Soutien en algorithmique et programmation

Séance 10 : listes doublement chainées

Introduction

On va travailler sur des listes doublement chainées. Chaque cellule d'une liste doublement chainée contient non seulement une valeur et un champ suivant comme pour les listes classiques, mais aussi un champ pointant sur la cellule précédente dans la liste. La classe `Cellule` que l'on utilisera sera donc définie par :

```
class Cellule:
    """
      Une cellule est constituee :
      - d'une valeur
      - d'un pointeur vers la cellule precedente
      - d'un pointeur vers la cellule suivante
    """
    # pylint: disable=too-few-public-methods

    def __init__(self, val, prec, suiv):
        """
          Constructeur
        """
        self.val = val
        self.prec = prec
        self.suiv = suiv

    def __str__(self):
        """ Afficheur """
        return f"{self.val} <-> "
```

On équipera nos listes d'éléments fictifs en tête et en queue (sentinelles), qu'on stockera dans un tableau de deux cases qu'on accèdera proprement grace à des constantes. On pourra tester les fonctions avec le programme principal suivant :

```
IX_T = 0
IX_Q = 1

def creer():
    """
      Cree une liste doublement chainee
    """
    liste = (Cellule('T', None, None), Cellule('Q', None, None))
    liste[IX_T].suiv = liste[IX_Q]
    liste[IX_Q].prec = liste[IX_T]
    return liste

def main():
    """
      Fonction principale
    """
    liste = creer()
    remplir(liste, [randint(0, 9) for _ in range(5)])
```

```
print("Liste initiale : ", end="")
afficher(liste)
print("Liste inversee : ", end="")
afficher(liste, True)
cell = liste[IX_T].suiv
while cell is not liste[IX_Q].prec:
    echanger(cell)
    print("Test echanger  : ", end="")
    afficher(liste)
trier(liste)
print("Liste triee    : ", end="")
afficher(liste)
print("Sens inverse   : ", end="")
afficher(liste, True)
```

Remplissage de la liste

Implanter une fonction `remplir(liste, tab)` qui remplit la liste avec les valeurs du tableau passé en argument, dans le même ordre. Cette fonction ne renvoie rien car les sentinelles de tête et de queue ne sont pas modifiées.

Affichage de la liste

Implanter une fonction `afficher(liste, inv=False)` qui affiche le contenu de la liste (i.e. les cellules contenant des valeurs significatives séparées par des <->). L'affichage se fera en partant de la fin ssi `inv == True` et en partant du début sinon.

Echange de deux éléments

Implanter une fonction `echanger(cell)` qui échange les cellules `cell` et `cell.suiv` dans la liste, et renvoie un pointeur vers l'ancienne cellule `cell.suiv`. **On procèdera par réorganisation des chainages, sans toucher aux valeurs** (i.e. vous n'avez pas le droit de simplement échanger les valeurs des deux cellules). On prendra comme hypothèse que la cellule `cell` n'est pas la dernière de la liste (i.e. il y a bien une cellule contenant une valeur significative après `cell`).

Tri de la liste

Implanter une fonction `trier(liste)` qui trie la liste par ordre croissant selon l'algorithme du nain de jardin, dont on rappelle le principe : un nain de jardin cherche à trier des pots de fleurs par taille croissante. Il regarde le pot devant lui : s'il est plus petit que le pot à sa droite, le nain avance d'un pas vers la droite (s'il n'est pas arrivé à la fin de la file de pots). Si le pot devant lui est plus grand que le pot à sa droite, le nain échange les deux pots, et recule d'un pas vers la gauche (s'il n'est pas revenu au début de la file de pots). On pourra prendre comme hypothèse que la liste n'est pas vide.