FIT 2 [INF224] – 2021/22

# Algorithmique et structures de données
# TP 07

Gulfem Isiklar Alptekin – Ozgun Pinarer

## Fonction de hachage

Une fonction particulière qui, à partir d'une donnée fournie en entrée, calcule une empreinte numérique servant à identifier rapidement la donnée initiale, au même titre qu'une signature pour identifier une personne.

La tâche de cette fonction est de générer un entier unique (clé) à partir d'une valeur qui lui est donnée. Mais dans le domaine de l'application, il est très difficile voire impossible de trouver la fonction appropriée qui générera toujours un numéro unique. Si le même nombre est produit à partir de deux valeurs différentes, cela s'appelle un conflit.

Propriétés d'une fonction de hachage :

- Il peut prendre n'importe quelle valeur de n'importe quelle longueur.
- Il devrait être capable de générer une clé de la longueur spécifiée en sortie.
- Il doit être unidirectionnel. La valeur donnée à la fonction ne doit pas être obtenue à partir de la clé générée par la fonction.

- RFC 1321 (algorithme MD5 Message-Digest, avril 1992)
- RFC 1828 (Authentification IP à l'aide de Keyed MD5, août 1995)
- RFC 2069 (An Extension to HTTP : Digest Access Authentication, janvier 1997)
- RFC 2537 (clés et SIG RSA/MD5 dans le système de noms de domaine (DNS), mars 1999)
- RFC 2104 (HMAC : Keyed-Hashing pour l'authentification des messages, février 1997)

```
#define MAX_ROW 16
int RSHash(char *str, int len)
{
   int b = 378551;
   int a = 63689;
   int hash = 0;
   int i = 0;
   for (i = 0; i < len; str++, i++)
   {
      hash = hash * a + (*str);
      a = a * b;
   }
   return (hash & 2147483647) % MAX_ROW;
}
```

MAX_ROW spécifie combien d'éléments nous allons stocker dans la table de hachage. Nous nous assurons que le numéro unique que nous obtiendrons avec ce processus de modulation se situe dans les limites de notre table.

## Le Table de hachage

Les listes chaînées ont un gros défaut lorsqu'on souhaite lire ce qu'elles contiennent : il n'est pas possible d'accéder directement à un élément précis. Il faut parcourir la liste en avançant d'élément en élément jusqu'à trouver celui qu'on recherche. Cela pose des problèmes de performance dès que la liste chaînée devient volumineuse. Les tables de hachage représentent une autre façon de stocker des données. Elles sont basées sur les tableaux du langage C. Elles permettent de retrouver instantanément un élément précis, que la table contienne 100, 1 000, 10 000 cases ou plus encore !

```
#define NAME_LENGTH    20
typedef struct _Row{
   int index;
   char name[NAME_LENGTH];
   struct _Row *pNext;
}Row;
```

index : Il stocke la valeur produite par la fonction de hachage.
name : Stocke les données fournies à la fonction de hachage.
pNext : Si la fonction de hachage produit la même valeur (conflit) pour des données différentes, elle représente le pointeur vers l'autre enregistrement de la même ligne.

```
typedef struct _Table {
   Row rows[MAX_ROW];
}Table;

Table table;
```

## Les Collisions

Quand la fonction de hachage renvoie le même nombre pour deux clés différentes, on dit qu'il y a collision.

Deux raisons peuvent expliquer une collision.

- La fonction de hachage n'est pas très performante. C'est notre cas. Nous avons écrit une fonction très simple (mais néanmoins suffisante) pour nos exemples. Les fonctions MD5 et SHA1 mentionnées plus tôt sont de bonne qualité car elles produisent très peu de collisions. Notez que SHA1 est aujourd'hui préférée à MD5 car c'est celle des deux qui en produit le moins.
- Le tableau dans lequel on stocke nos données est trop petit. Si on crée un tableau de 4 cases et qu'on souhaite stocker 5 personnes, on aura à coup sûr une collision, c'est-à-dire que notre fonction de hachage donnera le même indice pour deux noms différents.

## Bilan

- Les listes chaînées sont flexibles, mais il peut être long de retrouver un élément précis à l'intérieur car il faut les parcourir case par case.
- Les tables de hachage sont des tableaux. On y stocke des données à un emplacement déterminé par une fonction de hachage.
- La fonction de hachage prend en entrée une clé (ex. : une chaîne de caractères) et retourne en sortie un nombre.
- Ce nombre est utilisé pour déterminer à quel indice du tableau sont stockées les données.
- Une bonne fonction de hachage doit produire peu de collisions, c'est-à-dire qu'elle doit éviter de renvoyer le même nombre pour deux clés différentes.
- En cas de collision, on peut utiliser l'adressage ouvert (recherche d'une autre case libre dans le tableau) ou bien le chaînage (combinaison avec une liste chaînée).

Créez un menu et écrivez les fonctions :

- inserer, rechercher, supprimer

Developpez une application en utilisant le hachage. Expliquez l’application vous créez et implementez-le.

## IMPORTANT:

- Vous devez effectuer des études TP en tant que « fichier d'en-tête – fichier source – fichier de test » : écrivez les en-têtes des fonctions que vous avez écrites dans le fichier d'en-tête, les codes des fonctions dans le fichier source et les codes de test dans le fichier d'essai.
- Nous préférons faire des études de TP dans un environnement linux, sans avoir besoin d'IDE supplémentaire.
- Compressez vos fichiers et téléchargez-le sur le système en les nommant « NumeroEtudiant_Prenom_Nom_TPX.zip ».