Bases de la programmation impérative

Ensimag 1A

1. Listes doublement chaînées

On travaille aujourd'hui sur les listes doublement chaînées. Dans une liste doublement chaînée, chaque cellule contient deux pointeurs : un pointeur vers la cellule suivante et un autre vers la cellule précédente. Par exemple on peut dessiner la liste des entiers (1,3,2) comme figure 1.

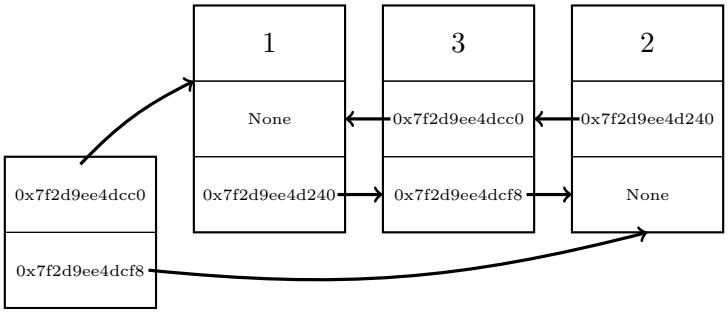

FIGURE 1 – liste doublement chaînée

Le type `Cellule` devient donc :

```
class Cellule:
    """
    une cellule d'une liste. contient une valeur, un pointeur
    vers la cellule suivante et un autre vers la cellule precedente.
    """
    # pylint: disable=too-few-public-methods
    def __init__(self, valeur, precedent, suivant):
        self.valeur = valeur
        self.suivant = suivant
        self.precedent = precedent
```

Le type `Liste` reste inchangé, avec un pointeur de tête et un pointeur de queue.

1.1. Écrire les méthodes d'ajout d'éléments : `ajouter_en_tete(self, valeur)` et `ajouter_en_queue(self, valeur)`. Pour cet exercice, ces méthodes seront les deux seules méthodes permettant la création de cellules.

1.2. Programmez un itérateur `cellules` itérant sur toutes les cellules de la liste (de la tête à la queue). Il devra être résistant aux modifications sur la cellule courante.

1.3. En utilisant `next`, `filter` et l'itérateur sur les cellules, programmez une méthode `recherche(self, valeur)` renvoyant la première cellule contenant la valeur donnée (`None` si pas trouvée).

1.4. Programmez une méthode `enlever_cellule(self, cellule)` enlevant la cellule donnée, qui fait partie de la liste.

1.5. En supposant que notre liste soit de longueur ≥ 2 et contienne des entiers, proposez une méthode `entiers_proches(self)` renvoyant un couple d'entiers les plus proches de la liste.

1.6. Écrire une méthode `supprimer_doublons`, éliminant tous les doublons de la liste.

1.7. Écrire une méthode `entrelacer(self, autre)` qui entrelace les deux listes dans `self`. Par exemple, entrelacer (0,2,4,6,8,10,12) et (1,3,5,7,9) donne (0,1,2,3,4,5,6,7,8,9,10,12) ; entrelacer (0,1,2,3) et (2,5,4,7) donne (0,2,1,5,2,4,3,7). On supposera qu'aucune liste n'est vide ; en sortie `autre` doit être vide.