Informatique

2
Dictionnaires et programmation dynamique

TD2-1

Dictionnaires

Exercices 1 et 2 inspirés du travail de Bernard SALAMITO

Exercice 1: Inversion

Soit le dictionnaire défini de la manière suivante :

```
dico_1 = {i:i%3 for i in range(10)}
```

Dans cette première question, il n'y a qu'une seule valeur pour chaque clé.

Question 1: Créer la fonction dico_inverse_1(dico) prenant en argument un dictionnaire et renvoyant un nouveau dictionnaire contenant pour clés les valeurs de dico, et pour valeurs ses clés

Vérifier :

```
>>> dico_inverse_1(dico_1)
{0: [0, 3, 6, 9], 1: [1, 4, 7], 2: [2, 5, 8]}
```

On considère maintenant le dictionnaire possédant pour chaque clé une ou plusieurs valeurs stockées dans une liste :

```
dico_2 = {0:[1,2],1:[3,6],2:[1],3:[5,5]}
```

Question 2: Créer la fonction dico_inverse_2(dico) prenant en argument un dictionnaire et renvoyant un nouveau dictionnaire contenant pour clés les valeurs de dico, et pour valeurs ses clés

Vérifier :

```
>>> dico_inverse_2(dico_2)
{1: [0, 2], 2: [0], 3: [1], 6: [1], 5: [3, 3]}
```

Exercice 2: Comptage

Soit la liste L d'entiers suivante :

```
from random import randint as rd
n = 10
L = [rd(-100,100) for _ in range(n)]
```

Depuis la version 3.7 de Python, lorsque l'on crée un dictionnaire, l'ordre d'ajout des éléments « clé:valeur » dans un dictionnaire est conservé. On utilisera cette propriété dans la suite.

Toutes les fonctions de cet exercice devront avoir une complexité linéaire O(n).

Question 1: Créer une fonction minmax(L) renvoyant en un seul parcours, les min et max de L
Question 2: Créer la fonction occurrences_1(L) prenant la liste L en argument, et renvoyant un dictionnaire contenant toutes les clés (int) de min(L) à max(L) et pour valeurs, le nombre d'occurrences de chaque clé dans L
Question 3: Créer la fonction tri_1(L) renvoyant une nouvelle liste reconstruite à partir du dictionnaire renvoyé par occurrences_1 correspondant à L triée
Vérifiez votre fonction sur des exemples.

On remarquera que le dictionnaire est rempli de clés inutiles. On propose de réaliser une nouvelle version de ce tri en ne créant pas les clés du dictionnaire à l'avance. Elles ne seront donc plus triées.

Question 4: Créer la fonction occurrences_2(L) prenant la liste L en argument, et renvoyant un dictionnaire pour clé les entiers présents dans L, et pour valeurs, le nombre d'occurrences de chaque clé dans L
Question 5: Créer la fonction tri_2(L) renvoyant une nouvelle liste reconstruite à partir du dictionnaire renvoyé par occurrences_2 correspondant à L triée

Exercice 3: Chemin

Soit le dictionnaire suivant en lien ici :

```
Dico = {(0, 1): [0, 0], (5, 7): [6, 7], (6, 7): [6, 8], (5, 2): [5, 3],
(9, 1): [8, 1], (4, 7): [5, 7], (9, 2): [9, 1], (8, 6): [9, 6], (9, 5):
[9, 4], (7, 8): [8, 8], (5, 4): [4, 4], (9, 3): [9, 2], (3, 9): [4, 9],
(9, 6): [9, 5], (4, 4): [3, 4], (9, 4): [9, 3], (8, 8): [8, 7], (1, 2):
[0, 2], (0, 0): 'RAS', (2, 2): [1, 2], (5, 1): [5, 2], (5, 3): [5, 4],
(4, 9): [4, 8], (3, 2): [2, 2], (6, 8): [7, 8], (6, 1): [5, 1], (8, 7):
[8, 6], (3, 4): [3, 3], (7, 1): [6, 1], (8, 1): [7, 1], (3, 3): [3, 2],
(0, 2): [0, 1], (4, 8): [4, 7]}
```

Il a été obtenu lors du parcours d'un labyrinthe carré de 100 cases (10x10) par succession de mouvements verticaux et horizontaux. Chaque case est représentée par la liste [L,C] ou son tuple (L,C) de sa ligne et colonne. Pour chaque case clé (tuple), on a stocké pour valeur la case de provenance (liste). Pour la première case, on a stocké la valeur `'RAS'`.
La case d'arrivée est la case L,C = 3,9.

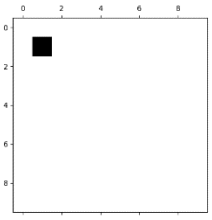
Remarque : J'ai mélangé les tuples et les listes pour que vous ayez conscience que votre habitude d'utiliser les listes risque de vous poser des problèmes avec les parcours de graphes. En effet, les clés ne peuvent être des listes, et cet exercice vous invite donc à jongler entre les différents types 😉

Rappel : pour transformer une liste en tuple, écrire :
```
>>> tuple([1,2])
(1, 2)
```

Question 1: Proposer un code affichant dans la console la liste des cases dans leur ordre de parcours

On souhaite afficher le trajet effectué. On utilisera un array (numpy) pour représenter le plateau. On peut simplement afficher les valeurs non nulles d'un array sous forme d'image en utilisant les commandes suivantes :

```
import numpy as np
import matplotlib.pyplot as plt
Plateau = np.zeros([10,10])
Plateau[1,1] = 1
plt.spy(Plateau)
plt.show()
```



Question 2: Afficher le trajet réalisé

Exercices 1 et 2 inspirés du travail de Bernard SALAMITO