Tables de hachage

1/ Ensembles et tableaux associatifs

Ensemble (fini) :
- Insertion d'un élément (qui n'existe pas encore)
- Recherche d'un élément
- Suppression d'un élément

Tableau associatif : ensemble fini de paires (k, v)
(k : clef, v : valeur)

Tableau associatif :
- Insertion d'une paire (dont la clef n'existe pas encore)
- Recherche d'une paire par sa clef
- Suppression d'une paire (par sa clef)

En Python : `set` pour l'ensemble (`x in s`)
`dict` pour un dictionnaire `{.....}`
(implémentés par une table de hachage)

Implémentation naïve

1) Tableau dynamique
   - Insertion : $O(1)$ amorti
   - Recherche : $\Theta(n)$
   - Suppression : $O(1)$ amorti
     après recherche (en échangeant la case contenant l'élément et la dernière case)

2) Tableau dynamique trié
   - Insertion : $\Theta(n)$
   - Recherche : $\Theta(\log n)$
   - Suppression : $\Theta(n)$

3) Liste dll chaînée
   - Insertion : $O(1)$
   - Recherche : $\Theta(n)$
   - Suppression : $O(1)$ après recherche

2/ Tables de hachage

Espace de hachage : $\{0, \dots, m-1\}$

Univers U : ensemble des (éléments possibles clefs)

En général : $|U| >> n$

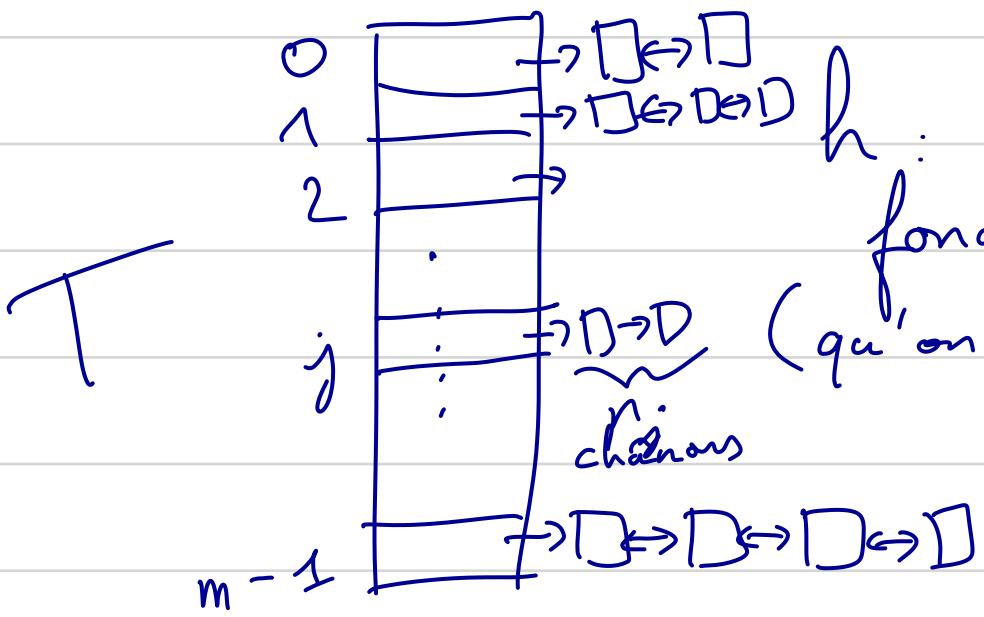
n ↑ nombres d'éléments à stocker

On va prendre m proche de n

$\alpha = \frac{n}{m}$. On va prendre $\alpha = O(1)$

facteur de charge (donc $m << |U|$).

Table de hachage : tableau de taille m fixée



$h : U \to \{0, \dots, m-1\}$

fonction de hachage fixée.

(qu'on suppose calculable en $O(1)$)

Insérer un nouvel élément $x \in U$:

→ Calcule $j = h(x)$

→ Accède à $T[j]$ $O(1)$

→ Ajoute x au début de la liste chaînée pointée par $T[j]$

Rechercher si x est dans l'ensemble

→ Calcule $j = h(x)$

→ Accède à $T[j]$

→ Recherche si x est dans la liste chaînée pointée par $T[j]$ $O(n_j)$

Suppression (après recherche)

$O(1)$ → Suppression ds la liste dblt chaînée

Dans le pire des cas

$h(x) = h(x')$ pour tout x, x' que l'on veut stocker : $n_{h(x)} = n$ et donc complexité en $\Theta(n)$

Cela se produit si :
- h est "mauvaise" (elle répartit les éléments non uniformément dans l'espace de hachage)
- on est "malchanceux" et tous les éléments à stocker ont la même valeur de hachage malgré un hachage uniforme

Hypothèse d'uniformité : $\Pr_{x \in \text{éléments à stocker}}(h(x) = j) = \frac{1}{m}$

pour $0 \leq j \leq m-1$

$$E[n_j] = \sum_{i=1}^{n} \Pr(h(x_i) = j) = \sum_{i=1}^{n} \frac{1}{m} = \frac{n}{m} = \alpha$$

facteur de charge.

Si :
- l'hypothèse d'uniformité est vérifiée
- $\alpha = O(1)$

Alors :
- Recherche en $O(1)$ *en moyenne*

Choix / Conception d'une fonction de hachage

$$h(x) = x \mod m$$

* Par exemple si $m = 256 = 2^8$

Si x est un entier sur 4 octets (entre 0 et $2^{32}-1$),
$h(x)$ est l'octet de poids faible.

* si $m = 1000$

et qu'on stocke des comptes de population qui sont souvent arrondies au millier, alors $h(x)$ sonvent $= 0$.

Du coup, on va choisir m un nombre premier, qui n'a pas de signification particulière pour les éléments de l'univers.

Hachage universel

On suppose qu'on a un ensemble de fonctions de hachage $H = \{h_1 \dots h_w\}$

pour $k \neq l$ $\in U$ $\#\{h(k) = h(l)\} \leq \dfrac{|H|}{m}$

On construit une table de hachage comme avant, mais en choisissant comme fonction de hachage h tiré uniformément aléatoirement parmi H.

On introduit la variable aléatoire $X_{kl} = \mathbb{1}_{h(k)=h(l)}$ pour $k \neq l$ arbitraires.

Y_k : nombre d'éléments dans la liste chaînée à la case $h(k)$

$$E[Y_k] \leq E\left[\sum_{\substack{l \in T \\ l \neq k}} X_{kl}\right] + 1$$

$$\leq \sum_{\substack{l \in T \\ l \neq k}} \underbrace{E[X_{kl}]}_{\leq \frac{1}{m}} + 1$$

$$\leq \sum_{\substack{l \in T \\ l \neq k}} \frac{1}{m} + 1$$

$$= O\left(\frac{n}{m}\right) = O(\alpha)$$

Choix d'un ensemble de fonctions de hachage universel :

Fixons $p > m$ nombre premier.

Je considère pour $0 < a < p$ et $0 \leq b < p$ la fonction de hachage $h_{a,b}$ définie par :

$$h_{a,b}(k) = \left[(ak + b) \bmod p\right] \bmod m$$

$$H = \left\{ h_{a,b} \mid 0 < a < p,\ 0 \leq b < p \right\}$$

$$|H| = p(p-1)$$

Prop $\#\{h(k) = h(l)\} \leq \frac{p(p-1)}{m}$

pour tous $k \neq l$

Soit $k \neq_{(\text{mod } p)} \ell \in U$.

$$r = (ak + b) \bmod p$$
$$s = (a\ell + b) \bmod p$$

On a $r \neq s$ car $r - s \equiv a(k - \ell) \bmod p$

(a et $(k-\ell)$ non nuls modulo p)

$$a = (r-s)\left[(k-\ell)^{-1} \bmod p\right] \bmod p$$
$$b = (r - ak) \bmod p$$

Il y a une bijection entre les choix possible pour les (a, b) avec $0 < a < p$ et $0 \leq b < p$ et les paires (r, s) avec $r \neq s$ et $0 \leq r, s < p-1$

Donc, pour $k \neq \ell$ en entrée, j'obtient toutes les paires (r, s) avec $r \neq s$ avec la même probabilité $\frac{1}{p(p-1)}$ en choisissant uniformément aléatoirement une paire (a, b) avec $0 < a < p$ et $0 \leq b < p$.

Donc $\Pr_h[h(k) = h(\ell)] = \underbrace{\Pr_{\substack{r,s \\ r \neq s}}[r \equiv s \bmod m]}_{(*)}$

Pour un $0 \leq r < p$ fixé,
Le nombre de $s \neq r$ t.q. $0 \leq s < p$ et $r \equiv s \bmod m$ est :

$$\leq \left\lceil \frac{p}{m} \right\rceil - 1 \leq \frac{p-1}{m}$$

$$(*) \leq \frac{\frac{(p-1)}{m}}{p-1} = \frac{1}{m}$$

□

En pratique :
→ soit on fait du hachage universel
→ soit on prend une "bonne" fonction de hachage et on fait l'hypothèse que les éléments à stocker ne sont pas biaisés par cette fonction de hachage

Et ça marche : $O(1)$ vérifié en pratique pour la recherche, l'insertion et la suppression

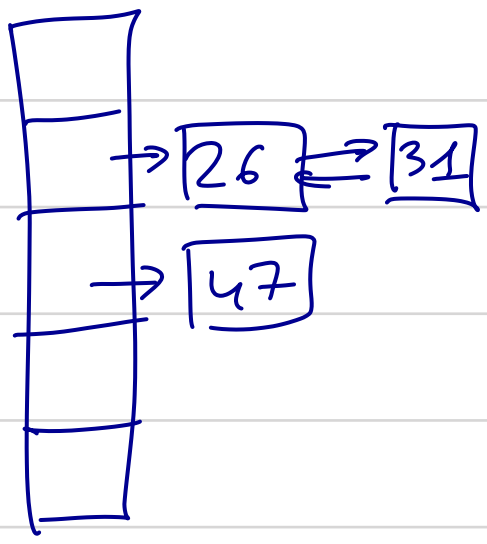
---

Jusqu'ici on a parlé de hachage par chaînage.
Inconvénient : la table de hachage ne tient pas dans une zone contigüe de mémoire

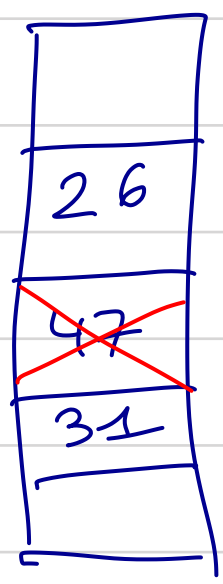
Alternative : hachage ouvert → tous les éléments sont directement ds la table de hachage, sans liste chaînée

$m = 5$

$h(DA) = x \bmod m$



26, 47, 31



Sondage linéaire
→ ajoute l'élément ds la prochaine case libre (parcours linéaire)
→ Suppression : remplace un élément par une valeur spéciale indiquant une suppression

Pb : tendance à faire apparaître des zones contiguës pleines

Hachage ouvert plus intelligent : double hachage.

h_1 , h_2 deux fonctions de hachage

$$h(k, i) = h_1(k) + i \times h_2(k) \mod m$$

(on veut $h_2(k)$ premier avec m).

On essaie d'insérer en $h(k, 0)$

si déjà rempli, en $h(k, 1)$

en $h(k, 2)$

⋮

jusqu'à trouver une case libre.

En pratique, très bonnes performances.

Tables de hachage dynamique
(dont la taille n'est pas fixée à l'avance)

* Faire comme les tableaux dynamique
(réallouer la table de hachage avec $m' = 2 \times m$ (ou une variante)

on prend une nouvelle fonction de hachage $U \to \{0, \dots, m'-1\}$, on réinsère les valeurs unes à unes. Coûteux ($O(n)$) mais rare.
Analyse amortie → $O(1)$ amorti.

* Faire plus intelligent
→ hachage extensible
→ hachage linéaire (Litwin)