Azzolini Riccardo 2019-04-16

Tabelle hash

1 Funzioni di hash modulari

Se si considera ogni chiave c come un intero (di $|c|$ bit), è possibile trasformarla in un indirizzo considerando il resto della divisione (modulo) per M:

$$H(c) = \langle c \rangle_M$$

In particolare, se le chiavi sono stringhe su un alfabeto di R caratteri, le si può interpretare come interi in base R. Ad esempio, considerando la codifica ASCII a 7 bit, e quindi $R = 128$:

$$\text{ora} = 111 \cdot 128^2 + 114 \cdot 128^1 + 97 \cdot 128^0 = 1833313$$
$$\text{averylongkey} = 97 \cdot 128^{11} + 118 \cdot 128^{10} + \cdots + 121 \cdot 128^0 > 128^{11} = 2^{77}$$

Da questi esempi si possono notare alcuni problemi:

- se $M = 128$, l'indirizzo dipende solo dall'ultimo carattere (ad esempio, $H(\text{ora}) = \langle 1833313 \rangle_{128} = 97$);
- il valore intero delle chiavi, prima dell'operazione di modulo, può essere eccessivamente grande.

Per soddisfare i requisiti delle funzioni di hash, sono allora necessari alcuni accorgimenti:

- si sfruttano alcune proprietà dell'aritmetica modulare,

$$\langle A + B \rangle_M = \langle \langle A \rangle_M + \langle B \rangle_M \rangle_M$$
$$\langle A \cdot B \rangle_M = \langle \langle A \rangle_M \cdot \langle B \rangle_M \rangle_M$$

  e la *regola di Horner*,

$$a_0x^n + a_1x^{n-1} + \cdots + a_nx^0 = a_n + x(a_{n-1} + x(a_{n-1} + \cdots x(a_1 + xa_0)\cdots))$$

per calcolare $H(c)$ in modo efficiente e senza overflow;

- l'indirizzo calcolato dipende da tutti i bit della chiave quando M e R sono primi tra loro, cioè non hanno fattori comuni (perciò conviene scegliere M primo).

1.1 Implementazione

```
public int hash(String v) {
    int l = v.length();
    int m = 0;
    for (int h = 0; h < l; h++) {
        char c = v.charAt(h);
        m = (DIM_ALFA * m + getNumericValue(c)) % DIM_TAB;
    }
    return m;
}
```

Il tempo di calcolo è $\Theta(|c|)$, perché vengono eseguite $|c|$ iterazioni, ciascuna a costo costante.

2 Collisioni

Siccome $M \ll k$, una funzione di hash è necessariamente molti a uno, quindi occorre gestire le **collisioni**, che si hanno quando a chiavi diverse corrisponde lo stesso indirizzo:

$$x, y \in U,\ x \neq y, \quad H(x) = H(y)$$

La tecnica utilizzata dipende dal tipo di tabella hash.

- In una tabella a concatenazioni separate, la soluzione è immediata: si inseriscono più nodi nella stessa lista.
- Se si utilizza l'indirizzamento aperto, invece, la soluzione è più complessa.

3 Hash statico con concatenazioni separate

Si utilizza una tabella T di dimensione fissa M: $\forall i,\ 0 \leq i < M$, $T[i]$ contiene un riferimento al primo nodo di una lista, la quale a sua volta contiene i dati con chiavi c tali che $H(c) = i$.

Si possono così implementare le operazioni dell'algebra eterogenea PA:

- Member(x) scorre la lista riferita da $H(x.chiave)$ fino a trovare x;
- Insert(x) effettua un inserimento in testa nella lista riferita da $H(x.chiave)$;
- Delete(x) scorre la lista riferita da $H(x.chiave)$ fino a trovare x, che viene quindi cancellato dalla lista.

3.1 Prestazioni

Se

- la funzione di hash rispetta le condizioni 1, 2 e 3
- T ha dimensione M e contiene n dati

allora la lunghezza media di ciascuna lista è $\frac{n}{M}$.

Poiché ciascuna operazione si effettua su una delle liste, il suo costo corrisponde a quello di un'operazione su una lista di n elementi, ma con fattore moltiplicativo $\frac{1}{M}$.

Se si ha una stima sul numero di dati attesi, scegliendo $M \approx n$ si ottiene un costo medio $O(1)$ senza sprechi di spazio dovuti a un numero eccessivo di riferimenti a liste vuote (perché ogni lista conterrà in media 1 elemento).

Nel caso peggiore, è comunque possibile (per quanto improbabile) che tutti i dati vengano inseriti in una sola lista, e in tal caso le prestazioni degenerano a $O(n)$.

3.1.1 Fattore di carico e lunghezza delle liste

Teorema: Sia $\alpha = \frac{n}{M}$ il **fattore di carico**. La probabilità che una data lista della tabella

- sia lunga k è minore di:

$$\frac{\alpha^k e^{-\alpha}}{k!}$$

- contenga più di $t\alpha$ dati (cioè abbia lunghezza pari o superiore a t volte quella media) è minore di:

$$\left(\frac{\alpha e}{t}\right)^t e^{-\alpha}$$

Ad esempio, se $\alpha = 20$, la probabilità che una lista contenga almeno 40 elementi ($t = 2$) è minore di:

$$\left(\frac{20e}{2}\right)^2 e^{-20} \approx 0.0000016$$