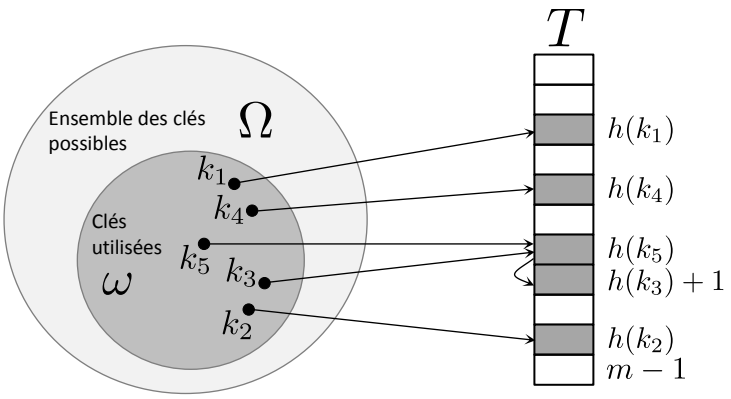
Algorithmique - TD6
Table de hachage à adressage ouvert

1 Introduction

Dans ce TP, vous allez concevoir et implémenter en langage C une table de hachage avec adressage ouvert.

Voici pour rappel le fonctionnement d'une telle table :



Lorsque la valeur de la fonction de hachage donne deux fois la même valeur, il faut utiliser une autre case du tableau (d'où l'aspect ouvert de l'adressage). La fonction de hachage/codage est donnée dans l'ossature de base sur moodle, voici son prototype :

```
unsigned int HashFunction(Key key, unsigned int size);
```

Cette fonction prend donc la clé `key` en argument ainsi que la taille de la table `size`.

Voici les structures de données qui sont préconisées pour la représentation d'une cellule et de la table elle-même :

```
/* état d'une case (peut être fait aussi avec un enum) */
#define EMPTY 0
#define SET 1
#define REMOVED 2

/* utile pour l'affichage */
const char * Labels[] = {"Empty","Set","Removed"};

/* une cellule de la table */
typedef struct {
   char * key;
   int    status;
```

```
   char * val;
} Data;

/* la table de hachage elle-même */
typedef struct {
  Data * tab;
  int    size;
} HashTable;
```

2 Travail à effectuer

Implémenter les opérations suivantes à l'aide de fonctions prenant en argument un pointeur sur `HashTable` :

- Initialiser la table (argument : la taille de la table sous forme d'entier). On veillera à bien initialiser l'état de chaque cellule. (fonction déjà fournie dans l'ossature)
- Insérer un élément à partir du couple (clé, valeur). Cette opération écrase la valeur associée si la clé était déjà présente dans le dictionnaire. Attention, dans le cas où on essaye de placer l'information dans une case ayant été supprimée, il faut être sûr que la clé n'était pas présente plus loin dans le tableau.
- Supprimer un élément à partir de sa clé. On fera passer la case à l'état supprimé.
- Effectuer une requête sur une clé.
- Afficher les statistiques de la table : nombre total de cases, nombre de cases remplies, supprimées et vides.
- Libérer la structure de données.

3 Description des entrées/sorties

Pour valider votre solution, les entrées et les sorties doivent respecter un format très précis qui sont déjà implémentées dans l'ossature de base donnée sous moodle. Les commandes possibles sont :

```
init <n>
insert <cle> <valeur>
delete <cle>
query <cle>
stats
destroy
bye
```

La commande `init` permet d'allouer la mémoire nécessaire à la table de hachage. Elle prend en argument le nombre de cases du tableau `<n>` un nombre entier. `<cle>` et `<valeur>` représentent des chaîne de caractères quelconques sans espace. La commande `insert` permet d'ajouter une paire (clé, valeur) au dictionnaire. La commande `delete` permet de supprimer un élément du dictionnaire. La commande `query` permet de faire une requête sur une clé. Si cette clé est présente, la valeur associée sera affichée, si elle ne l'est pas un message

spécial sera affiché. La commande `destroy` permet de libérer la mémoire. La commande `bye` termine l'exécution du programme, et la commande `stats` permet d'afficher les statistiques de la table (voir ci-dessous pour le format de sortie, attention à bien respecter le nombre d'espaces, ceux-ci ont été symbolisés par un symbole spécial dans le format de sortie). Chaque ligne affichée sera terminée par deux caractères (CR et LF), correspondant à la chaîne `"\r\n"`.

Un exemple d'entrée :

```
init 100
insert a b
insert a c
query a
query b
insert b d
delete a
query a
stats
destroy
bye
```

avec la sortie correspondante :

```
c
Not␣found
Not␣found
size␣␣␣␣:␣100
empty␣␣␣:␣98
deleted␣:␣1
used␣␣␣␣:␣1
```

Rappel : une ossature de programme en langage C vous est fournie sur moodle, elle permet l'interprétation des commandes décrites ci-dessus et vous donne une fonction de hachage efficace pour une chaine de caractères.