TD Informatique TSI2

Dictionary

EXERCICE 01

Considérons un dictionnaire `d1` et le code suivant :

```
In [1]: d2 = {}
        for c,v in d1.items():
            d2[v] = c
```

Le dictionnaire `d2` a-t-il la même longueur que le dictionnaire `d1` ?

Indication. *On pourra prendre par exemple pour commencer pour* ***`d1`*** *:*

```
In [1]: d1 = {'true': 'vrai', 'and': 'et', 'or ': 'ou'}
```

Puis pour ***`d1`*** *:*

```
In [2]: d1 = {'true' : 'vrai', 'and' : 'vrai', 'or' : 'ou' }
```

EXERCICE 02

Écrire une fonction `min_max` qui prend en paramètre une liste `nombres` de nombres non vide et renvoie un dictionnaire dont les clès sont les chaînes `"min"` et `"max"` avec pour valeurs respectives le minimum et le maximum des nombres de la liste.

Par exemple `min_max([8,5,9,3,1,7])` vaut `{"min": 1, "max": 9}`

Indication. *On initialisera le dictionnaire* ***`d`*** *avec :*
`d = {"min": nombres[0], "max": nombres[0] }` *et dans la boucle* ***`for n in nombres:`*** *on comparera* ***`n`*** *avec* ***`d["max"]`*** *et avec* ***`d["min"]`***

EXERCICE 03

1. Importer le module `sys` et taper `sys.hash_info`. Remarquer que si `width` vaut 64 (système 64 bits) alors `modulus` a pour valeur $2^{61} - 1$.
2. Créer une constante `M` de valeur `modulus`.
   Écrire une fonction `hachage` qui prend en paramètres deux entiers `p` et `q` avec `q` non nul, représentant un flottant $f = p/q$ et qui renvoie `int(abs(p) * M / abs(q)) % M` si f est positif et l'opposé de cette expression si f est négatif. Si f vaut -1, la fonction renvoie -2.
3. Comparer `hachage(3,2)` avec `hash(3/2)` et `hash(1.5)`. Puis comparer `hachage(115,100)` et `hash(1.15)` et enfin comparer `hachage(-7,3)`, `hachage(7,-3)` et `hash(-7/3)`.

T.S.V.P →

EXERCICE 04

Soit des polynômes à coefficients entiers de degré quelconque mais qui ne contiennent pas plus de cinq monômes. On utilise un tableau de longueur $16 = 8 \times 2$ pour stocker les couples (degré, coefficient) dans lequel on pourra stocker au maximum huit couples. Les places non occupées contiennent la valeur -1. La fonction de hachage `h` est la fonction identité : pour tout $n \in \mathbb{N}$, $h(n) = n$. Si n est le degré, $n \,\%\, 8$ donne un indice et à cet indice, on écrit le degré (la clé), suivi du coefficient, (la valeur).

Par exemple, $8 + 3x^{10} - 5x^{12}$ est stocké dans :

indice 0	0	8
indice 1	-1	-1
indice 2	10	3
indice 3	-1	-1
indice 4	12	-5

1. Donner le tableau correspondant au stockage du polynôme $2x^5 - 3x^{34} + 4x^{105}$.
2. Quel est le problème par exemple avec le polynôme $8 - 5x^2 + 3x^{10}$?
3. En cas de collision, on décide d'utiliser la première place libre suivante. Les monômes sont rentrés dans le tableau suivant l'ordre de lecture. Donner un exemple de polynôme de degré minimum qui génère une collision pour chaque monôme excepté le premier.
4. On envisage une autre possibilité de stockage avec deux tableaux, un tableau pour les couples (degré, coefficient) et un tableau pour les indices, les deux tableaux ayant pour capacité 8.
   Avec le polynôme $4x^3 - 2x^5 + 4x^9$, on obtient les deux tableaux de la manière suivante :

   dans le premier, on écrit chaque degré avec le coefficient correspondant suivant l'ordre des degrés et on complète le tableau avec des 0 ;

   dans le second, on calcule $d \,\%\, 8$, où d est un degré et on place à l'indice trouvé l'indice où on trouve le couple (degré, coefficient) dans le premier tableau. On complète avec des -1.

   Extraits des tableaux :

indice 0	3	4
indice 1	5	-2
indice 2	9	4
indice 3	0	0
...	...	...

indice 0	-1
indice 1	2
indice 2	-1
indice 3	0
indice 4	-1
indice 5	1

   Donner les deux tableaux correspondant au stockage du polynôme $3x^5 - x^{18} + 7x^{20}$.

EXERCICE 05

On utilise un dictionnaire pour comparer des listes de nombres qui sont tous du même type, soit du type `int` soit du type `float`.

1. On rappelle que si `d` est un dictionnaire, `d[n]` renvoie la valeur dont `n` est la clé et alors dans `d`, on a rajouté la structure `n : d[n]` qui correspond au couple `(n, d[n])`
   On considère ici la fonction `occurrences` suivante :

```
In [1]: def occurrences(nombres):
            d = { }
            for n in nombres:
                if n in d :
                    d[n] = d[n] + 1
                else :
                    d[n] = 1
            return d
```

Appliquer la procédure `occurrences` d'abord à la liste `[1,5,5,2,0,-5,-4,-5,-5,7])` puis ensuite à la liste `[3,5,-2,3,3,-2,3]`. Que renvoie cette fonction ?

2. Écrire une fonction `taille` qui prend en paramètre un dictionnaire obtenu comme ci-dessus et renvoie la longueur de la liste qui a été considérée.

   Indication. *On crée une variable locale `t` initialement égale à `0` et on se lance dans la boucle `n in d :` et on ajoute à `t` les valeurs `d[n]`*

3. Écrire une fonction `compare` qui prend en paramètres deux listes de nombres de même longueur et renvoie `True` si les deux listes contiennent les mêmes nombres, pas nécessairement dans le même ordre et `False` sinon. Utiliser la fonction `occurrences`.
   Quelle est la complexité en temps de cette fonction ?

   Indication. *Dans `d1` on rentre `occurrences(nombres1)` et dans `d2` on rentre `occurrences(nombres2)` puis on balaye les éléments de `d1` avec la boucle `for n in d1:` et on regarde si `n in d2`, si ce n'est pas le cas, c'est `False` et si c'est le cas, on compare `d1[n]` et `d2[n]`, après à vous de continuer.*

EXERCICE 06

On considère un polynôme écrit suivant les puissances croissantes, par exemple $P(x) = 3 - 7x + 5x^3$.

1. On représente un polynôme par la liste de ses coefficients suivant les puissances croissantes. Si $P(x) = 3 - 7x + 5x^3$ alors la liste est `[3,-7,0,5]` et ainsi le coefficient de x^n est placé à l'indice `n`
   Un polynôme constant $P(x) = K$ est représenté par `[K]`, le polynôme nul par `[0]`

   (a) Écrire une fonction `derive1` qui prend en paramètre une liste représentant un polynôme comme ci-dessus et renvoie la liste représentant le polynôme dérivé.
   Par exemple, la dérivée de $2 + 3x^2 + 5x^3$ est $6x + 15x^2$ donc `derive1([2,0,3,5])` a pour valeur `[0,6,15]`

   Indication. *Si `len(p)` vaut `1`, on renvoie `[0]` Puis on crée une variable locale `dp` qu'on initialise à la liste vide puis on parcourt la liste `p` et on nourrit `dp` avec les termes qu'il faut.*

   (b) Quelle est la complexité de la fonction `derive1` exprimée en fonction du degré n du polynôme ?

2. On représente ici un polynôme par un dictionnaire dont les clès sont les degrés des mônomes non nuls et les valeurs associées les coefficients correspondants. Si $P(x) = 3 - 7x + 5x^3$, le dictionnaire est `{0: 3, 1: -7, 3: 5 }` et on remarque que seuls les monômes non nuls sont mémorisés excepté pour le polynôme nul représenté par `{0: 0}`

   (a) Écrire une fonction `derive2` qui prend en paramètre un dictionnaire représentant un polynôme comme ci-dessus et renvoie le dictionnaire représentant le polynôme dérivé. Par exemple `derive2({0: 2, 2 : 3, 3 : 5 })` vaut ($\{1 : 6, 2 : 15\ \}$)

   Indication. *Au départ, si `len(p)` vaut `1` et `0 in p` alors on renvoie `{0: 0}`* puis ensuite on crée une variable locale `dp`, on l'initialise avec `{ }` et on parcourt les clès `deg` du dictionnaire `p` et si ces clés sont différentes de 0 alors dans `dp[deg-1]` on rentre `p[deg] * deg`

   (b) Quelle est la complexité de la fonction `derive2` exprimée en fonction du degré n du polynôme ?