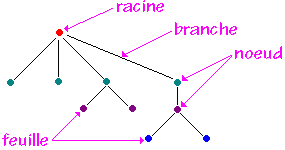
**Listes d’exercices**

Reprendre le code du TP sur les arbres de compétences, pour cela, il faut cloner le repository et utiliser la version avec parent <https://github.com/AurelienDidier/TP_SkillTree.git>

Pour ceux qui n’ont pas git, je vais vous les envoyer via discord.

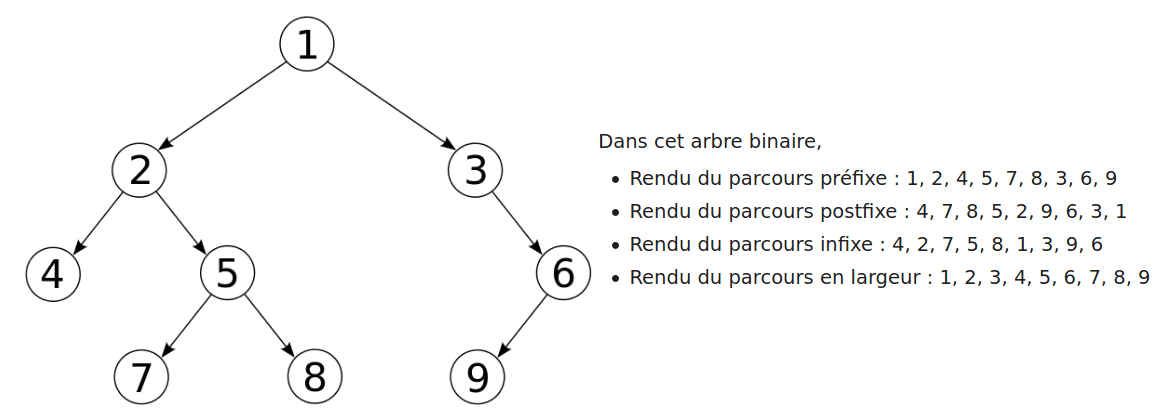
1. **Sur les arbres.**
2. Ecrire une fonction compte feuilles dans le fichier source et dans le header, qui compte le nombre de feuilles d’un arbre.



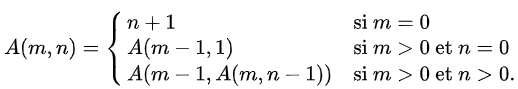
1. Changer les méthodes SetLeftSkillTree et setRightSkillTree pour éviter la création de cycle dans un arbre. (vérifier si le nœud que l’on veut ajouter a déjà un parent ou non)
2. Faire un constructeur SkillTree (SkillTree st, bool symetric) qui recréer le même arbre si symétrique est faux ou le symétrique du premier arbre si symetric est vrai.

Symétrique = les fils gauche deviennent des fils droit et inversement.

1. Ecrire un algorithme afficheEnLargeur qui affiche l’arbre en utilisant le parcours en largeur de l’arbre.



1. Faire une fonction transformeEnVector qui transforme un arbre en vecteur.
2. Faire la fonction inverse qui transforme un vecteur en arbre de sorte que l’arbre obtenu soit le même que l’arbre qu’on avait au début si on fait Arbre1 🡪 vector 🡪 Arbre2 alors Arbre 1 = Arbre2.
3. **Mémoisation**
4. Réécrire l’algo de calcul de la Fibonacci avec memoisation.
5. Faire l’algo de rendu de monnaie mais en utilisant la tabulation (plus simple)
6. Fonction d’Ackermann.
7. Ecrire la fonction pour calculer la valeur de la fonction d’Ackermann défini comme suit :



//Exemple en python:

def A(m,n):

if m == 0:

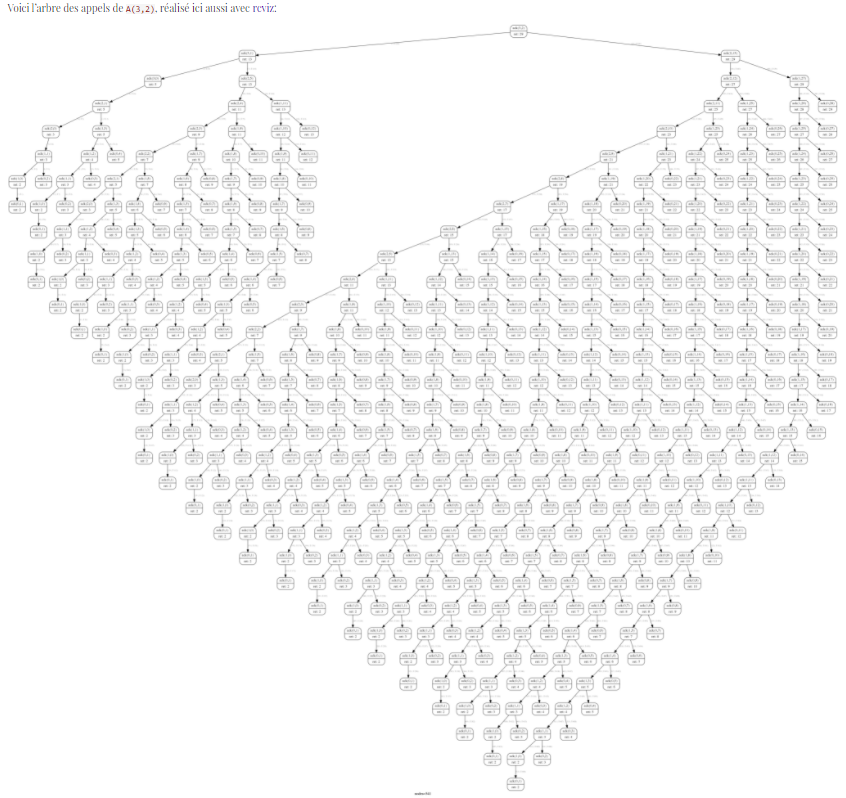
return n + 1

if n == 0:

return A(m - 1, 1)

return A(m - 1, A(m, n - 1))

1. Refaire l’exercice en utilisant la tabulation
2. Comparer les deux solutions en termes de temps de réponse



1. **Map**
2. Définir un classe Annuaire qui stocke une map de n de tel et de nom.

Et une méthode qui retrouve le numéro à partir du nom.

1. **Pointeurs**
2. Ecrire une fonction comparer qui compare deux pointeurs d’entiers (compare (int\* a, int\*b)) et qui renvoi « Egal » si deux pointeurs pointent vers des variables qui contiennent la même valeur, « Identique » si les deux pointeurs pointent vers la même variable ou « différent » si les deux pointeurs pointent vers des variables différentes avec des valeurs différentes.
3. Ecrire une fonction swap qui permet d’échanger deux entiers en utilisant le passage par valeur.
4. Ecrire une fonction swap qui permet d’échanger deux entiers avec des pointeurs en paramètres swap(int \*a, int \*b).