Lezione 21 aprile

Tuesday, April 21, 2020 4:07 PM

Dizionari

e = chiave / dati

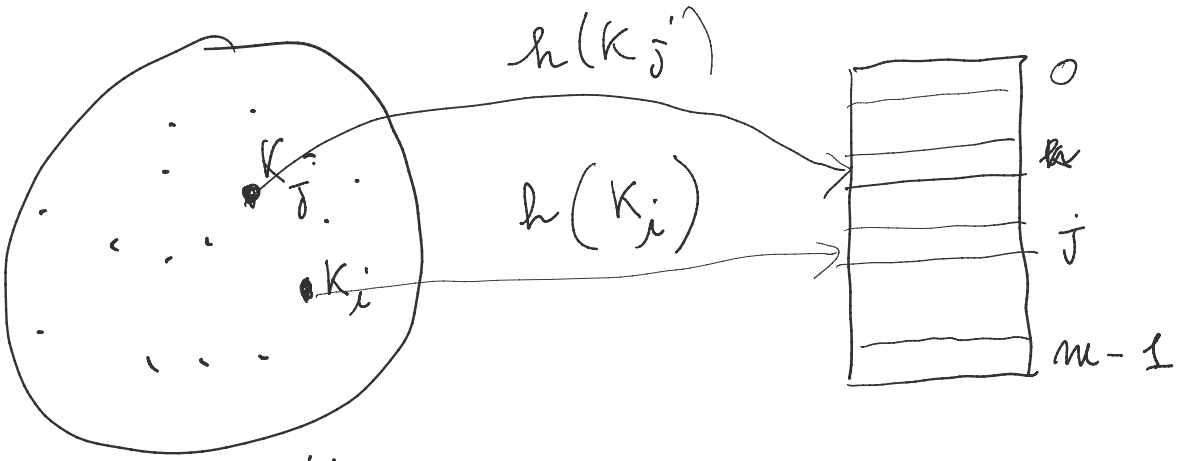
- ricerca (k)
- inserzione (e)
- cancellazione (e)

Tabelle hash

limite inferiore $\Omega(\log n)$ caso pessimo caso medio

⇓

ricerca per confronti



U h = funzione hash

K_i e K_j può accadere $h(K_i) = h(K_j)$

↑ COLLISIONE

Tabella hash

1) Dimensionare la tabella: m.
2) Scelta funzione hash
3) Metodo per la gestione delle collisioni

K può essere numerica o alfanumerica

codifica ASCII → ogni carattere in una cifra a base 16.

a) m: numero primo

$h(K) = K \bmod m$

b) $m = 2^r$

• $h(K) =$ r bit

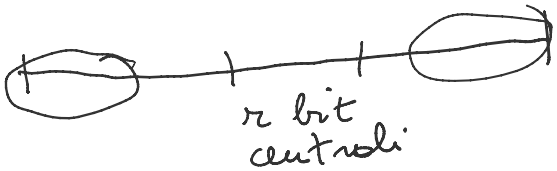
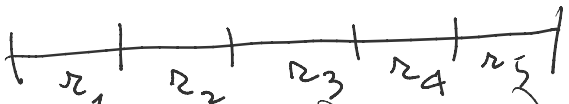


Tabella dei simboli di un compilatore è realizzata come una tabella hash

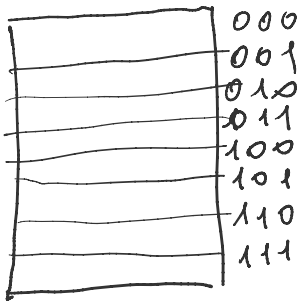
punt, ptr, punt1, punt2

h deve distribuire più uniformemente possibile le chiavi in tabella



K di s bit

$h(K) = (((r_1 \oplus r_2) \oplus r_3) \oplus r_4) \oplus r_5) = r$ bit



$r = 3$

$m = 2^3$

$r \rightarrow 2^r$

$$a \oplus b = \begin{cases} 0 & \text{se } a = b \\ 1 & \text{se } a \neq b \end{cases}$$

SHA 1

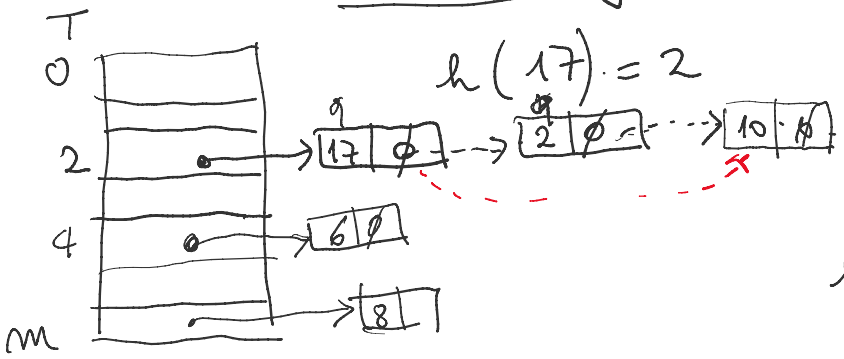
- Tabelle hash
- sistemi crittografici
- finger print
- sistemi peer to peer - sistemi distribuiti

Gnutella, Torrence, ...

$h(K_i) \neq h(K_j)$ situazione ideale

Risolvere le collisioni:

Chaining (liste di trabocco)



m = dimensione tabella n = n° el. in tabella

$\alpha = \frac{n}{m}$ fattore di riempimento

Ricerca(K) calcolo indirizzo hash $h(K)$

Ricerca in una lista:
- accesso alla lista di puntatore $h(K)$
- ricerca nella lista.

Ricerca nel caso pessimo? $O(n)$

caso pessimo: tutti gli elementi finiscono nella stessa lista

Tabelle funzionano male al caso pessimo!

Ricerca al caso medio?

Ricerca: numero medio di accessi $= (1 + \frac{1}{\alpha})$ per valori opportuni di α la ricerca è $\Theta(1)$.

Inserzione –

- calcolo indirizzo hash
- inserzione in fondo alla lista se K non è già presente.

- (oppure inserzione in testa dopo ricerca)

Cancellazione

Open hash ≡ Hashing aperto

Tutti gli elementi vengono memorizzati in tabella.

Sequenza di prove

$h(K,0) = (h(K), h(K,1), h(K,2), \dots h(K,m-1))$

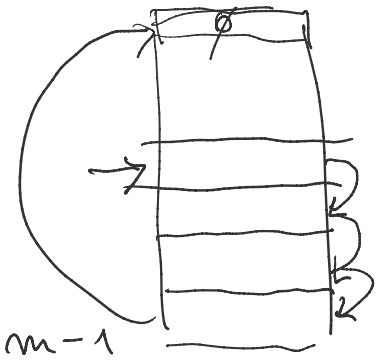
.

.

Se una cella = ϕ : ricerca senza successo

Inserzione: nella cella nulla

$h(K,i) = (h(K,0) + i) \bmod m$



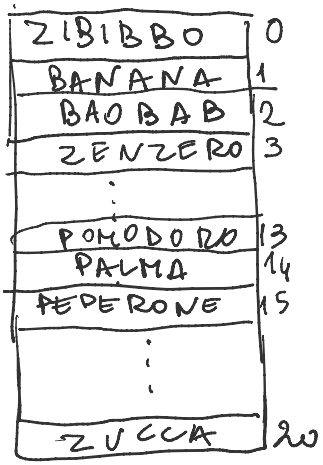
inserzione di K

$h(K)$

scansione lineare a passo 1

$h(K,i) = (h(K,0) + iq) \bmod m$

scansione lineare a passo q



$h(K)$ = posizione nell'alfabeto dell'iniziale

$h(\text{BANANA}) = 1$ $h(\text{BAOBAB}) = 1$

$h(\text{POMODORO}) = 13$

$h(\text{ZUCCA}) = 20$

$h(\text{ZIBIBBO}) = 20$

$h(\text{ZENZERO}) = 20$

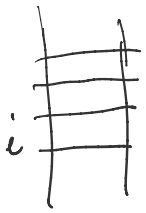
Cancellazione (LIBRO)

Ricerca K: si fa la sequenza di prove fino a trovare K o una cella vuota o fare m prove.

Inserimento (K) = Ricerca di K e si inserisce nella prima cella vuota

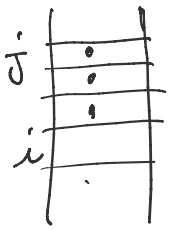
Gli agglomerati tendono a diventare più grandi.

P_{rr} (chiave K: vado nella cella i isolata)
$= pr(h(K) = i) = \frac{1}{m}$



P_{rr} (chiave K vado in una cella che segue un agglomerato) =

$P_{rr}(h(K) = j) + P_{rr}(h(K) = j+1) + \dots +$
$P_{rr}(h(K) = i) = \frac{i - j + 1}{m}$



Cancellazione

Usare marcature

Tempo medio di ricerca e inserzione

per OPEN HASH $\frac{1}{1-\alpha}$ caso medio

$\alpha = \frac{n}{m}$ α va mantenuto al di sotto del 90%

Algoritmo di cancellazione fun mort

- cancella l'elemento
- Considera tutti gli elementi che seguono nell'agglomerato e spostali nella cella vuota se il loro indirizzo hash è "minore" o uguale della cella vuota