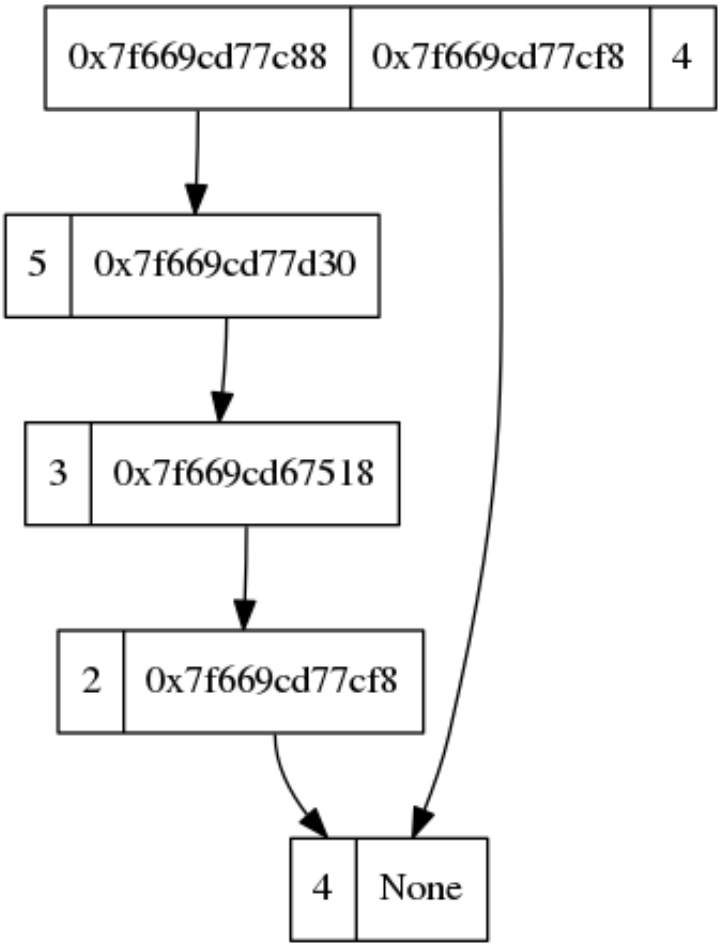
4.1 Listes simplement chaînées

On cherche à implémenter des listes simplement chaînées similaires à celles vues en TD. On dispose d'un type `Cellule` contenant une valeur et un pointeur vers une cellule suivante ainsi que d'un type `Liste` contenant un pointeur vers la première cellule, un autre vers la dernière cellule et enfin un compteur comptabilisant le nombre total d'éléments stockés dans la liste.

Une liste composée des entiers 5, 3, 2, 4 ressemble donc à :



On vous demande donc de compléter le fichier *listes.py* ci-dessous puis de tester l'exécution.

```
#!/usr/bin/env python3
"""
listes simplements chainees + quelques operations
"""
#from tycat import trace


class Cellule:
    """
    une cellule d'une liste. contient une valeur et un pointeur
    vers la cellule suivante.
    """
    # pylint: disable=too-few-public-methods
    def __init__(self, valeur, suivant=None):
        self.valeur = valeur
        self.suivant = suivant
```

```python
class Liste:
    """
    une liste simplement chainee.
    contient un pointeur sur la cellule en tete de liste et un autre sur
    la queue de liste.
    un compteur permet de savoir rapidement la taille de la liste.
    """
    def __init__(self):
        self.tete = None
        self.queue = None
        self.taille = 0

    #@trace
    def ajouter_en_tete(self, valeur):
        """
        ajoute une cellule en tete. cout : O(1).
        """
        # TODO
        pass

    #@trace
    def ajouter_en_queue(self, valeur):
        """
        ajoute une cellule en queue. cout : O(1).
        on peut le faire rapidement grace au pointeur de queue.
        """
        # TODO
        pass

    def cellules(self):
        """
        iterateur sur chaque cellule.
        """
        # TODO
        pass

    def recherche(self, valeur):
        """
        renvoie la premiere cellule contenant la valeur donnee.
        """
        # TODO
        pass

    #@trace
    def supprimer(self, valeur):
        """
        enleve la premiere cellule contenant la valeur donnee.
        """
        # TODO
        pass

    def __str__(self):
        """
        affiche (val1, val2, val3....)
        """
        # TODO
        return ""


def test_listes():
    """
    on teste toutes les operations de base, dans differentes configurations.
    """
```

```
    exemple = Liste()
    exemple.ajouter_en_tete(3)
    exemple.ajouter_en_tete(5)
    exemple.ajouter_en_queue(2)
    exemple.ajouter_en_queue(4)
    print("exemple : ", exemple)
    print("recherche : ", exemple.recherche(3).valeur)
    print("adresses des cellules : ",
          ",".join([hex(id(c))for c in exemple.cellules()]))
    exemple.supprimer(5)
    print("apres suppression de 5 : ", exemple)
    exemple.supprimer(4)
    print("apres suppression de 4 : ", exemple)

if __name__ == "__main__":
    test_listes()
```

listes.py

Pour vérifier votre code, on vous fournit le fichier de tests suivant : test_liste.py