Hachage

Méthodes de résolution des collisions

D.E ZEGOUR

École Supérieure d'Informatique

ESI

Hachage

Introduction

Rangement de données arrivant dans
un ordre quelconque dans un tableau.

Garder le tableau ordonné

Insertion avec décalages (O(n))

Recherche rapide (dichotomique: O(Log(n))

Garder le tableau non ordonné

Insertion à la fin (O(1))

Recherche lente (séquentielle : O(n))

Pour rechercher rapidement (O(Log(n)), il faudrait ranger lentement (O(n)) . Et
Pour ranger rapidement (O(1)) , il faudrait rechercher lentement (O(n)).

Hachage

Introduction

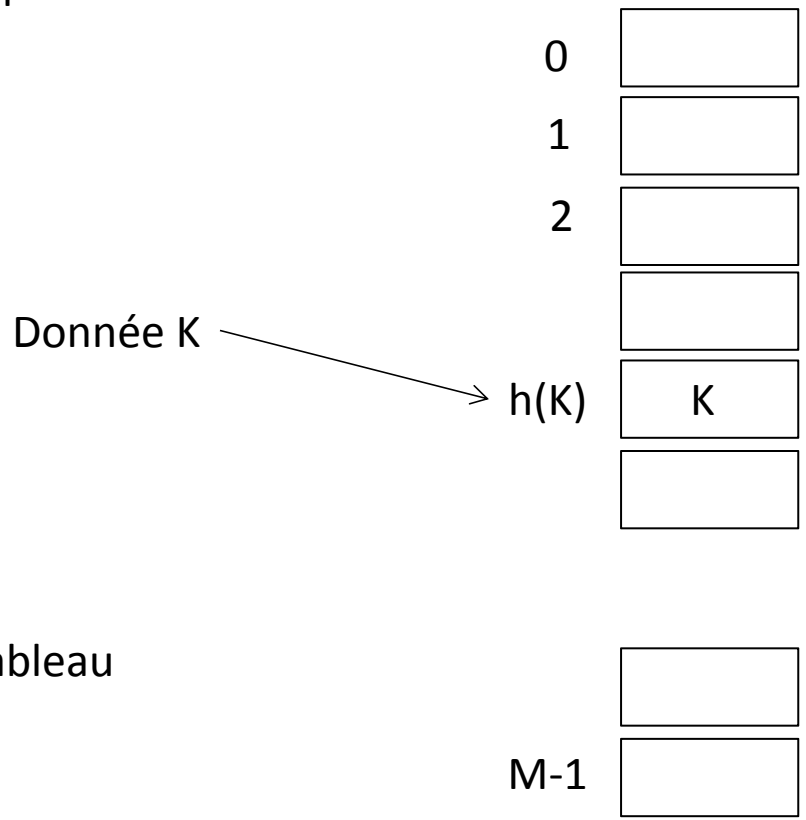
Une troisième possibilité de ranger une donnée dans un tableau

Donnée K rangée à un emplacement calculé par une fonction f (h(K))

Recherche et insertion rapides (O(1))

Problème : Détermination d'une fonction bijective

Bijection : attribue à chaque donnée un nouvel emplacement dans le tableau



Hachage

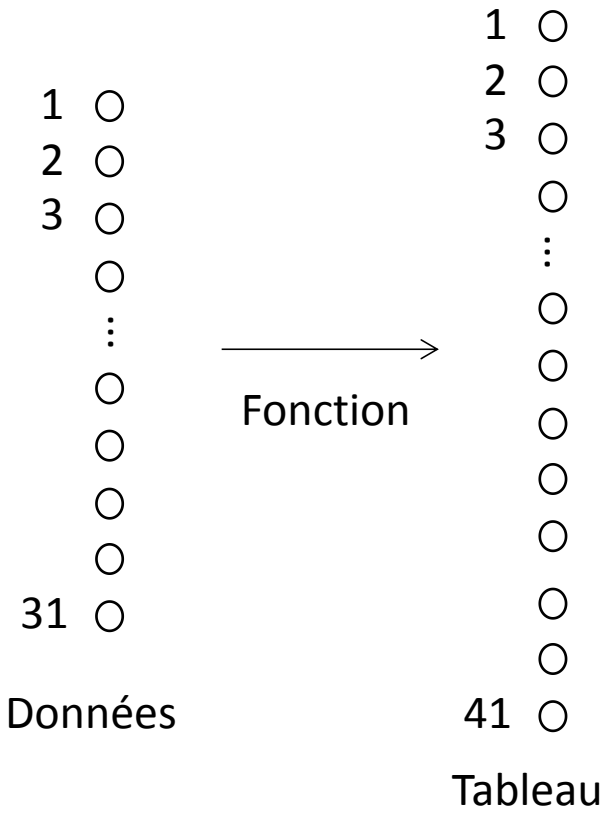
Introduction

Pas facile de découvrir une fonction bijective

Il y a 41^{31} =/ 10^{50} fonctions possibles d'un ensemble de 31 éléments dans un ensemble de 41 éléments (fonctions possibles)

Et seulement 41.40. .11=41! /10 ! = / 10^{43} de ces fonctions ont des valeurs distinctes pour chaque argument (fonctions bijectives)

Environ une fonction sur 10 millions est convenable (10^{43} / 10^{50})



Hachage

Introduction

Pas facile de découvrir une fonction bijective

Paradoxe de l'anniversaire:

<< Si 23 personnes ou plus sont présentes dans une salle, les chances sont bonnes pour que deux d'entre elles aient le même jour et le même mois de naissance>>

Fonctions d'un ensemble de 23 vers un ensemble de 365

Calculer ...

Probabilité = 0.4927

Hachage

Introduction

Ce type de rangement est appelée hachage (en anglais "Hashing") ou technique de rangement dispersé.

Il existe toujours des données distinctes K1, K2, ... pour lesquelles h(K1) = h(K2) = ...
Une telle situation est appelée collision.

En conclusion, pour utiliser une technique de hachage l'utilisateur doit définir :
- une fonction de hachage
- une méthode dite méthode de résolution des collisions.

Hachage

Terminologie

Les données qui ont la même image par la fonction de hachage sont appelées synonymes.
K1, K2, ..., Kn sont des synonymes si h(K1) = h(K2) = ... = h(Kn)

L'adresse primaire d'une donnée c'est l'adresse h(donnée).

Toute donnée qui n'est pas dans son adresse primaire est dite
donnée rangée en débordement (ou tous simplement débordement)
On dit aussi qu'elle est rangée dans une adresse secondaire.

Hachage

Fonctions de hachage

Il s'agit de trouver une fonction h telle que :

$$0 \leq h(K) < M$$

qui réduit au maximum le nombre de collisions.

L'idéal : h bijective.
Le pire des cas : toute donnée est hachée en une même adresse.

Une solution acceptable : certaines données partagent la même adresse(h est surjective).

Hachage

Exemple de fonction de hachage

a) Représenter la donnée sous forme numérique.

b) Concaténer et ajouter (Folk and Add).

c) Diviser par un nombre premier et utiliser le reste comme adresse.

Exemple : transformation de "ALGORITHME"

a) 65 76 71 79 82 73 84 72 77 69

b)

6576 + 7179	= 13755 Mod 20000	= 13755
13755 + 8273	= 22028 Mod 20000	= 2028
2028 + 8472	= 10500 Mod 20000	= 10500
10500 + 7769	= 18269 Mod 20000	= 18269

c) 18269 mod 101 = 89 C'est l'adresse ou sera logée la donnée.

Mod 20000 est utilisé afin qu'il n'y ait pas de débordement.

Hachage

Fonctions de hachage

Méthode de division
h(K)= K mod M

M : Taille du tableau
Bon choix : M nombre premier

Méthode "Middle square"

Elever la donnée au carré et prendre les chiffres du milieu

Exemple:

$(453)^2 = 205209$
h(453) = 52

Taille du tableau : 100

Bons résultats si le nombre au carré n'a pas de zéros.

Méthode "Transformation radix".

Convertir la donnée dans une certaine base de numération et prendre le reste de la division de la donnée transformée par la taille du tableau.

Exemple :

$453 = (382)_{11}$ (en base 11)
382 Mod 100 = 85
h(453) = 85

Taille du tableau : 100

Hachage

Méthodes de résolution des collisions

Plusieurs façons de résoudre les collisions.

Les plus classiques :

1. Essai linéaire
2. Double hachage
3. Chaînage interne
4. Chaînage séparé

Fonction de hachage utilisée par la suite : Modulo

Hachage

Essai linéaire

S'il se produit une collision sur la case k du tableau T[0..M-1], on insère la donnée, si elle n'existe pas, dans la première case libre fournie par la séquence cyclique :

h(k)-1, ..., 0, M-1, M-2, ...h(k)+1

En d'autres termes, on essaie de chercher linéairement une case libre dans la séquence ci-dessus.

D'ou l'appellation de la méthode

Hachage

Essai linéaire

L'insertion des données suivantes avec leurs transformées (entre parenthèses) :a(3), b(2), c(3), d(2), e(1) dans une table T de 6

- Insertion de a(3)
- Insertion de b(2)
- Insertion de c (3)
- Insertion de d (2)
- Insertion de e(1)

0	d
1	c
2	b
3	a
4	
5	e

Hachage

Essai linéaire

Algorithme :
- Recherche une donnée K dans une table T[0..M-1] de M éléments.
- Si K n'est pas trouvée et la table n'est pas pleine, alors K est insérée.

Utilisation d'une variable globale N : nombre de données insérées

Remarquer :la table est remplie quand N = M - 1 et non quand N = M.

L1. [Hacher]
i := h(K) { 0 <= i < M }

L2. [Comparer]
SI T(i) = K, l'algorithme se termine avec succès.
Autrement SI T(i) est vide aller à L4 FSI

L3. [Avancer au prochain]
i := i - 1
SI i < 0 i := i + M ALLERA L2 FSI

L4. [Insérer] {la recherche est sans succès}
SI N = M - 1
  l'algorithme se termine avec débordement
SINON
  N := N + 1
  T(i) := K
FSI

Hachage

Double hachage

Méthode presque analogue à la précédente

Autrement dit, s'il y a collision sur une case k, on calcule un pas p par une autre fonction de hachage et la séquence cyclique à consulter serait

h(k)-p, h(k)-2p, ...

Deux fonctions de hachage sont alors utilisées h(K) et h'(K). D'ou l'appelation de la méthode

Le choix de M est très important. Un mauvais choix peut entraîner une non couverture de l'ensemble des adresses possibles.

On démontre que << lorsque M est un nombre premier et la fonction de hachage est aléatoire, il y a couverture de l'ensemble des adresses>>

Hachage

Double hachage

Insertion de a(3), b(2), c(3), d(2), e(1) et en plus avec h'(c) = 3 ;
h'(d) = 1 ; h'(e) = 3 (h' étant la deuxième fonction de hachage)
dans une table T de 6 éléments

- Insertion de a(3)
- Insertion de b(2)
- Insertion de c (3)
- Insertion de d (2)
- Insertion de e(1)

0	c
1	d
2	b
3	a
4	e
5	

Hachage

Double hachage

Algorithme :
- Recherche une donnée K dans une table T[0..M-1] de M éléments.
- Si K n'est pas trouvée et la table n'est pas pleine, alors K est insérée.

Utilisation d'une variable globale N : nombre de données insérées

Remarquer :la table est remplie quand N = M - 1 et non quand N = M.

D1. [premier hachage]
i := h(K)

D2. [premier test]
SI T(i) vide ALLERA D6 FSI
SI T(i) = K l'algorithme se termine avec succès FSI

D3. [second hachage]
c := h'(K)

D4. [Avancer au prochain]
i := i - c ; SI i < 0 ALORS i := i + M FSI

D5. [Comparer]
SI T(i) est vide ALLERA D6 FSI
SI Donnée(i) = K l'algorithme se termine avec succès.
SINON ALLERA D4 FSI

D6. [Insérer]
SI N = M - 1 "débordement"
SINON N := N + 1 ; T(i) := K FSI

Hachage

Chaînage interne

Ranger les synonymes dans une liste linéaire chaînée représentée dans un tableau. D'ou l'appellation de la méthode.

En cas de collision sur une case k, parcourir la liste commençant en k.

Si la donnée n'est pas trouvée, chercher un emplacement vide dans le tableau. Cet emplacement sera rajouté à la liste.

Stratégie : rechercher une position vide à partir de la fin

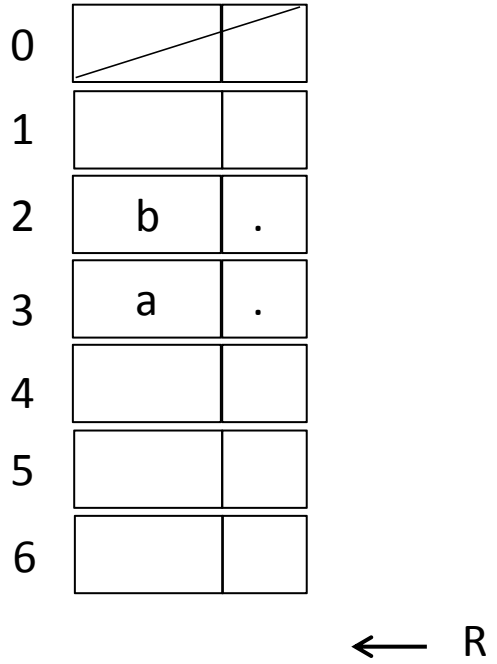
Remarque importante :
Une liste contient des groupes de synonymes.

Hachage

Chaînage interne

Insertion de a(3), b(2), c(3), d(2), e(1), f(6)
dans une table T de 6 éléments

- Insertion de a(3)
- Insertion de b(2)

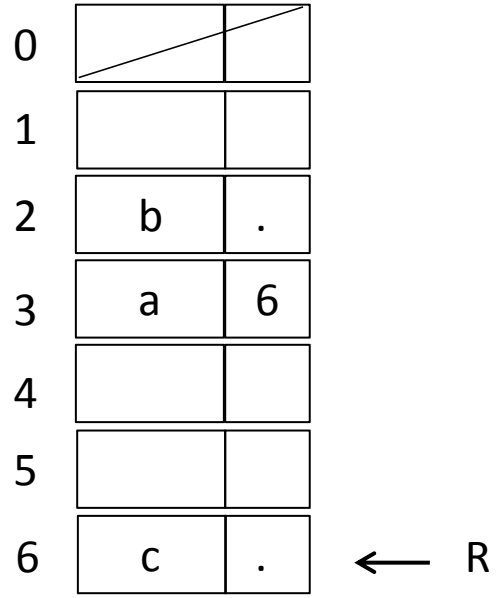


Hachage

Chaînage interne

Insertion de a(3), b(2), c(3), d(2), e(1), f(6)
dans une table T de 6 éléments

- Insertion de a(3)
- Insertion de b(2)
- Insertion de c (3)

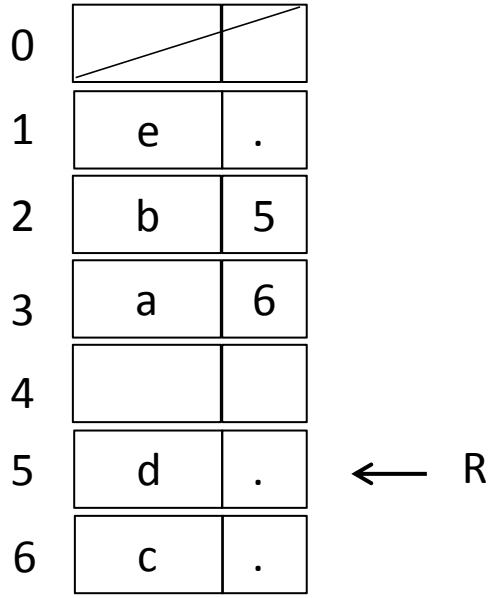


Hachage

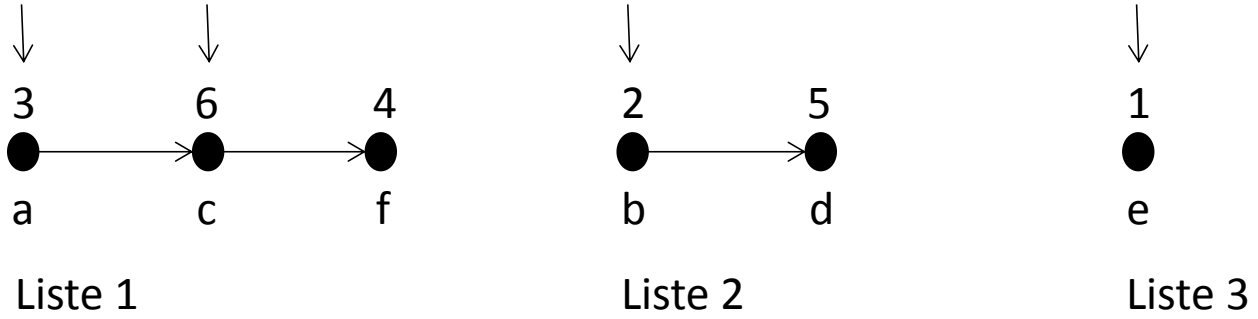
Chaînage interne

Insertion de a(3), b(2), c(3), d(2), e(1), f(6)
dans une table T de 6 éléments

- Insertion de a(3)
- Insertion de b(2)
- Insertion de c (3)
- Insertion de d (2)
- Insertion de e(1)



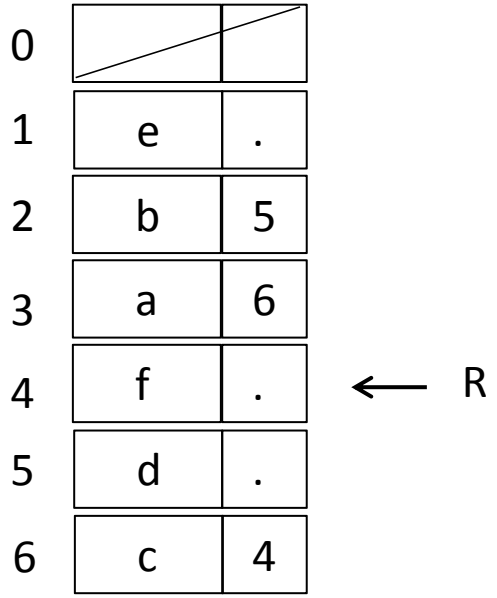
Hachage



Chaînage interne

Insertion de a(3), b(2), c(3), d(2), e(1), f(6)
dans une table T de 6 éléments

- Insertion de a(3)
- Insertion de b(2)
- Insertion de c (3)
- Insertion de d (2)
- Insertion de e(1)
- Insertion de f(6)



Hachage

Chaînage interne

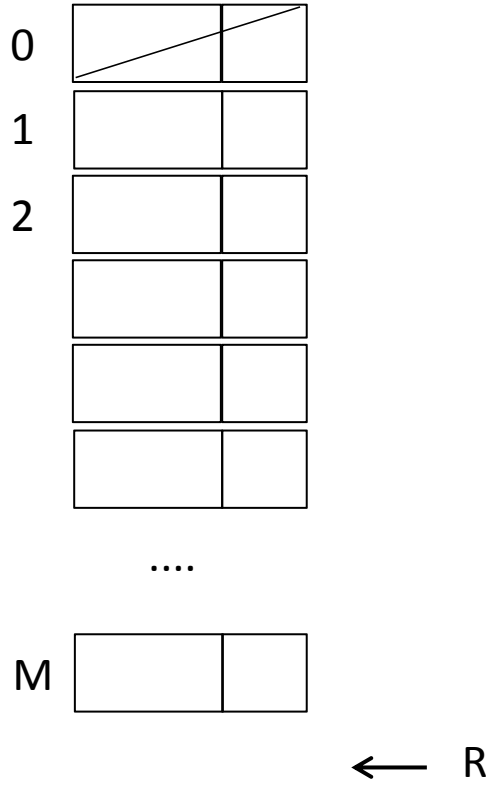
Algorithme :
- Recherche une donnée K dans une table T[0..M]de M éléments.
- Si K n'est pas dans la table et celle-ci n'est pas pleine alors K est insérée.

Element = 2 champs : Donnée et Lien

Une variable globale R est utilisée pour gerer l'espace mémoire utilisé. Quand la table est vide R= M + 1.

Après plusieurs insertions on a : T(j) occupé pour tout j tel que R ≤ j ≤ M.

Par convention T(0) sera toujours vide.



Hachage

Chaînage interne

Algorithme :
- Recherche une donnée K dans une table T[0..M]de M éléments.
- Si K n'est pas dans la table et celle-ci n'est pas pleine alors K est insérée.

Element = 2 champs : Donnée et Lien

Une variable globale R est utilisée pour gerer l'espace mémoire utilisé. Quand la table est vide R= M + 1.

Après plusieurs insertions on a : T(j) occupé pour tout j tel que R ≤ j ≤ M.

Par convention T(0) sera toujours vide.

C1. [Hash]
i := h(K) + 1   { donc 1 <= i <= M }

C2. [Existe-t-il une liste ?]
SI T(i) est vide ALLERA C6 FSI

C3. [Comparer]
SI K = T(i).Donnee, l'algorithme se termine avec succès FSI

C4. [Avancer au prochain]
SI T(i).Lien <> 0 alors i := LIEN(i)   ALLERA C3 FSI

C5. [Trouver le noeud vide]
Décrémenter R une ou plusieurs fois jusqu'à ce que T(R) soit vide.
SI R = 0 l'algorithme se termine avec débordement.
Autrement faire : T(i).Lien := R; i := R

C6. [Insérer]
T(i).Donnee := K ; T(i).Lien := 0

Hachage

Chaînage séparé

Les synonymes sont rangées dans une liste linéaire chaînée externe au tableau. D'ou l'appellation de la méthode.

Une liste ne contient qu'un seul groupe de synonymes.

Quand une collision se produit sur une case k, on parcourt la liste commençant en k.
Si la donnée n'est pas trouvée, on insère la donnée dans la liste (au début ou à la fin)

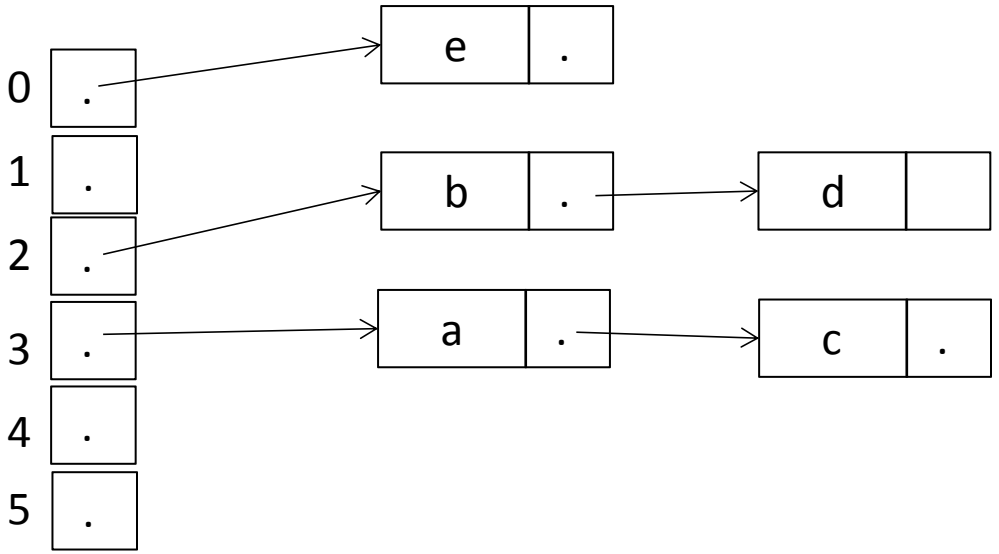
On peut ranger plus d'éléments que la taille du tableau

Hachage

Chaînage séparé

Insertion de a(3), b(2), c(3), d(2), e(1) dans une table T de 6 éléments

- Insertion de a(3)
- Insertion de b(2)
- Insertion de c (3)
- Insertion de d (2)
- Insertion de e(1)



Hachage

Chaînage séparé

Algorithme:
- Recherche une donnée K dans une table de M éléments. Si K n'est pas dans la liste correspondante, elle y est insérée.

Un élément du tableau contient un seul champ : Tête de liste

Initialement T(i) := Nil
pour tout i dans l'intervalle[0..M-1]

S1. [Hacher]
i := h(K)

S2. [Existe-t-il une liste ?]
SI T(i) est vide ALLERA S5
P := T(i)

S3. [Comparer]
Si K = Valeur(P) l'algorithme se termine avec succès FSI

S4. [Avancer au prochain]
Si Suivant(P) <> Nil : P := Suivant(P); ALLERA S3 FSI

S5. [Insérer au début de la liste]
Allouer (Q); Aff_val(Q, K); Aff_adr(Q, T(i))
T(i) := Q

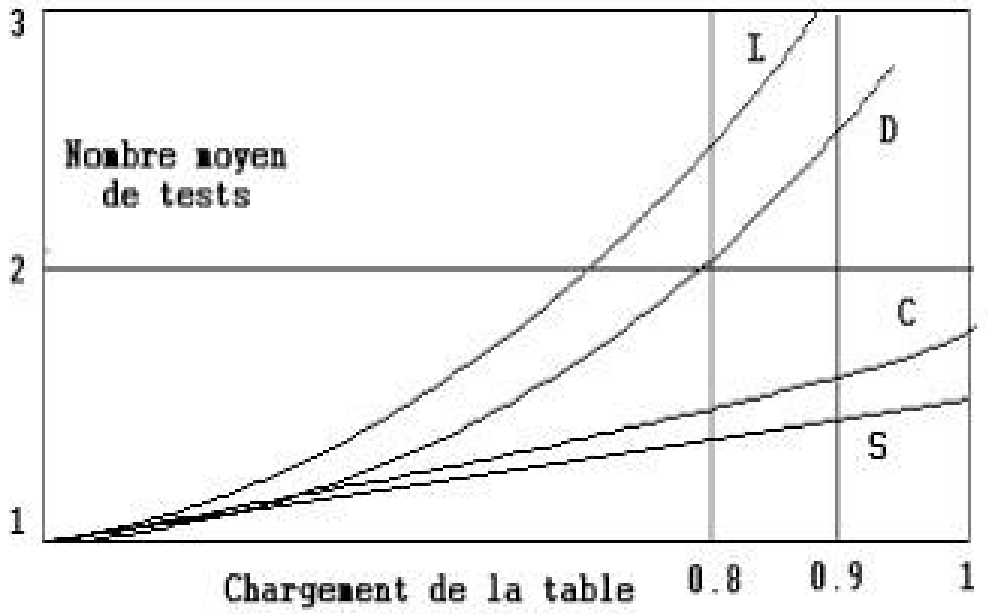
Hachage

Comparaison entre les différentes méthodes

Courbes des nombres moyens de tests pour une recherche par rapport au chargement de la table

(Chargement = N/M , N :nombre de données insérées et M la taille de la table

L désigne l'essai linéaire, D le double hachage, C le chaînage interne et S le chaînage séparé.



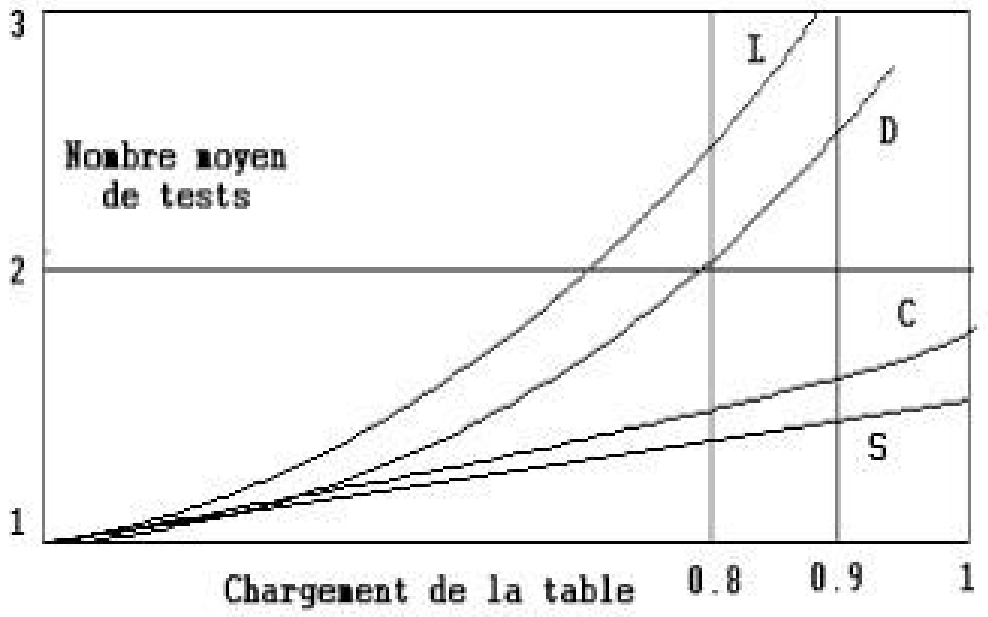
Hachage

Comparaison entre les différentes méthodes

S > C > D > L

Les méthodes de hachage ont de très bonnes performances.

Pour un chargement de 70-80% --> O(1).



Hachage

Synthèse

Généralisation: plus d'une donnée par case du tableau

Utilisation:
- Insertion avec contrôle(fixer un seuil)
- Bon compromis : chargement de 70 à 80 %
- Rehachage

Avantage : Accès rapide à l'information (O(1))

Inconvénients :
- Absence de l'ordre
- Limitation aux données statiques

Application : dictionnaire