Chapitre 9

Fonctions de Hachage

On aimerait stocker N éléments dans une table. Comment doit-on organiser ces éléments dans la table afin que d'une part on peut les trouver rapidement sans faire une recherche linéaire (parcourir un à un les éléments de la table) et d'autre part limiter au maximum l'espace mémoire utilisé pour préserver les éléments de la table (éviter donc une table de taille « infinie »).

Pour trouver un élément dans une table, il faudra donc rechercher son index si nous voulons éviter une recherche linéaire. Son index est déterminé en utilisant pour cela une fonction donnée. Par exemple, dans le cas des caractères, cette fonction peut retourner la valeur unicode du caractère i.e. (int) c, qui représente l'index de l'élément dans la table. Pour une chaîne de caractères de longueur n, c'est une fonction de tous les caractères de la chaîne. Par généralisation, pour un objet donné, x, l'appel x.hashcode() doit retourner un entier qui est l'index de l'objet dans la table.

En conclusion pour localiser l'élément dans sa table, il faut donc savoir comment calculer une fonction de l'élément qui donne un entier qui sera l'indice dans le tableau. Il faudra avoir une bonne fonction qui minimise les indices identiques pour des objets différents.
Si la fonction de hachage retourne l'index de la première lettre d'un mot, et si tous les mots de la table, commencent par la même lettre, alors la recherche d'un élément dans la table se transforme à une recherche linéaire, l'utilisation dans ce cas du mécanisme de hachage n'a pas de sens.

Soit T la taille du tableau. Tout élément dont l'index dépasse la valeur T sera calculé modulo cette taille. Ainsi nous aurons une meilleure distribution des éléments dans la table au lieu qu'ils soient complètement dispersés.

Soit l'ensemble des éléments entiers strictement positifs Z = {11, 59, 32, 44, 31, 26, 19} et un tableau de taille 10. Nous allons donc ranger les éléments de Z dans un tableau dont l'index varie entre 0 et 9 (<10).

La fonction de hachage pour l'élément z_i de l'ensemble Z est déterminée par :

$h(z_i) = z_i$ modulo [10].

z_i	$h(z_i)$
11	1
59	9
32	2
44	4
31	1
26	6
19	9

On constate dans cet exemple une duplication d'indices : 1 pour 11 et 31 ; 9 pour 59 et 19.

Plusieurs solutions sont possibles pour des éléments ayant le même indice : soit utiliser un des indices libres disponible dans la table moyennant certaines regèles à définir ; sinon utiliser des listes (chaînées) d'objets (buckets) à cet indice.

Dans le cas de l'utilisation de buckets l'algorithme de recherche d'un élément dans la table se résume ainsi :

- Obtenir l'index i dans la table
  - Calculer le hash code.
  - Réduire le hash code modulo la taille de la table pour obtenir l'index i.
- Itérer dans le bucket à l'adresse i.
- Pour chaque élément dans le bucket vérifier s'il est égal à x, l'élément recherché.
- Si on trouve un élément égal à x alors x est dans l'ensemble.
- Autrement, x n'est pas dans l'ensemble.

i	**0**	***1***	**2**	**3**	**4**	**5**	**6**	**7**	**8**	***9***
t[i]		*11*	32		44		26			*59*
		31								*19*

h(11) = h(31) = 1.

En utilisant la technique de la première case vide à droite de l'élément, nous obtenons la représentation suivante :

i	**0**	***1***	**2**	**3**	**4**	**5**	**6**	**7**	**8**	***9***
T[i]	**19**	*11*	32	**31**	44		26			*59*

Dans ce cas il faudra définir une fonction h qui tient compte de cette particularité.