TD hachage

Exercice 1

Montrer comment on réalise l'insertion des clés 5,28,19,15,20,33,12,17,10 dans une table de hachage où les collisions sont résolues par chaînage. On suppose que la table contient 9 alvéoles et que la fonction de hachage est $h(k) = k \mod 9$.

Exercice 2

On souhaite comparer plusieurs fonctions de hachage agissant sur les chaînes de caractères constituées uniquement des lettres en majuscule (`A` à `Z`). Nous commencerons par écrire ces quatre fonctions, puis nous estimerons la probabilité de collisions entre deux clés choisies au hasard.
Les quatre fonctions de hachage à coder sont les suivantes :

- h_1 : L'utilisation du code ASCII de la première lettre de la chaîne décalé (0 pour la lettre `A` et 25 pour la lettre `Z`, on se servira de la fonction `ord()`).
- h_2 : La somme des codes ASCII des lettres du mot.
- h_3 : La fonction qui à la chaîne ch associe $\sum_{k=0}^{len(ch)-1} ord(ch[k])26^k$.
- h_4 : La fonction `FNV` (dont une variante est utilisée pour le codage de la fonction de hachage de Python).
  - ➔ On part de l'entier $x = 2166136261$ qui va être modifié itérativement en parcourant les caractères de la chaîne.
  - ➔ Pour chaque caractère de la chaîne, on effectue un `xor` (ou exclusif bit à bit) entre la représentation en binaire du caractère et celle de x (cette opération s'obtient via la syntaxe `ord(caract)^x`, le symbole ^ correspond au ou exclusif sur les bits).
  - ➔ On multiplie le résultat par un entier premier, ici 16777619.
  - ➔ Enfin, on affecte à x le reste dans la division euclidienne de ce dernier résultat par 2^{32}.

1. Écrire ces quatre fonctions de hachage (h_i) en langage Python. Chaque fonction prendra en argument une chaîne de caractère et renverra un entier.
2. Écrire un script permettant d'ouvrir le fichier *mots.txt* et stockant son contenu dans une liste (chaque élément de la liste étant une ligne du fichier texte).
3. Écrire une fonction qui tire au hasard deux mot dans la liste (on peut utiliser la fonction `sample()` du module `random`).
4. Tirer deux mots au hasard, calculer leur indice dans une table de hachage de taille $n = 10000$ (rappel : $h(k)\%n$) et vérifier s'il y a collision.
5. Réitérer cette opération pour un grand nombre de tirages (environ 10^6) et calculer la probabilité de collision entre deux indices pour chacune des fonctions de hachage.
6. Quelle est la probabilité de collision pour un hachage uniforme simple ?
7. Conclure quant à la qualité du hachage.
8. Expliquer pourquoi la première fonction de hachage n'est pas une solution à garder.

Exercice 3

L'algorithme naïf de recherche d'un motif de longueur m dans un texte `T` de longueur n vu en première année a une complexité en $O(n \times m)$. On souhaite faire mieux à l'aide de fonction de hachage.
L'algorithme de Karp-Rabin permet de faire cette recherche avec une complexité en $O(n+m)$ en moyenne à l'aide de fonction de hachage déroulante, c'est à dire une fonction de hachage qui permet de calculer `h(T[i+1: i+m+1])` à partir de `h(T[i: i+m])` en temps constant et non en $O(m)$.
On a alors le pseudo code suivant :

```
rabin_karp(T, M)
 1.  n <-- longueur(T)
 2.  m <-- longueur(M)
 3.  hn <-- hach(T[0: m])
 4.  hm <-- hach(M[0: m])
 5.  pour i de 0 à n-m+1 faire
 6.    si hn = hm
 7.      si T[i: i+m] = M[0: m]
 8.        résultat trouvé : i
 9.    hn <-- hach(T[i+1: i+m+1])
10. résultat non trouvé
```

Comme fonction de hachage, on utilisera la fonction `h3` de l'exercice précédent, ce qui revient à voir un mot comme un nombre écrit en base 26. Puis prendre le reste dans la division euclidienne par un nombre premier p suffisamment grand (devant la taille du motif).
Ainsi multiplier par 26 revient à ajouter un 0 (c'est à dire `'A'`) à la fin de son écriture. De même ajouter un caractère à la fin on multiplie par 26 puis on ajoute la valeur du caractère ; et pour retirer le premier caractère, il suffit de soustraire la valeur du caractère fois 26^{m-1}.

1. Écrire une fonction `décaler` qui prend en paramètre un nombre qui représente `hm`, l'entier `m` et un caractère `c` (le caractère suivant dans le texte) et qui renvenvoie le hachage de la section suivante à considérer.
   Ne pas oublier la division euclidienne par p.
2. Écrire la fonction `karp_rabin`.