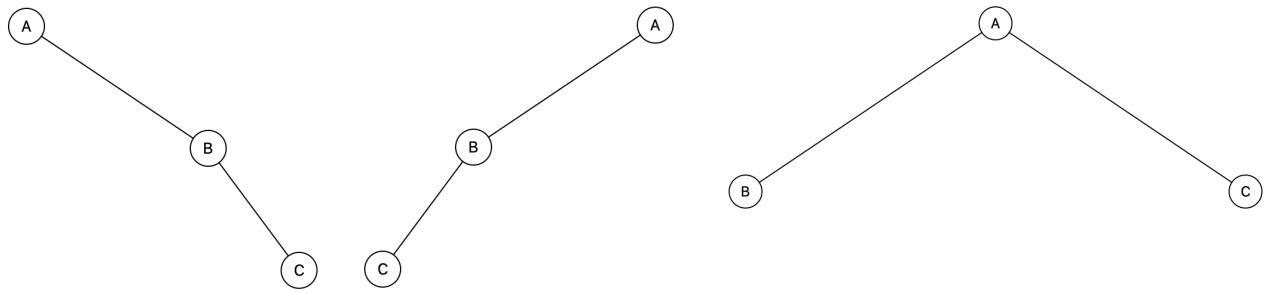
## Cours 16 décembre — Résumé de la séance

Nous avons présenté le sujet 0 de NSI et l'avons décrypté rapidement. Les exercices ne sont pas tous de la même difficulté : l'exercice 5 peut d'ailleurs se faire sans avoir fait le cours sur les réseaux ! L'exercice 2 est beaucoup plus sérieux et demande de la dextérité sur la récursivité.

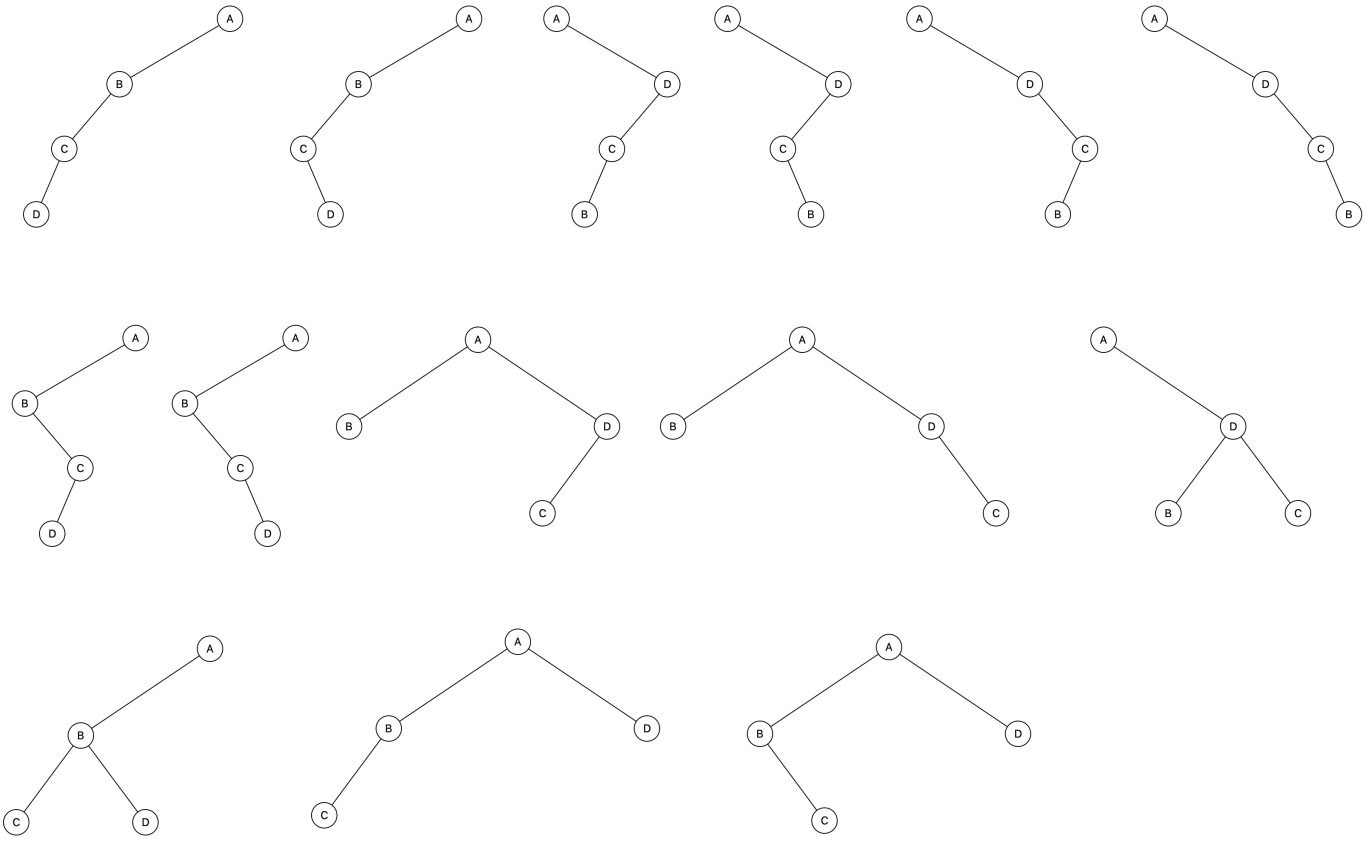
### 1) Arbres binaires — vocabulaire et démonstration — 1 heure

Nous avons commencé la séance par les arbres binaires :
☐ Correction exo 1 et 2 de la feuille d'exercices

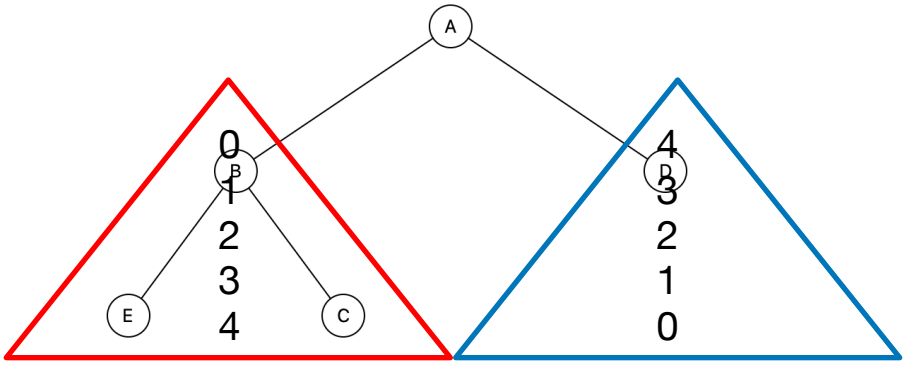
Ex 1 :
On liste tous les arbres à 3 et 4 noeuds :
**3 noeuds :**




**4 noeuds :**



Ex 2 :
L'idée est d'utiliser ce qu'on nous donne dans l'énoncé et de couper notre arbre en sous-arbres gauche et droit :




On met un noeud à la racine.
À gauche, nous pouvons donc mettre 4 noeuds, 3 noeuds, 2 noeuds, 1 noeud ou 0.
À droite, nous pouvons aussi mettre 0 noeuds, 1, 2, 3 ou 4.

Si on met 4 noeuds à gauche et 0 à droite, il y a 14 possibilités.
Si on met 3 noeuds à gauche et 1 à droite, il y a 5 possibilités à gauche et 0 à droite.
Si on met 2 noeuds à gauche et 2 à droite, il y a 2x2 possibilités.
Si on met 1 noeuds à gauche et 3 à droite, il y a 5 possibilités à droite et 0 à gauche.
Si on met 0 noeuds à gauche et 4 à droite, il y a 14 possibilités.

On fait la somme : 14+5+4+5+14 = 42 arbres possibles.

Pour le cours :
- [ ] Démonstration de l'encadrement de la taille d'un arbre en fonction de la hauteur de l'arbre à bien comprendre

**2) TP Arbres binaires — 1 heure**

Nous avons commencé le TP sur les arbres binaires que nous avancerons cette semaine.

TP :
- [ ] Algorithmes récursifs classiques

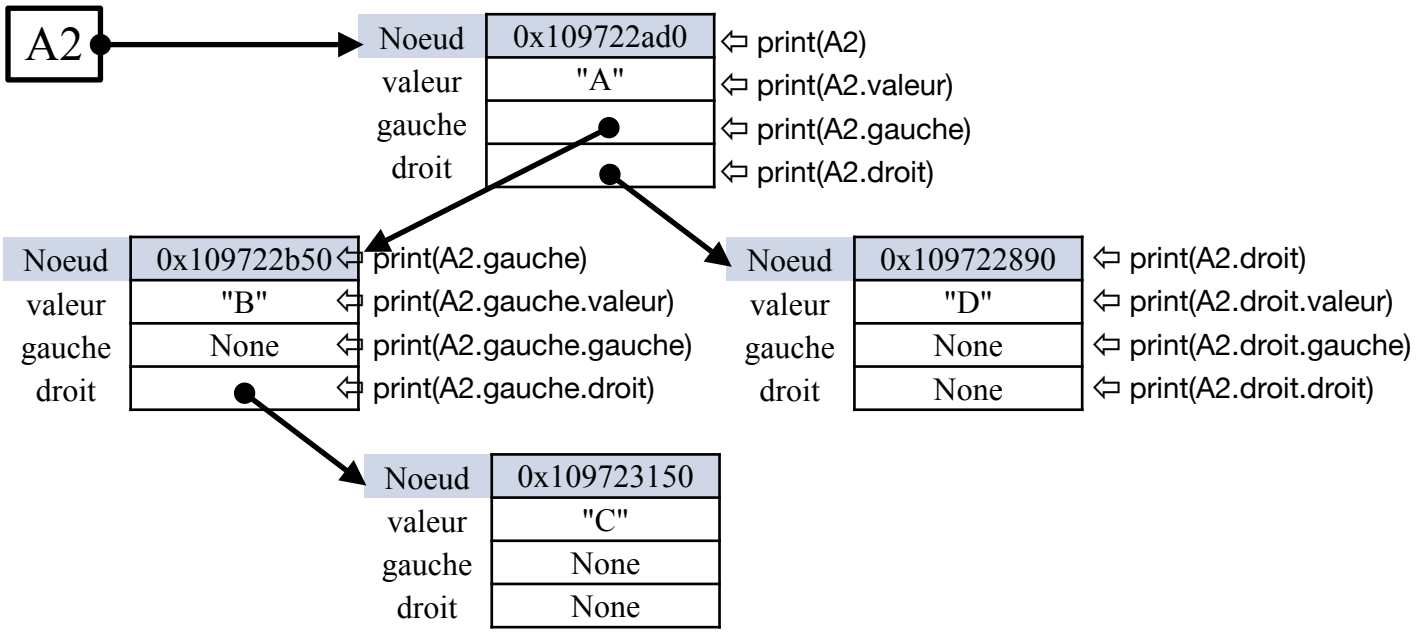
Quelques éléments de réponse :

```
class Noeud:

    def __init__(self, g = None, v=0, d=None):
        self.gauche = g
        self.valeur = v
        self.droit = d
```

```
A1 = Noeud(None, 'A', None)
```

```
A2 = Noeud(Noeud(None, 'B', Noeud(None, 'C', None)) , 'A',
Noeud(None, 'D', None))
```




Pour les fonctions récursives, nous allons utilisé le principe décrit sur le TP. Pour avoir la taille de l'arbre, il faut compter la racine et déterminer la taille du sous-arbre à gauche et du sous-arbre à droite. Cela donne :

```
def taille(self, G: Noeud):
    if G is None : return 0
    return 1 + self.taille(G.gauche) + self.taille(G.droit)
```

Même combat pour la fonction hauteur sauf que la hauteur se calcule en faisant 1 + maximum de la hauteur de l'arbre gauche et de l'arbre droit.

```
def hauteur(self, G: Noeud):
    if G is None : return 0
    return 1 + max(self.hauteur(G.gauche), self.hauteur(G.droit))
```