Capitolo 8

Tabelle hash

Sono strutture dati realizzate mediante vettori che permettono, nel caso medio, di eseguire operazioni in tempo costante. Sia U l'universo delle chiavi associabili agli elementi che vogliamo memorizzare; nel caso in cui U non sia l'insieme di numeri naturali $[0, m-1]$, l'utilizzo di tavole ad accesso diretto (ossia tavole che usano direttamente la chiave come indice per reperire l'elemento nella tabella) risulta troppo costoso, a causa del *fattore di carico* troppo basso: per questo motivo vengono usate le *funzioni hash* per la trasformazione delle chiavi in indici.

Definizione 8.1. Il *fattore di carico* di una tavola è definito come il rapporto $\alpha = n/m$ tra il numero n di elementi in essa memorizzati e la sua dimensione m.

Definizione 8.2. Una *funzione hash* è una funzione $h : U \to \{0, ..., m-1\}$ che trasforma chiavi in indici di una tavola.

Definizione 8.3. Una funzione hash h è perfetta se è iniettiva, ovvero $\forall u, v \in U, u \neq v \Rightarrow h(u) \neq h(v)$.

Affinché una funzione hash sia perfetta, occorre che $|U| \leq m$, ossia ci dev'essere spazio per tanti elementi quante sono le chiavi possibili: questo comporta un enorme spreco di memoria se l'insieme delle chiavi è molto grande. Se una funzione hash non è perfetta, allora potrebbe verificarsi una *collisione*, ossia si possono avere più chiavi diverse con lo stesso valore associato: per risolvere il problema, occorre operare delle strategie di risoluzione delle collisioni, che hanno lo svantaggio di ridurre le prestazioni.

8.1 Definizione di funzioni hash

Definizione 8.4. Sia

$$Q(i) = \sum_{k:h(k)=i} P(k)$$

la probabilità che, scegliendo una chiave, questa finisca nella cella i; una funzione hash h gode della proprietà di *uniformità semplice* se, $\forall i \in \{0, ..., m-1\}$,

$$Q(i) = \frac{1}{m}$$

Per definire funzioni hash con buone caratteristiche di uniformità, con l'assunzione che ogni chiave abbia la stessa probabilità di essere scelta, si usano spesso le seguenti tecniche.

Metodo della divisione: il metodo calcola il resto della divisione della chiave k per m, dove m è la dimensione della tabella hash; sebbene nella maggior parte dei casi si hanno buoni risultati, in altri potrebbero verificarsi molte collisioni: la bontà del metodo dipende dalla scelta di m, che sarebbe preferibile fosse un numero primo vicino ad una potenza di due, e dal fatto che la funzione hash dovrebbe dipendere da tutti i bit della chiave.

Metodo del ripiegamento: consiste nel dividere la chiave k in l parti e definire la funzione hash come l'applicazione di una funzione f, con codominio $\{0, ..., m-1\}$, sulle parti di chiave ottenute con la divisione; ossia:

$$h(k) = f(k_1, k_2, ..., k_l)$$

8.2 Risoluzione delle collisioni

8.2.1 Liste di collisione

Questo metodo consiste nell'associare a ciascuna cella della tabella hash una lista di chiavi, detta *lista di collisione*, di lunghezza media pari al fattore di carico α; per questo motivo, assumendo che la funzione di hashing goda della proprietà di uniformità semplice, si ha che il tempo medio necessario per un'operazione di ricerca o eliminazione è $O(1+\alpha)$, mentre l'inserimento può essere realizzato in tempo $O(1)$.

8.2.2 Indirizzamento aperto

Nel caso in cui la posizione $h(k)$ in cui inserire una chiave k sia già occupata, il metodo prevede di posizionarla in un'altra cella vuota, anche se quest'ultima potrebbe spettare di diritto ad un'altra chiave. Le operazioni vengono realizzate come segue:

Inserimento:

- se $v[h(k)]$ è vuota, inserisci la coppia (el, k) in tale posizione;
- altrimenti, a partire da $h(k)$, ispeziona le celle della tabella secondo una sequenza opportuna di indici $c(k,0)$, $c(k,1)$, ..., $c(k,m-1)$ e inserisci nella prima cella vuota; la sequenza, chiaramente, deve contenere tutti gli indici $\{0,...,m-1\}$.

Ricerca:

- se, durante la scansione delle celle, ne viene trovata una con la chiave cercata, restituisci l'elemento trovato;
- altrimenti, se si arriva a una cella vuota o si è scandita l'intera tabella senza successo, restituisci *null*.

Cancellazione: affinché la ricerca con il metodo appena descritto funzioni, occorre adottare una strategia particolare per la cancellazione, ossia utilizzare un valore speciale *canc* nel campo *el* dell'elemento che si vuole rimuovere: in particolare, l'inserimento tratterà tale cella come vuota e si fermerà su di essa, mentre la ricerca la oltrepasserà.

Le prestazioni delle operazioni implementate dipendono dalla particolare funzione $c(k,i)$ usata, ossia dal tipo di scansione scelto:

Scansione lineare:

$$c(k,i) = (h(k)+i) \bmod m \text{ con } 0 \leq i < m$$

Dopo un certo numero di inserimenti, tendono a formarsi degli agglomerati sempre più lunghi di celle piene, che comportano un decadimento delle prestazioni; si parla del problema di di *agglomerazione primaria*.

Scansione quadratica:

$$c(k,i) = \left\lfloor h(k) + c_1 \cdot i + c_2 \cdot i^2 \right\rfloor \bmod m, \text{ con } 0 \leq i < m \text{ e } c_1 \text{ e } c_2 \text{ opportuni}$$

Nonostante la scansione quadratica distribuisca le chiavi in modo da evitare l'agglomerazione primaria, ogni coppia di chiavi k_1 e k_2 con $h(k_1) = h(k_2)$ continua a generare la stessa sequenza di scansione; questo dà luogo all'*agglomerazione secondaria*.

Hashing doppio:

$$c(k,i) = \lfloor h_1(k) + i \cdot h_2(k) \rfloor \bmod m$$

con h_1 e h_2 funzioni hash distinte e m e $h_2(k)$ primi tra loro. Si tratta di un metodo che permette di eliminare virtualmente il fenomeno dell'agglomerazione, facendo dipendere dalla chiave anche il passo dell'incremento dell'indice usando una seconda funzione hash.

Complessità

Nell'ipotesi che le chiavi associate agli elementi di una tavola hash siano prese dall'universo delle chiavi con probabilità uniforme, il numero medio di passi richiesto da un'operazione di ricerca (contando anche le celle marcate come *canc*) è descritto nella seguente tabella:

Esito ricerca	*Scansione lineare*	*Scansione quadratica / Hashing doppio*
Chiave trovata	$\frac{1}{2} + \frac{1}{2 \cdot (1-\alpha)}$	$-\frac{1}{\alpha} \cdot \ln(1-\alpha)$
Chiave non trovata	$\frac{1}{2} + \frac{1}{2 \cdot (1-\alpha)^2}$	$\frac{1}{1-\alpha}$