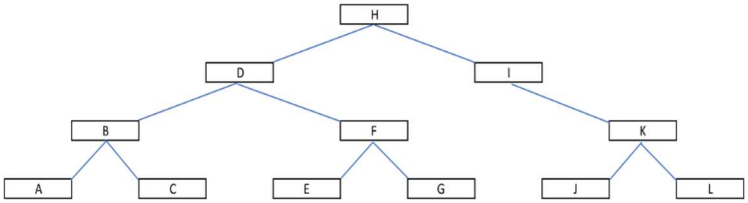
Algorithmes sur les **arbres binaires de recherche**

I. Définition , exemples, propriétés :

Un arbre binaire de recherche (ABR) (*binary search tree – BST*) est un arbre binaire tel que :

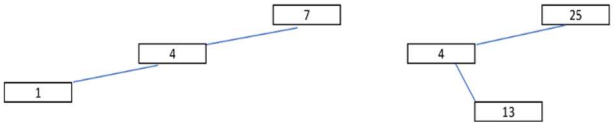
- **les clefs des nœuds doivent être ordonnables (il doit exister entre elles une relation d'ordre)**
- **pour chacun de ses nœuds:**
  - **chaque nœud du sous-arbre gauche a une clé inférieure (ou égale) à celle du nœud considéré,**
  - **chaque nœud du sous-arbre droit possède une clé supérieure (ou égale) à celle-ci**

Quelques exemples d'arbres binaires de recherche

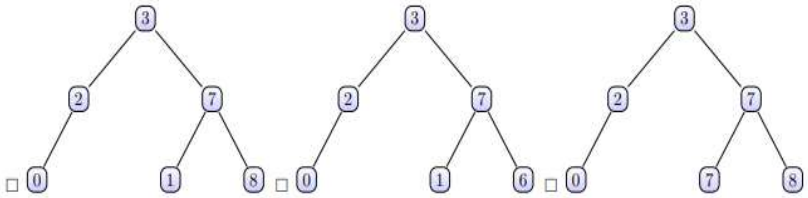


Chaque sous arbre d'un arbre binaire de recherche est donc un arbre binaire de recherche.

Un arbre binaire de recherche est donc une structure de données **récursive**.



Exercice 1 : Parmi les arbres suivants, entourer le s'il s'agit d'un arbre binaire de recherche (ABR) ?



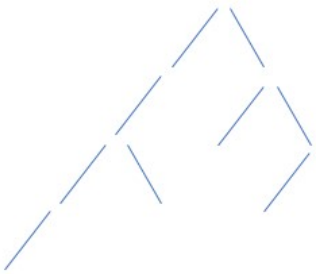
Exercice 2 : On donne ci-dessous une liste aléatoire de 14 nombres entiers :

25 60 35 10 5 20 65 45 70 40 50 55 40 15

Construire (dans l'ordre de la liste) l'arbre binaire de recherche associé :

Exercice 3 : On a représenté ci-contre le squelette d'un arbre de taille de 10.

1) Dessiner 2 autres squelettes différents d'arbre de taille 10
2) puis les compléter par les entiers de 1 à 10 tel qu'il soit des arbres binaires de recherche :



Que remarquez-vous ?.....

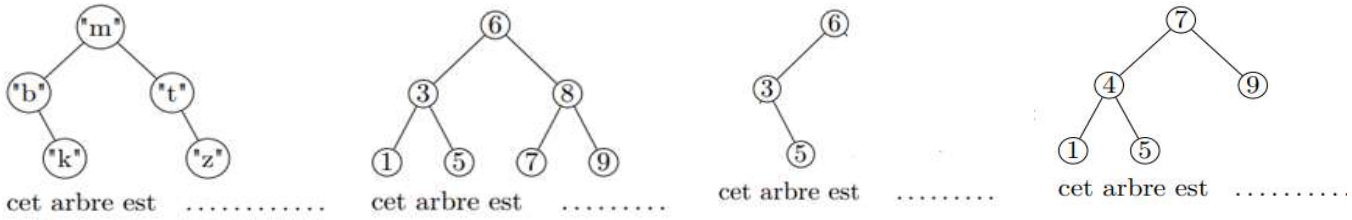
Propriétés :

- Dans un arbre binaire de recherche, où se trouve la plus petite valeur ? .
- Dans un arbre binaire de recherche, où se trouve la plus grande valeur ? .
- Quel type de parcours faut-il faire sur un ABR pour classer ses nœuds par ordre croissant ?

Autres définitions : Un arbre binaire de recherche est :

- **filiforme** : chaque nœud n'a qu'un fils sauf le dernier qui n'en a pas
- **complet** si tous les niveaux de l'arbre sont remplis sauf éventuellement le dernier qui contiendra ses nœuds le plus à gauche possible.
- **équilibré** si toutes ses feuilles possèdent la même hauteur.
- **parfait** si tous ses nœuds, excepté les feuilles, possèdent deux fils.

Exemples :



Si n est la taille de l'arbre et h sa hauteur, on a : $n = h + 1$ pour un filiforme et $n = 2^{h+1} - 1$ pour un équilibré

II. Algorithmes sur les arbres binaires de recherche

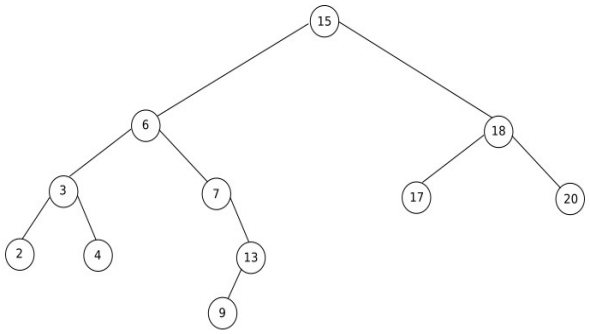
1. Rechercher la présence d'une valeur dans un arbre binaire de recherche (à connaitre)

Fonction recherche(arbre, valeur) -> booléen

Entrée : un arbre binaire de recherche arbre, une clé **valeur** comparable aux clé de l'arbre

Sortie : un booléen égal à Vrai si arbre contient la clé valeur et faux sinon

```
Début
    Si arbre ……..
        retourner ……..
    fin si
    Si ……..
        retourner Vrai
    fin si
    Si valeur ……
        retourner ……
    Sinon
        retourner ……..
    Fin si
Fin
```



Arbre 1

Appliquer l'algorithme à des exemples et donner le nombre d'opérations dans le meilleur et pire des cas

Etude de la **complexité** : Dans un arbre de recherche, la recherche est **en O(h) où h est la hauteur de l'arbre.**

Si n est la taille de l'arbre alors:

- si l'arbre est filiforme, $\boldsymbol{h = n - 1}$ où n est, donc le cout est linéaire
- mais si l'arbre est parfait, **h ≈ log₂ (n)** , le cout est logarithmique

En moyenne, on admet que le coût est **logarithmique**.

Un algorithme en O(log₂(n)) est plus "efficace" qu'un algorithme en O(n)

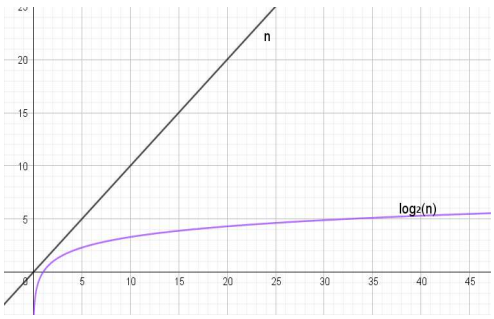
C'est quoi $log_2(n)$?

Un peu de math : $\log_2(x)$ est fonction réciproque de la fonction 2^x

exemple : $2^4 = 16 <=> \log_2(16) = 4$

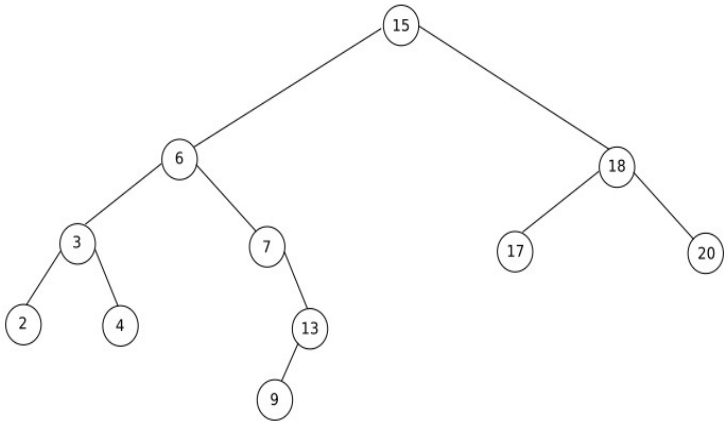
autre définition : $\log_2(n)$ est environ le nombre de bits pour écrire en binaire un entier n

$$42 = 101010_2 \ et \ log_2(42) \approx 5.4$$



2. Insérer une clé v dans un arbre binaire de recherche

A la main, insérer 5, puis 14, puis 19 dans l'arbre ci-dessous . et si on veut insérer 3 ?



L'insertion d'une clé va se faire au **niveau d'une feuille**, donc au bas de l'arbre.

Insérer une clé e dans un ABR A revient à construire un ABR qui contient e et toutes les clés de A.

Le principe est relativement simple :

- si l'ABR est vide, on construit un ABR possédant un unique noeud de clé e.
- si e est inférieure ou égale à l'étiquette de A il faut insérer la clé dans le sous-arbre gauche de A ce qui revient à créer un *nouvel* ABR dont :
  - l'étiquette est celle de A (inchangée) ;
  - le sous-arbre gauche est le sous-arbre gauche de A dans lequel on insère la clé e ⇒ appel récursif
  - le sous-arbre droit est celui de A (inchangé).
- si e est strictement supérieure à l'étiquette de A il faut insérer la clé dans le sous-arbre droit de A en procédant de manière similaire.

Tp notebook Arbre_binaire_recherche.ipynb

- Implémentation des fonctions **insérer** avec jeux de tests
- Écrire la fonction **recherche** qui cherche la présence d'une valeur dans un ABR
- Ecrire deux fonctions qui retournent le minimum et le maximum d'un ABR
- Ecrire une fonction qui à partir d'une liste d'éléments comparables renvoie l'arbre binaire associé.

  A tester avec : liste = ['cool','nsi','info','pc','arbre','alan','turing','souris','clavier']
- Ecrire une fonction chemin qui renvoie dans une liste les nœuds visités pour accéder à l'élément v d'un arbre binaire de recherche