Chapitre IV: Méthodes de Hachage

Chapitre IV: Méthodes de Hachage

Problème

- Il est impossible de réserver une place pour chaque élément de l'univers

Solution

- Utiliser des méthodes de hachage qui ont comme but de stocker des éléments en nombre relativement faible par rapport à l'univers auquel ils appartiennent

Chapitre IV: Méthodes de Hachage

Principes

- L'adresse de stockage d'un élément est calculé directement à partir de sa clé
- La recherche, l'adjonction ou la suppression s'effectue en temps constant.
- Le hachage s'effectue dans une zone primaire de stockage notée par l'intervalle de ses valeurs de hachage [0, m − 1].

Chapitre IV: Méthodes de Hachage

Table de hachage

• Une table de hachage(hash table anglais) est ŧstructure de données qui permet une association clé-élément, ŧimplémentation du type abstrait table de symboles. …

• On accède à chaque élément de la table via sa clé.

…

• L accès à un élément se fait en transformant la clé en une valeur de hachage (ou simplement hachage) par l'intermédiaire d'une fonction de hachage.

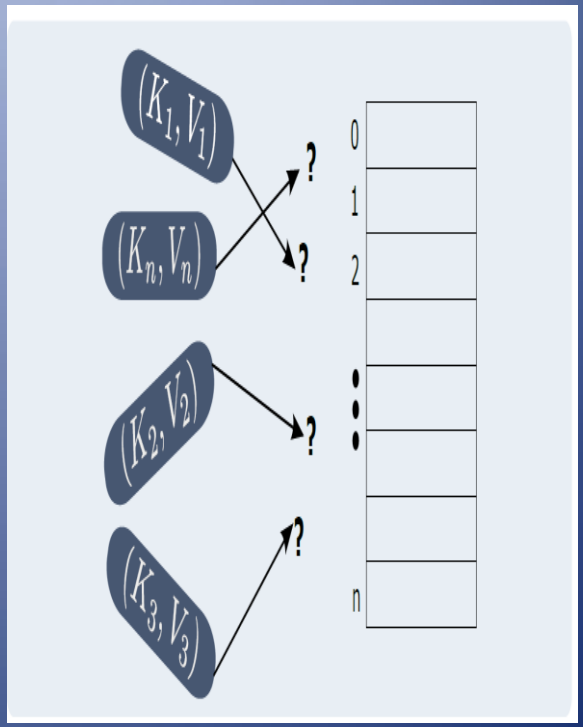
• .Le hachage est un nombre qui permet la localisation des éléments dans le tableau,. …

•Une case dans le tableau est appelée alvéole

Chapitre IV: Méthodes de Hachage

Table de hachage (Caractéristiques)

- Une structure de type dictionnaire
- Contenant des couples (clé , valeur)
- La clé identifie la valeur
- Pas de notion d'ordre
- Pas de doublon
- Basée sur les tableaux



Chapitre IV: Méthodes de Hachage

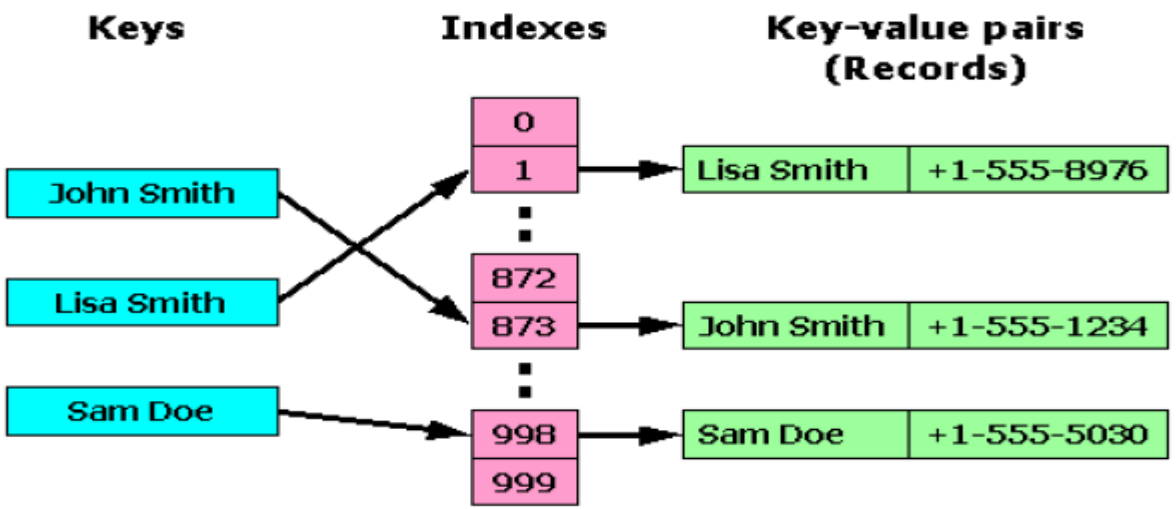
Table de hachage (Utilisation)

L'accès à un table de hachage peut être effectuer par les opérations suivantes

- Insérer(H,elt,clé) : insère dans la table de hachage H un élément elt dont la clef est clé .;.
- Recherche(H,clé) : recherche dans la table de hachage H si un élément est associé à la clef clé et renvoie cet élément ; …
- Booléen Appartient(H clé ,) : recherche dans la table de hachage H si un

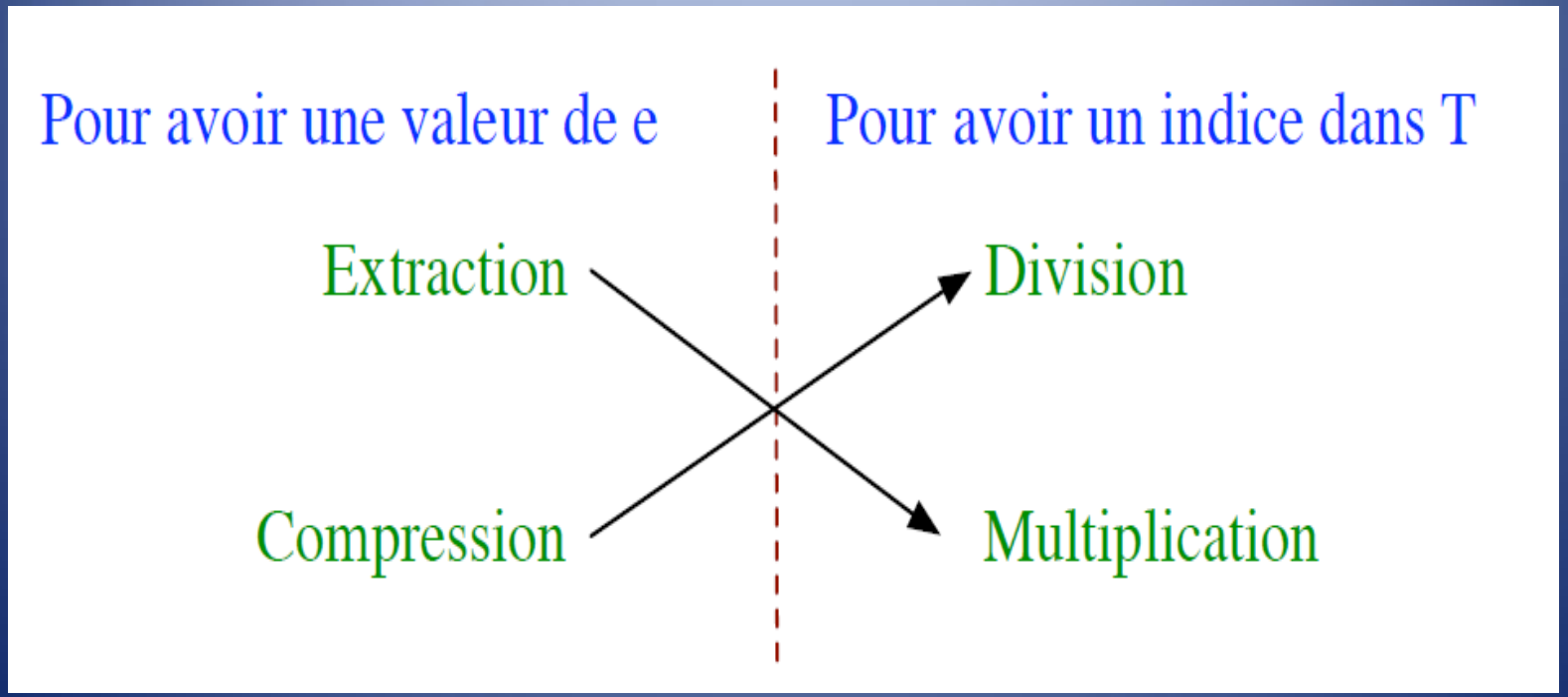
Chapitre IV: Méthodes de Hachage

Table de hachage (Exemple



Chapitre IV: Méthodes de Hachage

Méthodes principales de hachage



Chapitre IV: Méthodes de Hachage

Hachage par Extraction

Principe

Extrait une partie de la valeur (de la chaine de bits) de l'élément

Exemple

Extraction des bits 1,2,7 et 8 de l'´el´ement

élément	représentation	$h(\text{élément})_2$	$h(\text{élément})_{10}$
ET	0010110100	1000	8
OU	0111110101	1101	13
NI	0111001001	1101	13
IL	0100101100	0000	0

Chapitre IV: Méthodes de Hachage

Hachage par extraction

Avantages

- Calcul facile à mettre en œuvre
- Si $N = 2p - 1$ alors h → un indice dans T

Inconvénients

- Adaptée seulement à des cas particuliers:
  - Quand on connaît la valeur des éléments a priori
  - Quand on sait que certains bits ne sont pas significatifs
- En général, méthode par extraction pas de bons résultats car ne dépend pas de la totalité de la valeur de l'élément .

Règle : Une bonne fonction de hachage utilise toute la valeur de e

Chapitre IV: Méthodes de Hachage

Hachage par compression

Principe

Utilise tous les bits de e pour calculer son indice dans la table par exemple :

1 . découpe la chaîne de bits de l'élément en morceaux d'égale longueur
2. on additionne les morceaux.
   Pour éviter les débordements on peut utiliser, à la place de l'addition, l'opération booléenne "ou exclusif" (xor)

Exemple

Elément	Calcul	$h(\text{élément})_2$	$h(\text{élément})_{10}$
ET	00101 xor 10100	10001	17
OU	01111 xor 10101	11010	26
NI	01110 xor 01001	00111	7
CAR	00011 xor 00001 xor 10010	10000	16

Chapitre IV: Méthodes de Hachage

Hachage par extraction

Inconvénients

- Hache de la même façon toutes les permutations d'un même mot → h(CAR) = h(ARC)
- Vient du fait que toutes les sous chaines de bits sont des représentations de caractères → plusieurs sous-chaines peuvent être identiques

Règle : une "bonne" fonction de hachage doit briser les sous-chaînes de bits

Chapitre V : Méthodes de hachage

Hachage par compression

• Le hachage par compression partage la représentation binaire de la clé et opère une suite d' opérations .
• Partage de la clé en mots de 5 bits puis sommation bit-a bit par ou-exclusif :

h(ET) = 00101 xor 10100 = 10001
h(OU) = 01111 xor 10101 = 11010
h(NI) = 01110 xor 01001 = 00111
h(IL) = 01001 xor 01100 = 00101

Chapitre V : Méthodes de hachage

Hachage par division

- Le hachage par division divise la clé par la taille du tableau.
- Le hachage par division doit tenir compte de l'uniformité des données par rapport aux diviseurs de la taille du tableau.

Taille de tableau : 23
h(ET = 0010110100) = 180 mod 23 = 19
h(OU = 0111110101) = 501 mod 23 = 18
h(NI = 0111001001) = 457 mod 23 = 20
h(IL = 0100101100) = 300 mod 23 = 1

Pour éviter des phénomènes d'accumulation il faut choisir un diviseur premier.

Chapitre V : Méthodes de hachage

Hachage par multiplication

- Le hachage par multiplication multiplie la clé par un réel θ, garde la partie décimale puis multiplie par la taille du tableau et prendre la partie entière.

Taille de tableau : 20 et θ = 0.123456.

180 ∗ 0.123456 = 22.222080 h(ET) = 4
501 ∗ 0.123456 = 61.851456 h(OU) = 17
457 ∗ 0.123456 = 56.419390 h(NI) = 8
300 ∗ 0.123456 = 37.036800 h(IL) = 0

- Des valeurs qui repartissent uniformément les clés sont :

$$\theta = \frac{\sqrt{5}-1}{2} \text{ et } \theta = 1 - \frac{\sqrt{5}-1}{2}.$$

Chapitre V : Méthodes de hachage

Problème de collision

- Deux éléments possédant la même valeur primaire de hachage créent une collision primaire.

- Le hachage réunit en petits paquets les données ayant la même clé.

- Il est pratiquement impossible d' empêcher les collisions (fonction injective).
  Deux grandes techniques pour résoudre les collisions :
  par chaînage (méthode indirecte),
  par calcul (méthode directe).

- Une fonction de hachage efficace doit faire intervenir toute la représentation de la clé et en briser la structure.

Chapitre V : Méthodes de hachage

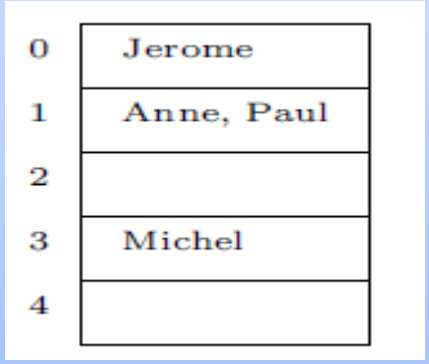
Résolution de collision par chaînage sépare

- La méthode par chaînage sépare construit pour chaque valeur de hachage une liste des clés.

- **Exemple**

Fonction de hachage h selon le rang dans l'alphabet de l'initial du mot modulo 5 (taille de la zone primaire de stockage).

h(Anne) = 1 mod 5 = 1,
h(Michel) = 13 mod 5 = 3,
h(Paul) = 16 mod 5 = 1,
h(J´erome) = 10 mod 5 = 0.



Chapitre V : Méthodes de hachage

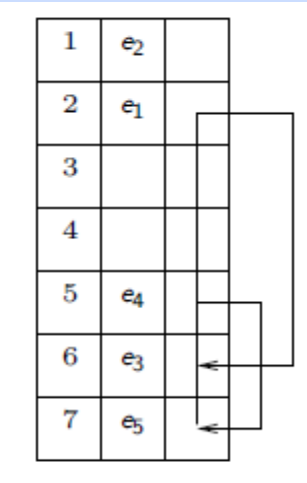
Résolution des collisions par chaînage coalescent

- L'espace maximal de mémoire de taille m est fixe a priori.
- Un espace de taille $m' \leq m$ est choisi comme espace primaire.
- L'espace de $m' + 1$ `a m est réservé aux collisions.

• **Exemple**

m = 7, m′ = 5.

h(e1) = 2 h(e2) = 1
h(e3) = 2 h(e4) = 5
h(e5) = 5 h(e6) = 2
h(e7) = 4



Chapitre V : Méthodes de hachage

Résolution des collisions par chaînage coalescent

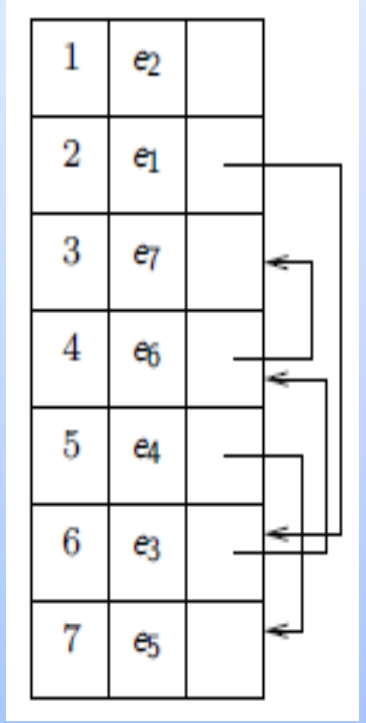
• La valeur e6 n'a pas pu être intégrée au tableau alors qu'il reste de la place.

• Idée : empiéter sur la zone primaire si la réserve est rempli

• Mais alors se créent des collisions secondaires entre e6 et e7.

• L'élément e7 (de valeur de hachage 4) est donc inséré dans la liste des collisions de valeur de hachage 2.

• Ainsi les listes de collisions pour les valeurs de hachage 2 et 4 sont coalescentes : elles sont réunies e n une seule liste.



Chapitre V : Méthodes de hachage

Résolution des collisions par calcul

• Pour un tableau de m valeurs de hachage, m fonctions de hachage h1 (hachage primaire), h2, . . . , hm.

• Pour tout élément x de l'univers des cl´es, h1(x), . . . , hm(x) doit être une permutation de [1..m].

• Les fonctions de hachage sont testées dans l'ordre jusqu'a ce que ce soit la clé (recherche positive) ou une place vide ou qu'il n'y ait plus de fonction de hachage (recherche négative).

Chapitre V : Méthodes de hachage

Résolution des collisions par calcul (Hachage linéaires)

- Hachage linéaire : calcul de la fonction de hachage hi suivant la fonction de hachage primaire : hi (x) = h1(x) + (i − 1).

- Résolution du problème du débordement de tableau par remplacement de l'addition par un calcul circulaire :

pour tout x, y ∈ {1, . . . ,m}, x $\oplus_m$ y = x + y si x + y ≤ m sinon x + y − m.

Les fonctions de hachage sont alors définies ainsi :

hi (x) = h1(x) $\oplus_m$ (i − 1)

Chapitre V : Méthodes de hachage

Résolution des collisions par calcul (Hachage linéaires)

- Hachage linéaire : calcul de la fonction de hachage hi suivant la fonction de hachage primaire : hi (x) = h1(x) + (i − 1).

- Résolution du problème du débordement de tableau par remplacement de l'addition par un calcul circulaire :

pour tout x, y ∈ {1, . . . ,m}, x $\oplus_m$ y = x + y si x + y ≤ m sinon x + y − m.

Les fonctions de hachage sont alors définies ainsi :

hi (x) = h1(x) $\oplus_m$ (i − 1)

Chapitre V : Méthodes de hachage

Résolution des collisions par calcul (Hachage linéaires)

h1(e1) = 6,
h1(e2) = 4,
h1(e3) = 7,
h1(e4) = 4,
h1(e5) = 8,
h1(e6) = 2,
h1(e7) = 5,
h1(e8) = 9,
h1(e9) = 8

1	2	3	4	5	6	7	8	9	10
e_9	e_6		e_2	e_4	e_1	e_3	e_5	e_7	e_8

- Les éléments en collisions forment des groupements qui ont tendance a croître, augmentant le nombre de fonctions de hachage a calculer.

Chapitre V : Méthodes de hachage

Résolution des collisions par calcul (Double Hachage)

- Pour éviter le phénomène d'accumulation du hachage linéaire, une deuxième fonction de hachage remplace l'incrément linéaire :

$hi(x) = h1(x) \oplus_m (i - 1)d(x)$.

- Cette fonction d doit être telle que pour tout x, d(x) soit premier avec m, sinon l'ensemble $(h1(x), \ldots, hm(x))$ n'est pas une permutation de $\{1, \ldots, m\}$.

m = 4, (pour tout x, d(x) = 2) et h1(x0) = 4 alors

$h2(x0) = h1(x0) \oplus_4 (2 - 1)d(x0) = 2$, h3(x0) = 4 et h4(x0) = 2.
(h1(x0), h2(x0), h3(x0), h4(x0)) n'est pas une permutation de [1..4].

Chapitre V : Méthodes de hachage

Suppression d'un élément

- Seule la méthode de chaînage est bien adaptée a la suppression d'un élément.

- La possibilité de supprimer des éléments dans un tableau rangé selon une méthode de hachage directe (calculée) modifie la recherche et l'adjonction dans une telle structure

1	2	3	4	5	6	7	8	9	10
e_9	e_6		e_2	e_4	e_1	e_3	e_5	e_7	e_8

- La suppression de l'élément e7 fait perdre l' accès a l ´élément e8.

- **Solution** : remplace l élément supprime par un marqueur libre (et non vide) qui indique que la recherche continue mais que l'adjonction est possible..