TD 3 : Tables de hachage

Exercice 1 – Empreintes Digitales

On estime à 1/64 milliards la probabilité que deux individus possèdent les mêmes empreintes digitales. A l'échelle de la population humaine (plusieurs milliards d'individus), cette probabilité est suffisante pour assurer une identification fiable. Une empreinte digitale est stockée sous forme d'un fichier image (JPEG) puis traitée afin de récupérer les informations nécessaires à l'identification. Ces informations sont représentées sous la forme d'une suite de six entiers compris entre 0 et 99 (par exemple, [02, 17, 24, 04, 77, 20]).

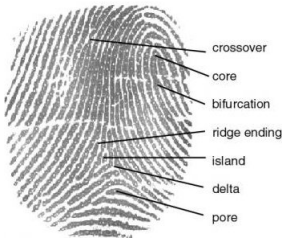

FIGURE 1 – Empreintes digitales

Les services de police disposent d'une banque de données associant une fiche personnelle (nom, prénom, âge, adresse, etc...) à une empreinte digitale.

Q 1.1 Quelle structure de données suggérez-vous pour le stockage des données en question ?

Q 1.2 La mémoire des ordinateurs dont dispose les services de polices étant limitée, la taille maximale des tableaux ne peut dépasser 100 000. Que peut-on en déduire sur la fonction de hachage à utiliser ?

Q 1.3 Soit $[x_1, x_2, x_3, x_4, x_5, x_6]$ le vecteur associé aux empreintes digitales d'un individu. Que pensez-vous de la fonction de hachage suivante ? Comment l'améliorer ?

```
1 int g(int x1,  int x2, int x3, int x4, int x5, int x6) {
2   return (x1*x2*x3*x4*x5*x6) % 100000;
3 }
```

Q 1.4 La banque de données des services de police contient les entrées suivantes :

$$[02, 04, 10, 01, 00, 17]$$
$$[02, 10, 00, 02, 09, 05]$$
$$[00, 01, 17, 04, 02, 10]$$

Calculer la valeur renvoyée par la nouvelle fonction de hachage pour chacune de ces clés. Qu'observe-t-on ? Que peut-on proposer afin de résoudre ce problème ?

Exercice 2 – Comparaison de différents types de tables

On considère l'ensemble de 13 mots du tableau suivant, et on suppose qu'il existe une fonction f qui associe à chaque mot une clé qui est valeur entière (donnée en hexadécimal dans le tableau).

	mot μ	clé $f(\mu)$ (en hexadécimal)
01	"le"	FF2E
02	"cours"	178DD38
03	"appelé"	75EA33
04	"structures"	35CE5
05	"de"	9AA8BF1
06	"données"	2738
07	"est"	A4C74
08	"absolument"	1CA4C74
09	"génial"	14D26
10	"j'adore"	5A38
11	"faire"	1BAE5
12	"ses"	65B4EE5
13	"TD/TME"	8C74

Q 2.1 Nous allons utiliser la fonction de hachage $g(k) = k \bmod 16$ pour chaque clé k associée à un mot. Donner en base décimale, la valeur hachée associée à chaque clé du tableau.

Q 2.2 Pour chaque type de table de hachage ci-dessous, indiquez ce que l'on obtiendrait si l'on partait d'une table vide de taille 16 et que l'on insérait dans l'ordre tous les mots ci-dessus avec la fonction de hachage $g(k) = k \bmod 16$.

1. Table de hachage avec résolution des collisions par chaînage.
2. Table de hachage avec adressage ouvert et probing linéaire $h(k, i) = g(k) + i$.
3. Table de hachage avec adressage ouvert et probing quadratique $h(k, i) = g(k) + \frac{i}{2} + \frac{i^2}{2}$.

Pour les adressages ouverts, calculez le nombre de probes à effectuer en moyenne pour obtenir un élément donné dans la table. Déduisez-en le meilleur des deux probings.

Exercice 3 – fonction de hachage

On considère l'ensemble de couples (clé,valeur) suivant :

	mot	clé (en décimal)
01	“le”	123
02	“cours”	22
03	“appelé”	88
04	“structures”	43
05	“de”	4
06	“données”	28
07	“est”	73
08	“absolument”	7
09	“génial”	13

Q 3.1 Calculez les valeurs hachées des clés obtenues pour chaque couple par les fonctions ci-dessous :

```
int f1(int x) {
  return x;
}
```

```
int f2(int x) {
  return 10*x;
}
```

```
int f3(int x) {
  return 2*x;
}
```

```
int f4(int x) {
  return (x==0)?0:(8 * f4(x/10) + x % 10);
}
```

Q 3.2 Supposons que l'on veuille insérer les mots ci-dessus dans une table de hachage de longueur 10. On veut donc hacher les clés des mots en utilisant une fonction de hachage $g(x) = f(x) \bmod 10$, où $f \in \{$`f1`,`f2`,`f3`,`f4`$\}$. Quel est le meilleur choix pour f ? Expliquez pourquoi.

Exercice 4 – Implémentation des tables de hachage par chaînage

Dans cet exercice, on considère les tables de hachage par chaînage, et on désire étudier comment bien les implémenter. On suppose ici que les éléments à stocker sont des entiers, et que chaque entier est associé à une clé entière.

Q 4.1 Quelle structure de données proposez-vous ? Donner le code C permettant de définir cette structure, de l'allouer et de l'initialiser.

Q 4.2 Les données sont stockées dans la table en utilisant une fonction de hachage `int g(int val)` donnée. Donner le code de la fonction d'insertion d'un élément dans la table de hachage.

Q 4.3 Selon l'application, il peut être important de pouvoir récupérer facilement le nombre d'éléments introduits dans la table. Comment proposez-vous de réaliser cette opération ?

Q 4.4 L'opération de suppression d'un élément dans une table de hachage est une opération également importante. Quelle modification de la structure permettrait d'améliorer l'implémentation d'une telle suppression ?