Complementi di Algoritmi e Strutture Dati

Hashing Universale

Docente: *Nicolò Cesa-Bianchi* versione 17 maggio 2021

Sia $\mathcal{U}$ un insieme con $|\mathcal{U}| = m \gg 1$. Consideriamo una tabella hash T di dimensione $1 < n \ll m$ ed una funzione di hash $h : \mathcal{U} \to \{0, \ldots, n-1\}$. Supponiamo di memorizzare un arbitrario sottoinsieme $S \subseteq \mathcal{U}$ nella tabella hash usando la funzione h. Ovvero, memorizziamo ogni $u \in S$ nella posizione $h(u)$ di T. Dato che la tabella hash è molto più piccola della cardinalità di $\mathcal{U}$, potranno avvenire delle collisioni. Ovvero, potranno esserci elementi distinti $u, v \in S$ tali che $h(u) = h(v)$. Assumendo quindi che le collisioni vengano gestite utilizzando, per esempio, delle liste associate ad ogni posizione di T, ogni ulteriore operazione di ricerca o inserimento sulla tabella richiederà tempo proporzionale alla lunghezza della lista associata alla posizione della tabella dove viene fatta l'operazione.

In un'analisi di caso peggiore, il tempo per eseguire un'operazione di ricerca o inserimento di un elemento $v \in \mathcal{U}$ è quindi proporzionale al massimo numero $\ell_{\max}$ di elementi in S che sono stati mappati da h nella stessa posizione di v. Più precisamente,

$$\ell_{\max} = \max_{v \in \mathcal{U}} \big|\{u \in S \,:\, h(u) = h(v)\}\big| \,.$$

Idealmente, vorremmo avere $\ell_{\max} = \mathcal{O}\big(|S|/n\big)$, in modo da minimizzare il tempo di esecuzione di un'operazione nel caso peggiore.

Possiamo ottenere questo risultato usando la randomizzazione. Supponiamo di usare una funzione hash h che sia la realizzazione di una variabile casuale H corrispondente all'estrazione casuale (con probabilità uniforme) di un elemento da una famiglia $\mathcal{H}$ di funzioni $h : \mathcal{U} \to \{0, \ldots, n-1\}$. Supponiamo ora che $\mathcal{H}$ soddisfa la condizione seguente

$$\mathbb{P}\big(H(u) = H(v)\big) \le \frac{1}{n} \qquad \text{per ogni } u, v \in \mathcal{U} \tag{1}$$

dove la probabilità è calcolata rispetto all'estrazione casuale della funzione hash H da $\mathcal{H}$.

La condizione (1) è sufficiente a garantire che

$$\max_{v \in \mathcal{U}} \mathbb{E}\Big[\big|\{u \in S \,:\, H(u) = H(v)\}\big|\Big] \le \frac{|S|}{n}$$

dove il valore atteso è calcolato rispetto all'estrazione di H da $\mathcal{H}$. Infatti, supponiamo di aver estratto H a caso da $\mathcal{H}$ e di averla usata per inserire $S = \{u_1, \ldots, u_s\}$ nella tabella. Sia $v \in \mathcal{U}$ un elemento arbitrario che vogliamo cercare o inserire nella tabella. Definiamo le variabili casuali $X_1, \ldots, X_s$ dove $X_i = \mathbb{I}\{H(u_i) = H(v)\}$ e $\mathbb{I}\{\cdot\}$ è la funzione indicatrice di un evento, definita come

$$\mathbb{I}\{A\} = \begin{cases} 1 & \text{se } A \text{ è vero} \\ 0 & \text{altrimenti.} \end{cases}$$

Allora,

$$\big|\{u \in S \,:\, H(u) = H(v)\}\big| = \sum_{i=1}^{s} X_i \ .$$

Il valore atteso di questo numero è calcolato come

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{s} X_i\right] &= \sum_{i=1}^{s} \mathbb{E}[X_i] \quad \text{per linearità del valore atteso} \\
&= \sum_{i=1}^{s} \mathbb{P}\big(H(u) = H(u_i)\big) \quad \text{per definizione di } X_i \\
&\leq \sum_{i=1}^{s} \frac{1}{n} \quad \text{per ipotesi su } H \\
&= \frac{|S|}{n} \ .
\end{aligned}$$

Non è difficile trovare famiglie $\mathcal{H}$ che soddisfano la condizione (1). Assumiamo per semplicità che $\mathcal{U} \equiv \{0, \ldots, m-1\}$ e consideriamo la classe $\mathcal{H}$ di tutte le funzioni $h : \{0, \ldots, m-1\} \to \{0, \ldots, n-1\}$. Ognuna di queste funzioni hash corrisponde a un vettore $\boldsymbol{h} = (h_0, \ldots, h_{m-1}) \in \{0, \ldots, n-1\}^m$ in modo che $h(u) = h_u$. Allora, il numero di vettori $\boldsymbol{h} \in \{0, \ldots, n-1\}^m$ tali che $h_u = h_v$ è n^{m-1}. Quindi, se estraggo H a caso da $\mathcal{H}$,

$$\mathbb{P}\big(H(u) = H(v)\big) = \frac{\big|\,\{\boldsymbol{h} \in \mathcal{H} \,:\, h_u = h_v\}\,\big|}{|\mathcal{H}|} = \frac{n^{m-1}}{n^m} = \frac{1}{n}$$

e la condizione (1) è soddisfatta. D'altra parte, la classe $\mathcal{H}$ non è utilizzabile in pratica in quanto mi servono $\lceil \log_2 |\mathcal{H}| \rceil = \Theta(m \log n)$ bit per memorizzare ciascuna $h \in \mathcal{H}$ e stiamo assumendo che $m \gg 1$.

Per eliminare situazioni di questo genere, aggiungiamo alla condizione (1) una richiesta ulteriore, ovvero che ogni $h \in \mathcal{H}$ possa essere rappresentata con al più $\Theta(\log m)$ bit (che è anche lo spazio che mi occorre per rappresentare un elemento arbitrario di $\mathcal{U}$) e calcolata in modo efficiente. Una famiglia $\mathcal{H}$ di funzioni $h : \mathcal{U} \to \{0, \ldots, n-1\}$ che soddisfa entrambe queste condizioni è detta una *famiglia universale di funzioni hash*.

Dimostriamo ora l'esistenza di famiglie universali. Oltre a $\mathcal{U} = \{0, \ldots, m-1\}$ assumiamo anche $n = p$ per un qualche p primo. Rappresentiamo ciascun elemento $u \in \{0, \ldots, m-1\}$ come un numero $[u]_p$ in base p. Ovvero, $[u]_p = x = (x_1, \ldots, x_r)$ dove $x_i \in [0, \ldots, p-1]$ e r è il più piccolo intero tale che $p^r \geq m$. Ovvero,

$$r = \left\lceil \frac{\log_2 m}{\log_2 p} \right\rceil \ .$$

Sia $\mathcal{A} = \{0, \ldots, p-1\}^r$ l'insieme usato per rappresentare gli elementi di $\mathcal{U}$. Introduciamo ora la famiglia di funzioni hash $\mathcal{H} = \{h_a \,:\, a \in \mathcal{A}\}$ di tipo $h_a : \mathcal{A} \to \{0, \ldots, p-1\}$ e definite come

$$h_a(u) = \left(\sum_{i=1}^{r} a_i x_i\right) \mod p \qquad \text{dove } x = [u]_p.$$

Si noti che posso rappresentare ogni h_a con $\Theta(\log m)$ bit e posso calcolare h_a in modo efficiente. Per verificare la condizione (1) utilizziamo il lemma seguente.

Lemma 1 *Per ogni primo p, per ogni α, β e z interi,*

$$z \not\equiv 0 \mod p \quad e \quad \alpha z \equiv \beta z \mod p \quad \textit{implicano} \quad \alpha \equiv \beta \mod p \ .$$

DIMOSTRAZIONE. Chiaramente, $\alpha z \equiv \beta z \mod p$ se e solo se $z(\alpha - \beta) \equiv 0 \mod p$. Inoltre, $z \not\equiv 0 \mod p$ e $z(\alpha-\beta) \equiv 0 \mod p$ implicano $(\alpha-\beta) \equiv 0 \mod p$. Infatti, se p è primo e z non contiene p come fattore, allora $\alpha-\beta$ lo deve contenere (se invece p non fosse primo, allora z e $\alpha-\beta$ potrebbero spartirsi i fattori primi di p). □

Siamo ora pronti a dimostrare che per $\mathcal{H}$ vale la proprietà 1. Dato che, come abbiamo già osservato, per la stessa classe valeva anche la proprietà 2, concludiamo che $\mathcal{H}$ è una famiglia universale di funzioni hash.

Teorema 2 *$\mathcal{H}$ è tale che per ogni $x, y \in \mathcal{A}$ distinti, la frazione di elementi $a \in \mathcal{A}$ tali che $h_a(x) = h_a(y)$ è al più $\frac{1}{p}$. Quindi, se H è estratta a caso da $\mathcal{H}$, allora*

$$\mathbb{P}\big(H(x) = H(y)\big) \leq \frac{1}{p} \ .$$

DIMOSTRAZIONE. Se $x \neq y$ allora esiste almeno una coordinata $j \in \{1, \ldots, r\}$ tale che $x_j \neq y_j$. Per estrarre H a caso in $\mathcal{H}$ prendiamo $a \in \mathcal{A}$ estraendo a caso ciascuna coordinata $a_i \in \{0, \ldots, p-1\}$. Per qualunque estrazione dei valori a_i sulle coordinate $i \neq j$, abbiamo che

$$\begin{aligned} h_a(x) = h_a(y) &\iff \left(\sum_{i=1}^{r} a_i x_i \equiv \sum_{i=1}^{r} a_i y_i \right) \mod p \\ &\iff a_j \underbrace{(y_j - x_j)}_{z} \equiv \underbrace{\sum_{i \,:\, i \neq j} a_i (x_i - y_i)}_{k} \mod p \\ &\iff a_j z \equiv k \mod p \ . \end{aligned}$$

Dimostriamo ora che esiste un unico $a_j \in \{0, \ldots, p-1\}$ tale che la congruenza $a_j z \equiv k \mod p$ sia soddisfatta. Per assurdo, supponiamo che esistano a_j, a'_j distinti che soddisfano entrambi la congruenza. Allora avremmo anche che $a_j z \equiv a'_j z \mod p$. Dato che $z \not\equiv 0 \mod p$ per ipotesi, il lemma implica che $a_j \equiv a'_j \mod p$. Siccome $0 \leq a_j, a'_j < p$, ciò implica che $a_j = a'_j$ e si ha una contraddizione. Quindi c'è un solo valore della coordinata j che rende $h_a(x) = h_a(y)$ e la probabilità di estrarre questo valore è al più $\frac{1}{p}$. □