# EA4 – Éléments d'algorithmique
# TD n° 9 : 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 (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 renvoie une liste constituée de tous les éléments de l'ensemble `E`. Quels sont les éléments qu'il faut prendre en compte pour estimer sa complexité ?
2. Écrire une fonction `union(E, F)` prenant en paramètre deux ensembles et renvoyant leur union. On supposera qu'on dispose d'une fonction `ensemble_vide(m,h)` qui initialise et renvoie un ensemble vide indexé par une fonction h sur une table de taille $m$, et d'une fonction `ajoute(E, elt)` qui ajoute un `elt` dans la table représentant `E`. Comment analyser sa complexité ? Essayez d'optimiser les choses dans le cas où il y aurait une contrainte sur le taux de remplissage.
3. Écrire une fonction `intersection(E, F)` prenant en paramètre deux ensembles et renvoyant leur intersection. On supposera qu'on dispose d'une fonction `contient(E, elt)` qui recherche `elt` dans la table représentant `E` et renvoie `True` ou `False` selon les cas. Quelle est sa complexité ?

### Exercice 3 : redimensionnement de table de hachage

Une table de hachage est vue ici comme un objet ayant six attributs :
- l'attribut `cles` est un tableau de (listes de) clés,
- l'attribut `h` est une fonction de hachage,
- l'attribut `taille` est la longueur du tableau `cles`,
- les attributs `tmax` et `tmin` sont les taux maximal et minimal de remplissage, et
- l'attribut `nbCles` est le nombre de clés du tableau.

Parmi ces six attributs, trois ne varient jamais : la fonction de hachage `h` et les taux de remplissage `tmax` et `tmin`. La fonction de hachage `h` 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 et une (nouvelle) taille `t`, et redimensionne `table` 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 $m$ la longueur d'une table après une succession d'insertions seulement ; des redimensionnements ont été effectués à chaque fois que le taux de remplissage a atteint `tmax` $= \alpha$. Combien de clés étaient présentes dans la table au moment de son remplissage maximal ?

Si la longueur de la table au moment de sa création était $\ell$, et que le coût de chaque redimensionnement est linéaire, quel est le coût cumulé des redimensionnements effectués ?

Déduire des résultats précédents une borne supérieure pour 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 4 : 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* ? Quelle est donc la complexité moyenne d'une recherche infructueuse ?

**3.** Lors d'une recherche réussie, quels éléments sont examinés avant l'élément cherché ?

**4.** Si $k$ éléments ont été insérés dans la table avant [1] l'élément cherché, quel est le nombre moyen d'insertions dans une boîte donnée ?

**5.** En déduire la complexité moyenne d'une recherche réussie.

1. ou après, selon votre réponse à la réponse précédente...