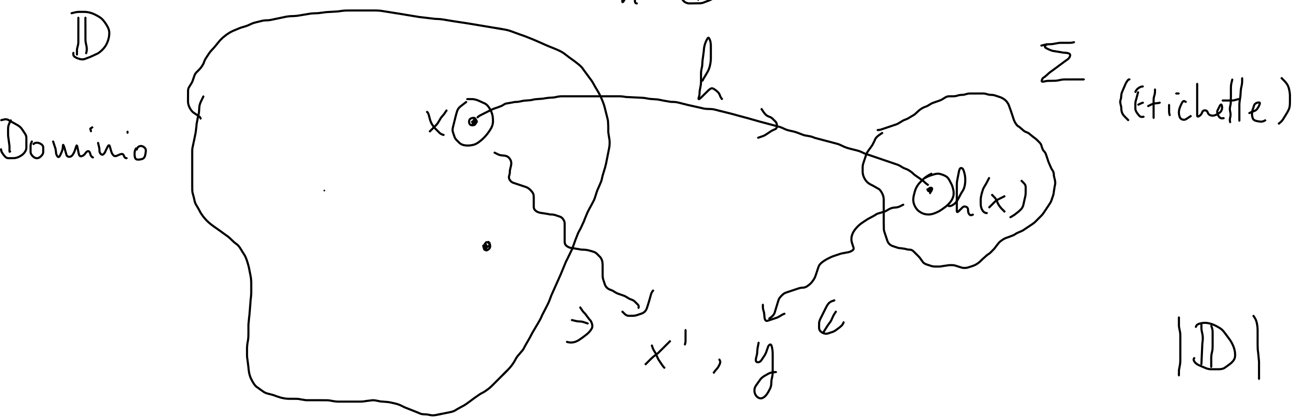
FUNZIONI HASH

$h : \mathbb{D} \to \Sigma$



Ricevo x' e y

Verifico se $h(x') = y$

Se $x' = x$, $y = h(x) \Rightarrow y = h(x')$

Proprietà desiderabili: (informale)

1) Facile da calcolare

2) Se $x \neq x'$, allora $h(x) \neq h(x')$

$|\mathbb{D}| \gg |\Sigma|$

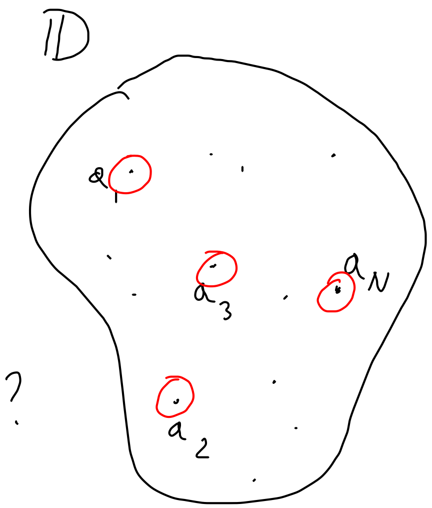
Controlli di integrità

Codice fiscale

Problema del dizionario:

Struttura dati per memorizzare N elementi del dominio $\mathbb{D}$

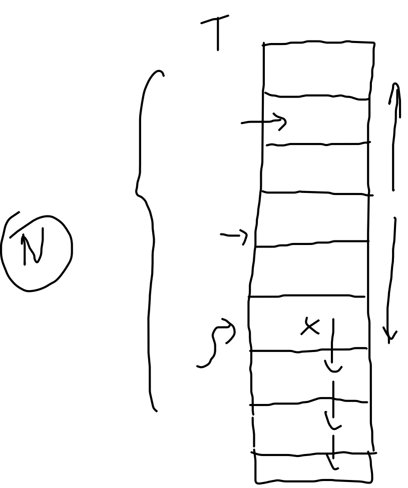
$(a_1, a_2, \ldots, a_N)$ e supportare le seguenti operazioni:



- ricerca(x), $x \in \mathbb{D}$: esiste un elemento memorizzato a_i tale che $a_i = x$?
- inserisci(x), $x \in \mathbb{D}$: aggiungi x agli elementi memorizzati
- cancella(x), $x \in \mathbb{D}$: rimuovi x dagli elementi memorizzati

Misure di efficienza: tempo delle operazioni, spazio di memoria richiesto

Soluzione 1. Mantieni $a_1, a_2, \ldots, a_N$ in un array ordinato T:



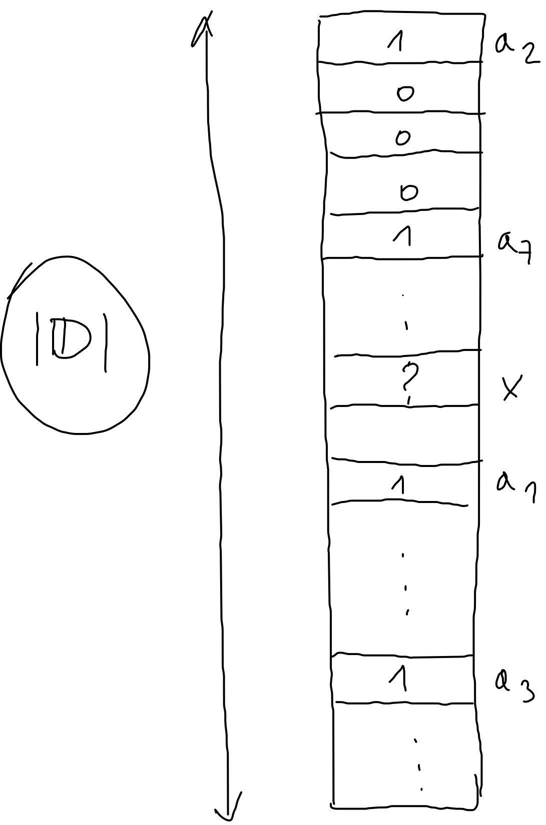
(Costruire T richiede tempo proporzionale a $N \cdot \log N$) (tempo di preprocessamento)

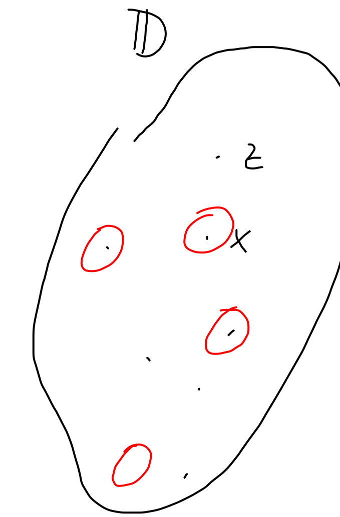
- ricerca(x): usa la ricerca binaria per determinare se $x \in T$ (tempo prop. a $\log N$)

* inserisci(x): aggiungi una riga, ricopiando le tabelle (tempo prop. a N)

* cancella(x): tempo prop. a N

Soluzione 2: Mantieni un array booleano B con un elemento per ogni $z \in \mathbb{D}$ ponendo $B[a_i] = 1$ per ogni $i = 1, 2, \dots, n$ (e $B[z] = 0$ per tutti gli altri $z \in \mathbb{D}$).





- ricerca(x): verifica se $B[x] = 1$ oppure no (tempo costante)
- inserisci(x): imposta $B[x] = 1$ (tempo ")
- cancella(x): imposta $B[x] = 0$ (tempo ")

Spazio richiesto è prop. a $|\mathbb{D}| \gg N$

(anche il pre-processamento richiede tempo $|\mathbb{D}|$)

Domanda: esiste una struttura dati che supporti ricerca, inserimento e cancellazione in tempo costante ma in spazio prop. a N?

Def. (Famiglia di funzioni hash)

Dati $\mathbb{D}$, Σ, $m \geq 1$, una famiglia hash è un insieme

$\mathcal{H} = \{h_1, h_2, \ldots, h_m\}$ con $h_i : \mathbb{D} \to \Sigma$, per ogni $i = 1, 2, \ldots, m$.

Def. (Famiglia di funzioni hash ε-universale)

La famiglia $\mathcal{H} = \{h_1, \ldots, h_m\}$ di funzioni hash è ε-universale $(0 < \varepsilon \leq 1)$

se per ogni $x \neq y$, $x, y \in \mathbb{D}$:

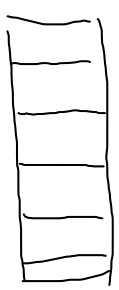
$$\Pr_i [\, h_i(x) = h_i(y) \,] \leq \varepsilon$$

i è scelto uniformemente a caso in $\{1, 2, \ldots, m\}$

Soluzione 3: simile alla soluzione 2, ma invece di mantenere un elemento dell'array per ogni $z \in \mathbb{D}$, ne mantengo uno per ogni possibile valore in Σ.

~~$|\mathbb{D}|$~~

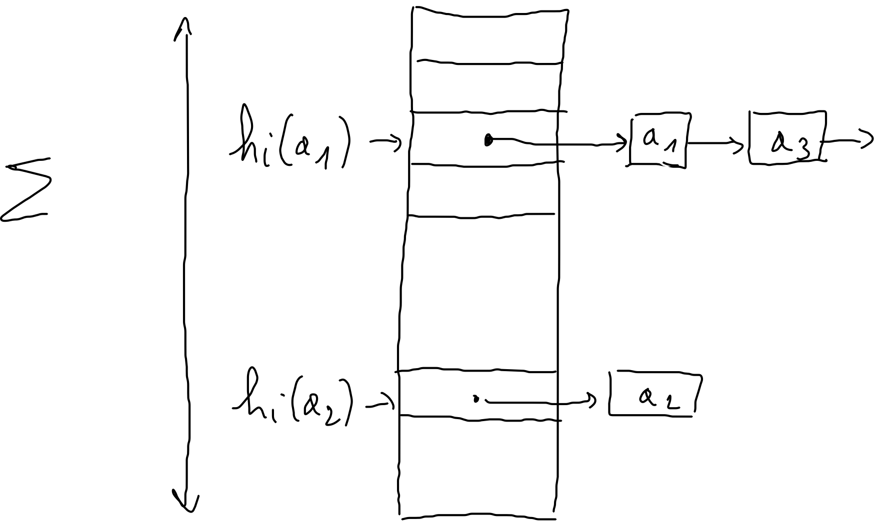
$|\Sigma|$



Sia $\mathcal{H}=\{h_1,\dots,h_m\}$ una famiglia hash ε-universale.

Costruisco un array di liste collegate, una per ogni $v\in\Sigma$.

Tabella hash



Preprocessamento: Prendo $h_i\in\mathcal{H}$ a caso.

Per ogni a_j ($j=1,2,\dots,\underline{N}$)

aggiungi a_j alla lista collegata in posizione $h_i(a_j)$

- ricerca(x): calcola $h_i(x)$ e cerca x nella lista collegata
- inserisci(x): calcola $h_i(x)$ e aggiungi x alla lista collegata
- cancella(x): calcola $h_i(x)$ e rimuovi x dalla lista collegata

$h_i(a_2)$

$h_i(a_3)=h_i(a_1)$

$h_i(a_3)$

Spazio complessivo: prop. a $|\Sigma|+N$

(se $|\Sigma|=O(N)$, lo spazio totale è $O(N)$)

Costo inserimento: tempo costante (aggiungi x in fondo alla lista)

Costo cancellazione: uguale al tempo della ricerca

- Costo ricerca(x): dipende dal numero di a_j tali che $\underbrace{h_i(a_j) = h_i(x)}_{\text{"collisione"}}$.

Prop. Sia $\mathcal{H} = \{h_1, \ldots, h_m\}$ una famiglia hash ε-universale.

Allora, per ogni $x, a_1, a_2, \ldots, a_N \in \mathbb{D}$ distinti,

$$\mathbb{E}_i \Big[\underbrace{\big| \{ a_j : h_i(a_j) = h_i(x) \} \big|}_{\#\text{ di collisioni}} \Big] \leq \varepsilon \cdot N$$

scelta aleatoria di i

Dim. Fissiamo j, $1 \leq j \leq N$. Per definizione di famiglia hash ε-universale,

$$\Pr_i \left[h_i(x) = h_i(a_j) \right] \leq \varepsilon.$$

Ma

$$\mathbb{E}_i [\#\text{ collisioni}] = \mathbb{E}_i \left[\sum_{j=1}^N \begin{cases} 1 & \text{se } h_i(a_j) = h_i(x) \\ 0 & \text{se } h_i(a_j) \neq h_i(x) \end{cases} \right]$$

$$= \sum_{j=1}^N \Pr_i [a_j \text{ collide con } x] \leq \sum_{j=1}^N \varepsilon = \varepsilon N. \quad \text{QED.}$$

Conclusione: se esiste una famiglia hash con dominio $\mathbb{D}$ e codominio Σ ε-universale, se $\varepsilon = O(1/N)$ e $|\Sigma| = O(N)$, allora esiste una struttura dati (randomizzata) che dati N elementi $a_1, a_2, \dots, a_N \in \mathbb{D}$ supporta ricerca, inserimento e cancellazione in tempo atteso costante, e in spazio proporzionale a N. $\leftarrow (|\Sigma| + N)$

Data $\mathcal{H} = \{h_1, \dots, h_n\}$, possiamo considerare il codice associato

$$C_{\mathcal{H}} : \mathbb{D} \to \Sigma^n \qquad \text{definito da} \qquad C_{\mathcal{H}}(x) = (\underset{\Sigma}{\underset{\cap}{h_1(x)}}, \underset{\Sigma}{\underset{\cap}{h_2(x)}}, \dots, \underset{\Sigma}{\underset{\cap}{h_n(x)}}) \in \Sigma^n$$

Viceversa, dato un codice $C(n, k)$ sull'alfabeto Σ, possiamo considerare

$$\mathcal{H}_C = \{h_1, \dots, h_n\} \quad \text{dove} \quad h_i : \Sigma^k \to \Sigma \quad \text{definita da} \quad h_i(x) \stackrel{\Delta}{=} [C(x)]_i \quad \text{per ogni } x \in \Sigma^k$$

parola di codice che codifica x

prendo la componente i-esima

Prop. Se $\mathcal{H}$ è ε-universale, allora $C_{\mathcal{H}}$ ha distanza minima $d_{min} \geq (1-\varepsilon)n$.

Viceversa, se C è un codice $C(n,k,\underbrace{\delta n}^{d_{min}})$ (per qualche $\delta \in (0,1)$) allora la famiglia $\mathcal{H}_C$ è una famiglia $(1-\delta)$-universale.

~~Dim~~

$\rightarrow$ Progettare una buona famiglia di funzioni hash significa progettare un buon codice di canale.