tabelle di hash

mercoledì 7 agosto 2024 10:02

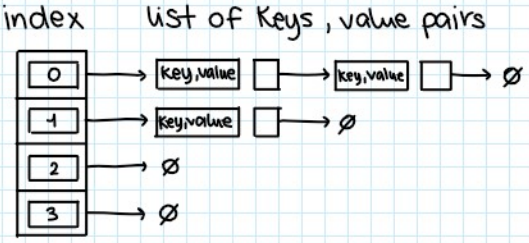
dizionari

chiamati anche "mappe" o "array associativi", servono ad implementare funzioni che non si possono calcolare "al fly". Ad esempio, la funzione che associa ad ogni parola il suo significato (chiave = parola, elem = significato)

TABELLE DI HASH PER IMPLEMENTARE DIZIONARI

Le tabelle di hash nascono per ovviare al problema dello spreco di spazio delle tabelle ad accesso diretto. Le chiavi possono non essere numeri e il loro ordine può essere casuale. Per accedere alla cella dell'array associata ad una chiave esistono delle funzioni (di hash) che trasformano attraverso un algoritmo la chiave in un numero. Ad esempio, se la chiave è una stringa, una funzione di hash potrebbe essere la parte intera del rapporto tra la somma dei valori ascii delle lettere componenti la stringa e la size dell'array.
Una buona funzione di hash dovrebbe distribuire gli elementi uniformemente all'interno dell'array. Ad esempio se hai 100 chiavi ed hai a disposizione 10 celle, una buona funzione di hash associa 10 chiavi ad ogni cella.
La funzione di hash deve essere calcolabile in tempo costante altrimenti si perde il vantaggio di usare questa struttura dati.
Se volessimo aggiungere un elemento nella tabella in una posizione già occupata si crea una collisione. Per risolverla, si creano delle liste semplici (di collisione o bucket) in cui vengono memorizzati tutti gli elementi associati alla stessa cella dell'array e all'interno di quest'ultima si inserisce il puntatore alla lista.
Una funzione di hash è perfetta quando non ci sono collisioni.
Una funzione di hash perfetta deve essere iniettiva e calcolabile in tempo costante.

Liste di collisione



Gli elementi sono contenuti in liste esterne alla tabella (chiamate bucket).
V[i] punta alla lista degli elementi tali che h(k) = i

grado di riempimento di una tabella:

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

n → numero elementi
m → numero celle tabella

ALGORITMI

- **inserimento:** Se v[h(k)] è vuota, inserisci la coppia (el, K) in tale posizione; atrimenti, a partire da h(k), ispeziona le celle della tabella e inserisci nella prima cella vuota
- **ricerca:** Se, durante la scansione delle celle, ne viene trovata una con la chiave cercata, restituisci l'elemento trovato; altrimenti, se si è scandita la tabella senza successo, restituisci null

esercizio su aulaweb

Si consideri una tabella di hash con liste di collisione che implementa un dizionario **D** in cui le chiavi sono sequenze di 4 cifre decimali che indichiamo con **a, b, c, d** e i valori sono stringhe.
La tabella di hash ha 7 bucket indicizzati da 0 a 6 e la funzione di hash è **h: $(a^2 + b + c + d) \bmod 7$**.

[Preparazione dei dati, senza voto MA LA PRESENZA DI DUE O PIU' ERRORI NEL CALCOLO DELLA FUNZIONE DI HASH COMPORTA L'ASSEGNAZIONE DI ZERO PUNTI ALL'INTERO ESERCIZIO]

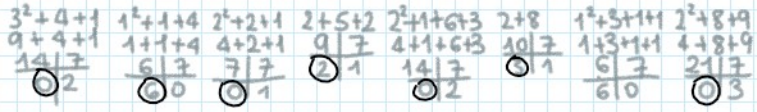
Completate su questo foglio lo schema mostrato sotto calcolando la funzione di hash delle chiavi scritte nella prima riga e riportandola nella corrispondente cella nella terza riga:

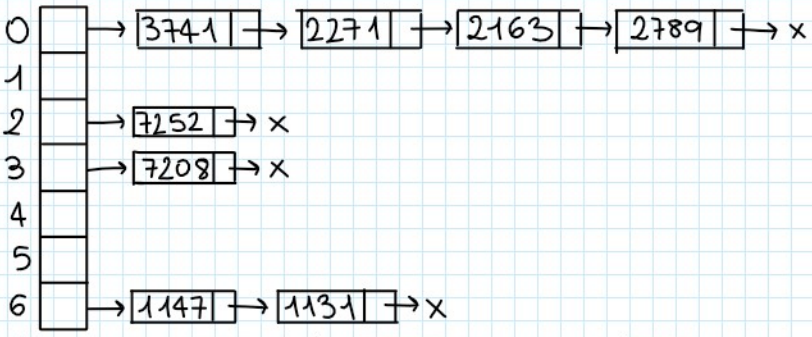
chiave = 3741	chiave = 1147	chiave = 2271	chiave = 7252	chiave = 2163	chiave = 7208	chiave = 1311	chiave = 2789
valore = "ma"	valore = "co"	valore = "gra"	valore = "pe"	valore = "le"	valore = "re"	valore = "ca"	valore = "giu"
h(chiave) =	h(chiave) =	h(chiave) =	h(chiave) =	h(chiave) =	h(chiave) =	h(chiave) =	h(chiave) =

- **D3.1:** Disegnate la tabella di hash che si ottiene dagli inserimenti delle stringhe "ma", "co", "gra", "pe", "le", "re", "ca", "giu" associate alle chiavi su quattro cifre riportate nello schema sopra, nell'ordine in cui compaiono nello schema da sinistra verso destra. Nel caso ci fossero delle **chiavi** duplicate, ignorate l'inserimento e scrivete esplicitamente sul foglio "questa coppia chiave-valore è stata ignorata perché la chiave è duplicata".

3741 ma	1147 co	2271 gra	7252 pe	2163 le	7208 re	1311 ca	2789 giu
0	6	0	2	0	3	6	0

h: $(a^2 + b + c + d) \bmod 7$.





Non ci sono chiavi duplicate ma solo collisioni

- **D3.2.1:** Quali sono le due principali "buone proprietà" che una funzione di hash deve avere?
- **D3.2.2:** La funzione di hash **h** applicata alle chiavi nello schema disegnato sopra gode della prima di queste due "buone proprietà"? Motivare esaurientemente la risposta.
- **D3.2.3:** La funzione di hash **h** applicata alle chiavi nello schema disegnato sopra gode della seconda di queste due "buone proprietà"? Motivare esaurientemente la risposta.

3.2.1:
- la prima proprietà è quella di essere calcolabile in un tempo costante
- deve essere uniforme, l'uniformità è la proprietà per cui tutte le chiavi vengono distribuite nel bucket. La funzione h non deve "sovrapporre" alcune celle della tabella, lasciandone alcune vuote

3.2.2:
La prima proprietà è soddisfatta perchè la nostra funzione di hash in h richiede operazioni aritmetiche semplici: somma, moltiplicazione e modulo. Queste operazioni sono generalmente considerate operazioni a tempo costante poichè il tempo di esecuzione non dipende dalla dimensione dell'input. Inoltre, in questo caso le chiavi sono sequenze di 4 cifre decimali che vanno da 0 a 9 quindi anche a livello di calcolo non sono numeri elevati.

Inoltre, in questo caso le chiavi sono sequenze di 4 cifre decimali che vanno da 0 a 9 quindi anche a livello di calcolo non sono numeri elevati.

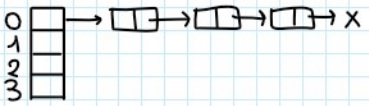
3.2.3: Non viene rispettata la seconda proprietà perchè alcune chiavi generano più frequentemente lo stesso valore di hash, come evidenziato dal fatto che alcuni bucket contengono più chiavi rispetto ad altri.

[Domanda 3.3, 1/3 del punteggio] Sia **m** la dimensione della tabella e sia **n** il numero di elementi nella tabella. Si assuma che **m** ed **n** siano entrambi diversi da 0.

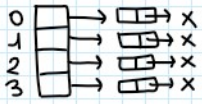
- **D3.3.1**: Si descriva l'algoritmo per l'inserimento di un elemento data la sua chiave; l'algoritmo deve esplicitamente e correttamente gestire il caso di chiavi duplicate.
- **D3.3.2 (la risposta sarà valutata solo se l'algoritmo descritto nella risposta D3.3.1 è corretto):** Sotto quali condizioni l'inserimento ha complessità Theta(n)?
- **D3.3.3 (la risposta sarà valutata solo se l'algoritmo descritto nella risposta D3.3.1 è corretto):** Sotto quali condizioni l'inserimento ha complessità Theta(n/m)?
- **D3.3.4 (la risposta sarà valutata solo se l'algoritmo descritto nella risposta D3.3.1 è corretto):** Sotto quali condizioni l'inserimento ha complessità Theta(1)?

3.3.1: L'algoritmo di inserimento calcola h(chiave), cerca se in quel bucket la chiave è già presente e se non lo è la inserisce

3.3.2: L'inserimento ha complessità theta(n) quando la funzione di hash smista tutte le chiavi in un solo bucket (caso peggiore)



3.3.3: L'inserimento ha complessità theta(n/m) quando la funzione di hash distribuisce le chiavi in maniera uniforme (caso medio)



3.3.4: L'inserimento ha complessità theta(1) quando la funzione di hash è iniettiva, o quando il numero di elementi n è molto più piccolo della dimensione della tavola m, portando a una bassa probabilità di collisioni, o quando è vuota (caso migliore)