Soutien en algorithmique et programmation

Séance 6 : listes chainées simples

Introduction

On va travailler pendant cette séance sur des listes chainées simples.

Rappel : il existe deux opérateurs de comparaison : == et `is`, qui se comportent de façons différentes notamment lorsqu'on travaille avec des références. En effet, == compare le **contenu** des deux variables comparées, alors que `is` compare les **adresses** (références) des cases contenant les valeurs. Pour comparer des références, on utilisera donc toujours `is` et jamais ==.

On définit un type `Cellule` par la classe suivante à copier dans un fichier `cellule.py` :

```
class Cellule:
    """
      Une cellule est composee d'une valeur et d'un pointeur vers la
        cellule suivante (ou None s'il n'y a pas de suivant)
    """
    # pylint: disable=too-few-public-methods

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

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

On pourra tester les fonctions écrites grace au programme principal suivant, à copier-coller dans un fichier `listes.py` :

```
def main():
    """
      Fonction principale
    """
    liste = None # creation d'une liste vide
    afficher(liste)
    for _ in range(10):
        tete = randint(0, 1) == 1
        val = randint(0, 5)
        if tete:
            print(f"Insertion en tete de {val} : ", end="")
            liste = inserer_tete(liste, val)
        else:
            print(f"Insertion en queue de {val} : ", end="")
            liste = inserer_queue(liste, val)
```

```
        afficher(liste)
        val = randint(0, 5)
        idx = rechercher(liste, val)
        if idx == -1:
            print(f"{val} est absente de la liste")
        else:
            print(f"Position de {val} : {idx}")
    afficher(liste)
    while not est_vide(liste):
        fictif = randint(0, 1) == 1
        val = randint(0, 5)
        print(f"Suppression de {val} : ", end="")
        if fictif:
            liste, supp = supprimer_fictif(liste, val)
        else:
            liste, supp = supprimer(liste, val)
        if supp:
            afficher(liste)
        else:
            print("valeur absente")
```

Les listes que l'on va utiliser dans cette séance sont les plus simples possibles : en pratique, une liste est tout simplement une référence vers la première cellule (ou `None` si la liste est vide). Ce modèle est plus simple que celui utilisé en TP : pas de référence sur la cellule de queue ni de compteur du nombre d'éléments.

Test de la liste vide

Implanter une fonction `est_vide(liste)` qui renvoie vrai ssi la liste est vide.

Insertion en tête

Implanter une fonction `inserer_tete(liste, val)` qui insère la valeur `val` en tête de la liste et renvoie la liste modifiée en résultat.

Insertion en queue

Implanter une fonction `inserer_queue(liste, val)` qui insère la valeur `val` en queue de la liste et renvoie la liste modifiée. On rappelle que dans cette variante des listes chainées, **on ne dispose pas d'une référence sur la cellule de queue**.

Affichage

Implanter une fonction `afficher(liste)` qui affiche le contenu de la liste sous le format de l'exemple suivant : `1 -> 3 -> 2 -> 7 -> FIN`.

Recherche d'une valeur

Implanter une fonction `rechercher(liste, val)` qui renvoie un entier correspondant à :

- la position dans la liste de la première cellule contenant la valeur `val` : par exemple, si on recherche la première occurrence de la valeur `5` dans la liste `2 -> 4 -> 5 -> 1 -> 5 -> 7 -> FIN`, la fonction renverra 3 ;

- -1 si la valeur `val` n'est pas présente dans la liste.

Suppression d'une valeur

Implanter une fonction `supprimer(liste, val)` qui supprime la première occurrence de la valeur `val` dans la liste (ou ne fait rien si la liste ne contient pas cette valeur). Si vous connaissez déjà la technique de l'élément fictif en tête, vous ne l'utiliserez pas dans cette question. La fonction renvoie la liste éventuellement modifiée et un booléen indiquant si on a supprimé une cellule ou pas.

Utilisation d'un element fictif en tête

Implanter une fonction `supprimer_fictif(liste, val)` équivalente à celle de la question précédente, mais en utilisant un élément fictif en tête de liste. Cette technique consiste à ajouter une nouvelle cellule (sans valeur pertinente) au début de la liste avant de commencer le parcours, afin de simplifier les cas particuliers.