Soutien en algorithmique et programmation

Séance 7 : réorganisation de listes chainées

Introduction

On continue à travailler sur des listes chainées simples, avec les mêmes types qu'à la séance précédente. On complètera donc le fichier `liste.py` de la dernière fois avec des nouvelles fonctions.

On pourra tester les fonctions écrites grace au programme principal suivant :

```
def init_liste(vals=None):
    """
      Initialise une liste pour tester
    """
    liste = None
    if vals is None: # on insere 10 chiffres aleatoires
        for _ in range(10):
            liste = inserer_tete(liste, randint(0, 9))
    else: # on insere les valeurs passees en argument en ordre inverse
        for val in vals:
            liste = inserer_tete(liste, val)
    return liste

def main():
    """
      Fonction principale
    """
    print("Liste initiale  : ", end="")
    liste = init_liste((2, 2, 4, 6, 6, 8))
    afficher(liste)
    print("Liste inversee  : ", end="")
    liste = inverser(liste)
    afficher(liste)
    print("Insertion triee : ", end="")
    for val in (1, 3, 3, 5, 7, 9, 9):
        liste = inserer_triee(liste, val)
    afficher(liste)
    print("Liste aleatoire : ", end="")
    liste = init_liste()
    afficher(liste)
    print("Tri (maximum)   : ", end="")
    liste = trier_max(liste)
    afficher(liste)
    print("Liste aleatoire : ", end="")
    liste = init_liste()
    afficher(liste)
    print("Tri (insertion) : ", end="")
    liste = trier_ins(liste)
    afficher(liste)
```

Inversion d'une liste

Implanter une fonction `inverser(liste)` qui inverse les éléments de la liste. Par exemple, si la liste initiale est `1 -> 2 -> 3 -> 4 -> FIN`, la liste inversée sera `4 -> 3 -> 2 -> 1 -> FIN`. Vous devez implanter cette fonction **sans aucune allocation de cellule** (ce qui interdit notamment d'utiliser la fonction d'insertion en tête). La fonction renvoie la liste inversée en résultat.

Insertion à sa place

Implanter une fonction `inserer_triee(liste, val)` qui insère la valeur à sa place dans une liste supposée triée par ordre croissant. Par exemple, si la liste initiale est `1 -> 2 -> 4 -> 5 -> FIN` et qu'on appelle `inserer_triee(liste, 3)`, la liste sera `1 -> 2 -> 3 -> 4 -> 5 - > FIN` à la fin de la fonction. La fonction renvoie la liste complétée en résultat.

Tri par sélection du maximum

Implanter une fonction `trier_max(liste)` qui trie la liste par ordre croissant en utilisant l'algorithme du tri par sélection du maximum. Le principe de cet algorithme est le suivant :

- on parcours la liste initiale pour rechercher l'élément maximum ;
- on insère la cellule correspondante en tête de la liste résultat et on la retire de la liste initiale ;
- le tri est terminé lorsque la liste initiale est vide.

Là encore, on ne doit pas allouer de nouvelles cellules, à part un élément fictif si nécessaire. La fonction renvoie la liste triée en résultat.

Tri par insertion

Implanter une fonction `trier_ins(liste)` qui trie la liste par ordre décroissant en utilisant l'algorithme du tri par insertion. Le principe de cet algorithme est le suivant :

- on détache l'élément de tête de la liste initiale et on parcours la liste résultat pour trouver l'endroit où insérer cet élément ;
- on insère l'élément à sa place dans la liste résultat ;
- le tri est terminé lorsque la liste initiale est vide.

La fonction renvoie la liste triée en résultat.

Note : on veut trier la liste par ordre décroissant, donc il n'est pas intéressant d'utiliser la fonction `inserer_triée` ici, mais on peut bien sûr s'en inspirer fortement.