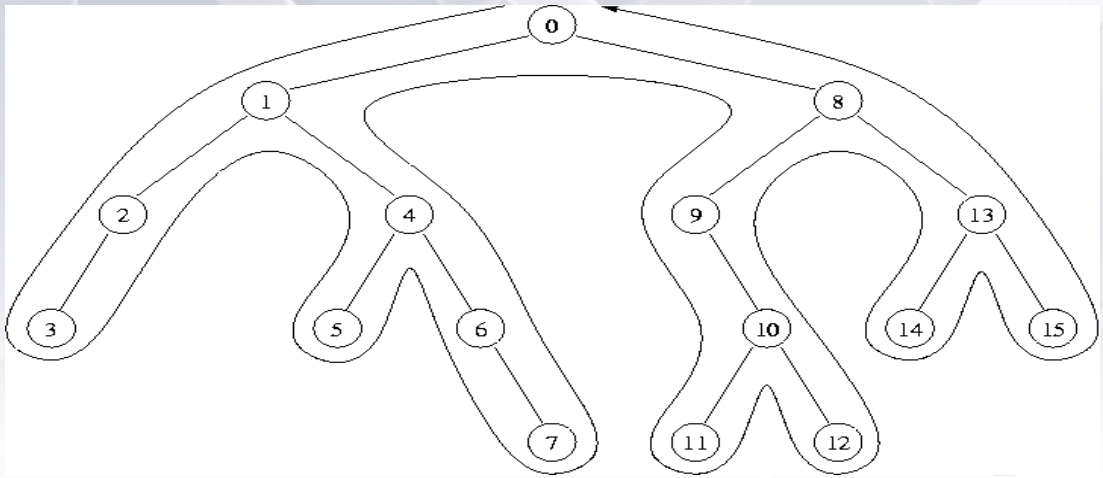
Université de Paris

Exercice 5

Question :

Il est fréquent que des sous arbres soient similaires au sein d'un arbre.
On considère le nombre de nœuds comme mesure de la taille d'un arbre.
Quelle est la taille du plus grand sous arbre apparaissant plusieurs fois (au sens de la similarité) ?

- Comment aborder cette question ?
- Lors d'un parcours on peut énumérer les racines des sous arbres, mais il faut encore les comparer entre eux.
- On va rencontrer une seconde fois un sous arbre similaire à un autre ... Si on dispose alors de ceux déjà rencontrés on peut effectuer un calcul.



Université de Paris

Exercice 5

Question :

Taille du plus grand sous arbre apparaissant plusieurs fois ?

Fichier Arbre.java

```
public int tailleSimilaire(){
 if (racine==null) return 0;
 List<Arbre> rencontrés=new LinkedList();
 return racine.plusGrandSimilaire(rencontrés); //transmettre ceux vus
}
```

Université de Paris

Exercice 5

Question :

Taille du plus grand sous arbre apparaissant plusieurs fois ?

Fichier Arbre.java

```
public int tailleSimilaire(){
 if (racine==null) return 0;
 List<Arbre> rencontrés=new LinkedList();
 return racine.plusGrandSimilaire(rencontrés); //transmettre ceux vus
}
```

Fichier Noeud.java

```
int plusGrandSimilaire(List<Arbre> rencontrés) {
 Arbre current=new Arbre(this); // passage de noeud à arbre
 int m=0;
 if (contient(rencontrés,current)) m=current.size();// écrire 'contient'
 else rencontrés.add(current);
 if (g!=null) m=Math.max(m,g.plusGrandSimilaire(rencontrés));
 if (d!=null) m=Math.max(m,d.plusGrandSimilaire(rencontrés));
 return m;
}
```

Question :

Taille du plus grand sous arbre apparaissant plusieurs fois ?

Fichier Arbre.java

```
public int tailleSimilaire(){
 if (racine==null) return 0;
 List<Arbre> rencontrés=new LinkedList();
 return racine.plusGrandSimilaire(rencontrés); //transmettre ceux vus
}
```

Fichier Noeud.java

```
int plusGrandSimilaire(List<Arbre> rencontrés) {
 Arbre current=new Arbre(this); // passage de noeud à arbre
 int m=0;
 if (contient(rencontrés,current)) m=current.size();// écrire 'contient'
 else rencontrés.add(current);
 if (g!=null) m=Math.max(m,g.plusGrandSimilaire(rencontrés));
 if (d!=null) m=Math.max(m,d.plusGrandSimilaire(rencontrés));
 return m;
}
```

- Remarquez la ressemblance avec l'exercice 2 !
  (sur le passage en argument)

Université de Paris

Exercice 5

pour être complêt, il restait à écrire :

Fichier Noeud.java

```
private static boolean contient(List<Arbre> rencontrés, Arbre current) {
 for (Arbre x:rencontrés)
  if (x.similaire(current)) return true;
 return false;
}
```

Vous pouvez vous arrêter ici pour cette semaine

Variations sur les arbres - Arbres n-aires

- Premier cas de figure : le degré est fixe, par exemple ternaire

Variations sur les arbres - Arbres n-aires

- Premier cas de figure : le degré est fixe, par exemple ternaire

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud filsG; // pour gauche
 private Noeud filsM; // pour milieu
 private Noeud filsD; // pour droit
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Variations sur les arbres - Arbres n-aires

- Premier cas de figure : le degré est fixe, par exemple ternaire

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud filsG; // pour gauche
 private Noeud filsM; // pour milieu
 private Noeud filsD; // pour droit
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Fichier Noeud.java

```
public void affichePréfixe(){
 System.out.println(content.toString()+", "); // supposé exister
 if (filsG!=null) filsG.affichePréfixe();
 if (filsM!=null) filsM.affichePréfixe();
 if (filsD!=null) filsD.affichePréfixe();
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Premier cas de figure : le degré est fixe, par exemple ternaire

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud filsG; // pour gauche
 private Noeud filsM; // pour milieu
 private Noeud filsD; // pour droit
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Fichier Noeud.java

```
public void afficheSuffixe(){
 if (filsG!=null) filsG.afficheSuffixe();
 if (filsM!=null) filsM.afficheSuffixe();
 if (filsD!=null) filsD.afficheSuffixe();
 System.out.println(content.toString()+", "); // supposé exister
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Premier cas de figure : le degré est fixe, par exemple ternaire

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud filsG; // pour gauche
 private Noeud filsM; // pour milieu
 private Noeud filsD; // pour droit
}
```

- Deux définitions possibles pour le parcours "infixe"

Fichier Noeud.java

```
public void afficheInfixeUn(){
 if (filsG!=null) filsG.afficheInfixeUn();
 System.out.println(content.toString()+", "); // supposé exister
 if (filsM!=null) filsM.afficheInfixeUn();
 if (filsD!=null) filsD.afficheInfixeUn();
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Premier cas de figure : le degré est fixe, par exemple ternaire

Fichier Noeud.java

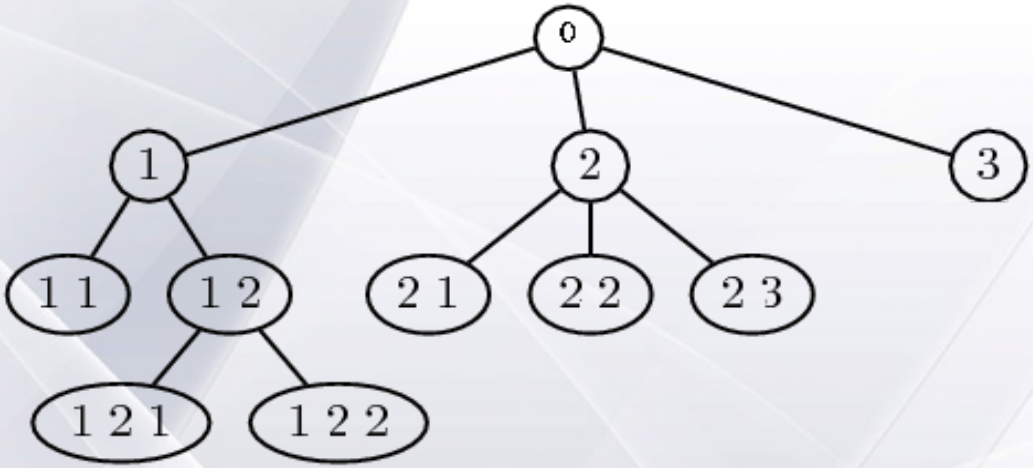
```
public class Noeud {
 private E content;
 private Noeud filsG; // pour gauche
 private Noeud filsM; // pour milieu
 private Noeud filsD; // pour droit
}
```

- Deux définitions possibles pour le parcours "infixe"

Fichier Noeud.java

```
public void afficheInfixeDeux(){
 if (filsG!=null) filsG.afficheInfixeDeux();
 if (filsM!=null) filsM.afficheInfixeDeux();
 System.out.println(content.toString()+", "); // supposé exister
 if (filsD!=null) filsD.afficheInfixeDeux();
}
```

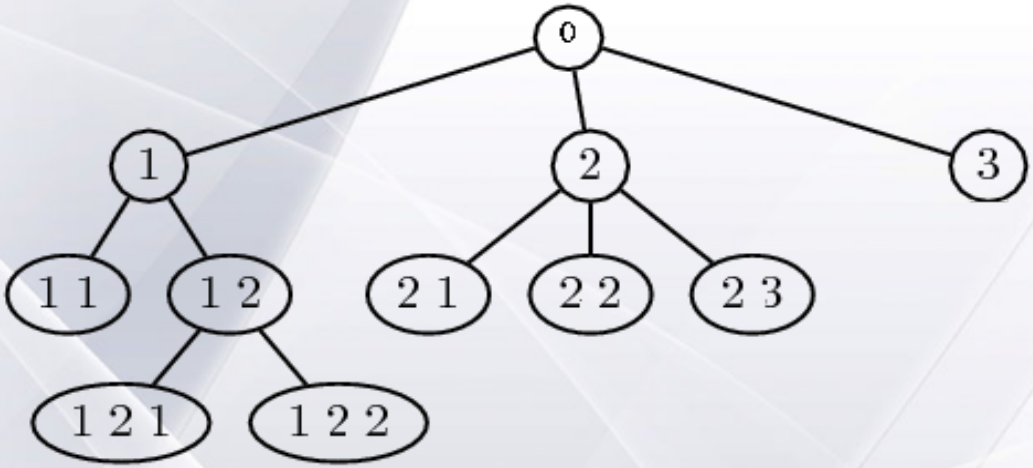
Université de Paris



Fichier Noeud.java

```
public void afficheInfixeDeux(){
if (filsG!=null) filsG.afficheInfixeDeux();
if (filsM!=null) filsM.afficheInfixeDeux();
System.out.println(content.toString()+", "); // supposé exister
if (filsD!=null) filsD.afficheInfixeDeux();
}
```

- Donne ici :



Fichier Noeud.java

```
public void afficheInfixeDeux(){
if (filsG!=null) filsG.afficheInfixeDeux();
if (filsM!=null) filsM.afficheInfixeDeux();
System.out.println(content.toString()+", "); // supposé exister
if (filsD!=null) filsD.afficheInfixeDeux();
}
```

- Donne ici : $11, 121, 122, 12, 1, 21, 22, 2, 23, 0, 3$

Université de Paris

Variations sur les arbres - Arbres n-aires

- Second cas de figure : le degré est borné

Variations sur les arbres - Arbres n-aires

- Second cas de figure : le degré est borné

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud[] fils;
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Variations sur les arbres - Arbres n-aires

- Second cas de figure : le degré est borné

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud[] fils;
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Fichier Noeud.java

```
public void affichePréfixe(){
 System.out.println(content.toString()+", "); // supposé exister
 for (int i=0;i<fils.length;i++)
  if (fils[i]!=null) fils[i].affichePréfixe();
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Second cas de figure : le degré est borné

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud[] fils;
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Fichier Noeud.java

```
public void afficheSuffixe(){
 for (int i=0;i<fils.length;i++)
  if (fils[i]!=null) fils[i].afficheSuffixe();
 System.out.println(content.toString()+", "); // supposé exister
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Second cas de figure : le degré est borné

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private Noeud[] fils;
}
```

- On peut éventuellement définir un parcours infixe, très général

Fichier Noeud.java

```
public void afficheInfixe(int k){ // k=0 : prefixe // k=length : suffixe
 int i;
 for (i=0 ; i<Math.min(k,fils.length);i++)
  if (fils[i]!=null) fils[i].afficheInfixe(k);
 System.out.println(content.toString()+", "); // supposé exister
 for ( ; i< fils.length;i++) // reprend là où on en était
  if (fils[i]!=null) fils[i].afficheInfixe(k);
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Troisième cas de figure : le degré est changeant

Variations sur les arbres - Arbres n-aires

- Troisième cas de figure : le degré est changeant

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private List<Noeud> fils;
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Variations sur les arbres - Arbres n-aires

- Troisième cas de figure : le degré est changeant

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private List<Noeud> fils;
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Fichier Noeud.java

```
public void affichePréfixe(){
 System.out.println(content.toString()+", "); // supposé exister
 for (Noeud x:fils)
  if (x!=null) x.affichePréfixe();
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Troisième cas de figure : le degré est changeant

Fichier Noeud.java

```
public class Noeud {
 private E content;
 private List<Noeud> fils;
}
```

- Les parcours préfixes et suffixes se définissent de la même façon

Fichier Noeud.java

```
public void afficheSuffixe(){
 for (Noeud x:fils)
  if (x!=null) x.afficheSuffixe();
 System.out.println(content.toString()+", "); // supposé exister
}
```

Université de Paris

Variations sur les arbres - Arbres n-aires

- Troisième cas de figure : le degré est changeant

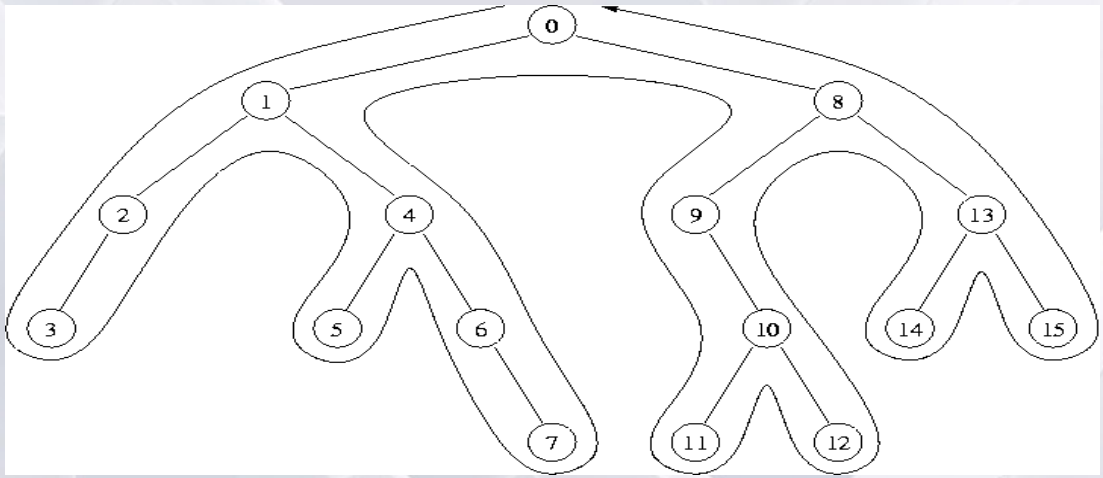
Fichier Noeud.java

```
public class Noeud {
 private E content;
 private List<Noeud> fils;
}
```

- Je vous laisse écrire un parcours infixe(k), et préciser sa définition

Exercice 6 - Créer un arbre rapidement

- En TP particulièrement on voudrait construire des arbres de test
- Solution : traduire un arbre vers un tableau, et vice-versa.
- La traduction s'appuie sur le parcours en largeur (-1 représentera un noeud null)



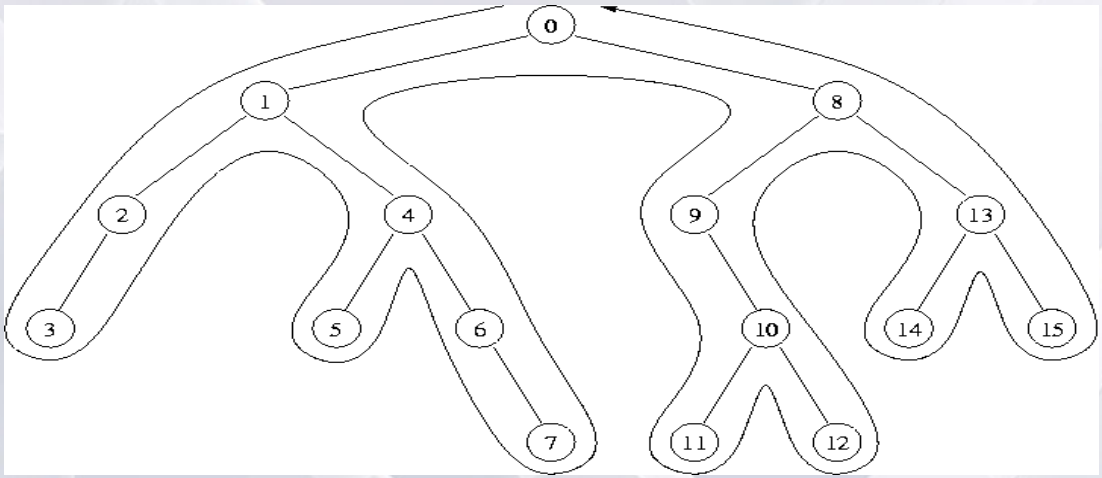
serait associé à

$[0; 1; 8; 2; 4; 9; 13; 3; -1; 5; 6; -1; 10; 14; 15; -1; -1; -1; -1; -1; 7; 11; 12]$

- Quel est le lien entre l'indice i dans le tableau et ses fils ?

Exercice 6 - Créer un arbre rapidement

- En TP particulièrement on voudrait construire des arbres de test
- Solution : traduire un arbre vers un tableau, et vice-versa.
- La traduction s'appuie sur le parcours en largeur (-1 représentera un noeud null)



serait associé à

$[0; 1; 8; 2; 4; 9; 13; 3; -1; 5; 6; -1; 10; 14; 15; -1; -1; -1; -1; -1; 7; 11; 12]$

- Quel est le lien entre l'indice i dans le tableau et ses fils ?

  réponse : fils gauche situé en $2i + 1$ et fils droit en $2i + 2$

Université de Paris

Exercice 6 - Array to Tree

Fichier Arbre.java

```
public static Arbre array2Tree(int [] tab){
 return new Arbre (Noeud.array2Noeud(tab,0));
}
```

Exercice 6 - Array to Tree

Fichier Arbre.java

```
public static Arbre array2Tree(int [] tab){
 return new Arbre (Noeud.array2Noeud(tab,0));
}
```

Fichier Noeud.java

```
public static Noeud array2Noeud(int[] tab, int pos) {
 if (pos>= tab.length || tab[pos]==-1) return null;
 return new Noeud( tab[pos] ,
  array2Noeud(tab,2*pos+1) ,
  array2Noeud(tab,2*pos+2)
 );
```

- Et c'est tout !

Université de Paris

Exercice 6 - Tree to Array (L'opération inverse)

Fichier Arbre.java

```
public int [] tree2Array(){
 int t[]=new int [(int)Math.pow(2,hauteur())-1];
 // il faut réserver cet espace, faire comme si l'arbre était complet
 for (int i=0;i<t.length;i++) t[i]=-1; // à priori
 if (racine!=null) racine.fill(t,0); // on délègue aux noeuds; 0 est la
     position courante
 return t;
}
```

Exercice 6 - Tree to Array (L'opération inverse)

Fichier Arbre.java

```
public int [] tree2Array(){
 int t[]=new int [(int)Math.pow(2,hauteur())-1];
 // il faut réserver cet espace, faire comme si l'arbre était complet
 for (int i=0;i<t.length;i++) t[i]=-1; // à priori
 if (racine!=null) racine.fill(t,0); // on délègue aux noeuds; 0 est la
     position courante
 return t;
}
```

Fichier Noeud.java

```
public void fill(int[] t, int i) {
 t[i]=val;
 // le noeud et son argument i changent de manière synchronisée
 if (g!=null) g.fill(t, 2*i+1);
 if (d!=null) d.fill(t, 2*i+2);
}
```

- Et c'est tout !

Université de Paris