EA4 – Éléments d'algorithmique
TD n° 10 : tables de hachage

Exercice 1 : échauffement

On considère une table de hachage `T` à adressage fermé avec résolution des collisions par chaînage, de longueur $m = 11$, utilisant la fonction de hachage $h(k) = k \mod m$.
Insérer successivement les clés 5, 28, 19, 15, 20, 33, 37, 17, 26 et 10 dans `T`, puis supprimer ensuite les clés 19, 37 et 17.
On fixe maintenant pour la table un taux de remplissage maximal de $\frac{3}{4}$. Comment sont alors réalisées ces insertions et suppressions ?

Exercice 2 : hachage et statistiques

On souhaite déterminer quels sont les mots utilisés par Proust dans la *Recherche du temps perdu*, et leur fréquence. Proposer une solution utilisant un `dict`ionnaire Python, en supposant qu'on dispose d'un générateur `proust()` qui permet d'itérer sur tous les mots du roman, l'un après l'autre, par une simple boucle `for mot in proust()`.
Pouvez-vous estimer la complexité de votre solution, sachant que la longueur du texte est d'environ 10 millions de caractères pour 1.5 million de mots ? Comparer avec d'autres méthodes.

Exercice 3 : hachage et (autres) opérations ensemblistes

On représente ici des ensembles d'entiers naturels par des couples `(T, h)`, où `T` est une table de hachage à adressage fermé avec résolution des collisions par chaînage, et `h` la fonction de hachage associée.

1. Écrire une fonction `liste_elements(E)` qui retourne une liste constituée de tous les éléments de l'ensemble `E`. Quelle est sa complexité ?
2. Écrire une fonction `union(E, F)` prenant en paramètre deux ensembles et retournant leur union. On supposera qu'on dispose d'une fonction `ensemble_vide()` qui initialise et retourne un ensemble vide, et d'une fonction `ajoute(E, elt)` qui ajoute un `elt` dans la table représentant `E`. Quelle est sa complexité ?
3. Écrire une fonction `intersection(E, F)` prenant en paramètre deux ensembles et retournant leur intersection. On supposera qu'on dispose d'une fonction `contient(E, elt)` qui recherche `elt` dans la table représentant `E` et retourne `True` ou `False` selon les cas. Quelle est sa complexité ?

Exercice 4 : redimensionnement de table de hachage

On représente ici une table de hachage par une liste de 6 éléments :

`[cles, h, taille, tmax, tmin, nbCles]`,

où `cles` est un tableau de (listes de) clés, `h` est une fonction de hachage, `taille` est la taille du tableau `cles`, `tmax` et `tmin` les taux maximal et minimal de remplissage autorisés, et `nbCles` est le nombre de clés du tableau. Parmi ces 6 éléments, trois ne varient jamais, la fonction de hachage `h` et les taux de remplissage `tmax` et `tmin`. La fonction de hachage associe à une clé un (grand) entier naturel, qui est considéré modulo `taille` pour obtenir le hachage de la clé.

1. Écrire une fonction `redimensionne(table, t)` qui prend en paramètre une `table` de hachage par chaînage et une (nouvelle) taille `t`, et retourne la table de hachage redimensionnée à la taille `t`.

   Pour quelles valeurs de `t` cette fonction est-elle utilisée en général ? Dans quelles circonstances ?
2. Quelle est la complexité de cet algorithme ?
3. Écrire une fonction `ajoute(table, elt)` qui ajoute un élément dans la `table` en effectuant un redimensionnement si nécessaire.
4. Soit n le nombre de clés présentes dans une table après une succession d'insertions et de suppressions ; si des redimensionnements ont été effectués à chaque fois que le taux de remplissage a atteint un certain α fixé, quelle est la complexité amortie de ces redimensionnements ?
5. Écrire une fonction `supprime(table, elt)` qui supprime `elt` de la `table`, en procédant à un redimensionnement si nécessaire.

Exercice 5 : analyse du hachage par chaînage

On considère une table de hachage de m boîtes contenant n éléments, supposés insérés dans un ordre aléatoire. On suppose par ailleurs que la résolution des collisions est faite par chaînage, et que chaque élément a une probabilité uniforme d'être haché vers l'une des m boîtes (hypothèse de *hachage uniforme simple*).

1. Quel est le nombre moyen d'éléments par boîte ?
2. Quel est alors le nombre moyen de tests faits lors d'une recherche *infructueuse* ?
3. Quelle est donc la complexité moyenne d'une recherche infructueuse ?
4. Lors d'une recherche réussie, quels éléments sont examinés avant l'élément cherché ?
5. Si k éléments ont été insérés dans la table après l'élément cherché, quel est le nombre moyen d'insertions dans une boîte donnée ?
6. Quel est alors le nombre moyen d'éléments insérés après l'élément cherché dans la même boîte que lui ?
7. Quelle est donc la complexité moyenne d'une recherche réussie ?