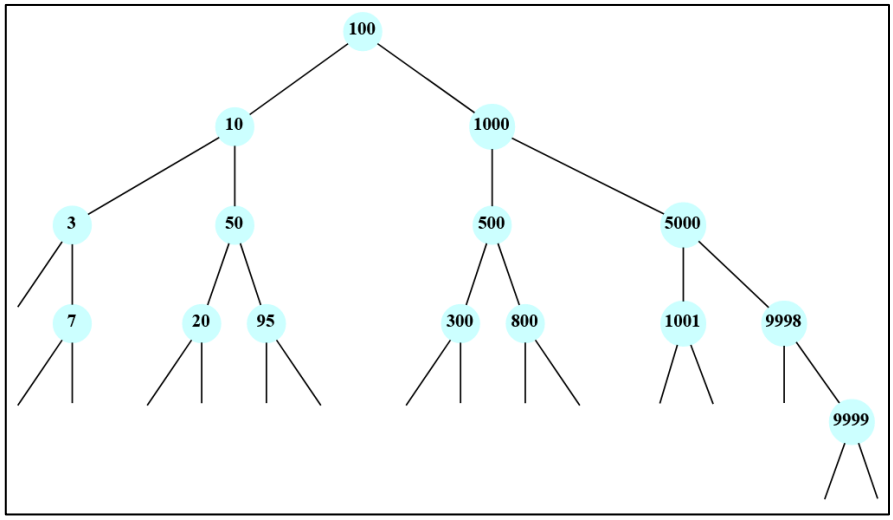
# Arbres binaires de recherche : exercices

**Exercice 1 :**

On considère l'arbre binaire de recherche ci-contre.




1) On décide d'y insérer, avec l'algorithme vu en cours, la valeur $29$ : où sera-t-elle insérée ?

2) Même question avec $900$.

3) Même question avec $2$ puis $-1$ puis $-7$ puis $-12$.

4) Est-e que l'ordre dans lequel on insère les valeurs a une influence sur l'arbre binaire de recherche obtenu ?

**Exercice 2 :**

Construire deux arbres binaires de recherche comportant les valeurs $1, 2, 3, 4, 5, 6$ et $7$ : le premier de hauteur $7$ et le second de hauteur $3$.

**Exercice 3 :**

1) Avec une complexité en $O(\log N)$, par combien est approximativement multipliée le temps d'exécution d'un algorithme lorsque la taille de l'entrée est multipliée par $1\ 000\ 000$ ?
2) Et avec une complexité en $O(N)$ ?
3) Donner un exemple d'ABR A de taille 1000000 et d'un ABR B de taille 1000 tel que la recherche de l'élément $777$ dans A est plus rapide que dans B.

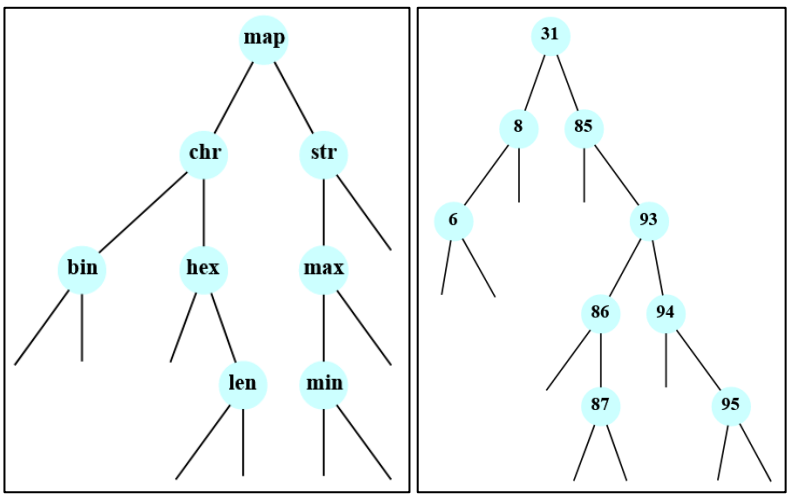
**Exercice 4 :**

1) Les deux arbres binaires ci-contre sont-ils des ABR ?

1) On parcourt l'arbre de gauche avec un parcours en largeur. Dans quel ordre les noeuds sont-ils parcourus ?
   Et avec un parcours infixe ?

2) Même question pour l'arbre de droite.

3) Donner un algorithme de parcours d'arbre binaire de recherche tel que les nœuds sont parcourus dans l'ordre décroissant.




**Exercice 5 :**

Donner un algorithme permettant de renvoyer la valeur maximale d'un arbre binaire de recherche.
Même question avec la valeur minimale.

**Exercice 6 (difficile sur la fin)** :

On suppose disposer d'une classe `ABR` disposant uniquement d'un attribut `_racine` ainsi que des méthodes `insere` et `recherche` (comme en TP). On suppose en outre que la méthode `insere` permet de construire un arbre *équilibré*.

1) L'utilisateur de la classe `ABR` a-t-il le droit d'utiliser l'attribut racine dans son code ? Peut-il le faire ?

2) Ecrire un algorithme `tab_2_abr` qui prend en argument un tableau d'entiers ou de chaînes de caractères et renvoie un arbre binaire de recherche comportant tous les éléments du tableau.

3) Ecrire un algorithme `abr_2_tab` qui prend en argument un arbre binaire de recherche et un tableau et mute le tableau pour qu'il contienne toutes les valeurs présentes dans l'ABR triées par ordre croissant (voir exercice 4 question 2 si besoin).

4) En utilisant ce qui précède, montrer que l'on peut ainsi disposer d'une méthode de tri efficace. Quelle est sa complexité ?

5) Vous implémentez cette méthode. Le tableau non trié est de taille $N = 3407$, le tableau trié est de taille $N' = 3341$. D'où peut venir le problème ?