Algorithmique

Cristina Sirangelo, ENS-Cachan

Préparation à l'option Informatique
de l'agrégation de mathématiques

Plan

1. Analyse et complexité des algorithmes (S.Haddad)
2. Types abstraits et structures de données

   Dictionnaires

      Implémentation par table de hachage
3. Algorithmes de tri
4. Techniques classiques de conception d'algorithmes
5. Algorithmes de graphes

Dictionnaires et tables de hachage

Implémentations vues des dictionnaires:

	Complexité de la recherche	
	cas moyen	cas pire
Tableau* - Liste	O(n)	O(n)
Arbres binaires de recherche	O(log n)	O(n)
Arbres équilibrés	O(log n)	O(log n)

* Sauf dans le cas de tableau trié

Implémentation par table de hachage

Objectif: O(1) dans le cas moyen et, sous certaines hypothèses, aussi dans le cas pire

Tables de hachage: une généralisation des *tableaux associatifs*

Tableaux associatifs

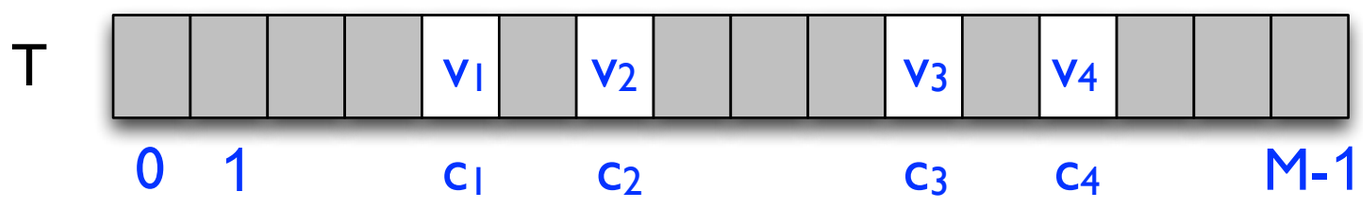
Une implémentation triviale des dictionnaires
complexité O(1) - cas pire des opérations de recherche/insertion/suppression

Hypothèse: Tclef = {0, .., M-1}
(ou, plus généralement, chaque clef peut être interprétée comme un entier dans {0, .., M-1}; *ex. l'entier correspondant à la représentation binaire de la clef*)

Données:
Un dictionnaire sur (Tclef, Tval): un tableau d'éléments de type Tval indexé par Tclef

Opérations:

INSERER (Dictionnaire T, Tclef c, Tval v)	T[c] ← v
CHERCHER (Dictionnaire T, Tclef c): Tval	return T[c]; *T[c] peut être UNSET*
SUPPRIMER (Dictionnaire T, Tclef c)	T[c] ← NIL



Souvent:

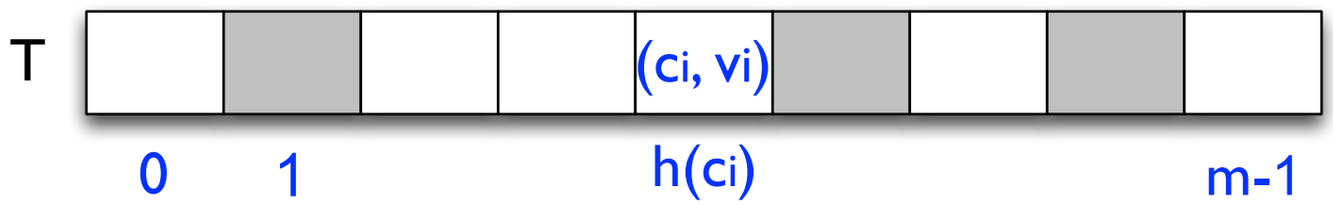
- Tclef est "grand" ⇒ mémoriser un tableau de dimension | Tclef | n'est pas pratique/ possible
- Tclef >> taille du dictionnaire ⇒ gaspillage d'espace

Tables de hachage

Implémentation d'un dictionnaire par table de hachage (idée)

- Dictionnaire: un tableau T[0..m-1] de couples (clef, valeur)
- l'index de T où l'élément de clef c est mémorisé n'est pas c, mais il est *calculé* à partir de c, par une fonction:

h: Tclef → {0,..,m-1} fonction de hachage



h(c) calculable en temps constant

- Pour avoir m<<|Tclef| (m de l'ordre de la taille du dictionnaire)

  ⇒ h en générale non-injective ⇒ collisions possibles (h(ci)= h(cj) pour ci ≠ cj)

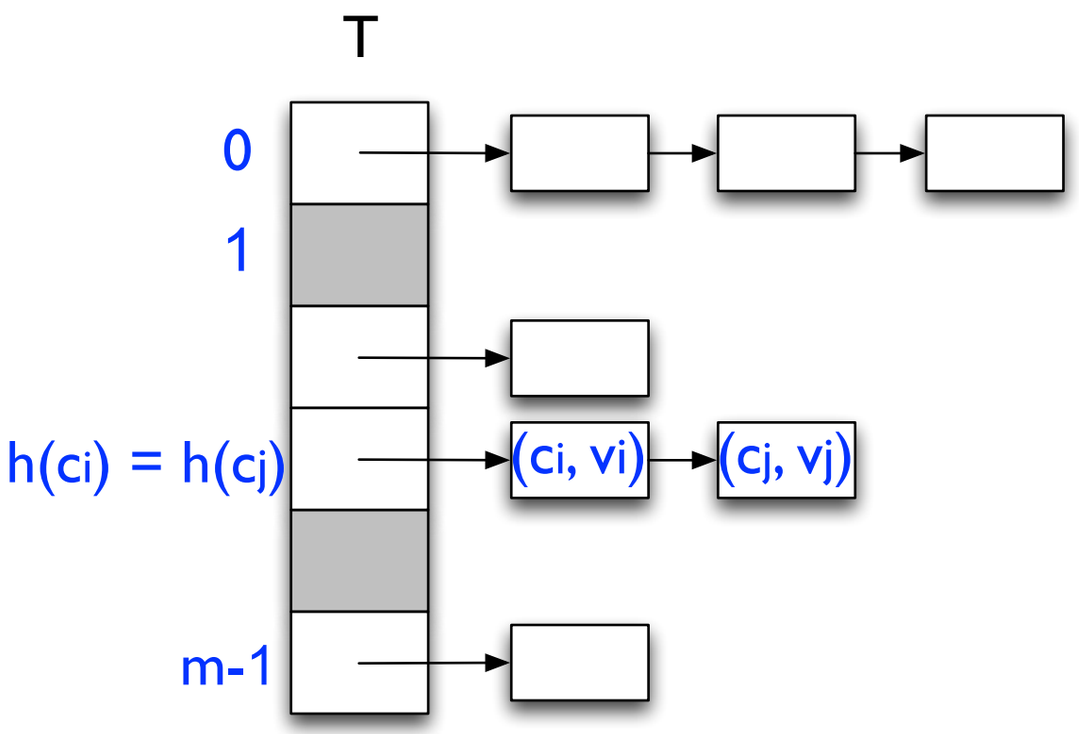
Résolution des collisions - deux techniques principales:

chaînage (aussi appelé *chaînage séparé*)

adressage ouvert

Chaînage

- Chaque élément du tableau T contient un pointeur vers une liste de couples (clef, valeur)
- Les éléments dont les clefs ont valeur de hachage j sont placé dans la liste pointée par T[j]



Listes doublement chainées
pour plus d'efficacité

- **Recherche/Suppression** d'une clef c : parcours de la liste T[h(c)]
- **Insertion** de (c,v): insertion en tête de la liste T[h(c)]

Chaînage

Analyse du coût - *par simplicité supposer absence de données satellites (champ valeur)*

Cas pire

Insertion: O(1)

Recherche/Suppression: proportionnel à la longueur des listes - O(n)

Cas moyen

- La longueur des listes dépend de la "qualité" de la fonction de hachage
- Une "bonne" fonction de hachage minimise les collisions et garanti coût O(1) en moyenne.

  Formellement:

  - bonnes propriétés de la fonction de hachage (par rapport à la distribution des clefs) ↔ hypothèse de *hachage uniforme simple*

  - On démontre:
    Sous l'hypothèse de hachage uniforme simple, le coût moyen de la recherche (suppression) dans une table de hachage à n éléments où les collisions sont résolues par chaînage et $m=\Omega(n)$ est $O(1)$ (ex. $m \geq n/3$)

Chaînage

Analyse du cas moyen

Hypothèse de hachage uniforme simple:

Informellement: *la probabilité de hacher une clef vers chaque position de la table est la même, indépendamment des valeurs de hachage des autres clefs*

Plus formellement (Pour une fonction de hachage donnée h: Tclef $\rightarrow$ {0,..., m-1}):

Hypothèse de hachage uniforme simple pour h sur une séquence aléatoire de n clefs distinctes (avec une certaine distribution de probabilité - pas nécessairement uniforme):

1) Pr(Hi = j) = 1/m pour tout j $\in$ {0, ..., m-1}, pour tout Hi

2) Hi et Hj sont indépendantes, pour tout i$\neq$j

Où **Hi, i=1..n** est la variable aléatoire qui représente la valeur de hachage de la i-eme clef

Chaînage - recherche infructueuse

Analyse du cas moyen

facteur de remplissage d'une table de hachage de taille *m* avec *n* clefs $\alpha = \frac{n}{m}$

Table de hachage de n clefs construite aléatoirement: table de hachage construite par insertions successives d'une séquence aléatoire de *n* clefs distinctes

Théorème
Le coût moyen de la recherche infructueuse d'une clef aléatoire dans une table de hachage construite aléatoirement où les collisions sont résolues par chaînage, est $\Theta(1+\alpha)$
sous l'hypothèse de hachage uniforme simple pour la fonction de hachage sur toutes les clefs

Preuve. Variables aléatoires:

Kj : La j-ème clef insérée dans la table, j=1,..n
K: La clef à chercher (K est absent de la table)
C: Coût de la recherche de K
Li= Longueur de la liste T[i], i = 0, .., m-1

$$E[C] = \sum_{i=0}^{m-1} E[C \mid h(K) = i] \cdot Pr[h(K) = i]$$

Chaînage - recherche infructueuse

Sachant h(K) = i, le coût C = 1+ Li (coût O(1): calcul du valeur de hachage)

- Hypothèse de hachage uniforme simple ⇒ Pr[h(K)=i] = 1/m et h(K) est indépendant de Li

$$\Rightarrow \qquad E[C] = \frac{1}{m} \sum_{i=0}^{m-1} 1 + E[Li]$$

- Longueur moyenne d'une liste de la table:

Soit Xij la variable indicatrice de l'événement h(Kj) = i (Xij = 1 si h(Kj) = i, Xij = 0 sinon) :

$$Li = \sum_{j=1}^{n} Xij \quad \Rightarrow \quad E[Li] = \sum_{j=1}^{n} Pr[h[Kj] = i] = \frac{n}{m}$$

par hypothèse de hachage uniforme simple.

$$E[C] = 1 + E[Li] = 1 + \frac{n}{m}$$

□

Chaînage - recherche fructueuse

Théorème
Le coût moyen de la recherche fructueuse d'une clef aléatoire dans une table de hachage construite aléatoirement où les collisions sont résolues par chaînage, est $\Theta(1+\alpha)$
sous les hypothèses:

1) hachage uniforme simple pour la fonction de hachage sur les n clefs de la table

2) clef à chercher distribuée uniformément parmi les n clefs de la table

Preuve. Variables aléatoires:

Ki : La i-eme clef insérée dans la table, i = 1,..n
K: La clef à chercher (K est dans la table)
C: Coût de la recherche de K

$$E[C] = \sum_{i=1}^{n} E[C \mid K = Ki] \cdot Pr[K = Ki] = \frac{1}{n} \sum_{i=1}^{n} E[C \mid K = Ki]$$

Chaînage - recherche fructueuse

Sachant K=Ki :

C= 1 *(calcul du valeur de hachage)*

\+ nombre de clefs Kj, j> i *(clefs insérées après Ki)* avec h(Kj) = h(Ki)

Soit Xij la variable indicatrice de l'événement h(Ki) = h(Kj):

$$C = 1 + \sum_{j=i+1}^{n} Xij$$

$$E[C|K = Ki] = 1 + \sum_{j=i+1}^{n} E[Xij]$$

Par l'hypothèse de hachage uniforme simple (uniformité *et indépendance* de h(Ki) et h(Kj))

$$E[Xij] = Pr[h(Ki) = h(Kj)] = \frac{1}{m}$$

Alors $E[C|K = Ki] = 1 + \frac{(n-i)}{m}$ et

$$E[C] = \frac{1}{n} \sum_{i=1}^{n} E[C \mid K = Ki] = \frac{1}{nm} \sum_{i=1}^{n} (m + n - i)$$

Chaînage - recherche fructueuse

$$E[C] = \frac{1}{nm} \sum_{i=1}^{n} (m + n - i)$$

$$= 1 + \frac{1}{nm} (n^2 - \sum_{i=1}^{n} i)$$

$$= 1 + \frac{1}{nm} (n^2 - \frac{n(n+1)}{2})$$

$$= 1 + \frac{n-1}{2m}$$

$$= 1 + \frac{\alpha}{2} - \frac{1}{2m} = \Theta(1 + \alpha)$$

□

Chaînage

Analyse du cas moyen

Sous l'hypothèse de hachage uniforme simple:

Insertion $O(1)$

Recherche fructueuse/infructueuse: $\Theta(1+\alpha)$

Suppression fructueuse/infructueuse (même coût que la recherche) : $\Theta(1+\alpha)$

Si la taille $m \geq c\,n$ pour une constante $c > 0$, α est $O(1) \Rightarrow$

Toutes les opérations de dictionnaire en temps $O(1)$ en moyenne

Exemple. $m \geq n/3$

nombre moyen d'éléments examinés dans un recherche (infructueuse) : 3

Chaînage

Choix de la fonction de hachage

- Une "bonne" fonction de hachage doit satisfaire (approximativement) l'hypothèse de hachage uniforme simple sur la distribution des clefs
- Distribution des clefs difficile à prévoir
- Fonctions de hachage souvent basées sur des heuristiques. Exemples:
  - méthode de la *division*
  - méthode de la *multiplication*
- Fonctions de hachage randomisés (*hachage universel*)
  - la fonction de hachage est choisie aléatoirement ;
  - la distribution des valeurs de hachage est déterminée par la distribution des fonctions de hachage, et non pas par la distributions des clefs;
  - peuvent garantir coût O(1) en moyenne *pour chaque input*;
  - évitent la vulnérabilité des fonctions de hachage fixes
    (un adversaire qui connaît la fonction de hachage fixe peut choisir n clefs avec la même valeur de hachage)

Chaînage - fonctions de hachage

Assomption: Les clefs sont des nombres naturels (ou peuvent être interprétées comme tels)

Méthode da la division

$$h(c) = c \bmod m$$

- calculé efficacement (une seule division);
- *m* ne doit pas être une puissance de 2
  - $m=2^p \Rightarrow$ h(c) = les p dernier bits de la représentation binaire de c : favorise les collisions
- une bonne heuristique: *m* premier, et loin d'une puissance de 2

Chaînage - fonctions de hachage

Méthode de la multiplication

$$h(c) = \lfloor \frac{m}{W} \; c \; s \bmod W \rfloor \qquad s, W: \text{ deux entiers } s < W$$

- Valeur de *m* pas critique
- Typiquement
  - $m = 2^p$, $W = 2^w$ où w est le nombre de bits d'un mot (ex. w=32), $p < w$, $0 < s < W$
- Facile à implémenter avec ces valeurs (en supposant c représentable sur w bits)
  - calculer c s sur 2w bits
  - extraire les derniers w bits ($c \; s \bmod W$)
  - extraire les premier p bits de ceux-là ($\lfloor \frac{c \; s \bmod W}{2^{w-p}} \rfloor = h(c)$)
  - choix de s : tel que $s/W \approx (\sqrt{5} - 1)/2$ [Knuth]

Chaînage - hachage universel

Hachage universel

Soit $\mathcal{H}$ une collection de fonctions de hachage Tclef → {0,...m-1}

et soit H une fonction tirée aléatoirement avec distribution uniforme sur $\mathcal{H}$

Définition. $\mathcal{H}$ *universelle*:

Pour tout c, c' ∈ Tclef, c ≠ c' Pr[H(c) = H(c')] ≤ 1/m

- i.e comme si H(c) et H(c') étaient distribués uniformément sur {0..m-1} et indépendants
- analogue de l'hypothèse de hachage uniforme simple

Théorème

Soit $\mathcal{H}$ une famille universelle, et soit H tirée aléatoirement avec distribution uniforme parmi les fonctions de $\mathcal{H}$
Soit $c_1, ..c_n$ une séquence de n clefs distinctes dans Tclef et soit c ∈ Tclef
Soit T une table de hachage de taille m construite par insertion de $c_1,..., c_n$ en utilisant la fonction de hachage H.

Le coût moyen de la recherche/ suppression/ insertion de c dans T est O(1+ α)

Chaînage - hachage universel

Preuve

Insertion: O(1)

Recherche/Suppression:

Variables aléatoires C: coût de la recherche/suppression de c

Y: nombre de clefs parmi cj, j=1..n, telles que cj ≠ c et H(cj) = H(c),

1) c absente de la table ⇒ C=1+Y

2) c dans la table ⇒ C ≤ 1 + (Y+1)

Soit Xj la variable indicatrice de l'événement H(cj) = H(c): $Y = \sum_{cj \neq c} Xj$

$E[Y] = \sum_{cj \neq c} Pr[H(cj) = H(c)] \leq \sum_{cj \neq c} \frac{1}{m}$ ($\mathcal{H}$ universelle) $E[Y] \leq \sum_{i=1}^{n} \frac{1}{m} = \frac{n}{m}$

$\Rightarrow E[C] = O(1 + \alpha)$ □

Chaînage - hachage universel

Une famille universelle de fonctions de hachage:

Soit $p > 1$ un nombre premier tel que pour tout $c \in$ Tclef $c < p$

Soit a, b deux entiers $1 \leq a \leq p-1$ $0 \leq b \leq p-1$

$\mathcal{H} = \{ h_{ab} \mid 1 \leq a \leq p-1 \quad 0 \leq b \leq p-1 \}$ où $h_{ab}(c) = ((a \cdot c + b) \bmod p) \bmod m$

$|\mathcal{H}| = p(p-1)$

Théorème $\mathcal{H}$ est universelle

Preuve

Fonction choisie uniformément dans $\mathcal{H} \leftrightarrow$ (a,b) choisi uniformément dans $\{1, .., p-1 \} \times \{0, ..p-1\}$

Soit $c, k \in$ Tclef, $c \neq k$

on montre que le couple (r , s)

$r = (a\ k + b) \bmod p$

$s = (a\ c + b) \bmod p$

a une distribution uniforme sur toutes les couples $\{ (i,j) \in \{0, .., p-1\}^2 , i \neq j \}$

Chaînage - hachage universel

Il suffit de démontrer que la fonction qui transforme (a, b) en (r, s) est une bijection

f: {1, .., p-1 } × {0, ..p-1} → { (i,j) ∈ {0, .., p-1}^{2} | i≠ j } *(Exercice)*

(Probabilité de collision entre k et c:) Pr[h_{ab} (k) = h_{ab}(c)] = Pr[r = s (mod m)]

Pour tous r ∈ {0, .., p-1},

- au plus (p-1) valeurs de s ≠ r
- au plus (p-1)/ m valeurs de s ≠ r satisfont r = s (mod m)

$$\Pr[r = s \pmod m] \leq \frac{p(p-1)}{m\ p(p-1)} = \frac{1}{m}$$

Alors pour tout c, k ∈ Tclef, c≠ k : Pr[h_{ab} (k) = h_{ab}(c)] ≤ 1/m

⇒ $\mathcal{H}$ est universelle

□

Adressage ouvert

Résolution des collisions:

- Toutes les clefs sont stockées dans la table (pas de listes externes)
- Insertion d'une clef: plusieurs alvéoles sont *sondées*, jusqu'à trouver une alvéole libre (ou table pleine)
  - La suite d'alvéoles à sonder est calculée à partir de la clef (les pointeurs sont évités)
- Recherche d'une clef: la même suite d'alvéoles utilisée pour l'insertion est sondée
- Absence de pointeurs ⇒ utilisation de la mémoire plus efficace que dans le chaînage⇒ tables plus larges ⇒ potentiellement moins de conflits
- La taille de la table borne le nombre de clefs: $\alpha \leq 1$

Adressage ouvert

Soit T une table de taille *m*, et Tclef le domaine des clefs.

Fonction de hachage étendue: associe à chaque clef une *séquence* de valeurs de hachage (la séquence d'alvéoles à sonder)

$$h: Tclef \times \{ 0,.., m-1 \} \rightarrow \{ 0, ..., m-1 \}$$

Pour $c \in Tclef$, $< h(c, 0), h(c, 1), ..., h(c, m-1) >$ est appelée *séquence de sondage* pour c

Restriction sur h: la séquence de sondage doit être une permutation de < 0, .., m-1 >

(toutes les alvéoles de la tables doivent pouvoir être sondées)

Insertion d'une clef c. Sonder la séquence d'alvéoles T[h(c, i)], i =0, ...m-1 jusqu'à ce que T[h(c, i)] est libre:

```
i← 0   première sondage
while (i < m  et  T[ h(c, i) ] occupé ) do  i ← i+1 prochain sondage
if ( i=m ) then retourner table pleine
T[ h(c, i) ] ← c
```

Adressage ouvert

Suppression d'une clef c (dont on connait l'alvéole p). T[p] ← DELETE

DELETE est une valeur spéciale, considérée comme non-occupé dans l'insertion

Recherche d'une clef c. Sonder la séquence d'alvéoles T[h(c, i)] , i =0, ...m-1 jusqu'à ce que T[h(c, i)]= c ou T[h(c, i)]= UNSET

i← 0 *première sondage*

while (i < m et T[h(c, i)] ≠ c T[h(c, i)] ≠ UNSET) **do** i ← i+1 *prochain sondage*

if (i=m ou T[h(c, i)] = UNSET) **then** *retourner clef absente*

return c

Correction de l'algorithme de recherche:

Si c a été insérée dans la table:

- c est dans une alvéole p de la séquence de sondage de c
- toutes les alvéoles qui précèdent p dans la séquence étaient occupées au moment de l'insertion ⇒ occupées ou DELETE (non-UNSET) au moment de la recherche

Adressage ouvert

Analyse du cas moyen

Hypothèse de hachage uniforme:

Informellement: *la probabilité que la séquence de sondage d'une clef soit une quelconque des m! permutations des alvéoles de la table est la même, indépendamment des séquences de sondage des autres clefs*

Plus formellement (Pour une fonction de hachage donnée h: Tclef × { 0,.., m-1 } → { 0, ...,m-1 }):

Hypothèse de hachage uniforme pour h sur une séquence aléatoire de n clefs distinctes (avec une certaine distribution de probabilité - pas nécessairement uniforme):

1) $Pr(Si = \sigma) = 1/m!$ pour toute permutation σ de <0, .., m-1> et pour tout Si

2) Si et Sj sont indépendantes, pour tout $i \neq j$

Où **Si, i=1..n** est la variable aléatoire qui représente la séquence de sondage de la i-ème clef

Adressage ouvert - recherche infructueuse

Analyse du cas moyen

Théorème
Le nombre moyen de sondages pour la recherche infructueuse d'une clef aléatoire dans une table de hachage à adressage ouvert construite aléatoirement avec $\alpha = n/m < 1$ est au plus $1 / (1-\alpha)$ *sous l'hypothèse de hachage uniforme pour la fonction de hachage sur toutes les clefs*

Remarque: La table est construite par insertion d'une séquence aléatoire de n clefs distinctes - pas de suppressions. En cas de suppression l'analyse est plus complexe (le coût de la recherche ne dépend pas uniquement du facteur de remplissage)

Preuve.

Variables aléatoires:

S: séquence de sondage de la clef à chercher

Si: premières i alvéoles de S , i=1..m

Ω: ensemble des n alvéoles occupées de la table

C: nombre de sondages pour la recherche de la clef (coût de la recherche)

Adressage ouvert - recherche infructueuse

- Soit Ai l'événement: *"les alvéoles de Si sont occupées"* Alors :

$$\Pr[\, C \geq i \,] = \Pr[\, A_{i-1} \,] \qquad i = 2,\ldots, m$$

$$\Pr[\, C \geq 1 \,] = 1 \qquad \textit{(au moins un sondage est nécessaire)}$$

$$\Rightarrow \Pr[\, C \geq n+2 \,] = 0 :$$

$$\Pr[\, C \geq n+2 \,] = \begin{cases} \Pr[\, A_{n+1} \,] = 0 & \text{si } n+2 \leq m \quad \textit{(au plus n alvéoles sont occupées)} \\ 0 & \text{si } n+2 > m \end{cases}$$

- $\Pr[\, A_i \,]$, $i=1..n$:

Ai est l'événement $Si \in \Omega^i$ $\Rightarrow$ Pour un ensemble quelconque $\omega \subseteq \{ 0, .., m-1 \}$ $|\omega| = n$:

$$\Pr[\, A_i \mid \Omega = \omega \,] = \Pr[\, Si \in \Omega^i \mid \Omega = \omega \,]$$

$$= \Pr[\, Si \in \omega^i \,] \quad \text{(puisque Si et } \Omega \text{ sont indépendants par l'hypothèse de hachage uniforme)}$$

Adressage ouvert - recherche infructueuse

- Pr[Si $\in \omega^i$], pour un ensemble $\omega \subseteq \{ 0, .., m-1 \}$ $|\omega| = n$:
  - Valeurs possibles de Si : dispositions (sans répétition) de { 0.. m-1} sur i places
    (équiprobables par l'hypothèse de hachage uniforme)

    $m\ (m-1)\ ..(m-i+1)$
  - Valeurs possibles de Si dans ω^i : dispositions sans répétition de ω sur i places

    $n\ (n-1)\ ..(n-i+1)$

$$Pr[\ Ai \mid \Omega = \omega\ ] = Pr[\ Si \in \omega^i\ ] = \frac{n\ (n-1)\ ..(n-i+1)}{m\ (m-1)\ ..(m-i+1)} = Pr[\ Ai\ ]$$

- Probabilité du nombre de sondages:

$$Pr[\ C \geq i\ ] = \frac{n\ (n-1)\ ..(n-i+2)}{m\ (m-1)\ ..(m-i+2)} \leq \alpha^{i-1} \qquad i = 2,..., n+1$$

$$Pr[\ C \geq 1\ ] = 1$$

$$Pr[\ C \geq n+2\ ] = 0$$

Adressage ouvert - recherche infructueuse

- Coût moyen :

$$E[C] = \sum_{i=1}^{\infty} Pr[\ C \geq i\ ]$$

$$\leq 1 + \sum_{i=2}^{n+1} \alpha^{i-1}$$

$$= \sum_{i=0}^{n} \alpha^{i} \leq \sum_{i=0}^{\infty} \alpha^{i} = \frac{1}{1-\alpha}$$

□

Adressage ouvert - recherche fructueuse

Analyse du cas moyen

Théorème

Le nombre moyen de sondages pour la recherche fructueuse d'une clef dans une table de hachage à adressage ouvert construite aléatoirement avec $\alpha = n/m < 1$ est au plus $\frac{1}{\alpha} \ln \frac{1}{1-\alpha}$ *sous les hypothèses:*

1) hachage uniforme pour la fonction de hachage sur les n clefs de la table

2) clef à chercher distribuée uniformément parmi les n clefs de la table

Preuve.

Variables aléatoires:

Ki : La i-eme clef insérée dans la table, i=1,..n
K: La clef à chercher (K est dans la table)
Ci: Nombre de sondages pour l'insertion de Ki
C: Nombre de sondages pour la recherche de K (coût de la recherche)

$$E[C] = \sum_{i=1}^{n} E[C \mid K = Ki] \cdot Pr[K = Ki] = \frac{1}{n} \sum_{i=1}^{n} E[C \mid K = Ki]$$

Adressage ouvert - recherche fructueuse

- Sachant K=Ki : C= Ci

$$E[C \mid K = Ki] = E[Ci \mid K = Ki] = E[Ci]$$

Ci: Coût de l'insertion de la clef Ki =

Coût de la recherche infructueuse de Ki dans un table construite aléatoirement sur K1, .. Ki-1

- Hachage uniforme sur K1, .. Kn $\Rightarrow$ Hachage uniforme sur K1, .. Ki $\Rightarrow$ $E[Ci] \leq \frac{1}{1 - \frac{i-1}{m}}$

$$E[C] = \frac{1}{n}\sum_{i=1}^{n} E[Ci] \quad \leq \quad \frac{1}{n}\sum_{i=1}^{n} \frac{m}{m-i+1} \quad = \quad \alpha \sum_{k=m-n+1}^{m} \frac{1}{k} =$$

$= \alpha\ (H_m - H_{m-n})$ où H_j est le j-ème *nombre harmonique* $Hj = 1 + \frac{1}{2} + \cdots + \frac{1}{j}$

$$H_m - H_{m-n} \leq \ln \frac{m}{m-n} \Rightarrow \quad E[C] \quad \leq \quad \alpha \ln \frac{1}{1-\alpha}$$

□

Adressage ouvert

Analyse du cas moyen

- Sous l'hypothèse de hachage uniforme, en absence de suppressions, et $\alpha < 1$:

  Recherche infructueuse: $O(\frac{1}{1-\alpha})$ fructueuse: $O(\frac{1}{\alpha} \ln \frac{1}{1-\alpha})$

  Insertion $O(\frac{1}{1-\alpha})$ (Insertion = recherche infructueuse + insertion de la clef dans le première alvéole libre)

- Si la taille $m \geq c\,n$ pour une constante $c > 1$, α est $O(1) \Rightarrow$

  Opérations de dictionnaire en temps O(1) en moyenne

- Exemple: 0.9 m ≥ n (table pleine à 90%)

  nombre moyen de sondage dans la recherche infructueuse (fructueuse) ≤ 10 (≤ 2.559)

- En présence de suppressions, la technique de chaînage est plus adaptée

Adressage ouvert

Choix de la fonction de hachage:

- L'hypothèse de hachage uniforme est difficile à garantir
  (difficile d'obtenir m! séquences de sondages distinctes)

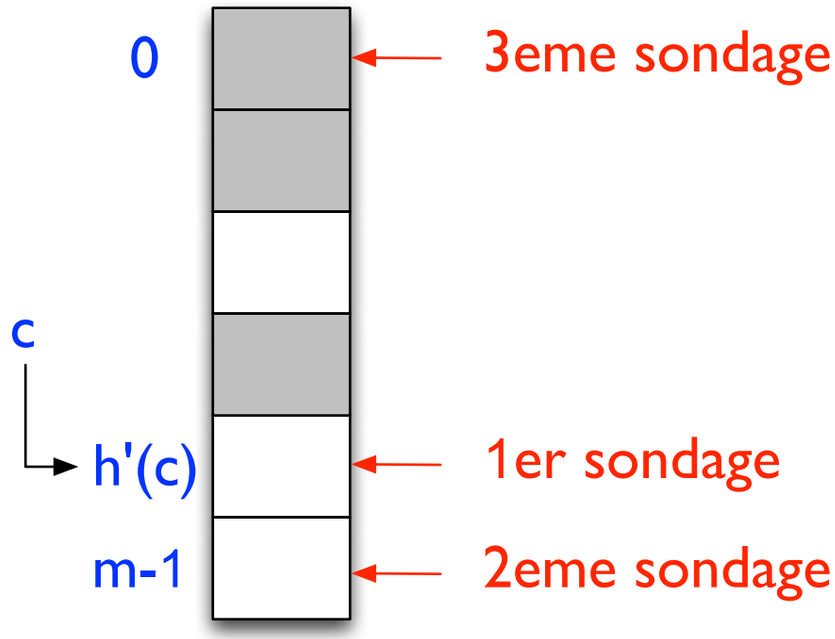
- Approximations (Tclef = $\mathbb{N}$)
  - *sondage linéaire* (m séquences de sondage)
  - *sondage quadratique* (m séquences de sondage)
  - *double hachage* (m^2 séquences de sondage) ← meilleure méthode

Sondage linéaire

$$h(c, i) = (h'(c) + i) \bmod m$$

où h' est une fonction de hachage ordinaire : Tclef → {0, .., m-1}

- Intervalle entre deux sondages d'une séquence: 1
- < h(c, 0), ..., h(c, m-1) > est une permutation de < 0, ..., m-1 >
- Nombre de séquences de sondage: *m*
  (= nombre de valeurs de h'(c), c ∈ ℕ)



- Problème de la *grappe forte* (*primary clustering*):

  longues séquences d'alvéoles occupées contiguës

  - Donnée une séquence de *k* alvéoles occupées: probabilité d'occuper la suivante = *k*/*m*
    (*sous l'hypothèse de hachage uniforme simple pour h'*)

Sondage quadratique

$$h(c, i) = (\ h'(c) + k_1 i + k_2 i^2\ ) \bmod m$$

- Contrainte sur k_1, k_2 et m:
  < h(c, 0), ..., h(c, m-1) > doit être une permutation de < 0, ..., m-1 >
  - trouver des valeurs de k_1, k_2 et m qui garantissent cette propriété (Exercice)
- Intervalle entre deux sondages: dépend (linéairement) de *i*
- meilleures performances que le sondage linéaire

- Nombre de séquences de sondage: *m*
  (= nombre de valeurs de h'(c))
- Problème de la *grappe faible* (*secondary clustering*):
  - clefs avec la même valeur h'(c) suivent le même chemin d'insertion

Double hachage

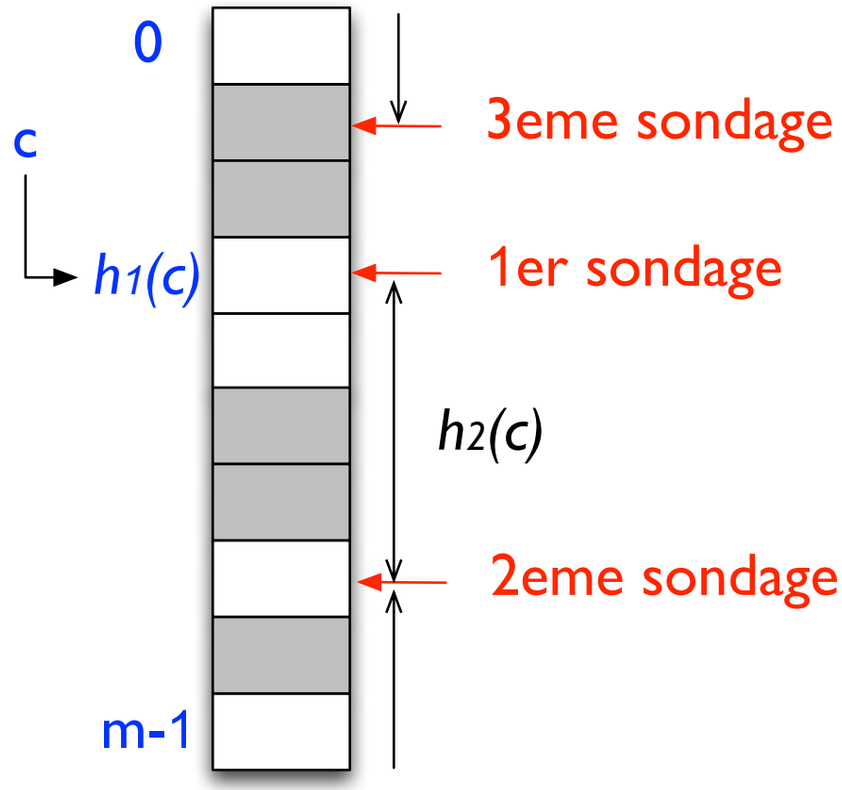
$$h(c, i) = (h_1(c) + i\, h_2(c)) \bmod m$$

où h_1, h_2 sont deux fonction de hachage ordinaires

- Intervalle entre deux sondages: $h_2(c)$

- Nombre de séquences de sondage: $\Theta(m^2)$
  (= nombre de valeurs de < $h_1(c), h_2(c)$ >)
  - m^2 si h_1 et h_2 sont "indépendants"

- Prestations comparable au hachage uniforme, en pratique



Double hachage

$$h(c, i) = (\ h_1(c) + i\ h_2(c)\ ) \bmod m$$

- Contraintes sur h_2 et m: si $h_2(c)$ est premier avec m (pour tout c), $< h(c, 0), ..., h(c, m-1) >$ est une permutation de $< 0, ..., m-1 >$:

  $h(c, i) \neq h(c, j)$ pour $i \neq j$ puisque l'équation linéaire modulaire:

  $$h_1(c) + i\ h_2(c) = b \pmod m$$

  a *une seule* solution (modulo *m*) pour *i* quand $h_2(c) > 0$ *et* $m > 1$ *sont premiers entre eux*

- m et $h_2(c)$ premiers entre eux, pour tout c, si:
  - *m: puissance de 2* et *$h_2(c)$: impaire*, pour tout c, ou
  - *m: premier* et $0 < h_2(c) < m$, pour tout c
- Exemple: m premier et

  $h_1(c) = c \bmod m$

  $h_2(c) = 1 + (c \bmod m')$ avec $m' < m$ et m' proche de m *(ex. m' = m-1)*

Hachage statique et hachage dynamique

Hachage statique. Chaînage et adressage ouvert sont des techniques de hachage statique

- Adressage ouvert: la taille *m* de la table borne la taille *n* de l'ensemble de clefs
- Chaînage: pas de borne explicite, mais $n \leq \alpha m$ (avec α constant) pour des bonnes performances

Hachage dynamique.

- Si l'ensemble dynamique des clefs ne peut pas être borné:
  - tables de hachage combinées avec une technique de gestion de tables dynamiques (doubler la taille de la table lorsque le facteur de remplissage atteint un certain seuil)
  - clefs re-insérées dans la nouvelle table avec une nouvelle fonction de hachage (*rehashing*)
  - coût moyen amorti constant
- Le *rehashing* est trop couteux pour des ensembles en mémoire externe (ex. indexes de bases de données)
- Hachage dynamique en mémoire externe [voir Silberschatz et al. "Database System Concepts"]
  - *hachage linéaire* (à ne pas confondre avec le sondage linéaire)
  - *hachage extensible*

Hachage parfait

Permet d'obtenir complexité de la recherche O(1) dans le cas pire

sous l'hypothèse que l'ensemble des clefs est fixé
(les clefs, une fois insérées dans la table ne changent plus)

Voir

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein.
Introduction to Algorithms. 3e édition, The MIT Press 2009

Bibliographie

1) Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein.
Introduction to Algorithms. 3e édition, The MIT Press 2009

2) D. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching* (2nd Edition)
Addison-Wesley

3) A.V. Aho, J.E. Hopcroft, J.D. Ullmann.
Data Structures and Algorithms. Addison-Wesley.

4) R.Sedgewick, Ph. Flajolet *An Introduction to the analysis of algorithms* Addison-Wesley

5) A.Silberschatz, H.F. Korth, S. Sudarshan *Database System Concepts* Third Ed. McGraw-Hill

Union-Find: 1), 3) et

6) Danièle Beauquier, Jean Berstel, Philippe Chrétienne.
Éléments d'Algorithmique.
Masson, 1992. http://www-igm.univ-mlv.fr/~berstel/Elements/Elements.html