TP 13 : dictionnaires

PTSI 2025/2026

Exercice 1 : Niveau :

Écrire une fonction `reverse` qui prend en entrée un dictionnaire formée de clefs et de valeurs et qui renvoie le dictionnaire dont les clefs sont les valeurs du premier dictionnaire et les valeurs la liste des clefs correspondantes dans le premier dictionnaire. Exemple :

```
{ 2:'a', 3:'a', 7:'x'}
```

devient :

```
{'a':[2,3] , 'x':[7]}
```

Exercice 2 : *"Mémoisation"* Niveau : 2

1. a. Écrire une fonction Python récursive `suite` qui prend en entrée les paramètres `a,b,u0,u1,n` et qui calcule le terme de rang n de la suite récurrente $u_{n+2} = au_{n+1} + bu_n$ à partir des valeurs initiales u_0, u_1.
   b. Rappeler pourquoi cette fonction n'est pas efficace.
2. Écrire une fonction récursive `suite_memo` effectuant la même tache mais en utilisant un dictionnaire pour stocker les résultats déjà calculés et éviter leur recalcul. Consignes :
   - La fonction prend en entrée les coefficients `a,b`
   - les valeurs initiales sous forme d'un dictionnaire : `dico = {0:u0, 1:u1}`
   - l'indice `n` du terme qu'on veut calculer.
3. Comparer les temps d'exécution.

Exercice 3 : *Taxicab le retour* Niveau :

Utiliser la structure de dictionnaire pour programmer une fonction `taxicab(n)` prenant en entrée un entier naturel n et renvoyant le premier entier naturel qui s'écrit de `n` manières comme une somme de deux cubes d'entiers naturels non nuls.

(les couples cherchés seront de la forme (a, b) avec $1 \leqslant b \leqslant a$).

Exercice 4 : *Matrices parcimonieuses* Niveau : 2

On s'intéresse ici aux matrices parcimonieuses, c'est-à-dire dont la plupart des coefficients sont nuls. Une telle matrice M de dimensions (n, p) pourra être codée par un dictionnaire ayant pour couples clefs/valeurs :

- `'dim'` : (n, p), qui donne les dimensions de la matrice ;
- (i, j) : M_{ij}, qui donne pour chaque couple (i, j), la valeur de l'élément $M_{ij} \neq 0$ de la matrice.

1. Donner le dictionnaire qui code la matrice :

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 \end{pmatrix}.$$

2. Proposer une fonction d'addition `somme_mat(M1:dict, M2:dict) -> dict` de deux matrices parcimonieuses.
3. Si la première matrice contient c coefficients non nuls, et la seconde c', quelle est la complexité temporelle de cet algorithme ?