université
PARIS-SACLAY
DIU EIL
Bloc 4

Listes chaînées

Dans tous l'énoncé le mot « liste » désigne une liste chaînée.

1 Structure de données

Une liste chaînée est faites de *cellules*. Ces dernières peuvent être de deux types :
— la cellule vide, qui permet de représenter la fin d'une liste (et en particulier la liste vide) ;
— une cellule intermédiaire, contenant une valeur et une référence vers la cellule suivante.

On choisit de représenter la cellule vide par `None` et une celulle intermédiaire par un objet `Cellule` possédant deux attributs :

`valeur` contenant la valeur stockée dans la cellule

`suivante` contenant la cellule suivante

1. Définir une fonction `longueur` qui renvoie le nombre d'éléments de la liste (donner deux définitions possibles, l'une récursive, l'autre impérative).
2. Écrire une fonction `nieme_element (l, n)` qui renvoie le $n^{ème}$ élément d'une liste. La encore, la fonction peut être écrite récursivement et impérativement. Gérer correctement les cas d'erreur en levant une exception.
3. Définir une fonction `chaine (l)` qui convertit la liste `l` en chaîne de caractères. La liste est délimitée par des crochets et les éléments séparés par des virgules.
   Note : cette fonction peut servir dans la suite pour débugger les listes créées avec `print(chaine(l))`.

2 Opérations avancées

1. Écrire une fonction récursive `concatener(l1, l2)` qui renvoie la concaténation de `l1` et `l2`. Quelle partie de la liste résultante est partagée en mémoire avec ses arguments ?
2. Écrire une fonction itérative `concatener_mod(l1, l2)` qui effectue la concaténation « en place » de `l1` avec `l2`. Si `l1` est vide, la fonction renvoie `l2`. Sinon l1 est parcourue et sa dernière cellule est modifiée pour pointer vers `l2`. Dans tous les cas, la liste résultante est envoyée comme résultat de la fonction.
   Proposer quelques exemples permettant de montrer la différence entre `concatener` et `concatener_mod`.
   Que se passe t'il si on exécute `concatener_mod(l, l)` pour une liste `l` non vide ? (deviner sans tester). Ne pas tenter d'afficher le résultat avec **`print`**`(chaine(concatener_mod(l,l)))`, ça peut planter la machine.
3. Quelle est la compléxité de la fonction `concatener(l1, l2)` ?
4. (difficile à faire proprement) Écrire une fonction `concatener_i(l1, l2)`, itérative, qui renvoie la concaténation de `l1` et `l2` comme une nouvelle liste.

3 Algorithmes sur les listes

1. Écrire une fonction `renverser(l)` qui renvoie une liste contenant les éléments de `l` dans l'ordre inverse. On aura tout intérêt à écrire directement une version itérative de cette fonction.
2. Considérons la définition récursive suivante :

```
def renv_r (l):
    if l is None:
      return None
    else:
      return concatener(renv_r(l.suivante), Cellule(l.valeur, None))
```

Que calcule cette fonction et quelle est sa complexité ?

3. Écrire une fonction `est_trie (l)` qui vérifie que les éléments de la liste `l` sont triés (selon l'opérateur Python <=).

4 Interface objet

On souhaite encapsuler les fonctions précédentes dans une classe `Liste` Cette dernière contient une référence vers une `Cellule` ou `None`.

1. Rajouter au code de la classe `Liste` les méthodes `longueur`, `est_vide` et `nieme_element`.
2. Ajouter les méthodes `ajoute` et `retire`. La méthode `retire` supprime le premier élément de la liste et renvoie cet élément. Comment gérer le cas de la liste vide ?
3. Ajouter la méthode `concat`.

On souhaite maintenant rendre la classe `Liste` compatible avec la bibliothèque standard Python.

4. Ajouter une méthode `__str__` à la classe `Liste`, qui renvoie le contenu de la liste sous forme d'une chaîne de caractères.
5. Faire de même avec les méthodes `__add__`, `__getitem__` et `__len__`.

5 Exercices avancés

Les exercices ci-dessous permettent d'aller un peu plus loin sur les listes chaînées.

1. Écrire une fonction `fusion(l1, l2, f)` qui fusionne deux listes supposées triées selon un ordre `f` donné en paramètre. Ici, `f(v1, v2)` renvoie `True` si et seulement si `v1` $\leq$ `v2`.
2. Écrire une fonction `separe(l)` qui partage `l` deux sous listes dont la taille diffère d'au plus un (la fonction sépare la liste en deux moitié, mais elle n'est pas obligée de préserver l'ordre des éléments).
3. Écrire une fonction `tri_fusion(l, f)` qui réalise l'algorithme de tri fusion sur la liste `l` en utilisant la fonction de comparaison `f`.
4. En Python, il est possible d'utiliser une boucle `for` sur n'importe quel objet « itérable ». Un objet est « itérable » s'il possède une méthode `__iter__()` renvoie un itérateur. Un itérateur est lui-même un objet possédant une unique méthode `__next__()`. Cette méthode renvoie la prochaine valeur à considerer dans l'itération ou lève l'exception `StopIteration` s'il n'y a plus d'objet. Modifier la classe `Liste` pour qu'elle soit itérable.