

Algoritmi e Strutture Dati

Lezione 30

1° dicembre 2025

Tabelle Hash

TABELLE HASH : idea generale

- DIZIONARIO memorizzato in un array (tabella)
- La POSIZIONE di un elemento viene applicando una funzione alla chiave



funzione hash

FUNZIONE HASH

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

trasformazione di chiavi

in indici (posizioni)

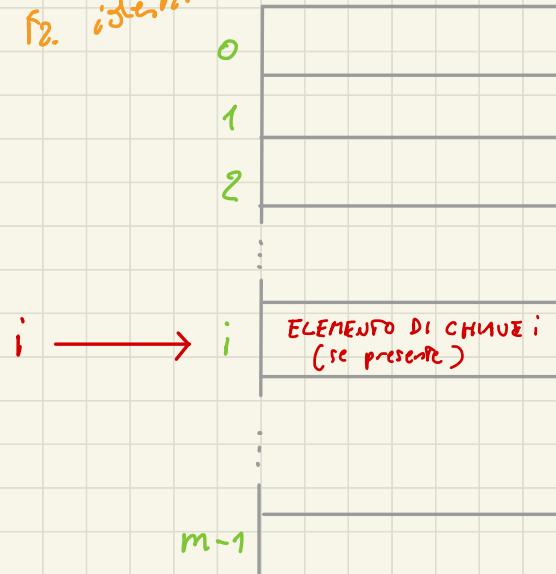
chiave \xrightarrow{h} posizione

0	
1	
2	
3	
⋮	
$m-1$	

TABELLE AD ACCESSO DIRETTO : un caso semplice

- Chiavi univoche intere nel range $0..m-1$ $\cup = \{0, \dots, m-1\}$

\Rightarrow ARRAY di m posizioni
indice \leftrightarrow chiave $h = f_2.$ identific.



RICERCA
INSERIMENTO
CANCELLAZIONE } Tempo $O(1)$

Studenti \rightarrow n° matricola n° nel range 0..999999

\downarrow
n studenti

che seguono obiettivo ≤ 200

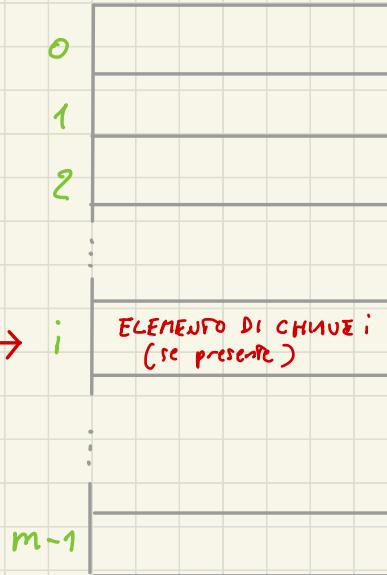
$$U = \{0, \dots, 999999\}$$

$$m = 1000000$$

$$n = 200$$

$$\frac{n}{m} = \frac{200}{1.000.000} = 0.02\%$$

$i \longrightarrow i$ ELEMENTO DI CHIAVE
(se presente)



FATTORE DI CARICO α

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

quando la tabella è piena

$n \ll m \rightarrow \alpha \approx \frac{1}{m}$

$\alpha = 1$ tabella piena

$\alpha = 0$ tabella vuota

FUNZIONI HASH

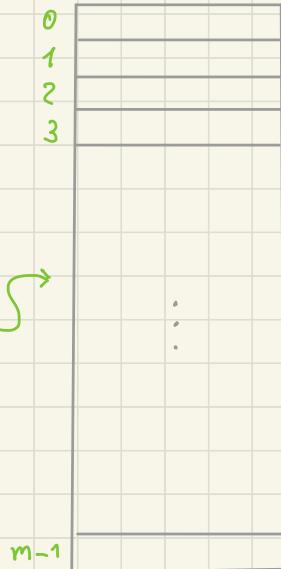
es $V = \{950\cdot000, \dots, 999\cdot000\}$ $n = 200$

- $m = 50\cdot000$

$$h(x) = x - 950\cdot000$$

$$\alpha = \frac{n}{m} = \frac{200}{50\cdot000} = \frac{2}{500} = 0.4\%$$

chiave \xrightarrow{h} posizione



$n = 20$ persons

CHIUSE: cognome

$U = \text{Cognomi}$

initially

'A' → 0

'B' → 1

'C' → 2

⋮

'z' → 25

FUNZIONI HASH

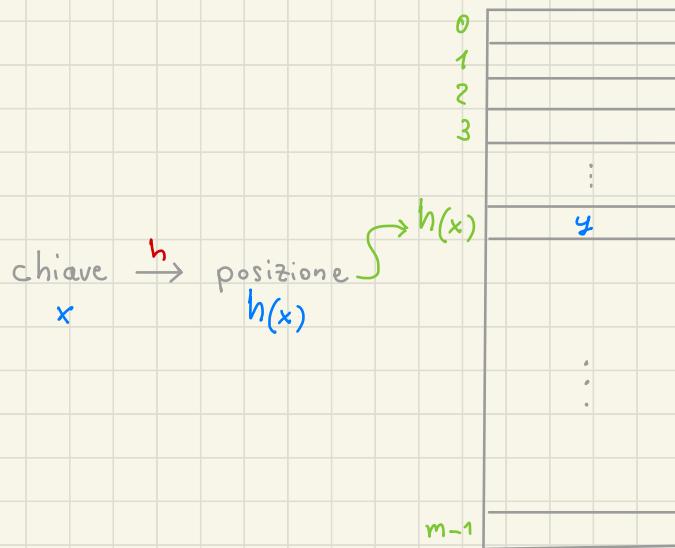
Nella pratica, salvo casi particolari:

- il numero di chiavi possibili ($\#U$)
è molto più grande del numero di chiavi attese
- la dimensione della tabella (m) viene scelta paragonabile
al numero di chiavi attese

Cognomi \leftrightarrow Stringhe $U = \text{Stringhe di 10 lettere}$

$$\#U = 26^{10} > 10^{14} \approx 141167 \text{ miliardi}$$

COLLISIONI



COLLISIONE:

Quando dovrando inserire o cercare una chiave x , la posizione $h(x)$ è occupata da un'altra chiave y .

CONVIVERE CON LE COLLISIONI

- FARE IN MODO CHE AVVENGANO RARAMENTE

In deve "sparagliare" le chiavi nella tabella
in modo che le collisioni siano poco probabili

⇒ SCELTA DELLA FUNZIONE HASM

- UNIFORME
- VELOCE DA CALCOLARE
- AVERE UNA STRATEGIA PER GESTIRE LE COLLISIONI
nel caso esse accadano

Funzioni Hash

SPARAGLIAMENTO e UNIFORMITA'

- $h: U \rightarrow \{0, \dots, m-1\}$ funzione hash

- Per $x \in U$:

$P(x)$: probabilità che, scegliendo a caso una chiave da U , si scelga x

- Per $i = 0, \dots, m-1$:

$\varphi(i) = \sum_{x: h(x)=i} P(x)$ probabilità che una chiave scelta a caso da U abbia valore hash i

La funzione hash h è UNIFORME se $\varphi(i)$ è la stessa

per ogni i , cioè $\varphi(i) = \frac{1}{m}$

SPARPAGLIAMENTO e UNIFORMITA'

Esempio

$$U = [0, 1) \quad \text{numeri reali}$$

$$m = 10$$

$$h: U \rightarrow \{0, \dots, 9\}$$

$$h(x) = \lfloor 10x \rfloor$$



h è uniforme se ogni x con $0 \leq x < 1$ ha la stessa probabilità di essere scelto

SPARAGLIAMENTO e UNIFORMITA'

Esempio

$U = \text{Cognomi}$

$m = 26$

$h: U \rightarrow \{0, \dots, 25\}$

$h(x) = \text{ord}(\text{1}^{\text{a}} \text{ lettera di } x)$

$$\text{ord}('A') = 0$$

$$\text{ord}('B') = 1$$

$$\text{ord}('C') = 2$$

\vdots

$$\text{ord}('Z') = 25$$

NON UNIFORME

REQUISITI DI UNA BUONA FUNZIONE HASH

- UNIFORME
- VELOCE DA CALCOLARE

ESEMPI DI FUNZIONI HASH

Metodo della divisione

- ① TRASFORMO LA CHIAVE IN UN INTERO
- ② DIVIDO PER LA DIMENSIONE DELLA TABELLA
e CONSIDERO IL RESTO

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

↑ in binario

di serie
potenza di 2

ESEMPI DI FUNZIONI HASH

Metodo della divisione

- ① TRASFORMO LA CHIAVE IN UN INTERO
- ② DIVIDO PER LA DIMENSIONE DELLA TABELLA
e CONSIDERO IL RESTO

$U = \text{parole lingua italiana}$

$m = 2^{10}$

$x = \text{GATTO} = 010001110100000101010100010101000100111$

The binary string '010001110100000101010100010101000100111' is shown with green curly braces underlining groups of five bits each. Below it, five labels are aligned with these groups: 71(G), 65(A), 84(T), 84(T), and 79(O). This illustrates how each character in the string is mapped to a specific index in the hash table.

71(G) 65(A) 84(T) 84(T) 79(O)

$$h(\text{'GATTO'}) = 79 = 0001001111$$

FACILE di colpire

UNIFORME? NO

- In genere la dimensione della tabella è una potenza di 2
- Il valore della funzione hash deve dipendere da tutta la chiave e non solo da una parte di essa

HASH \equiv sminuzzare / triturare

Esempio

$U = \text{parole lingua italiana} \quad m = 2^{10}$

$s = s_0 s_1 \dots s_{n-1}$ parola di n lettere

$$h(s) = \sum_{i=0}^{n-1} s_i \cdot 31^{n-i-1} \bmod 2^{10}$$

$s = \text{GATTO} = \underbrace{01000111}_7 \underbrace{01000001}_5 \underbrace{01010101}_9 \underbrace{001010101000}_8 \underbrace{0100111}_2$

71(G) 65(A) 89(T) 89(T) 79(O)

$$\begin{aligned} h(\text{'GATTO'}) &= (71 \cdot 31^4 + 65 \cdot 31^3 + 89 \cdot 31^2 + 89 \cdot 31^1 + 79 \cdot 31^0) \bmod 2^{10} \\ &= 67589813 \bmod 2^{10} = 693 \end{aligned}$$

BUONA UNIFORMITÀ

TEMPO DI CALcolo ELEVATO

Metodo del ripiegamento

- m potenza di 2, $m = 2^k$
- x chiave in binario
- suddivisa in blocchi x_1, \dots, x_k di k bit ciascuno
- $f(x) = x_1 \oplus x_2 \oplus \dots \oplus x_k$

$$m = 2^{10}$$

$x = \text{GATTO} = 010001110100000101010100001010001001111$

$$\begin{cases} x_1 = 0100011101 \\ x_2 = 0000010101 \\ x_3 = 0000010101 \\ x_4 = 0001001111 \end{cases}$$

blocchi di lunghezza 10

$$f(x) = 0101010010 = 338$$

Facile da calcolare

buona uniformità

(OR-exclusivo: ottima funzione blanda)

Gestione delle collisioni

GESTIONE DELLE COLLISIONI

Collisione

La posizione che dovrebbe essere utilizzata per una chiave x è già occupata da una chiave differente y

Tecniche di gestione

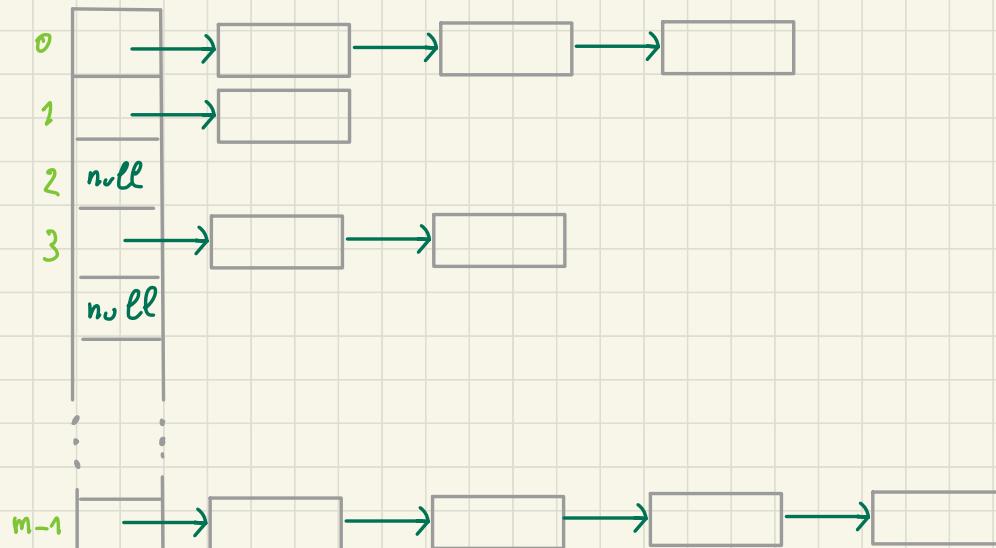


esterne

interne

GESTIONE DELLE COLLISIONI ESTERNA

Liste di collisione (concatenamento esterno)



Posizione i : contiene ogni record la cui chiave x ha
valore hash i , cioè $h(x) = i$

L1STE DI COLLISIONE: esempio

0 :
⋮ :

14

15

16

17

18

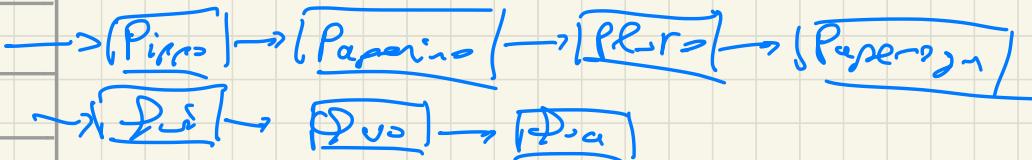
19

20

21

22

23



$U = \text{Stringhe di Lettere}$

$h(x) = \text{ord}(\text{1}^{\text{a}} \text{ Lettera di } x)$

$\text{ord}('A') = 0$

$\text{ord}('B') = 1$

⋮

$\text{ord}('P') = 15$

$\text{ord}('Q') = 16$

$\text{ord}('R') = 17$

$\text{ord}('S') = 18$

$\text{ord}('T') = 19$

⋮

$\text{ord}('Z') = 25$

inserire

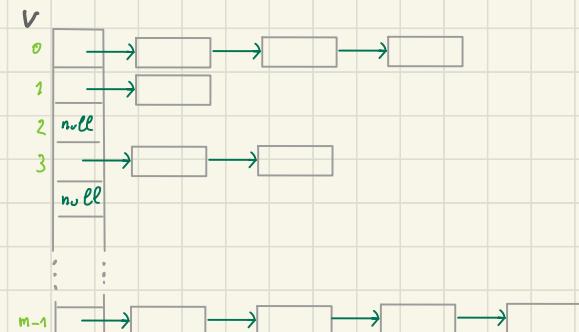
Pippo, Paperino, Topolino, Qui

Pluto, Duo, Qua, Paperoga

LISTE DI COLLISIONE

Operazioni

- Inserimento (elemento e, chiave k)
 $O(1)$ se inseriamo all'inizio



- Ricerca (chiave k) \rightarrow elemento

passi: dipende dalla lunghezza delle liste

Caso peggiore $O(n)$

in media lunghezza lista $\frac{n}{m} = \alpha$

- Cancellazione (chiave k)

$$\rightarrow T_{avg}(n) = O\left(1 + \frac{n}{m}\right)$$

tempo medi.

(α possibile avere $\alpha > 1$)

AGGLOMERAZIONE

↓

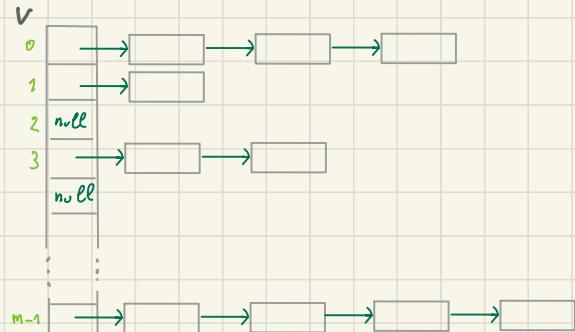
presenta di zone in cui si concentrano molti dati.

Se la funzione hash non s'impaglia
bene si potrebbero formare alcune
liste molto lunghe

Esempio

Chiavi \leftrightarrow cognome

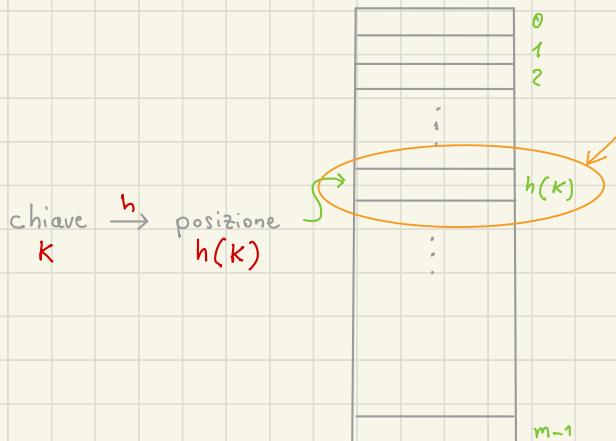
F₂ hash \leftrightarrow ord (iniziale)



GESTIONE DELLE COLLISIONI INTERNA

Indirizzamento aperto

inserimento di elemento di chiave K



se $h(K)$ è già occupata
si cerca un'altra posizione
libera utilizzando una
strategia predefinita

SCANSIONE LINEARE;

scopo: prima posizione
libera successiva [mod m]

SCANSIONE LINEARE: esempio

0	.
:	:
14	
15	Pippo
16	Paperino
17	Qui
18	Pluto
19	Topolino
20	Queso
21	Qua
22	Paperoga
:	:
25	

U = Stringhe di lettere

$h(x) = \text{ord}(\text{1}^{\text{a}} \text{ lettera di } x)$

$\text{ord}('A') = 0$

$\text{ord}('B') = 1$

:

$\text{ord}('P') = 15$

$\text{ord}('Q') = 16$

$\text{ord}('R') = 17$

$\text{ord}('S') = 18$

$\text{ord}('T') = 19$

:

$\text{ord}('Z') = 25$

inserire

Pippo, Paperino, Topolino, Qui

Pluto, Queso, Qua, Paperoga

AGGLOMERAZIONE PRIMARIA

$$c(k, i) = (h(k) + i) \bmod m$$

INDIRIZZAMENTO APERTO

Funzione ausiliaria $c(K, i)$

chiave intero ≥ 0
 ordine nella scansione

La funzione ausiliaria deve verificare:

$$c(K, 0) = h(K)$$

$$\{c(K, 0), c(K, 1), \dots, c(K, m-1)\} = \{0, \dots, m-1\}$$

FUNZIONI AUSILIARIE : Esemp:

Scansione Lineare

$$c(k, i) = (h(k) + i) \bmod m$$

Possibilità di AGGLOMERAZIONE PRIMARIA

FUNZIONI AUSILIARIE: Esemp:

scansione quadratica

$$c(k, i) = \lfloor h(k) + c_1 i + c_2 i^2 \rfloor \text{ MOD } m$$

i	$\frac{k}{2}; \frac{i}{2}; i^2$
0	0
1	1
2	3
3	6
...	...

con c_1, c_2 opportuni (es. $c_1 = c_2 = 0.5$)

NOTA

Se $h(k_1) = h(k_2)$ la sequenza di celle scansionata per k_1 e k_2 è la stessa

Possibilità di AGGLOMERAZIONE SECONDARIA

HASHING DOPPIO

$$c(k, i) = (h(k) + i h'(k)) \bmod m$$

dove $h' : U \rightarrow \{0, \dots, m-1\}$ è una seconda funzione hash

Situazione ideale:

$$h(k_1) = h(k_2) \Rightarrow h'(k_1) \neq h'(k_2)$$

Possibilità di AGGLOMERAZIONE SECONDARIA

Inserimento (elemento e, chiave k)

$v[0..m-1]$ tabella

$i \leftarrow 0$

WHILE $i < m$ AND $v[c(k, i)]$ è occupata DO

$i \leftarrow i + 1$

IF $i < m$ THEN $v[c(k, i)] \leftarrow (e, k)$ // inserisci elemento

ELSE errore // tabella piena

Ricerca (chiave k) \rightarrow elemento

$v[0..m-1]$ tabella

$i \leftarrow 0$

WHILE $i < m$ AND $v[c(k, i)]$ è occupata DO

AND $v[c(k, i)].chiave \neq k$ DO

$i \leftarrow i + 1$

IF $i = m$ OR $v[c(k, i)]$ è libera THEN

RETURN null // non c'è

ELSE

RETURN $v[c(k, i)].elemento$ // trovato

Cancellazione (chiavi K) ?


cancellazione logico

Esempio

$$f : \{a, \dots, z\} \rightarrow \{0, \dots, 15\}$$

x	f(x)
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7
j	7
k	7
l	8
m	9
n	10
o	10
p	11
q	12
r	12
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

funzione hash:

$$h(k) = f(\text{prima lettera di } k)$$

scansione lineare

inserire nell'ordine:

salmone

tonno

rombo

aringa

cernia

aragosta

totano

ostrica

nasello

palombo

anguria

merluzzo

tonno

0	ARWEG
1	ARAGOSTA
2	CERNIA
3	PALOMBO
4	
5	
6	
7	
8	
9	
10	OSTRICA
11	NASELLO
12	TONNO
13	SALMONE
14	TONNO
15	TOTANO

Esempio

← prossima lezione!

$$f : \{a, \dots, z\} \rightarrow \{0, \dots, 15\}$$

x	f(x)
a	0
b	1
c	2
d	3
e	4
f	5
g	6
h	6
i	7
j	7
k	7
l	8
m	9
n	10
o	10
p	11
q	12
r	12
s	13
t	13
u	14
v	14
w	15
x	15
y	15
z	15

funzioni hash:

$$h(k) = f(\text{prima lettera di } k)$$

$$g(k) = \begin{cases} \text{più piccolo numero } p \\ \text{t.c. } p \text{ è primo, } p \neq 2 \text{ e} \\ p \geq f(\text{terza lettera di } k) \end{cases}$$

hashing doppio con funzione:

$$c(k, i) = (h(k) + i \cdot g(k)) \bmod 16$$

inserire nell'ordine:

salmone
 tonno
 rombo
 aringa
 cernia
 aragosta
 totano
 ostrica
 nasello
 palombo

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	