Hachage

Hachage

TAD :
opérations $\mathsf{insert}(k)$, $\mathsf{delete}(k)$, $\mathsf{find}(k)$ où $k \in U$ est une **clé**

(U : univers de clés)

Tableau de hachage : taille M

on veut y placer N éléments

facteur de remplissage (*load factor*) $\alpha = N/M$

Fonction de hachage

Fonction de hachage : $h : U \mapsto \{0, 1, \ldots, M-1\}$

collision : $h(k) = h(k')$ mais $k \neq k'$

Birthday paradox (clés aléatoires)

Probabilité d'aucune collision

$$p = 1\left(1 - \frac{1}{M}\right)\left(1 - \frac{2}{M}\right)\cdots\left(1 - \frac{N-1}{M}\right)$$
$$< \prod_{i=0}^{N-1} e^{-i/M} = \exp\left(-\frac{(N-1)N}{2M}\right)$$

$N > \sqrt{M 2 \ln 2} \approx 1.18\sqrt{M}$, $\Rightarrow$ au moins une collision avec probabilité $\geq 1/2$.

Fonctions de hachage

on veeut éviter les collisions : disperser les éléments autant que possible

idéalement : $h(k)$ a la distribution uniforme sur $\{0, \ldots, M-1\}$.

en pratique : on ne sait pas la distribution de k

Fonctions de hachage (cont)

Méthode de la division : $h(k) = k \bmod M$

Choix de M : on veut éviter la réduction de l'espace de valeurs de hachage à cause des données non-aléatoires

- ne devrait pas être $M = 2^k$ (les derniers k bits de la clé determinents la valeur de hachage)
- puissances de 10 à éviter (nombres décimaux)
- ne devrait pas être un multiple de 3 (permutations d'une clé binaire donnent des valeurs de hachage qui diffèrent par multiples de 3)

Règle : choisir un prime qui est loin de 2^k et 10^m.

Fonctions de hachage (cont)

Méthode de la multiplication : $h(k) = \Big(\lfloor M\{\gamma k\}\rfloor\Big)$
où w est la longueur d'un mot-machine et $\{x\} = x - \lfloor x\rfloor$

Choix de M : $M = 2^p$ (p bits de l'ordre supérieur de la moitié inférieure de Ak)

Poser $\gamma = \frac{A}{2^w}$
Choix de A : prime relatif à 2^w

- Ak devrait être distribué «uniformement»

$A \approx \frac{\sqrt{5}-1}{2}2^w$ est un bon choix (le prime relatif le plus proche)

Hachage universel

Choisir une fonction de hachage aléatoire :
écrire la clé de $(r+1)$ «caractères» comme $x = \langle x_0, x_1, \ldots, x_r \rangle$ et calculer

$$h(x) = \left(\sum_{i=0}^{r} h_i(x_i) \right) \bmod M$$

où h_i sont aléatoires

et M est un prime

P.e., $h_i(x) = a_i x \bmod M$, $a_i \in \{0, \ldots, M-1\}$ aléatoire uniforme

Collision entre $x = \langle x_0, \ldots, x_r \rangle$ et $y = \langle y_0, \ldots, y_r \rangle$:

$$\sum_{i=0}^{r} a_i(x_i - y_i) \equiv 0 \pmod{M}$$

Il existe M^r solutions pour $\langle a_0, \ldots, a_r \rangle$
$\Rightarrow$ probabilité de collision $\frac{M^r}{M^{r+1}} = \frac{1}{M}$. (indépendamment de x, y)

Résolution de collisions

1. chaînage : stocker une liste pour chaque valeur de hachage

2. adressage ouvert : séquence de sondage

Chaînage

Liste chaînée

Performance : longueur d'une liste en moyenne est α

Nombre de comparaisons pour recherche avec succès $1 + \alpha/2$

Permet $\alpha > 1$

Adressage ouvert

Sondage d'une séquence de positions : dépend de la clé à insérer

On essaie les cases $h_0(k)$, $h_1(k)$, ...

Avec une fonction f : $h_i(k) = h(k) + f(i,k) \bmod M$

f : stratégie de résolution de collision

Méthodes :

- sondage linéaire $f(i,k) = i$
- sondage quadratique $f(i,k) = i^2$
- double hachage $f(i,k) = ih'(k)$ avec fonction de hachage auxiliaire h'

Ne permet pas $\alpha > 1$

Sondage linéaire

(*Linear probing*) : $h(k), h(k)+1, h(k)+2, \ldots$

Problème : grappe forte (*primary clustering*) — blocs de cases occupées

Nombre moyen de sondages lors de recherche fructueuse : $\approx \frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$

sondages lors de recherche infructueuse ou insertion : $\approx \frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right)$

$\Rightarrow$ utiliser jusqu'à $\alpha < 0.75$

Sondage quadratique

(*Quadratic probing*) : $h(k), h(k)+1, h(k)+4, h(k)+9, \ldots$

Problème : grappe faible (*secondary clustering*) — suite de cases occupées par clés avec la même valeur de hachage

N'est pas certain que l'on trouve une case vide lors du sondage que si $\alpha < 0.5$

Double hachage

Généralisation de sondage linéaire : $h(k), h(k)+c, h(k)+2c, h(k)+3c, \ldots$
c dépend de la clé k : $c = h'(k)$

Exemples (éviter $c = 0$!) : $h'(x) = 1 + x \bmod M'$ avec $M' < M$
$h'(x) = M' - (x \bmod M')$

Très proche d'une résolution idéale

Nombre moyen de sondages lors d'insertion ou recherche infructueuse :
$\approx 1 + \alpha + \alpha^2 + \cdots \approx \frac{1}{1-\alpha}$

sondages lors de recherche fructueuse :
$\approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{1}{1-i/M} \approx \frac{1}{\alpha} \ln \frac{1}{1-\alpha}$

Deletion

suppression avec adressage ouvert : très difficile

utiliser *lazy deletion* : marquer la case «supprimée»

$\Rightarrow$ facteur de remplissage inclut les éléments supprimés

Ensemble statique

On veut stocker un ensemble statique de M clés d'un univers de taille $n = |U|$

On ne veut que l'opération $\mathsf{find}(k)$ — vérifier si k est là

Exemple : utiliser un tableau des valeurs triées $\rightarrow$ recherche binaire pour find

Problème théorique : on fixe le nombre de sondages, quels choix de M et n sont possibles ?

Recherche binaire : $\lceil \lg(M+1) \rceil$ sondages — est-ce qu'on doit toujours stocker les valeurs triées ?

Exemple : stocker deux clés de $U = \{1, 2, 3\}$

$$\{1, 2\} \rightarrow [1, 2]$$
$$\{2, 3\} \rightarrow [2, 3]$$
$$\{1, 3\} \rightarrow [3, 1]$$

Sondage

Exemple : stocker $M > 2$ clés de $U = \{1, 2, \ldots, n\}$ avec $n = 2M - 2$

Personnes $P_1, \ldots, P_n$: $P_1, \ldots, P_M$ des hommes et $P_{M+1}, \ldots P_{2M-2}$ des femmes

On a $M - 2$ couples : (P_1, P_{M+1}), (P_2, P_{M+2}), $\ldots$, (P_{M-2}, P_{2M-2}) et deux célibataires P_{M-1}, P_M

Bicycles $C_1, \ldots, C_M$: $C_1, \ldots, C_{M-2}$ appartiennent aux couples, C_{M-1} et C_M appartiennent aux célibataires

Règles pour l'excursion de M personnes :

1. si seulement un/e membre d'une couple est là, elle/il prend sa bicycle
2. si tous les deux d'une couple sont là, la femme doit prendre une bicycle sans propriétaire présent
3. tous ceux qui restent (les célibataires et les hommes des couples) partagent les bicyles qui restent mais personne ne prend son propre vélo

Ensembles statiques

Pour vérifier si P_i est là : regarder son vélo

1. si P_i est sur son vélo, il/elle est là
2. si c'est une femme sur son vélo, P_i n'est pas là
3. si c'est un homme sur son vélo, P_i est là