TP2
Dictionnaires

1 Dictionnaires

Exercice 1.1. *Le Scrabble est un jeu de lettres où l'on forme des mots et l'on marque des points en faisant la somme des valeurs de chaque lettre composant le mot.*
Dans la version française, les points sont les suivants :

- *1 point pour A, E, I, L, N, O, R, S, T et U ;*
- *2 points pour D, G et M ;*
- *3 points pour B, C et P ;*
- *4 points pour F, H et V ;*
- *8 points pour J et Q ;*
- *10 points pour K, W, X, Y, Z.*

1. *Donner la commande permettant de créer un dictionnaire dont les clés sont les lettres (type* `str`*) et les valeurs sont les points correspondant suivant le barème ci-dessus.*
2. *Écrire une fonction* ***score(mot)*** *qui prend en argument une chaîne de caractères* `mot` *(en majuscules sans accent) et qui renvoie la valeur de ce mot.*
   On utilisera bien sûr le dictionnaire défini dans la question précédente.
3. *On souhaite créer une fonction de hachage pour un mot, fondée sur le même barème que le Scrabble. On définit la valeur de hachage d'un mot par :*

$$h(mot) = \left(\sum_{lettre \in mot} points[lettre] \right) \times len(mot)$$

   Autrement dit, on multiplie le score Scrabble du mot par sa longueur. Écrire une fonction ***hachage(mot)*** *qui renvoie cette valeur de hachage.*
4. *On désire maintenant stocker des mots dans une table de hachage de taille $T = 20$.*

   On utilise la fonction de hachage :

$$h(mot) = (score(mot) \times len(mot)) \bmod T$$

(a) *Écrire une fonction* ***hachage_index(mot, T=20)*** *qui renvoie l'indice de hachage d'un mot.*

(b) *Pour gérer les collisions, on utilisera la méthode du chaînage : chaque case de la table contient une liste de mots ayant le même indice.*
Écrire une fonction ***inserer(table, mot)*** *qui insère un mot dans la table de hachage.*

(c) *Donner un exemple d'utilisation : création d'une table de taille 20 puis insertion des mots* `PYTHON`, `LION`, `OURS` *et* `JEU`.

Exercice 1.2. *Le tiroir des chaussettes est sens dessus dessous : elles ne sont pas appariées et il en manque peut-être certaines pour faire des paires. On représente les chaussettes dans une liste de chaînes de caractères représentant leur couleur, par exemple :*

```
tiroir = ['blanche', 'noire', 'noire', 'bleue', 'noire', 'blanche']
```

Écrire une fonction ***paires(tiroir)*** *qui renvoie le nombre de paires de chaussettes de même couleur que l'on peut former.*

Par exemple, avec la composition du tiroir donnée précédemment, le résultat sera 2.
En effet : une paire noire et une paire blanche, les deux restantes ne s'apparient pas.

2 Fonction de hachage

Exercice 2.1. *Des chaines vers les entiers*
On propose ici un exemple simple de pré-traitement des chaînes de caractères avant hachage.
Nous allons nous servir du codage ASCII qui est donné sous Python par la fonction ***ord(c)*** *qui renvoie le code ASCII sur 8 bits du caractère c.*
Inversement, l'instruction ***chr(code)*** *renvoie le caractère associé à un code ASCII.*

1. *Écrire une fonction* ***chaineversentier(ch)*** *qui prend une chaîne de caractères en argument et renvoie la valeur :*

$$\sum_{k=0}^{n-1} \text{ord}(ch[k])256^k.$$

si n désigne la longueur de la chaîne de caractères.

Cette fonction renvoie des entiers de grande taille, par exemple :

chaineentiers ('exemple de taille modérée')

renvoie :

639711425003567419095171883946955047862310670182554716698725.

2. *On souhaite réserver une table de hachage de longueur m, la fonction de hachage étant la fonction f précédente.*
   Écrire une fonction ***chaineentiersmodulo(ch,m)*** *qui prend une chaîne de caractères ainsi qu'un entier m en arguments et renvoie la valeur $g : ch \longrightarrow f(p)\%m$ (reste de la division euclidienne).*
3. (a) *Générer une liste de* 1000 *chaînes de caractères de longueur* 5 *de façon aléatoire.*
   (b) *Pour $m = 12$, vérifier sur les* 1000 *chaînes générées si la distribution de la fonction de hachage* ***chaineentiersmodulo*** *est uniforme.*

Exercice 2.2. ***Table de hachage et gestion des collisions***

On considère une table de hachage de taille $T = 17$, représentée en Python par une liste de 17 cases initialisées à `None`.
*Pour insérer des éléments, on utilisera l'****adressage ouvert*** *avec* ***sondage linéaire*** *: en cas de collision, si une case est occupée, on essaie la suivante, puis encore la suivante, en revenant au début si nécessaire.*

1. *On définit la fonction de hachage suivante pour une chaîne de caractères* `mot` *composée de lettres majuscules :*

$$h(mot) = \left(\sum_{k=0}^{n-1} \left(ord(mot[k]) - 64 \right) \right) \bmod T$$

   où n est la longueur du mot.
   (a) *Calculer la valeur de hachage des mots* `"CHAT"`, `"LION"` *et* `"TIGRE"`.
   (b) *Écrire une fonction Python* `hachage(mot, T)` *correspondant à cette définition.*
2. *On souhaite insérer un mot dans la table en utilisant le sondage linéaire :*

$$i,\ i+1,\ i+2,\ \ldots \pmod T$$

   tant qu'une case est occupée.
   (a) *Écrire une fonction* `inserer(table, mot)` *qui effectue cette insertion.*
   (b) *Montrer que, dans le pire des cas, le nombre de comparaisons est $O(T)$.*
3. *On souhaite maintenant insérer successivement les mots :*

`LION`, `OURS`, `TIGRE`, `LOUP`, `LYNX`

   (a) *Donner, à chaque étape, l'état de la table (indices et mots stockés).*
   (b) *Indiquer le nombre total de collisions rencontrées.*
4. *Discuter brièvement les avantages et inconvénients du sondage linéaire par rapport au chaînage.*

Exercice 2.3. *Multiplication*
Une méthode consiste à construire des fonctions de hachage de la façon suivante :

- *on se donne un réel (ou un flottant)* $\theta \in]0,1[$ *et un entier* $m > 0$.
- *on définit* $h : e \in \mathbb{N}^* \longrightarrow \lfloor m \times ((e \times \theta) \bmod 1) \rfloor$ *(où* $x \bmod 1 = x - \lfloor x \rfloor$*).*

On se propose d'en explorer quelques propriétés.

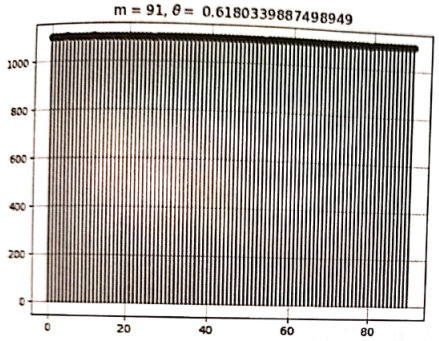
1. *Quelle sera la taille d'une table de hachage associée à une fonction multiplicative de cette forme ?*
2. *On décide de tester la répartition des clés dans la table en fonction de* θ.
   (a) *Définir une fonction* ***hachage (m, theta, c)*** *qui prend en arguments* m *un entier strictement positif,* $0 <$ *theta* < 1 *un flottant,* c *entier et qui réalise le hachage de* c.
   (b) *Écrire une fonction* ***testhachage(m, theta)*** *qui renvoie pour chaque entier* i *compris entre 0 et* $m-1$ *le nombre de valeurs* $c \in [\![0, 10^5]\!]$ *telles que* $h(c) = i$*, soit :*

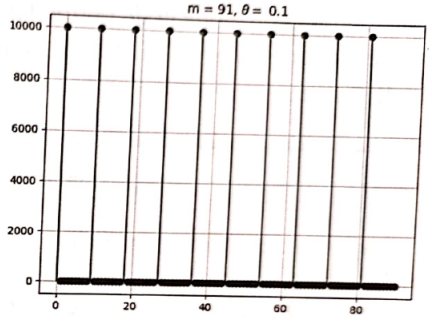
$$d_i = card(\{c \in [\![0, 10^5]\!] / h(m, \theta, c) = i\}).$$

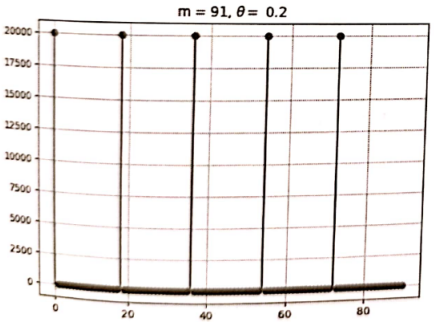
   Ces valeurs seront stockées dans un dictionnaire que renverra la fonction.
   (c) *Les quatre histogrammes ci-dessous représentent les répartitions obtenues (i en abscisse, et* d_i *en ordonnée) avec :*

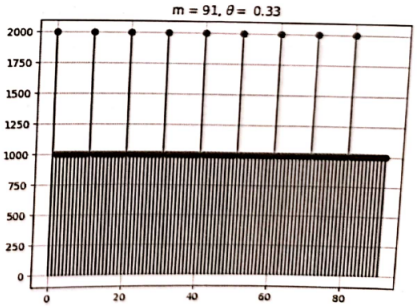
$$m = 91 \text{ et } \theta \in \left\{ \frac{\sqrt{5}-1}{2}, 0.1, 0.2, 0.33 \right\}.$$

   Expliquer les résultats lorsque $\theta = 0.1, 0.2$.









(d) Justifier que $c \longrightarrow h(m, a/b, c)$ *prend au plus* $\min(m, b)$ *valeurs où* a *et* b *sont deux entiers tels que* $a < b$.
Est-ce cohérent avec ce que l'on observe ?

Conseil : *Il pourra être intéressant d'observer la répartition des collisions lorsque* $\theta = 0.33$ *et pour différentes valeurs de* m *(plus petites, plus grandes que 100).*