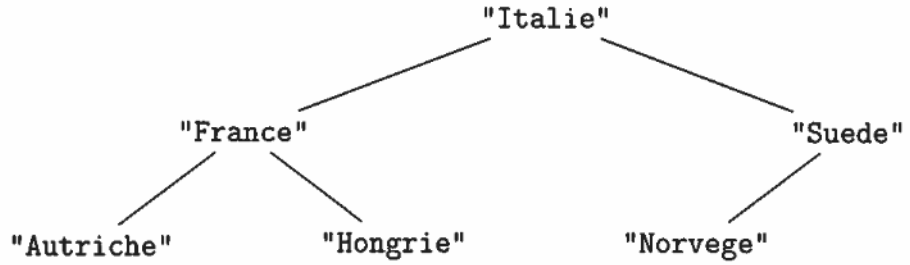
Exercice 3 (4 points).

Cet exercice porte sur les arbres binaires de recherche et leurs algorithmes associés.

Les arbres binaires de recherche considérés ici sont des arbres binaires où les nœuds désignent des chaînes de caractères et pour lesquelles la valeur de chaque nœud est supérieure à celles des nœuds de son enfant gauche, et inférieure à celles des nœuds de son enfant droit.
La relation d'ordre notée $<$ est ici la relation d'ordre alphabétique.

Dans cet exercice, on utilisera la convention suivante : la hauteur d'un arbre binaire ne comportant qu'un nœud est 1.
Dans cet exercice les arbres binaires de recherche ne contiennent que des noms de pays tous distincts.

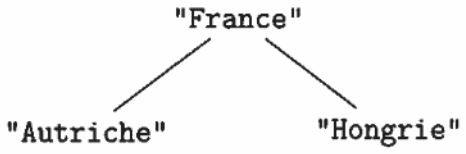
On considère l'arbre binaire de recherche suivant :



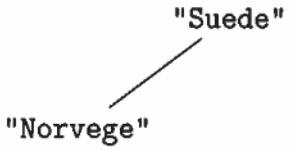
1. (a) Donner sans justification la hauteur de cet arbre.
   (b) Donner sans justification la valeur booléenne de l'expression `"Allemagne" < "Portugal"`.
   (c) Recopier l'arbre après l'ajout de `"Allemagne"`, de `"Portugal"` et de `"Luxembourg"` dans cet ordre.

Pour les questions 2, 3 et 4, on traite l'arbre initial, donc sans l'ajout de `"Allemagne"`, `"Portugal"` et `"Luxembourg"`.

2. On souhaite parcourir l'arbre. Indiquer l'ordre de visite des nœuds lors d'un parcours en largeur.
3. On souhaite écrire une fonction pour déterminer si le nom d'un pays est dans l'arbre.
   On dispose pour cela de :
   - la fonction `est_vide` qui prend en paramètre un arbre `arb`. Cette fonction renvoie `True` si l'arbre `arb` est vide, `False` sinon ;
   - la fonction `gauche` qui prend en paramètre un arbre `arb` et renvoie son sous-arbre gauche.
     Exemple : si `A` est notre arbre initial, `gauche(A)` renvoie



   - la fonction `droite`, qui prend en paramètre un arbre `arb` et renvoie son sous-arbre droit.
     Exemple : si `A` est notre arbre initial, `droite(A)` renvoie :



— la fonction `racine`, qui prend en paramètre un arbre `arb` et qui renvoie la valeur de la racine de l'arbre.

Exemple : `racine(A)` renvoie `"Italie"`.

Recopier, en complétant les lignes, 2, 6, 7 et 10, la fonction `recherche` donnée ci-dessous et écrite en Python. Cette fonction prend en paramètre un arbre `arb` et une valeur `val`. L'appel `recherche(arb, val)` renvoie un booléen (`True` si la valeur `val` est dans l'arbre `arb`, `False` sinon).

```
1  def recherche(arb, val):
2      """--------------------"""
3      if est_vide(arb):
4          return False
5      if val == racine(arb):
6          return ________
7      if val _________________:
8          return recherche (gauche(arb), val)
9      else:
10         return ________
```

4. Écrire une fonction récursive `taille` permettant de déterminer le nombre de pays présent dans un arbre.

   Cette fonction prendra en paramètre un arbre `arb` et renverra un entier.