DICTIONNAIRES

TP 9

Structure de données - v1.0

Lycée La Martinière Monplaisir, 41 Rue Antoine Lumière, 69372 Lyon

1 Comptage des entiers d'une liste

On considère une liste `L` d'entiers naturels, par exemple : `L = [2, 5, 8, 2, 0, 2, 5, 4, 8, 2, 4, 4, 2]`.

Si l'on fait l'inventaire des éléments de cette liste, sans tenir compte de l'ordre dans lequel ils apparaissent, on peut écrire qu'il y a :

- 1 fois le nombre 0
- 0 fois le nombre 1
- 5 fois le nombre 2
- 0 fois le nombre 3
- 3 fois le nombre 4
- 2 fois le nombre 5
- 0 fois le nombre 6
- 0 fois le nombre 7
- 2 fois le nombre 8

Cette information peut être stockée sous la forme de la liste `M` suivante : `M = [1, 0, 5, 0, 3, 2, 0, 0, 2]` dans laquelle on trouve en position `i` le nombre de fois où l'élément `i` apparaît dans la liste initiale `L`.

Question 1 Quel lien y a-t-il entre la longueur de la liste `M` et la liste initiale `L` ?

Question 2 Écrire une fonction `inventaire(L)`, prenant en paramètre une liste `L` d'entiers naturels (`L` est supposée non vide), qui construit et renvoie la liste `M` faisant l'inventaire des éléments de `L` tel que décrit ci-dessus.

Question 3 Tester la fonction avec la liste donnée en exemple puis avec la liste [23, 2021, 5, 23].

Question 4 Commenter le second exemple.

L'utilisation d'un dictionnaire va permettre de palier l'inconvénient rencontré avec les listes.

Question 5 Écrire une fonction `inventaire_dict(L)`, prenant en paramètre une liste `L` d'entiers naturels (`L` est supposée non vide), qui construit et renvoie le dictionnaire dont les clés sont les différents éléments de `L` et les valeurs associées leur nombre d'occurrences.

Question 6 Tester la fonction. Vérifier en particulier que `inventaire_dict([23, 2021, 5, 23])` renvoie est bien le dictionnaire `{23: 2, 2021: 1, 5: 1}`.

2 Suppression des doublons

Question 1 Créer une fonction `ensemble(L)` qui renvoie une liste des éléments de la liste `L` sans doublon.

Dans la plupart des cas, le test `elt in L` correspond à une opération de coût n, la longueur de la liste `L`. Pour une raison liée à l'implémentation d'un dictionnaire que vous verrez en deuxième annnée, `elt in d` à un coût unitaire pour tester l'appartenance d'une clé dans un dictionnaire.

Question 2 Créer une fonction `ensembleDic(L)` qui renvoie une liste des éléments de la liste `L` sans doublon, mais utilisant un dictionnaire pour être plus efficace.

Question 3 Existe-t-il une restriction sur l'utilisation de cette deuxième version ? (Rappel : limitation sur les clés d'un dictionnaire).

3 Introduction à la mémoïsation sur la suite de Fibonnacci

La suite de Fibonacci $(F_n)_{n\in\mathbb{N}}$ est définie par $F_0 = 0$, $F_1 = 1$, et la relation de récurrence $F_n = F_{n-1} + F_{n-2}$.

Question 1 Écrire une fonction récursive `fibo(n)` la plus simple possible pour calculer le n^e terme de la suite de Fibonacci.

Vous pouvez constater que le calcul devient vite très lent (vous pouvez tester la valeur 34 par exemple) : cela est lié à ce que beaucoup de termes sont calculer plusieurs fois indépendamment. Pour améliorer cela, on va utiliser un dictionnaire qui va mémoriser les résultats.

Question 2 Réécrire une fonction `fiboMem(n)` pour calculer le n^e terme de la suite de Fibonacci. À l'appel initiale de cette fonction, on doit créer un dictionnaire de mémorisation : `d = {0: 0, 1: 1}` et définir une fonction interne récursive qui utilisera ce dictionnaire pour améliorer sa performance.

Pour cette question, on fera attention à :

- commencer par vérifier si le résultat est en mémoire ou pas ;
- ne pas renvoyer de résultat avant de l'avoir ajouté au dictionnaire.

Question 3 Constater qu'on a une amélioration importante de la performance temporelle avec cette nouvelle fonction.

4 Notation aux échecs

Aux échecs, les lignes sont codées par des chiffres de 1 à 8 et les colonnes sont codées par des lettres de a à h. La base b4 correspond donc à la quatrième ligne, deuxième colonne (toujours vu des blancs).

Question 1 Écrire une fonction `decodage(code)` qui renvoie les coordonnées `(i, j)` (numérotation à partir de 0) à partir de la chaine de caractères de longueur 2 correspondant au codage. Vous pouvez vous aider de la fonction `ord` qui renvoie le numéro Unicode correspondant à un caractère et de la fonction `chr` qui renvoie le caractère correspondant à un numéro Unicode.

Question 2 En déduire deux dictionnaires : un de décodage et un de codage.

5 Codage César

La cryptographie a pour but de cacher le contenu d'un message. Afin de le rendre incompréhensible, on brouille le message suivant un protocole mis au point préalablement par l'expéditeur et le destinataire. Ce dernier n'aura qu'à inverser le procédé pour rendre le message lisible, alors que l'ennemi, s'il ne connaît pas le protocole de brouillage, trouvera difficile, voire impossible, de rétablir le texte original.

Au premier siècle de notre ère est apparu un chiffrement par substitution connu sous le nom de code de César car l'empereur en a été l'un des plus assidus utilisateurs. Le chiffrement de César consiste à assigner à chaque lettre de l'alphabet une autre lettre, résultant du décalage de l'alphabet d'un certain nombre de lettres.

Par exemple, avec le décalage suivant :

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e

le texte "vous suivez le cours de python" devient "'atzx xznaje qj htzwx ij udymts".

Question 1 Mettre en place le codage et le décodage du code César en utilisant des dictionnaires par compréhension.