# TP 8 – Tables, tables de hachage

Année 2019-2020

Ce TP permet de créer des structures de données de tables d'association ou tables de hachage et d'écrire des algorithmes dans le cadre d'applications qui les utilisent. Dans une première partie vous implémenterez vous-même ces tables à partir de listes chaînées simples. Dans une seconde partie vous exploiterez la structure de données *dictionnaires* en *python*.

## I. Première partie – Implémentation des tables d'association

### Exercice 1 – Implémentation d'une table à l'aide d'une liste chaînée

Cet exercice a été partiellement traité en cours.

1. Écrivez une classe `Table` qui implémente le TAD table d'association présenté en cours en utilisant une liste chaînée. Vous pouvez utiliser une liste chaînée que vous avez implémentée lors d'un précédent TP (fichier `liste.py` téléchargeable sur Moodle). Chaque cellule de la liste chaînée sera définie par deux champs : `data` (données) et `suivant` (lien vers la cellule suivante). Les données `data` seront représentées par une classe `TableCell` comprenant deux attributs `key` et `value` définissant chaque élément d'une table d'association.

2. Ecrivez les méthodes `print_keys`, `print_values` et `print_table` qui permettent d'afficher respectivement la liste des clés, des valeurs, et la liste des éléments de la table d'association.

3. Écrivez la méthode `get_elt(self, key)` qui retourne l'élément de clé `key`.

4. Écrivez la méthode `remove(self, key)` qui retire un élément de clé `key` de la table. Cette méthode retourne `None` si l'élément n'est pas présent dans la table.

### Exercice 2 – Utilisation d'une table d'association

1. Écrivez une classe `Person` qui définit une personne représentée par son nom, son prénom et son n° de téléphone.

2. En vous appuyant sur l'implémentation de la `Table` effectuée à l'exercice 1, écrivez une application de répertoire téléphonique : un répertoire est une table de personnes. La clé permettant d'accéder à un élément de type `Person` est une chaîne de caractères constituée du nom et du prénom de la personne. Testez les différentes méthodes sur ce répertoire.

## II. Deuxième partie – Dictionnaires en python, tables de hachage

Un mini-cours sur les dictionnaires en python est disponible sur Moodle (CM8-Dictionaries).
**Rappels** – En python, les tables de hachage sont appelées `dictionnaires` (dictionary en

anglais). Dans un dictionnaire, on associe une valeur à une clé. Les clés peuvent être de presque n'importe quel type : entiers, chaînes de caractères, fonctions, éléments d'une classe, etc. (mais pas une liste python).

- **Création d'un dictionnaire** : On peut créer un dictionnaire de plusieurs manières :
  - d = {'a' : 12, "leo" : "toulouse", 42 : [1,2,3]}
  - d = {}

  d['a'] = 12
  d["leo"] = "toulouse"
  d[42] = [1,2,3]

  - l = [('a' ,12), ("leo", "toulouse"), (42, [1,2,3])]

  d = dict(l)

- **Accession aux éléments du dictionnaire** Pour accéder à la valeur d'un dictionnaire `d` correspondant à la clé `k` on utilise la syntaxe `d[k]`. Par exemple, si `d` est le dictionnaire défini précédemment, `d["leo"]` vaut la chaîne de caractères "toulouse".
- **Accession aux clés du dictionnaire** Pour obtenir la liste de toutes les clés du dictionnaire, on utilise la syntaxe `d.keys()` qui renvoie une liste python contenant les clés.
- **Accession aux valeurs du dictionnaire** Pour obtenir la liste de toutes les valeurs du dictionnaire, on utilise la syntaxe `d.values()` qui renvoie une liste python contenant les valeurs.

## Exercice 3 – Exercices simples sur les dictionnaires

1. Choisissez 5 mots de la langue française et créez un dictionnaire qui associe à chacun de ces mots sa traduction en anglais.
2. Ajoutez une entrée au dictionnaire de la question précédente (un nouveau mot et sa traduction).
3. Écrivez une fonction `ajoute(mot1, mot2, d)` qui prend en argument un mot en français, sa traduction en anglais et ajoute ces deux mots dans le dictionnaire `d` uniquement si `mot1` n'est pas une clé du dictionnaire. Si `mot1` apparaît dans `d` la fonction modifie la valeur du mot s'il est différent de `mot2`, sinon elle ne fait rien.
4. Écrivez une fonction qui affiche à l'écran toutes les valeurs correspondant aux clés qui sont dans votre dictionnaire (ici, tous les mots en anglais qui apparaissent dans votre dictionnaire).
   **Indication** : on exécute une boucle `for` sur tous les éléments de `d.keys()` et l'on renvoie pour chacun la valeur qui lui est associée ; écrivez également la fonction avec `d.values()` qui renvoie une liste contenant les valeurs du dictionnaire.
5. Pour supprimer une entrée du dictionnaire, on peut utiliser la fonction `del` (qui permet de supprimer une variable quelconque de manière générale). Ainsi, pour supprimer l'entrée correspondant à la clé "leo" on peut utiliser l'instruction `del (d["leo"])`.
   Écrivez une fonction `delete(car, dict)` qui prend en argument un caractère `car` et un dictionnaire `dict` et supprime du dictionnaire toutes les entrées correspondant à des clés qui commencent par la lettre `car`.

## Exercice 4 – Fonction de hachage et anagrammes

On considère des clés sur un ensemble de 256 caractères (l'aphabet ASCII 8 bits par exemple) et l'on associe à chaque clé l'entier qu'elle représente en base 256. Ainsi, par exemple, puisque les caractères `B`, `l`, `o`, `p` correspondent aux valeurs 66, 108, 111 et 112 respectivement, la clé `"Blop"` est associée à l'entier

$$66.256^3 + 108.256^2 + 111.256 + 112 = 1114402672 \tag{1}$$

1. Écrivez une fonction python *str_to_int* qui prend en entrée une chaîne de caractères en ASCII 8 bits et renvoie l'entier associé.
   **Indication** : vous pourrez utiliser la fonction `ord(c)` qui renvoie la valeur ASCII du caractère `c`.
   **Remarque** : Pour plus d'efficacité vous pourrez utiliser le principe de la méthode de Horner pour évaluer le polynôme :

   $$a_0.X^p + a_1.X^{p-1} + ... + a_{p-1}.X + a_p \tag{2}$$

   en X = 256, où $a_0, a_1, ..., a_p$ sont les entiers correspondant aux lettre de la chaîne `s` passée en argument.

2. On considère la fonction de hachage qui permet de calculer à partir d'un grand nombre $x$ un nombre $y$ compris entre 0 et 255 :

   $$h : x \to (x \ mod \ 255) \tag{3}$$

   Pour tout mot `s` correspondant à $x$ (par la fonction `str_to_int`), si un mot `w` est obtenu à partir de `s` par permutation de ses lettres (mêmes lettres, même nombre d'occurrences, mais l'ordre est quelconque), alors $hachage(s) = hachage(w)$.
   Écrivez la fonction `hachage` en python qui prend en argument une chaîne de caractères, la convertit en entier puis le hache par la fonction $h$ et vérifiez la propriété annoncée sur quelques exemples. Vous remarquerez en particulier que `hachage("chien")` et `hachage("niche")` renvoient la même valeur (en l'occurrence 9).

3. Expliquez.

4. Utilisez la fonction `hachage` précédente pour déterminer si deux mots sont des anagrammes.

## Exercice 5 – Table de hachage – Le paradoxe des anniversaires

1. Si l'on considère un groupe de $N$ personnes, quelle est la probabilité que deux d'entre elles soient nées le même jour de l'année ? Pour cette question, vous donnerez simplement une expression de la probabilité (commentaire dans le programme Python).
   **Indications**

   - Le jour est déterminé par un nombre entre 1 et 365. Vous compterez systématiquement une année de 365 jours (en négligeant le 29 février).
   - Vous considèrerez qu'il y a équi-probabilité pour qu'une personne soit née un jour quelconque de l'année compris entre 1 et 365.

- Vous pourrez d'abord exprimer la probabilité pour que 2 personnes soient nées un jour différent (364/365), puis la probabilité pour que la troisième soit née un jour différent des deux précédentes, etc.
- Vous en déduirez la probabilité que tous soient nés un jour différent.

2. En utilisant le résultat de la question 1, écrivez la fonction `anniversaire(n)` qui calcule la probabilité qu'il existe 2 personnes parmi un groupe de $n$ personnes ayant leur anniversaire le même jour. En particulier, combien vaut cette probabilité pour un groupe de 23 personnes?

3. Quel lien peut-on établir avec les tables de hachage ?

4. Généralisez la fonction `anniversaire` en écrivant la fonction `collision` qui calcule la probabilité que l'on obtienne une collision en ajoutant $n$ valeurs dans une table de hachage de taille $m$.

5. Pour une taille de table donnée (`m` = 100 par exemple), tracez la courbe qui donne la probabilité qu'il y ait une collision en fonction du nombre de valeurs en entrée (`n` variable).

6. Donnez une estimation (expérimentale) du nombre de clés à insérer pour avoir une probabilité à peu près constante d'avoir une collision.
   **Indications** A l'aide de la fonction `collision` on peut vérifier expérimentalement que le nombre de clés à insérer pour avoir une probabilité fixe d'avoir une collision est de l'ordre de $\sqrt{m}$. Le plus simple est de faire varier `n` pour des valeurs de plus en plus grandes et de remarquer que la probabilité converge (tracer la courbe).