Lezione 5

INTRODUZIONE ALLE TABELLE HASH E FUNZIONI HASH

Una tabella hash è una struttura dati efficace nell'implementare dizionari. Questa struttura dati supporta infatti le operazioni INSERT, SEARCH, DELETE.

Una tabella hash, in genere, impiega un array di dimensione proporzionale al numero di chiavi memorizzate, che è relativamente piccolo rispetto al numero di chiavi possibili.

L'efficienza delle tabelle hash è legata all'uso di funzioni hash che abbiano opportune proprietà.

Notazione:

U : universo delle chiavi
$\cup|$
K : chiavi memorizzate , $|K| = n$
m : dimensione della tabella (# slot).

DEF: Una funzione hash mappa l'universo U delle chiavi negli slot di una tabella $T[0, 1, \dots, m-1]$

$$h : U \longrightarrow \{0, 1, \dots, m-1\},$$

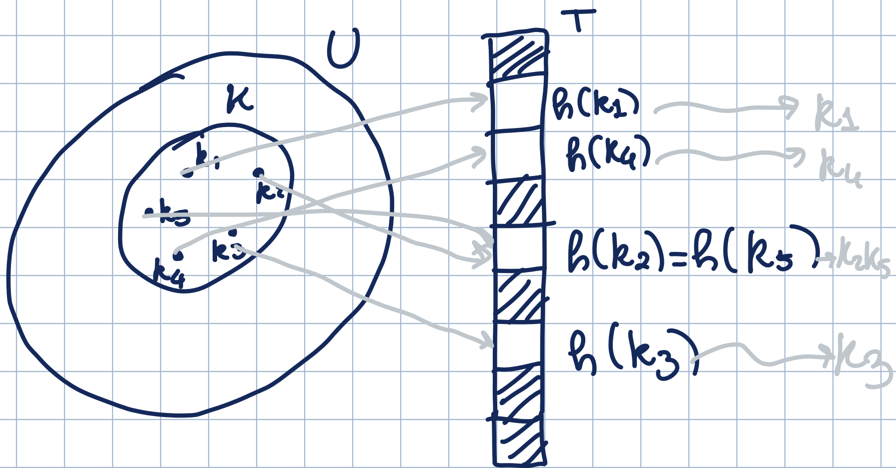
$$k \longmapsto h(k)$$ (chiamiamo $h(k)$ l'hash di k)

(l'elemento con chiave k è memorizzato nello slot $h(k)$).

In generale, $m << |U|$.

Esempio: $h : U \longrightarrow \{0, \dots, m\}$, $h(k) = k \bmod (m)$.

OSS: Due chiavi possono avere lo stesso hash, ovvero possono $\exists\, k_1, k_2 \in U : h(k_1) = h(k_2)$. In tal caso diciamo che k_1 e k_2 collidono.



Evitare totalmente le collisioni è impossibile. Perciò abbiamo bisogno di risolverle.

Risoluzione mediante chaining (concatenazione): L'insieme delle chiavi è suddiviso in sottoinsiemi di dimensione $\frac{n}{m}$.
La funzione hash determina a quale sottoinsieme appartiene una chiave. Ogni sottoinsieme è gestito indipendentemente come una lista.

Una funzione hash ben costruita contribuisce a evitare le collisioni.

?

Una funzione hash uniforme e indipendente è tale che $\forall\, k \in U$, $h(k)$ sia scelto in maniera uniforme e indipendente da $\{0,1,\dots,m-1\}$.

uniforme: $\forall\, k \in U\ \forall\, j \in \{0,\dots,m-1\}$, $Pr[h(k)=j] = \frac{1}{m}$;

indipendente: $\forall\, k_1, k_2 \in U,\ k_1 \neq k_2$, $\forall\, j_1, j_2 \in \{0,\dots,m-1\}$

$$Pr[h(k_1)=j_1 \mid h(k_2)=j_2] = Pr[h(k_1)=j_1]$$

o, equivalentemente

$$Pr[h(k_1)=j_1 \wedge h(k_2)=j_2] = Pr[h(k_1)=j_1]\, Pr[h(k_2)=j_2].$$

~~Una funzione di hash indipendente e uniforme viene anche chiamata random oracle.~~

Le funzioni di hash indipendente e uniforme sono una astrazione teorica, ma sono difficili da implementare in pratica. Tuttavia, è possibile avere funzioni di hashing che sono approssimativamente uniformi e indipendenti.

Per l'analisi degli algoritmi di ricerca nelle tabelle hash assumeremo l'hashing uniforme e indipendente.

fattore di carico

$$\alpha = \frac{n}{m} \quad \left(\begin{matrix}\text{\# medio di elementi}\\ \text{memorizzati in ogni slot}\end{matrix}\right)$$

OSS: Nel caso di tabelle hash dove le collisioni sono risolte per concatenazione $\alpha \lessgtr 1$.

In una tabella hash a indirizzamento aperto, ogni elemento della tabella contiene o un elemento dell'insieme dinamico o NIL. Le tabelle hash a indirizzamento aperto possono essere riempite così che nessun nuovo inserimento sia possibile ovvero il fattore di carico non può superare 1.

Quando un elemento deve essere inserito nella tabella, viene allocato nella posizione "prima scelta". Se la posizione è già occupata il nuovo elemento è allocato nella posizione di "seconda scelta" e così via.

Per cercare un elemento bisogna esaminare sistematicamente gli slot nell'ordine prefissato relativo all'elemento, finché o si trova l'elemento cercato o si trova uno slot vuoto, in quest'ultimo caso si ha una ricerca senza successo.

Il vantaggio dell'indirizzamento aperto è che si evitano del tutto i puntatori.

Nell'analisi degli algoritmi di ricerca in una tabella hash a indirizzamento aperto, assumeremo che la funzione di hashing sia un hashing di permutazione indipendente e uniforme.

Hashing di permutazione

$$\begin{array}{l} \qquad\qquad\qquad\qquad\qquad\qquad\qquad \rightarrow \text{sono } m! \\ h: K \longrightarrow \text{Permutazioni di } \{0, \dots, m-1\} = \{\sigma_1, \dots, \sigma_{m!}\} \end{array}$$

$$h(k) = \{\underbrace{h(k,1)}_{\text{prima scelta}}, \dots, \underbrace{h(k,m)}_{m\text{-esima scelta}}\}$$

Un hashing di permutazione uniforme e indipendente è tale che $\forall\, k \in U$, $h(k)$ sia scelto in maniera uniforme e indipendente da $\{\sigma_1, \dots, \sigma_{m!}\}$

uniforme: $\forall\, k \in U$, $\forall\, \sigma$ permutazione di $\{0, \dots, m-1\}$, $Pr[h(k) = \sigma] = \dfrac{1}{m!}$;

indipendente: $\forall\, k_1, k_2 \in U$, $k_1 \neq k_2$, $\forall\, \sigma_1, \sigma_2$ permutazioni di $\{0, \dots, m-1\}$

$$Pr[h(k_1) = \sigma_1 \mid h(k_2) = \sigma_2] = Pr[h(k_1) = \sigma_1]$$

o, equivalentemente

$$Pr[h(k_1) = \sigma_1 \wedge h(k_2) = \sigma_2] = Pr[h(k_1) = \sigma_1]\, Pr[h(k_2) = \sigma_2].$$

<u>OSS</u>: Nelle tabelle hash a indirizzamento aperto $n \leq m \Rightarrow \alpha \leq 1$.