Vade-Mecum: ‘**find\_smallest\_common\_ancestor()**’.

* Fonctionnement attendu de la fonction :
* Description :

Fonction ‘**find\_smallest\_common\_ancestor()**’, recherche dans un arbre binaire trié quel est le parent commun le plus proche de deux nœuds et renvoie ce nœud.

* Données d’entrée :
  + ‘root’ : Node, racine/point de départ dans l’arbre binaire trié dans lequel la recherche sera effectuée.
  + ‘left\_child’ : Node, premier des deux enfants du nœud recherché.
  + ‘right\_child’, Node, deuxième des deux enfants du nœud recherché.
* Pré-conditions :
  + Les deux enfants font parties de l’arbre.
  + L’enfant de gauche est plus petit que l’enfant de droite.
  + La ‘root’ est différente des 2 enfants.
  + Les enfants sont deux nœuds de valeurs différentes.
* Données de sortie :
  + ‘ancestor’, Node, le plus petit ancêtre commun.
* Post-conditions :
  + L’arbre n’est pas modifié.
  + Le plus petit ancêtre commun est plus grand que l’enfant de gauche.
  + Le plus petit ancêtre commun est plus petit que l’enfant de droite.
* Prototype et docstring :

*def* find\_smallest\_common\_ancestor(root, left\_child, right\_child):  
 *"""  
 Recherche dans un arbre binaire 'tree' trié quel est le parent commun le plus proche de deux nœuds et renvoie ce nœud.* ***:param*** *root: Node, racine/point de départ dans l'arbre binaire trié dans lequel la recherche sera effectuée.* ***:param*** *left\_child: Node, premier des deux enfants du nœud recherché.* ***:param*** *right\_child: Node, deuxième des deux enfants du nœud recherché.* ***:return****: ancestor, Node, le plus petit ancêtre commun.  
 """*

* Plan de tests :
* L’ancêtre est trouvé.
* Enfant(s) n’appartenant pas à l’arbre.
* Root différent des enfants.
* Les enfants sont des nœuds de valeurs différentes.
* Idée :
* Description :

L’élément recherché doit être plus grand que son fils de gauche et plus petit que son fils de droite. En partant de la racine, on cherche si les fils sont plus grands, plus petits ou répartis des deux côtés de la racine.

* Organigramme ou pseudo-code :

SI left\_child et right\_ child sont dans l’arbre :

SI root est plus grand que left\_ child et root est plus petit que right\_ child:

Retourner root

SINON

SI left\_ child et right\_ child sont plus petits que root:

Appel récursif de la fonction, où root prend pour paramètre ‘root.left’.

SINON SI left\_ child et right\_ child sont plus grands que root:

Appel récursif de la fonction, où root prend pour paramètre ‘root.right’.

SINON

Retourner un message.

* Données de travail éventuelles :

Aucune.