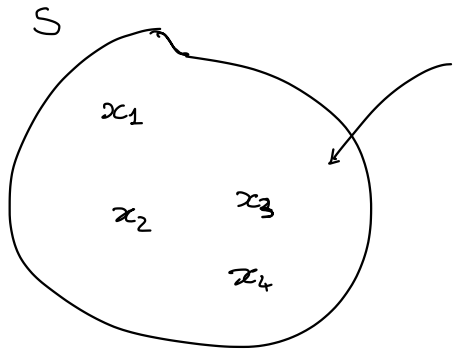
Algoritmi e Strutture Dati (26/10/2021)

* Hash Tables

collezione di oggetti x $\begin{cases} x.key \\ \text{dati satellite} \end{cases}$



Insert (S, x)
Delete (S, x)
Search (S, key)

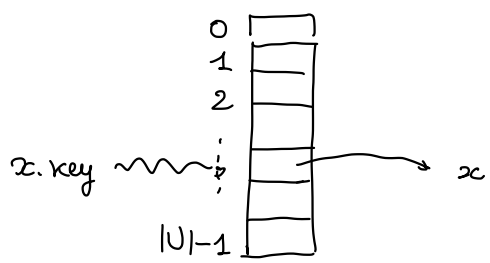
efficiente:
→ costo medio $\Theta(1)$
→ " peggiore $\Theta(n)$

* Indirizzamento diretto

insieme (universo) di chiavi $U = \{0, 1, \dots, |U|-1\}$

idea: collezione come un array $T[0..|U|-1]$

un elemento x è inserito in $T[x.key]$



$$T[j] = \begin{cases} \text{l'elemento } x \text{ della collezione t.c. } x.key = j & \text{(se c'è)} \\ nil & \text{altrimenti} \end{cases}$$

problemi:
1. non è possibile avere oggetti con la stessa chiave
2. ok se il numero $|U|$ è "piccolo"

Insert (T, x)
$T[x.key] = x$ $\Big\}$ $\Theta(1)$

Search (T, k)
return $T[k]$ $\Big\}$ $\Theta(1)$

Delete (T, x)
$T[x.key] = nil$ $\Big\}$ $\Theta(1)$

problema :

U = stringhe di 8 caratteri

1 car. = 8 bit 2^8 caratteri

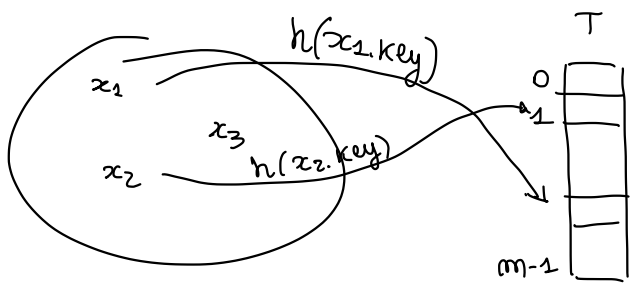
$\underbrace{2^8 \times \cdots \times 2^8}_{8}$ stringhe = 2^{64} stringhe

$\cong 16 \cdot 10^8$

* Hash Tables

obiettivo : usare quantità di memoria proporzionale al # di elementi da memorizzare

Idea : uso tabella T di dimensione $m \ll |U|$



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

funzione di hashing

* Collisione : k_1 e k_2 t.c. $h(k_1) = h(k_2)$

inevitabile se $m \ll |U|$

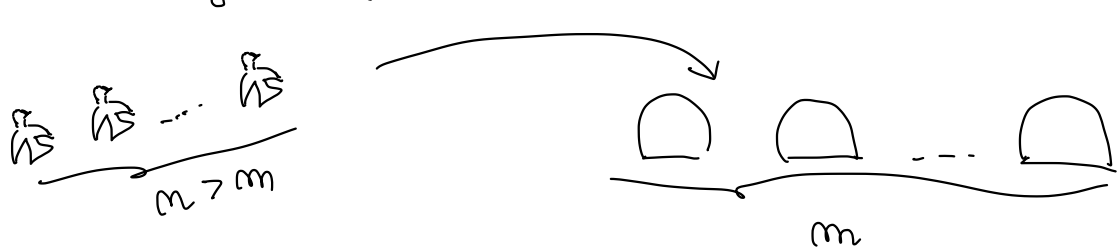
indico con n = num. di elementi memorizzati

se $n > m$ $\Big\} \to$ collisione

PIGEON HOLE PRINCIPLE

Dati A, B finiti $|A| > |B|$

ogni funzione ogni $f : A \to B$ non iniettiva



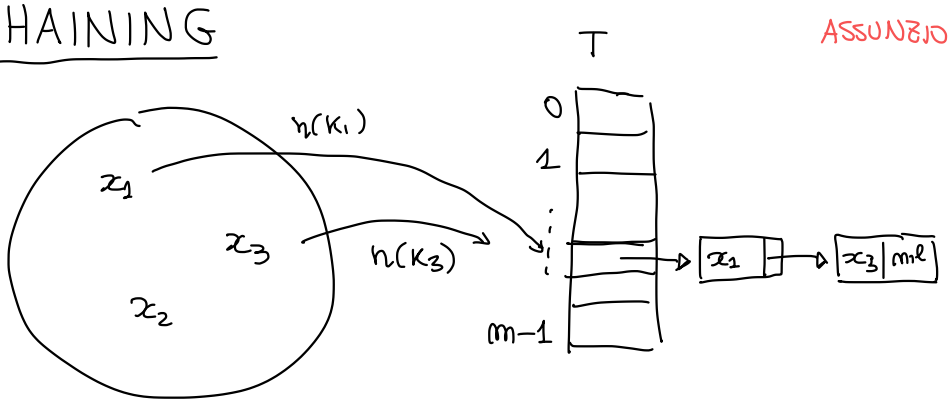
2 possibili soluzioni

(A) chaining

(B) open addressing

(A) CHAINING

ASSUNZIONE: h (funzione di hash) si calcola in $O(1)$



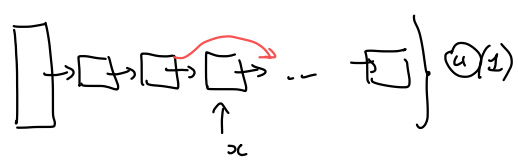
$T[0, \ldots, m-1]$ contiene liste

$T[j]$ = lista degli elementi x t.c. $h(x.key) = j$

Operazioni

Insert (T, x)
inserisce x nella lista $T[h(x.key)]$ (in testa) } $\Theta(1)$

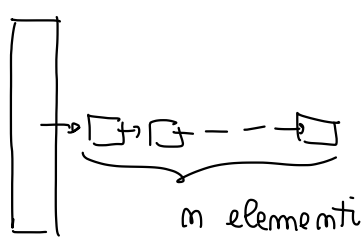
Delete (T, x)
cancella x dalla lista $T[h(x.key)]$



} $\Theta(1)$

Search (T, k)
cerca un elemento con chiave k nella lista $T[h(k)]$

caso peggiore $\Theta(n)$



CASO MEDIO ?

m = # celle tabella
n = # elementi inseriti

$$\boxed{\alpha = \frac{n}{m}}$$ < 1, $= 1$, > 1

↑ fattore di carico

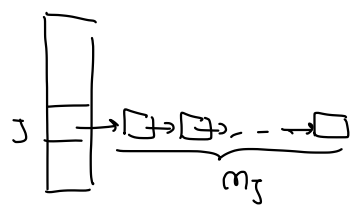
↳ qualità della funzione hash

distribuzione dell'input

<u>ipotesi</u> : HASHING UNIFORME SEMPLICE

ogni elemento x in input ha la stessa probabilità $1/m$ di essere "indirizzato" in una qualunque delle m celle

* indicato con n_j = lunghezza lista in $T[j]$



$x_1 \quad x_2 \; \dots \; x_m$

$$E[n_j] = \sum_{i=1}^{m} \frac{1}{m} \cdot 1 = \frac{n}{m} = \alpha$$

* <u>costo medio search</u>

(•a•) Ricerca di una chiave non presente k

→ calcolo $h(k) = j$
→ accedo alla lista $T[j]$ } $\Theta(1)$
→ scorro la lista $T[j]$ fino in fondo (n_j elementi) } n_j

$$\Theta(1 + n_j)$$

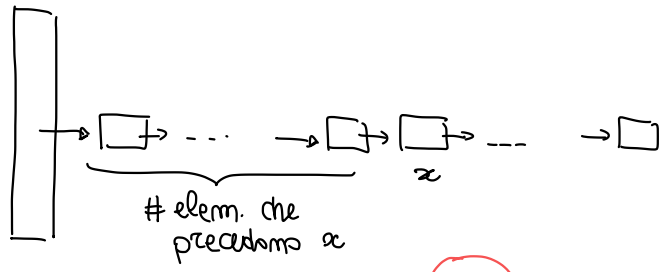
media $\Theta(1 + \alpha)$

(•b•) Ricerca di una chiave presente

→ calcolo $h(k) = j$
→ accedo alla lista $T[j]$ } $\Theta(1)$
→ scorro la lista $T[j]$ fino a trovare l'elem. con chiave k

$$1 + \frac{n_j}{2} \xrightarrow{\text{media}} 1 + \frac{\alpha}{2}$$

più precisi: cerco x con chiave $x.key = k$



elem. che precedono x
= # elem. inseriti dopo di x con stesso hashing di chiave

siano $x_1 \dots x_m$ gli elementi inseriti

tempo medio per cercare un elemento presente è

$$\frac{1}{m}\sum_{i=1}^{m} \underbrace{\text{tempo atteso per trovare } x_i}_{\text{valore atteso del \# elem. inseriti dopo } \underbrace{\text{di}}_{x_{i+1} \dots x_m} x_i \text{ con stesso hashing}}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\sum_{j=i+1}^{m} \underbrace{\text{prob } h(x_i.key) = \underbrace{h(x_j.key)}}_{1/m}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\underbrace{\sum_{j=i+1}^{m}}_{m-i \text{ termini}} \frac{1}{m} = \frac{1}{m}\sum_{i=1}^{m}\frac{m-i}{m} = \frac{1}{mm}\underbrace{\sum_{h=0}^{m-1} h}_{\frac{m(m-1)}{2}}$$

$$= \frac{m(m-1)}{2mm} = \overset{\alpha}{\frac{m}{2m}} - \overset{\alpha}{\frac{1 \cdot m}{2m \cdot m}} = \frac{\alpha}{2} - \frac{\alpha}{2m}$$

$$\Theta\left(1 + \frac{\alpha}{2}\right) = \Theta(1+\alpha)$$

IN CONCLUSIONE: costo della ricerca (in media)

- elemento assente $\Theta(1+\alpha)$
- elemento presente $\Theta\left(1+\frac{\alpha}{2}\right)$

$\Theta(1)$

NOTA: se assumiamo $n \leq c\,m$ (c costante)

$\alpha = \frac{n}{m} \leq c$

* Come definire (buone) funzioni di hash?

NOTA: se le chiavi fossero numeri reali $K \in [0,1)$

$$h(K) = \lfloor m K \rfloor$$

* Metodo della divisione

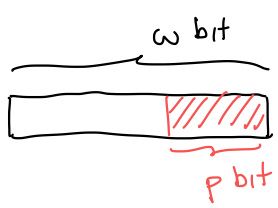
chiavi $U = \{0, 1, \dots, |U|-1\}$

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

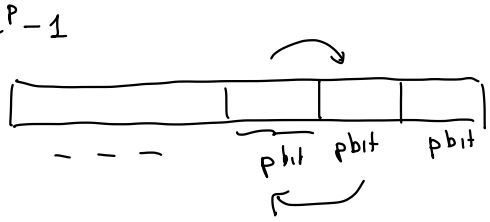
$$h(K) = K \bmod m$$

→ scelta di m è critica

→ $m = 2^p$



→ $m = 2^p - 1$



$h(K) = h(K')$ se K si ottiene "permutando" K'

numero divisibile per 9 sse Σ cifre divisibile per 9

numero in base b

$$a_1 \dots a_n \bmod (b-1) = \left(\sum_{i=1}^{n} a_i\right) \bmod (b-1)$$

ESERCIZIO

* Metodo della moltiplicazione

se $x \in [0,1)$ $h(x) = \lfloor m x \rfloor$

ma la nostra chiave è $K \in \{0, 1, \dots, |U|-1\}$

dato K

→ fisso costante $A \in (0, 1)$ $\quad (0 < A < 1)$

→ $K \cdot A = \text{-----}, \underbrace{\text{-----}}$

$\boxed{K \cdot A \mod 1}$

$(K \cdot A - \lfloor K \cdot A \rfloor)$

$$h(K) = \lfloor m \, (K \cdot A \mod 1) \rfloor$$

PRO :

→ m non critico

→ A " " $\quad \left(A = \frac{\sqrt{5} - 1}{2} \right)$

CONTRO : costo ?

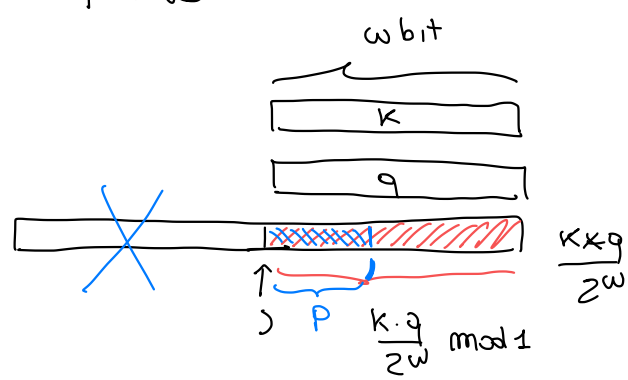
NO

se $m = 2^p$

$A = \frac{q}{2^w}$ $\qquad 0 < q < 2^w$

w = # bit parola

$K \cdot A = K \cdot \frac{q}{2^w}$



$$h(K) = \lfloor m \, (K A \mod 1) \rfloor$$

$$= \lfloor 2^p \underbrace{\frac{Kq}{2^w} \mod 1} \rfloor$$

= p bit più significativi della parola meno significativa di $K \times q$