Titolo nota 17/03/2016

Dizionario

Ricerca
Inserzione
Cancellazione

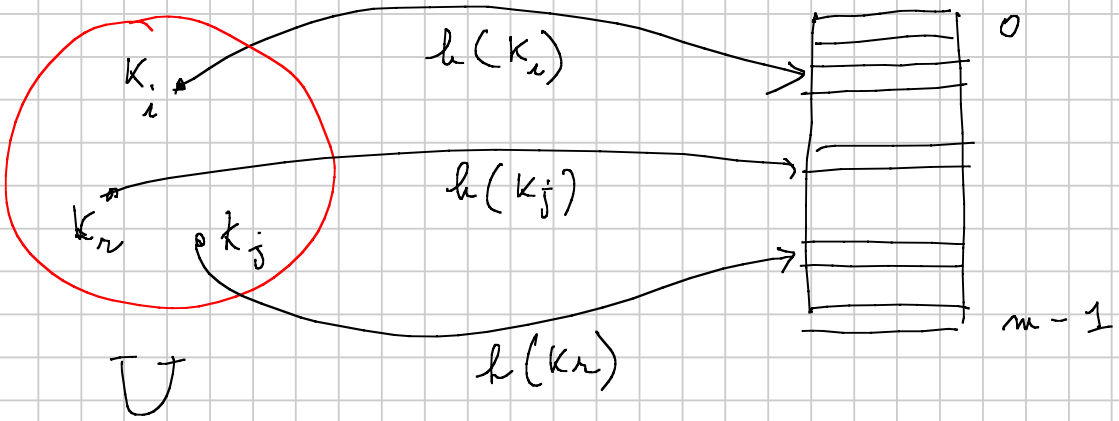
Heap Ricerca (max o min)

x x.key x.info

Tabelle Hash (Tavole Hash)
Hash Table

Titolo nota 17/03/2016

n numero di elementi m dimensione della tabella



$n \ll m$

h: funzione hash

U dimensione molto grande $\quad h : U \longrightarrow [0, 1, \ldots, m-1]$

16 $\quad k_i, k_j \quad k_i \neq k_j \quad h(k_i) = h(k_j)$ collisione

Titolo nota 17/03/2016

Tabella hash

- dimensionamento della tabella : m
- funzione hash
- tecnica di risoluzione delle collisioni

INDIRIZZAMENTO DIRETTO → capita raramente

universo possibili chiavi ⟷ coincide con le posizioni della tabella

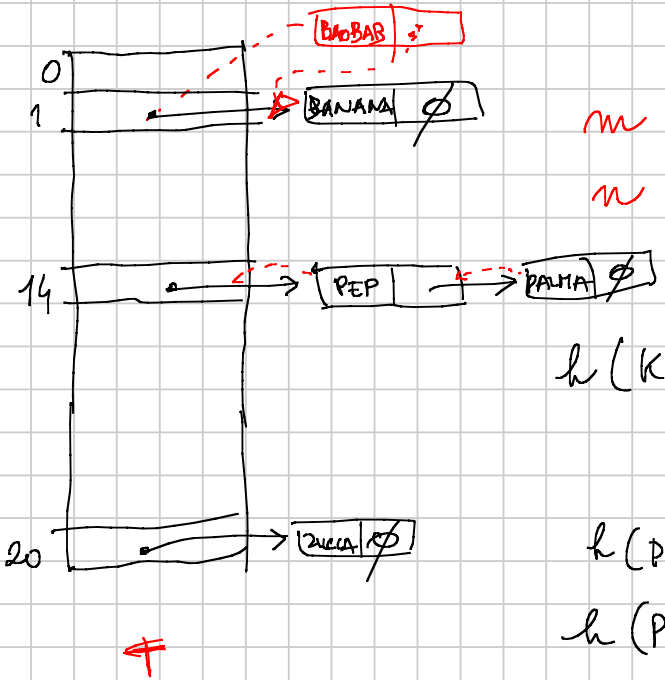
$$\forall \quad k_i \neq k_j \Rightarrow h(k_i) \neq h(k_j)$$

Ricerca
Inserzione
Cancellazione
} un accesso alla tabella $\Theta(1)$

$k_i \neq k_j$ $\quad h(k_i) = h(k_j)$

metodo di concatenazione

x x.key
   x.info



n
m
$\frac{n}{m}$

CHAINED-HASH-SEARCH(T, k)
ricerca nella lista $T[h[k]]$

CHAINED-HASH-INSERT(T, x)
inserzione in testa di x nella lista $T[h[x.key]]$

Ricerca ?

$O(n)$

caso pessimo funzionano molto male

$n > m$

$h(k_i)$ = posizione nell'alfabeto italiano della lettera iniziale

$h(\text{PALMA}) = 14$

$h(\text{PEPERONE}) = 14$

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

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

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

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

$=$ fattore di carico

ricerca senza successo:

caso $n = 2m$ ricerca $= 3$

accesso alla tabella
scansione completa della lista
$\Rightarrow 1 + \frac{n}{m} = 1 + \alpha$

ricerca
el.
inserzione
hanno costo $\Theta(1)$ se n e m sono scelti opportunamente

Titolo nota 17/03/2016

Come si definisce la funzione hash

chiave numerica .

codifica ASCII

ogni carattere è un numero

in base 128

pt $p = 112$ $t = 116$

Metodo della divisione

$h(k) = K \% m$

m numero primo

$h(K)$ r bit

$m = 2^r$

Titolo nota 17/03/2016

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

K si divide in s blocchi di r bit ciascuno e si calcola l'OR esclusivo ($\oplus$) dei blocchi

$r = 4$

$X \oplus Y =$

K_0	K_1	K_2	K_3
0011	0101	1001	0000

$h(K)$ di 168 bit

SHA-1

```
0011
0101
0110
1001
1111
```

```
1001
0000
1001
```

- h deve poter colpire tutte le posizioni della tabella

deve distribuire uniformemente i dati nella tabella. Ciascun indirizzo di T deve

probabilità $\frac{1}{m}$ di essere generato.

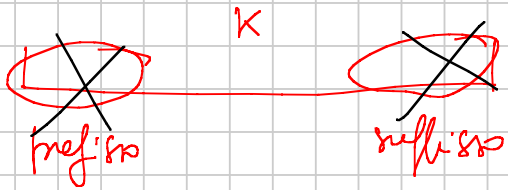


tabella dei simboli
Compilatore

La funzione hash deve essere indipendente da qualsiasi forma di regolarità ci sui sui dati.

OPEN HASH
TABELLE A INDIRIZZAMENTO APERTO

} tutto viene memorizzato all'interno della tabella

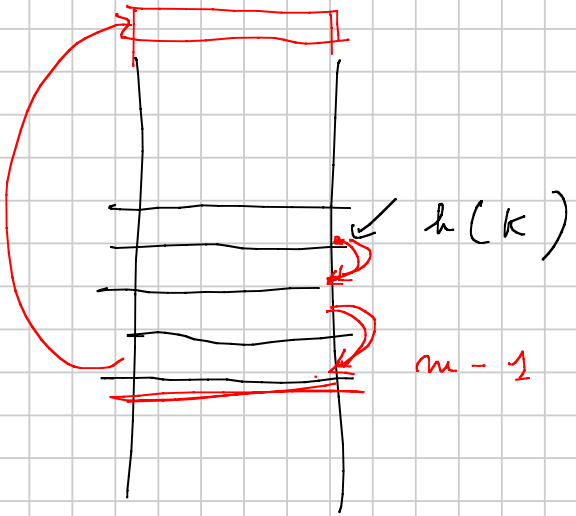
$n < m$

non vogliamo che la tabella si riempia troppo

✓

Sequenza di scansione : sequenza di posizioni della tabella da controllare

$h(K) =$



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

↑

funzione hash

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

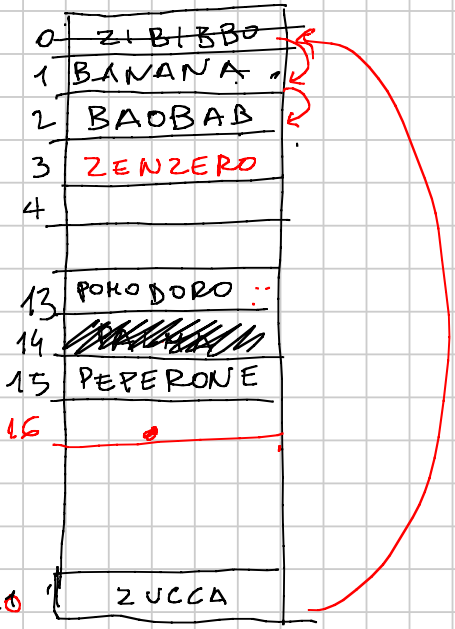
lineare a passo 1

Titolo nota 17/03/2016

$m = 21$

T

Cancella Zibibbo



$Pr(h(k) = i) = \frac{1}{m}$

Cancello PALMA

Ricerco PEPERONE

NIL

$Pr(h(k) = j \; + \; h(k) = j+1 \; , \; \dots \; + \; h(k) = i) = \frac{i - j + 1}{m}$

$h(k)$ = posizione iniziale nell'alfabeto

è una buona funzione hash ??

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

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

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

$(h(k_0) + i) \% m$

$h(\text{BANANA})$

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

~~$h(\text{ZIBIBBO}, 1) = (20 + 1) \bmod 21 = 0$~~

$h(\text{ZENZERO})$

$h(\text{BAOBAB})$

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

Ricerca di BIETOLA

elemento ≡ corrisponde alla chiave

```
HASH-INSERT (T, k):
i = 0;
repeat {
  j = h(k, i);
  if T[j] == NIL {
      T[j] = k;
      return j;
  } else i++;
} until i == m
error overflow tabella
```

$O(n)$

```
HASH-SEARCH (T, k)
i = 0;
repeat {
  j = h(k, i);
  if T[j] == k return j
      else i++
} until (i == m or T[j] == NIL)
return NIL;
```

$O(n)$ caso pessimo

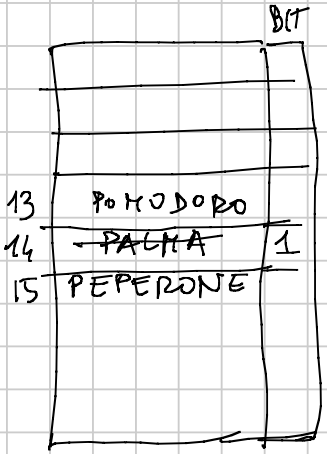
Titolo nota 17/03/2016

CANCELLAZIONE:

1) cancellazione logica

⇓

ristrutturazione con getta una tantum



2) Ristrutturazione degli agglomerati

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

OPEN HASH

scansione lineare

{ inserzione
ricerca

~~cancellazione~~