TP 5 : Arbres binaires - arbres binaires de recherche

Ce TP se base sur trois fichiers que vous devrez compléter :
- `arraystack.py` qui implémente une pile mutable de taille maximale fixée avec un tableau
- `bintree.py` qui implémente un arbre binaire mutable
- `treeset.py` qui implémente un ensemble mutable d'entiers avec un arbre binaire de recherche

1 Pile

Complétez l'implémentation de `arraystack.py` avec la fonction `copy(self)` qui renvoie une copie profonde de la pile.

2 Arbre binaire

On souhaite représenter un arbre binaire avec une collection de *nœuds*. Chaque nœud possède une valeur entière dans son champ `val`, et un fils gauche (champ `left`) et un fils droit (champ `right`) qui sont eux-mêmes des nœuds (ou la valeur spéciale `None` si les fils correspondants sont vides).

Un objet de type arbre (`BinTree`) contient deux champs.
- le champ `current` contient le nœud courant. Ce champ vaut `None` si et seulement si l'arbre est vide. Le nœud courant n'est pas nécessairement la racine de l'arbre.
- le champ `parents` est une pile. Cette pile est vide si l'arbre est vide, ou si le nœud courant est la racine de l'arbre. Dans le cas contraire, la pile contient les ancêtres ordonnés du nœud courant, de façon à ce que le sommet de la pile soit le père du nœud courant de l'arbre, l'élément suivant soit le grand-père, etc... Cette pile sera utile pour remonter dans l'arbre.

Complétez le fichier `bintree.py` pour les fonctions suivantes (le détail de la spécification de chaque fonction est dans le fichier).
- `is_empty(self)` teste si l'arbre est vide.
- `is_left_empty(self)` teste si le fils gauche du nœud courant est vide.
- `is_right_empty(self)` teste si le fils droit du nœud courant est vide.
- `goto_father(self)` positionne le nœud courant sur le père du nœud courant.
- `goto_root(self)` positionne le nœud courant à la racine de l'arbre.
- `goto_left(self)` positionne le nœud courant sur son fils gauche.
- `goto_right(self)` positionne le nœud courant sur son fils droit.
- `add_left(self,v)` ajoute un fils gauche au nœud courant. Le nouveau nœud a la valeur `v`. Le nœud courant devient ce nouveau nœud.
- `add_right(self,v)` ajoute un fils droit au nœud courant. Le nouveau nœud a la valeur `v`. Le nœud courant devient ce nouveau nœud.

3 Arbre binaire de recherche

On souhaite représenter un ensemble fini d'entiers avec arbre binaire de recherche. Un arbre binaire de recherche est un arbre où la valeur de chaque nœud
- est strictement supérieure à la valeur des nœuds de son fils gauche
- est strictement inférieure à la valeur des nœuds de son fils droit

Un objet de type `TreeSet` contient un seul champs `tree` qui contient un objet de type `BinTree`.

Complétez le fichier `treeset.py` pour les fonctions suivantes.
- `mem(self,v)` teste si la valeur `v` appartient à l'ensemble.
- `add(self,v)` ajoute la valeur `v` à l'ensemble si elle n'y appartient pas déjà.

4 Bonus

Ajoutez les fonctions suivantes à la classe `BinTree`.
- `remove(self)` supprime le nœud courant et se positionne sur son père, si il existe. Si le nœud courant a un fils (unique), il prend sa place dans l'arbre.
- `swap_ancestor(self,i)` permute les champs `val` du nœud courant et de son i-eme ancêtre.

Ajoutez la fonction suivante à la classe `TreeSet`.
- `remove(self,v)` supprime la valeur `v` de l'ensemble si elle y appartient.