

R3.02 : Développement efficace

Table de Hachage

J-F. Kamp

Septembre 2024

Notion de table

Ensemble formé d'un nombre variable, éventuellement nul, de données sur lequel on peut effectuer les opérations suivantes :

- Ajout d'une nouvelle donnée (tuple en BDD)
- Recherche d'une donnée par sa clé
- Consultation de l'info associée à une clé
- Suppression du tuple d'après sa clé
- Test si la table est vide

Notion de table

On voudrait être efficace :

- Pour l'ajout : être aussi rapide qu'une liste chaînée ($O(1)$)
- Pour la recherche : être aussi rapide qu'un tableau indicé (si on connaît l'indice c'est $O(1)$)
- Suppression : c'est comme une recherche

Exemple de table

Un annuaire téléphonique :

- On suppose la clé = nom du correspondant
- Pour la recherche du numéro : on donne le nom (clé) et on retrouve le numéro
- Solution : utiliser un tableau indicé par les noms. En donnant le nom on accède immédiatement à la bonne case ($O(1)$).

Exemple de table

Un annuaire téléphonique :

- Solution : utiliser un tableau indicé par les noms. En donnant le nom (clé) on accède immédiatement à la bonne case ($O(1)$)
- Problème : un nom (chaîne de caractères) n'est pas un indice
- Solution : utiliser une fonction de hachage h qui converti la clé en un indice de tableau (un entier compris entre 0 et $n-1$)
$$indice = h(clé)$$

Exemple de table

Un annuaire téléphonique :

Exemple de fonction de hachage :
utiliser le code ASCII de chaque
caractère de la chaîne et additionner

Soit T le tableau des caractères de la
chaîne de longueur L :

$$h(\text{clé}) = \text{ASCII}(T[0]) + \dots + \text{ASCII}(T[L-1])$$

clé = Dupont

$$h(\text{clé}) = 68 + 117 + 112 + 111 + 110 + 116$$

$$h(\text{clé}) = 634$$

En pratique pour une table de hachage

- Il faut un tableau indicé

0 1 2 3 ... n - 1

?	?	O1	?	...	?
---	---	----	---	-----	---

- Pour insérer / rechercher une information (un objet O1 = numéro de tél. par ex.) associée à une clé (nom) on calcule l'indice

indice = h (clé) (= 2 dans l'exemple)

Cet indice sert A LA FOIS pour l'ajout dans le tableau ET la recherche ($O(1)$).

Problèmes de la table de hachage

1. Le tableau indicé a une taille de départ fixée (n)

0 1 2 3 4 ... $n - 1$

?	?	?	?	01	?	?
---	---	---	---	----	---	---

=> pour être certain que l'indice calculé ($h(\text{clé})$) soit compris entre 0 et $(n - 1)$ il faut prendre le modulo

$$\text{indice} = h(\text{clé}) \bmod n$$

exemple pour un tableau $n = 10$:

clé = Dupont

$$\text{indice} = 634 \bmod 10 = 4$$

Problèmes de la table de hachage

2. Rien ne garantit que l'indice calculé par hachage ne corresponde pas à une case DÉJÀ occupée = la COLLISION

0 1 2 3 4 ... n - 1

?	?	?	?	O1	?	?
---	---	---	---	----	---	---

Exemple : insérer un objet O2 (un autre numéro de tél.) associée à la clé Dupot

$$h(\text{clé}) = 68 + 117 + 112 + 111 + 116$$

$$h(\text{clé}) = 524$$

En supposant $n = 10$:

$$\text{indice} = h(\text{clé}) \bmod n = 4$$

Problèmes de la table de hachage

3. Trouver la fonction de hachage qui exploite au mieux TOUTES les cases du tableau

0 1 2 3 4 ... n - 1

O10	O2	O4	O12	O1	...	O3
-----	----	----	-----	----	-----	----

Ce n'est pas un problème simple.

Par exemple : 95% de chance d'avoir une collision dans un tableau de 1 million de cases avant d'avoir ajouter 2500 tuples.

Il existe d'autres fonctions de hachage bien + performantes que le code ASCII.

Solutions au problème de collision

Règle de départ

Démarrer avec un tableau de taille n
 $= 2 \times$ le nombre de tuples à insérer
(densité de 50%).

Arrondir la taille n au nombre **premier**
le plus proche (pour éviter d'avoir trop
souvent l'indice zéro).

Si la densité approche 75%, doubler
la taille du tableau et retour à une
densité de 37%.

Solutions au problème de collision

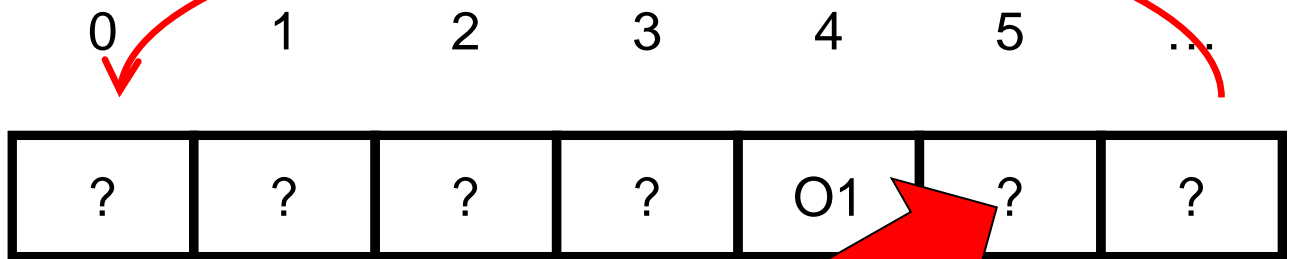
Solution1

Adressage ouvert simple.

1. Calcul de l'indice avec la fonction de hachage
2. Si l'emplacement à l'indice calculé est libre alors placer le tuple à cet endroit
3. Sinon chercher la première place vide et y placer le tuple. Cette recherche se fait en balayant le tableau de façon circulaire.

Solutions au problème de collision

Solution1 - Exemple



Insérer un objet O2 (un autre numéro de tél.) associée à la clé Dupot

$$h(\text{clé}) = 68 + 117 + 112 + 111 + 116$$

$$h(\text{clé}) = 524$$

En supposant $n = 10$:

$$\text{indice} = h(\text{ clé }) \bmod n = 4$$

==> prochaine place libre est en
 $\text{indice} + 1 = 5$

Solutions au problème de collision

Solution2

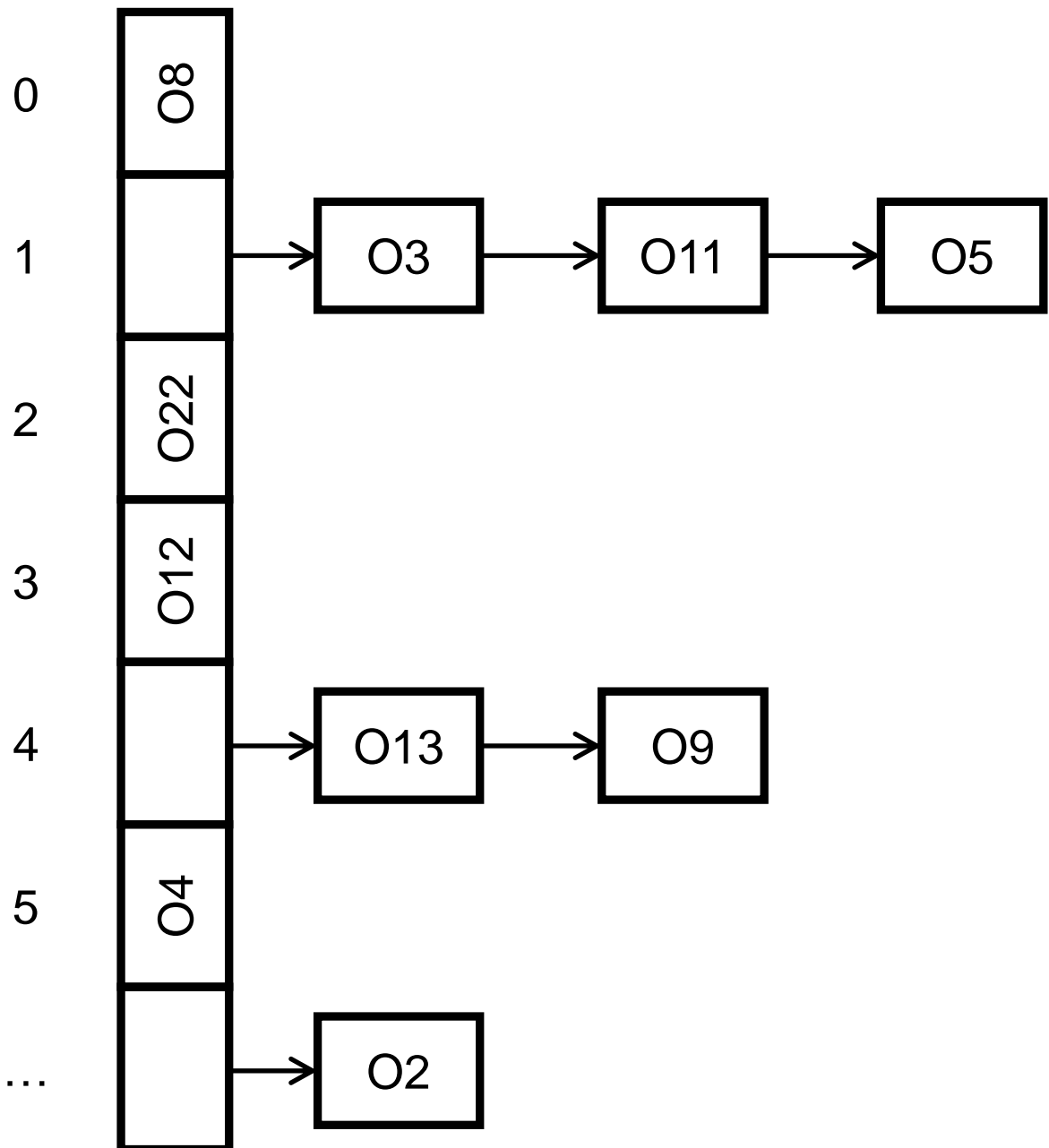
Chaînage des collisions.

Si l'emplacement à l'indice calculé est occupé alors placer le tuple au début d'une liste chaînée qui démarre à cet emplacement là.

Les tuples qui ont le même indice sont placés dans la MÊME liste (chaînée).

Solutions au problème de collision

Solution2 - Exemple



Solution2 - Caractéristiques

- ajout en $O(1)$ car on calcule un indice puis on insère en tête de liste (uniquement si collision)
- recherche dans le pire des cas en $O(m)$ (m = nombre de tuples) car si $h(clé)$ renvoie toujours le même indice \Rightarrow on se ramène à une « simple » liste chaînée
- cette solution retarde le moment où la taille du tableau doit être multipliée (par 2)

La table de hachage idéale

Dans l'idéal, il faut garder l'efficacité en $O(1)$ pour l'ajout et la recherche (la suppression à la même efficacité que la recherche) :

- retarder le doublement du tableau pour mieux utiliser la mémoire
- utiliser une fonction de hachage qui réparti uniformément les tuples dans le tableau
- utiliser le chainage lorsqu'il y a collision
- lorsqu'on arrive à 100% de remplissage du tableau ou lorsque les chaînes deviennent trop longues => on double le tableau