TD n° 9 - Arbres équilibrés

Dans tout ce TD, on considère que les arbres sont définis comme il a été vu en cours à l'aide des classes suivantes :

```
class Arbre:
    pass
def arbre(n=None):
    a = Arbre()
    a.racine = n
    return a
class Noeud:
    pass
def noeud(x):
    n = Noeud()
    n.valeur = x
    n.parent = None
    n.gauche = None
    n.droit = None
    return n
```

Exercice 1. *Rotations*

En algorithmique, on appelle *rotation* une opération qui transforme un arbre binaire de recherche en modifiant la position de certains nœuds tout en conservant l'ordre des éléments.
Une rotation touche principalement deux nœuds, l'un étant le fils de l'autre et permet de faire remonter le fils tout en descendant le père pour inverser leur niveau dans l'arbre. Il existe deux rotations : droite et gauche, illustrées sur la figure 1.

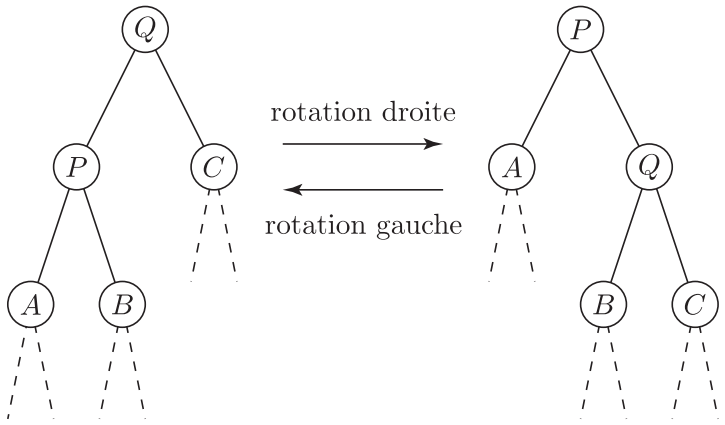

FIGURE 1 – Rotations droite et gauche sur un arbre binaire.

Remarque : On ne modifie pas les sous-arbres correspondant aux nœuds A, B et C.

1. Écrivez les fonctions `rotation_droite(q)` et `rotation_gauche(p)` qui prennent en argument un nœud et effectuent la rotation correspondante.

2. Quelle est la complexité des rotations ?

3. En supposant que la hauteur du sous-arbre enraciné en B (sur la figure 1) soit inférieure ou égale aux hauteurs des sous-arbres enracinés en A et C, comment varient les hauteurs des sous-arbres gauches et droit de l'arbre lorsque l'on effectue une rotation droite ?

Exercice 2. *Arbres équilibrés*

Les arbres équilibrés (ou arbres AVL du nom de leurs inventeurs G.M. Adelson-Velskii et E.M. Landis) sont des arbres binaires tels que pour tout nœud de l'arbre, la différence entre les profondeurs du sous-arbre gauche et du sous-arbre droit est d'au plus 1. On suppose de plus que chaque nœud connaît la différence entre les profondeurs de ses sous-arbres droit et gauche (on rajoute un champ `n.delta` aux nœuds qui contient cette différence, et qui sera mis à jour lorsque l'on modifie l'arbre).

1. En notant $N(h)$ le nombre minimal de sommets d'un arbre équilibré de profondeur h, montrez que $N(h+2) \geq 2N(h) + 1$.

2. En déduire par récurrence que pour tout $h \geq 1$, $N(2h) \geq 2^h$, et que la profondeur d'un arbre équilibré de taille n (la taille d'un arbre est son nombre de sommets) est inférieure à $O(\log_2(n))$.

3. Quelle est la complexité de la recherche d'un élément dans un arbre binaire de recherche dans le pire des cas en fonction de la hauteur de l'arbre ? En déduire la complexité de la recherche dans un arbre binaire de recherche équilibré dans le pire des cas en fonction du nombre de sommets de l'arbre.

Lorsque l'on insère un nouveau nœud dans un arbre équilibré, il est possible de déséquilibrer les sous-arbres d'un nœud de telle sorte que l'arbre ne soit plus équilibré. Il faut alors transformer l'arbre à l'aide de rotations pour le ré-équilibrer.

4. Écrivez la fonction `inserer(x, n)` qui insère la valeur `x` dans un arbre binaire de recherche dont la racine est `n` et qui renvoie le nouveau nœud contenant la valeur `x`.

5. Quels sont les nœuds qui peuvent avoir été déséquilibrés lorsque l'on insère une valeur dans un arbre binaire équilibré ? (comment les trouver si l'on connaît le nœud qui vient d'être ajouté)

Afin de ré-équilibrer l'arbre après une insertion, on décide donc de remonter à travers tout l'arbre pour vérifier que les ancêtres du nœud inséré sont équilibrés.

6. Écrivez une fonction `verifie(n)` qui met à jour les champs `delta` de chacun des ancêtres du nœud `n` qui vient d'être inséré et renvoie le premier nœud pour lequel la différence est de ± 2 (ou `None` s'il n'y en a aucun).

Supposons que l'on ait trouvé le premier nœud n (en partant du nœud que l'on vient d'insérer) dont la différence des profondeurs est 2 (le sous-arbre droit est plus profond que celui de gauche). Il va alors falloir remonter son fils droit n_d à l'aide d'une rotation gauche entre les nœuds n et n_d.

7. Que se passe-t-il si la différence des profondeurs des fils droit et gauche de n_d est égale à -1 (le fils gauche est plus profond que le fils droit) lorsque l'on fait une rotation gauche entre les nœuds n et n_d.

Remarque : Il est fortement conseillé de faire un schéma.

Pour éviter le problème de la question précédente, on s'assure avant de faire la rotation entre n et n_d que le sous-arbre droit de n_d est au moins aussi profond que son sous-arbre gauche. Si ce n'est pas le cas, on effectue une rotation droite entre n_d et son fils gauche.

8. Écrivez la fonction `corrige(n)` qui effectue les rotations nécessaires pour corriger une erreur d'équilibrage rencontrée en un nœud n.

Indication : Il faut commencer par regarder lequel des fils de n doit être remonté, puis regarder s'il faut d'abord effectuer une rotation au niveau de ce fils avant de faire la rotation au niveau de n.

9. Écrivez la fonction `inserer_AVL(x, n)` qui insère une valeur `x` dans un arbre binaire de recherche équilibré tout en le maintenant équilibré.

10. Quelle est la complexité de l'opération d'insertion dans un arbre binaire de recherche équilibré dans le pire des cas en fonction du nombre de nœuds de l'arbre.