Problema del DIZIONARIO

S : collezione di elementi

ogni elemento è identificato, in modo univoco, attraverso una chiave

U : universo delle chiavi

x = puntatore all'elemento del dizionario

$k = x.key$

OPERAZIONI

- Insert(S, x) $\quad S \rightarrow S \cup \{x\}$
- Delete(S, x) $\quad S \rightarrow S \setminus \{x\}$
- Search(S, k)

$$Search(S, k) = \begin{cases} x & x \in S \text{ t.c. } x.key = k \\ NIL & \text{se } S \text{ non contiene un elemento di chiave } k \end{cases}$$

U totalmente ordinato

$Min(S)$ l'elemento di chiave minima di S

$Max(S)$ // // // massima

$Successore(S, x)$ restituisce un puntatore y all'elemento con la più piccola chiave $\geq x.key$

$Predecessore(S, x)$ puntatore y all'elemento con la chiave più grande ~~[illegible]~~ $\leq x.key$

TAVOLE a INDIRIZZAMENTO DIRETTO

$U \in \mathbb{N}$

idea: la chiave → posizione dell'elemento nella tabella

$x \in S \qquad T[x.key] = x$

T array di dimensione $|U|$

$$T[k] = \begin{cases} x & x \in S,\ x.key = k \\ NIL & \forall x \in S,\ x.key \neq k \end{cases}$$

Search(T, k) $\Theta(1)$

return $T[k]$

Insert(T, x) $\Theta(1)$

$T[x.key] = x$

Delete(T, x) $\Theta(1)$

$T[x.key] = NIL$

Svantaggio $|S| << |U|$

SPRECO DI MEMORIA

TABELLE HASH

idea: uso la chiave per calcolare la posizione degli elementi in una tabella ad accesso diretto

T dim $T \simeq |S| << |U|$

K = insieme delle chiavi degli elementi di S

$K \subseteq U$

dim T = m

$T[0, 1, \ldots, m-1]$

funzione hash: $h : U \to [0, m-1]$

$|U| >> m$

$|U| >> m$ $\quad h : U \to [0, m-1]$

- h NON PUÒ ESSERE INIETTIVA

$\exists\ k_1, k_2 \in U \quad k_1 \neq k_2$ e $h(k_1) = h(k_2)$

"collisione"

- h deve essere facile da calcolare
- h deve essere SURIETTIVA (× usare tutte le celle di T)
- h deve essere deterministica
- h deve DISTRIBUIRE UNIFORMEMENTE LE CHIAVI SULLE CELLE di T
- OSSERVAZIONE $k_1 < k_2 \not\Rightarrow h(k_1) < h(k_2)$ non è detto

Realizzazione del dizionario con tabelle hash

T di dim m.

$h : U \to [0, m-1]$

$T[h(k)] = \begin{cases} \text{puntatore a } x \in S \text{ t.c. } x.key = k \\ \text{NIL} \end{cases}$

S non contiene elementi chiave k.

HASHING PERFETTO

non ci sono collisioni

h è iniettiva su K

$\forall k_1, k_2 \in K \subseteq U,\ k_1 \neq k_2 \Rightarrow h(k_1) \neq h(k_2)$

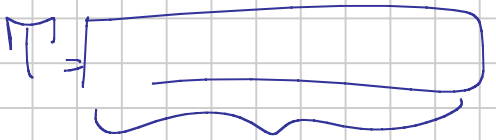
METODO della DIVISIONE

$U = \mathbb{N}$ $\quad h : \mathbb{N} \to [0, m-1]$

$k \in U$

$h(k) = k \bmod m$

$m = 19$ $\quad k = 100$



$h(k) = 100 \bmod 19 = (95+5) \bmod 19 =$

$= (19 \cdot 5 + 5) \bmod 19 = 5$

T

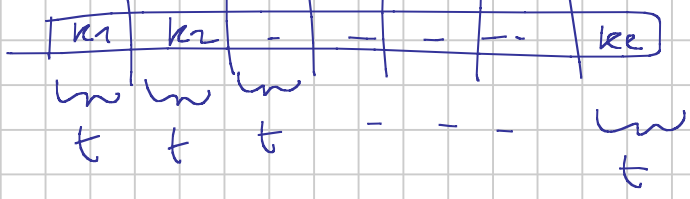
→ m numero primo

→ m non troppo vicino a una potenza di 2.

⇝ $h(k)$ dovrebbe dipendere da tutti i bit della chiave

METODO ITERATIVO

$k \rightarrow$ si scrive k in binario



$m = 2^t$

uso t bit per indicizzare le posizioni di T

$h(k) = k_1 \oplus k_2 \oplus \dots \oplus k_e$ → sequenza di t bit che corrisponde a una posizione della tabella

OR ESCLUSIVO bit a bit

$k = (310)_{10} = (100|110|110)_2$

$m = 2^3 = 8$

$t = 3$

$h(k) = (100 \oplus 110 \oplus 110)_2 =$

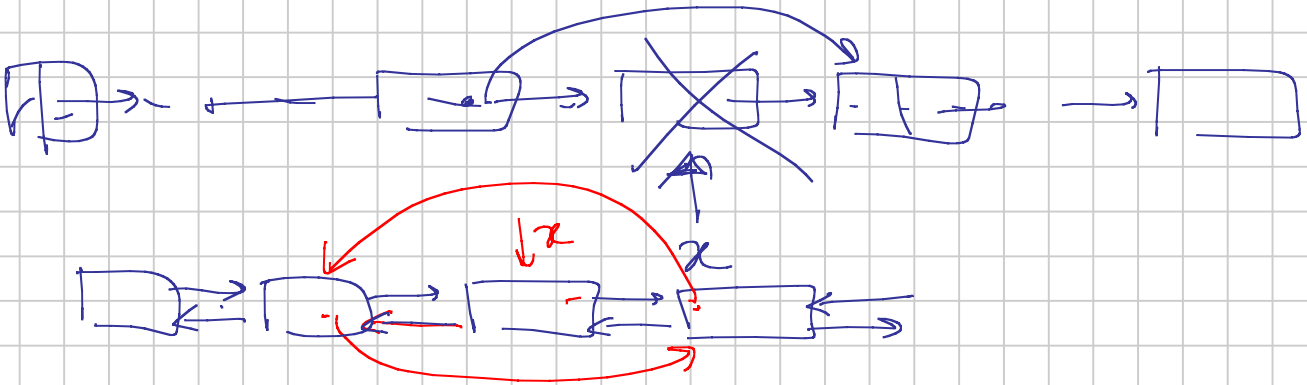
$= (010 \oplus 110)_2 = (100)_2 = (4$

① TABELLE HASH con concatenamento (o liste di trabocco)

idea

$T[j]$ = puntatore alla lista di tutti gli elementi di S che hanno chiave il cui valore hash è j

T: array di liste doppie dim $T = m$



Operazioni di dizionario

m = dim T

n = $|S|$ = # elementi memorizzati in T

CHAINED-HASH-INSERT (T, x)

inserisci x in testa alla lista, doppia $T[h(x.key)]$ $\quad T(n) = \Theta(1)$

(n potrebbe essere anche $> m$)

CHAINED-HASH-DELETE (T, x)

rimuovi x dalla lista doppia $T[h(x.key)]$ $\quad T(n) = \Theta(1)$

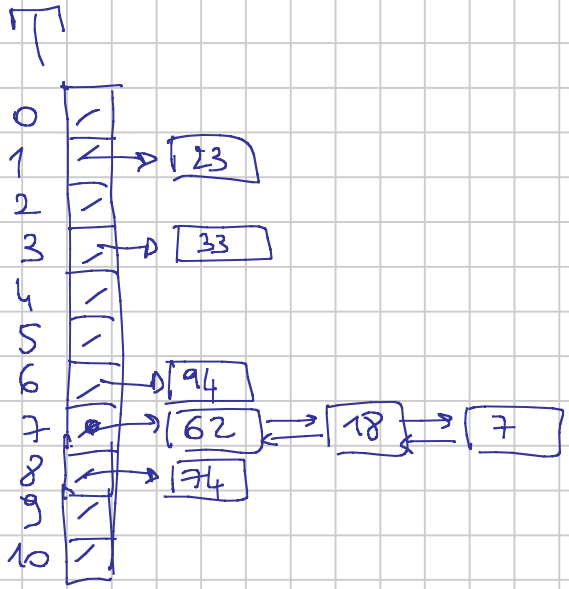
CHAINED-HASH-SEARCH (T, k)

cerca un elemento di chiave k nella lista $T[h(k)]$ caso peggiore $\quad T(n) = \Theta(n)$

ESEMPIO

$K = \{7, 94, 36, 18, 23, 62, 74\}$

$h(k) = k \bmod 11$ $\quad m = 11$



$$\alpha = \frac{n}{m} = \text{fattore di carico della tabella hash}$$

$$= \frac{|S|}{\dim T}$$

Analisi della ricerca al caso medio

TEOREMA

In una tabella hash in cui le collisioni sono risolte con il concatenamento la ricerca richiede tempo $\Theta(1+\alpha)$ al caso medio, nell'ipotesi di "HASHING UNIFORME SEMPLICE".

↳ qualsiasi elemento ha la stessa probabilità di essere mandato in una qualsiasi delle m celle della tabella, indipendentemente dalle celle in cui sono mandati gli altri elementi.

Dimostrazione (ricerca senza successo)

ipotesi di hashing uniforme semplice ⇒ gli n elementi di S sono distribuiti uniformemente sulle m liste

⇓

le liste hanno una lunghezza media pari a $\alpha = \frac{n}{m}$

$k \rightarrow h(k) \rightarrow$ scorro la lista $T[h(k)]$ che ha lunghezza α

$T_{medio}(n, m) = \Theta(1 + \alpha)$

1: calcolo funzione hash

α: ricerca nella lista $T[h(k)]$

se α è $\Theta(1)$

$\Rightarrow$ la ricerca ha costo $\Theta(1)$ al caso medio.

Nella pratica si cerca di tenere $\alpha \simeq \frac{1}{2}$

Costo dipende dal rapporto $\frac{n}{m}$