ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

STRUTTURE DATI:
Dizionari e Tabelle Hash

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dizionari

Durante tutto il corso abbiamo assunto che i nodi del grafo fossero numerati da 1 a n, ma se così non fosse?

Esempio: il grafo rappresenta città e autostrade che le collegano

Abbiamo bisogno di un **DIZIONARIO**:
implementa il concetto matematico di relazione univoca

$$r : D \rightarrow C$$

fra gli elementi di un insieme DOMINIO D e gli elementi di un insieme CODOMINIO C

CHIAVI **VALORI**

Memorizza coppie:(CHIAVE,VALORE)

Altri esempi: (nome, numero telefono) - (indirizzi IP, utenti) …

Dizionario

Memorizza coppie (CHIAVE,VALORE)
le coppie sono indicizzate dalla chiave,
il valore è un dato satellite

PRIMITIVE DIZIONARIO:

- `new_dictionary()`: restituisce un dizionario vuoto
- `lookup(D,k)`: restituisce il valore associato in D alla chiave k se esiste, NIL altrimenti
- `insert(D,k,v)`: associa l'associazione (k,v) (inserendo o sovrascrivendo eventuali associazioni precedenti)
- `remove(D,k)`: rimuove da D l'associazione della chiave k

Dizionari

POSSIBILI IMPLEMENTAZIONI

	ARRAY NON ORD.	ARRAY ORD	LISTA NON ORD	LISTA ORD	BST	IDEALE
INSERT	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(1)$
LOOKUP	$O(n)$	$O(\log n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
REMOVE	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$

ALTERNATIVA: TABELLA HASH

Tabelle Hash

- LE COPPIE (K,V) VENGONO MEMORIZZATE IN UN ARRAY $H[0..m-1]$
- LA POSIZIONE DELLA COPPIA (K,V) DIPENDE DA K
- SI USA UNA FUNZIONE HASH h PER DETERMINARE LA POSIZIONE DELLA COPPIA (K,V) IN H, IN FUNZIONE DI K

Tabelle Hash

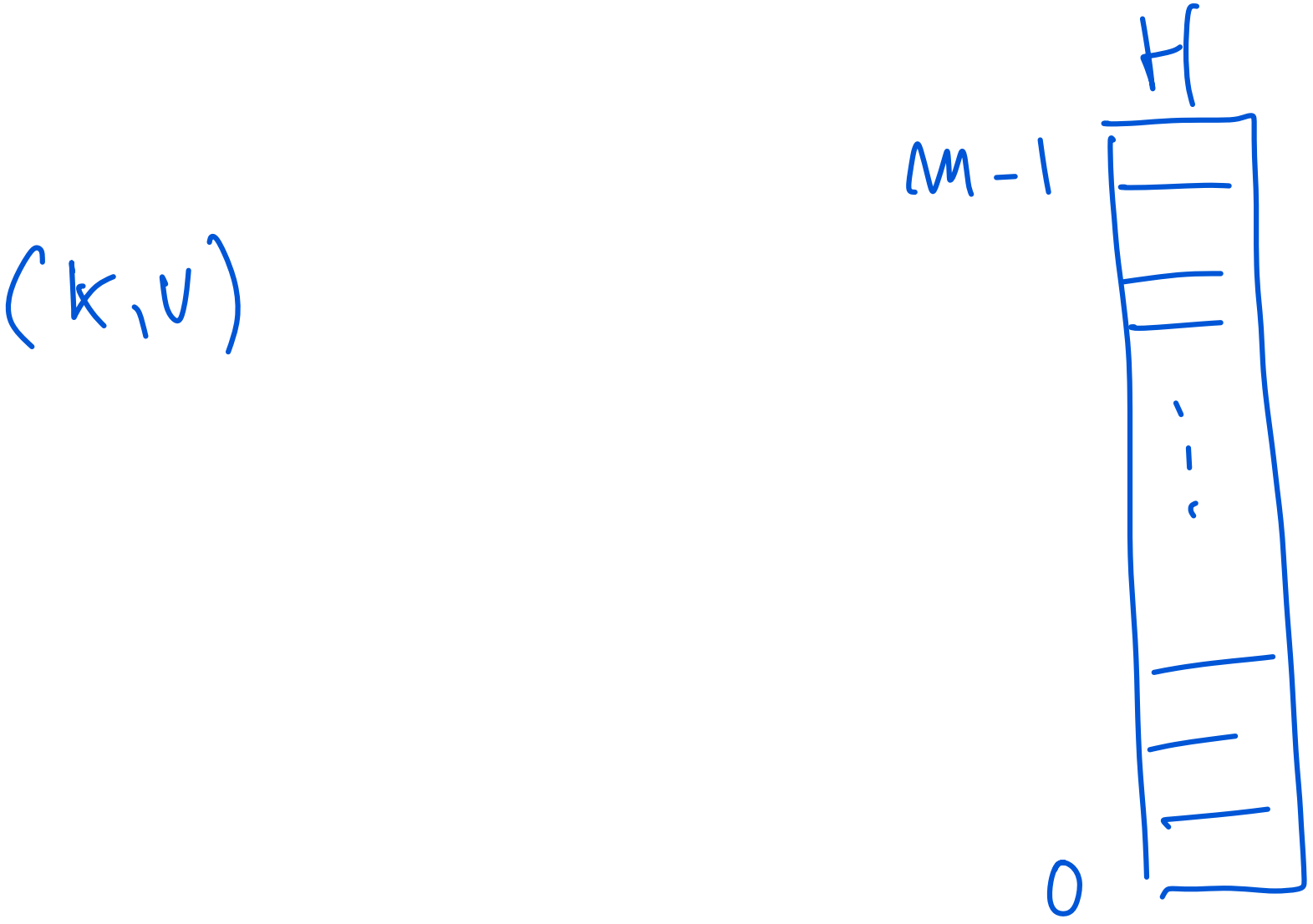


Tabelle Hash

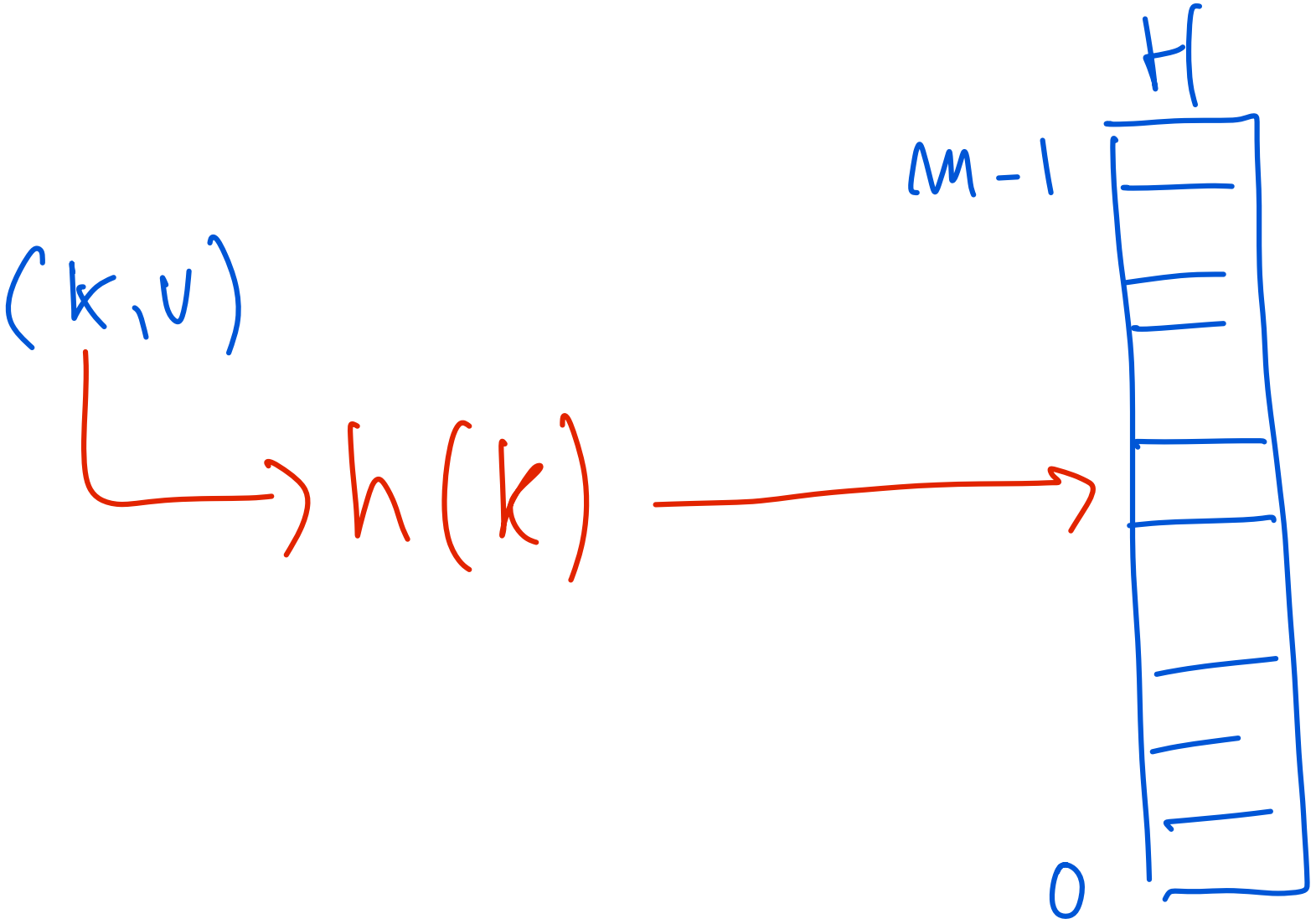
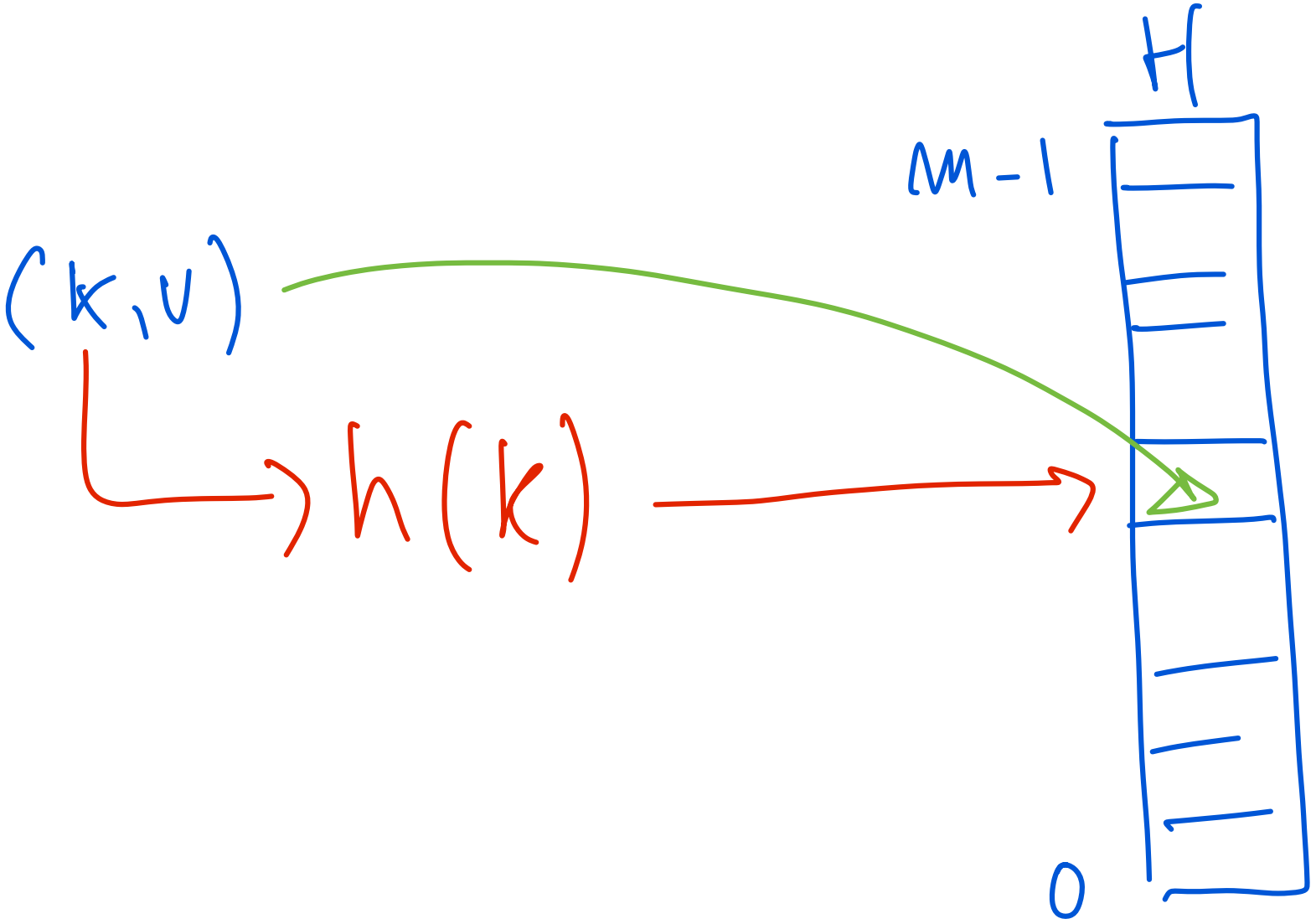
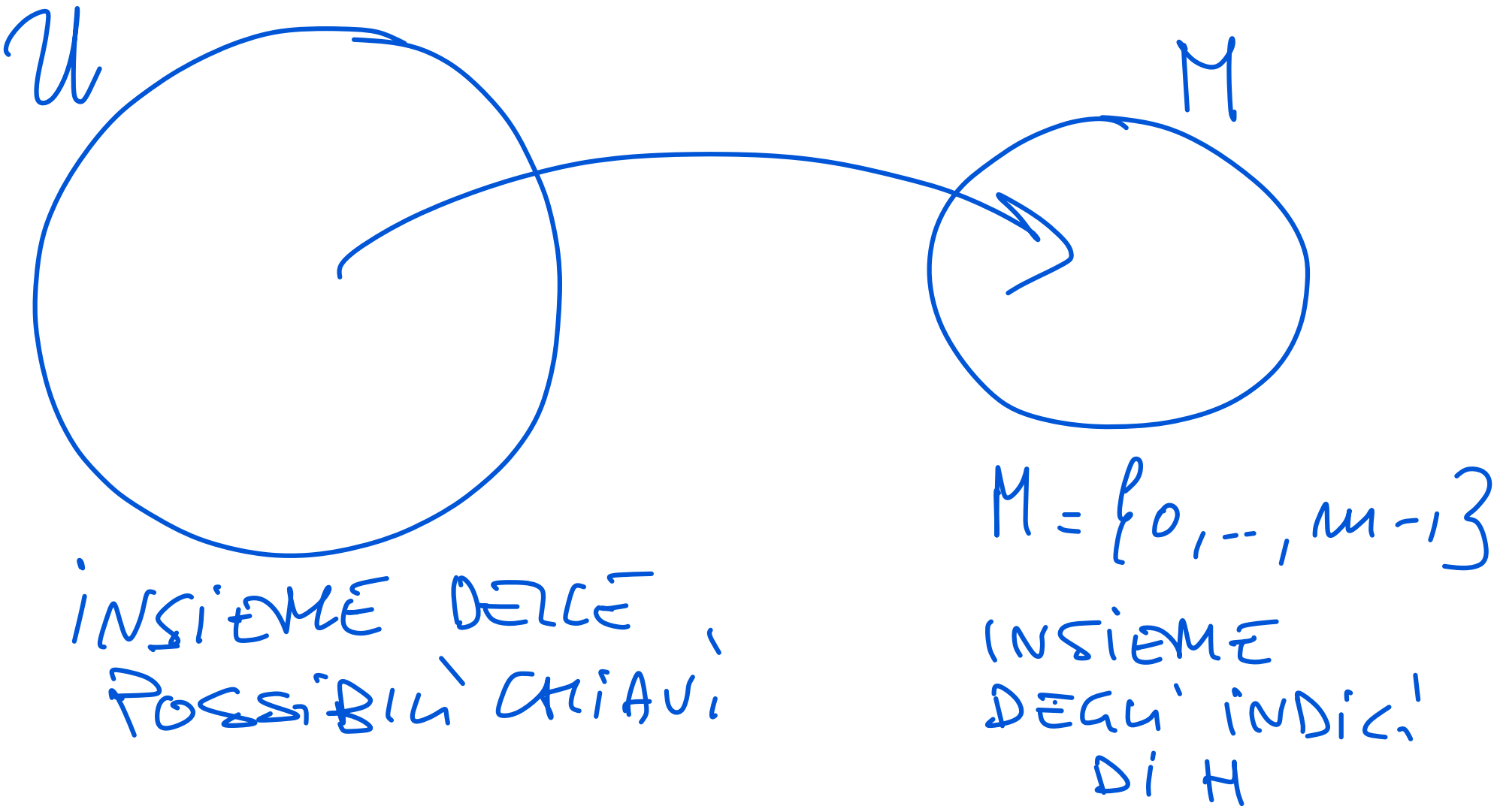


Tabelle Hash



Funzione Hash

COSA FA LA FUNZIONE HASH?



Funzione Hash

COME SI PUÒ REALIZZARE UNA FUNZIONE HASH?

ASSUNZIONE:

QUALUNQUE SIA IL TIPO DI CHIAVE (È UNA SEQUENZA DI CARATTERI), QUESTE POSSONO SEMPRE ESSERE TRADOTTE IN VALORI NUMERICI NON NEGATIVI ED ESSERE MANIPOLATI COME NUMERI

ATTENZIONE: QUESTA È UNA CODIFICA, NON LA FUNZIONE HASH

Funzione Hash

COME SI PUÒ REALIZZARE UNA FUNZIONE HASH?

ASSUNZIONE: ESEMPIO

bin(k) = RAPPRESENTAZIONE BINARIA DELLA CHIAVE k, CONCATENANDO LE RAP. BINARIE DEI SUOI CARATTERI

bin(DOG) = 01000100 01001111 01000111

D O G

USANDO CODICE ASCII

Funzione Hash

COSA FA LA FUNZIONE HASH?

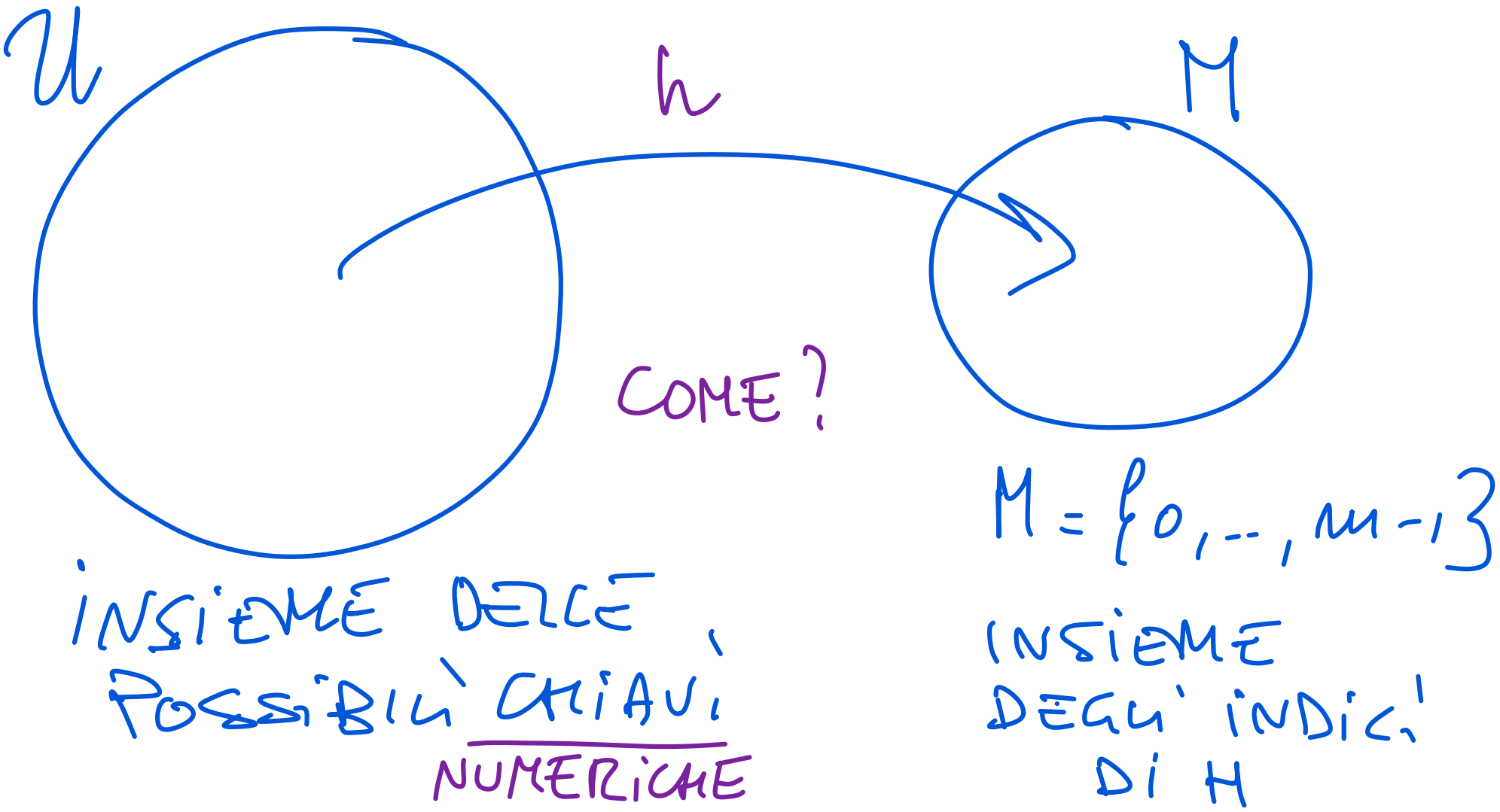


Tabelle ad accesso diretto

SE L'INSIEME UNIVERSO U È' PICCOLO

ES: - GIORNI DELL'ANNO (366)
- L'INSIEME DEI CARROZZIERI DI PROVINCIA (?)

FUNZIONE HASH IDENTITÀ:

$$h(k) = k$$

$$m = |U|$$

PROBLEMI

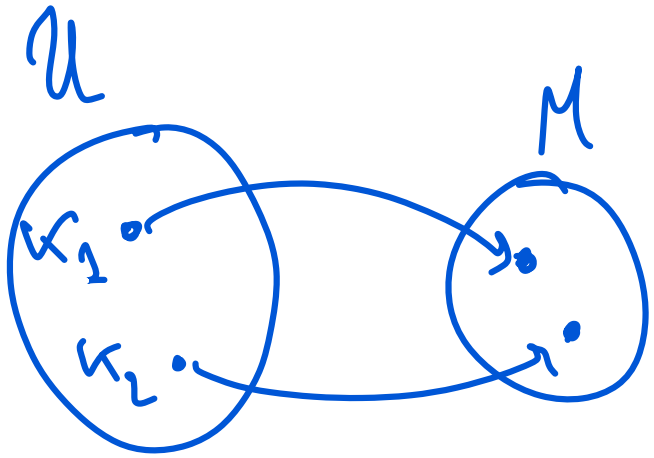
- FUNZIONA SOLO SE U È' PICCOLO
- SE LE CHIAVI USATE SONO POCHE → SPRECO DI MEMORIA

Funzioni hash perfette

DEFINIZIONE:

UNA FUNZIONE HASH SI DICE PERFETTA SE

$\forall k_1, k_2 \in \mathcal{U} : k_1 \neq k_2$ ABBIAMO $h(k_1) \neq h(k_2)$



CHIAVI DIVERSE "FINISCONO" IN POSIZIONI DIVERSE

ESEMPIO

STUDENTI IMMATRICOLATI NEL 2015/2020

CHIAVE = # MATRICOLA

$\mathcal{U} = \{234714, 234715, \ldots, 23800\}$

$h(k) = k - 234714$

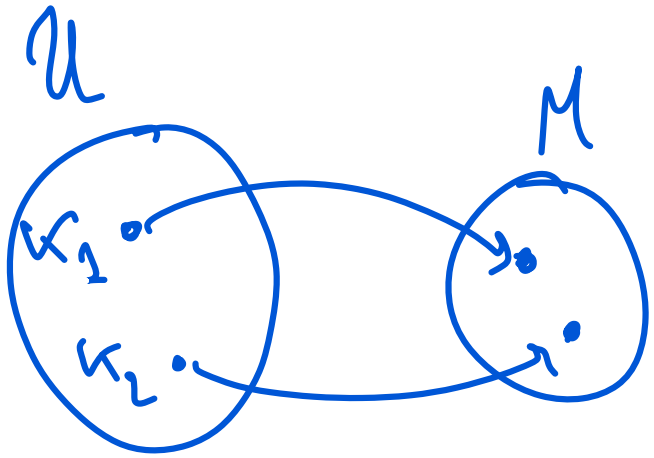
$m = 23800 - 234714 + 1$

Funzioni hash perfette

DEFINIZIONE:

UNA FUNZIONE HASH SI DICE PERFETTA SE

$\forall k_1, k_2 \in \mathcal{U} : k_1 \neq k_2$ ABBIAMO $h(k_1) \neq h(k_2)$



CHIAVI DIVERSE "FINISCONO" IN POSIZIONI DIVERSE

PROBLEMI

- SPESSO DIFFICILE DA IMPLEMENTARE
- $|\mathcal{U}| = M$ PUO' ESSERE GRANDE
- LE CHIAVI EFFETTIVAMENTE USATE POCHE

Esempi di funzione hash

ESTRAZIONE

$m = 2^p$ potenza di 2

$h(k)$ = sottoinsieme di p bit nella rappresentazione binaria di k

ESEMPIO: $p = 16$ $m = 2^{16}$

ESTRAZIONE degli ultimi 16 bit

bin(DOG) = 01000100 01001111 01000111

h(bin(DOG)) = 0100111101000111

INDICE DI H

Esempi di funzione hash

XOR

$m = 2^p$

$h(k)$ = SOMMA MODULO 2 (bit a bit) DI SOTTOINSIEMI DI p BIT DELLA RAPPRESENTAZIONE BINARIA DI k

ESEMPIO: $p = 8$ $\quad m = 2^8$

$bin(DOG) =$ 01000100 | 01001111 | 01000111

```
  01000100
+ 01001111
+ 01000111
----------
  01001100
```

$h(bin(DOG)) =$ 01001100

INDICE DI H

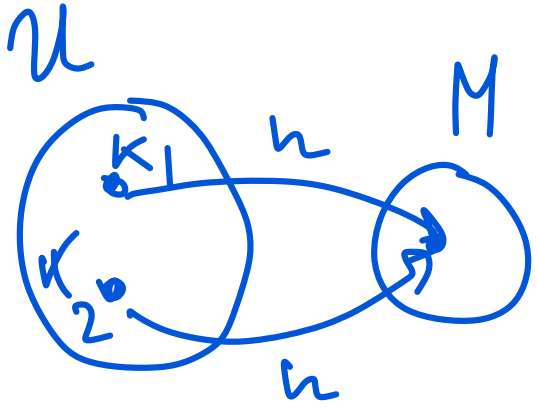
Gestione dei conflitti

PROBLEMA:

IL NUMERO DI CHIAVI È MAGGIORE DEL NUMERO DI POSIZIONI IN H

⇒ ESISTONO CHIAVI $K_1 \neq K_2$ TALE CHE

$$h(K_1) = h(K_2)$$



PUÒ ESSERCI COLLISIONE SULLE CELLE DI H

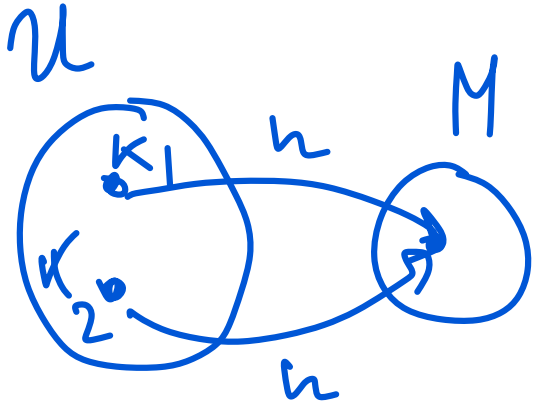
Gestione dei conflitti

PROBLEMA:

IL NUMERO DI CHIAVI È MAGGIORE DEL NUMERO DI POSIZIONI IN H

⇒ ESISTONO CHIAVI $K_1 \neq K_2$ TALE CHE

$$h(K_1) = h(K_2)$$



ESEMPIO

ESTRAZIONE

$$h(\text{bin}(\text{DOG})) = h(\text{bin}(\text{FOG}))$$

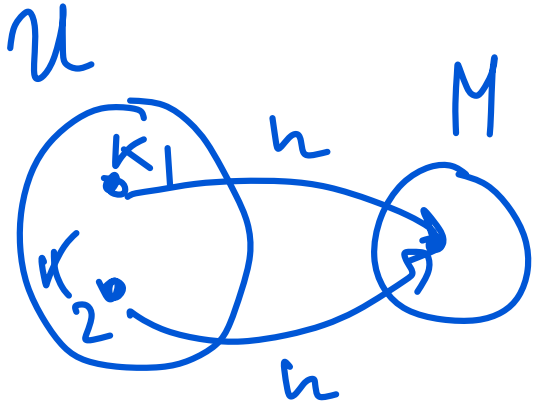
Gestione dei conflitti

PROBLEMA:

IL NUMERO DI CHIAVI È MAGGIORE DEL NUMERO DI POSIZIONI IN H

⇒ ESISTONO CHIAVI $K_1 \neq K_2$ TALE CHE

$$h(K_1) = h(K_2)$$



COME MANTENERE MEMORIZZATE IN H ENTRAMBE LE COPPIE?

Funzione hash uniforme

UNA FUNZIONE HASH SI DICE UNIFORME SE DISTRIBUISCE UNIFORMEMENTE LE CHIAVI DI $\mathcal{U}$ IN M

UNIFORMITÀ SEMPLICE:

$P(k)$ = PROB k INSERITA NELLA TABELLA

$Q(i)$ = PROB CHE UNA CHIAVE VADA NELLA CELLA i DELLA TABELLA

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

VOGLIAMO CHE $Q(i) = 1/m \quad \forall i = 0, \dots, m-1$

Funzione hash uniforme

UNA FUNZIONE HASH SI DICE UNIFORME SE DISTRIBUISCE UNIFORMEMENTE LE CHIAVI DI U IN M

NON SI ELIMINANO LE COLLISIONI, MA SI CERCA DI LIMITARNE IL NUMERO

PROBLEMA: LA PROBABILITÀ P(k) DEVE ESSERE NOTA

Gestione dei conflitti

PROBLEMA:

CHIAVI DIVERSE POSSONO FINIRE NELLA STESSA POSIZIONE DI H

→ PREVEDERE POSIZIONI ALTERNATIVE PER LE CHIAVI

→ STABILIRE COME CERCARE LE CHIAVI NELLE POSIZIONI ALTERNATIVE

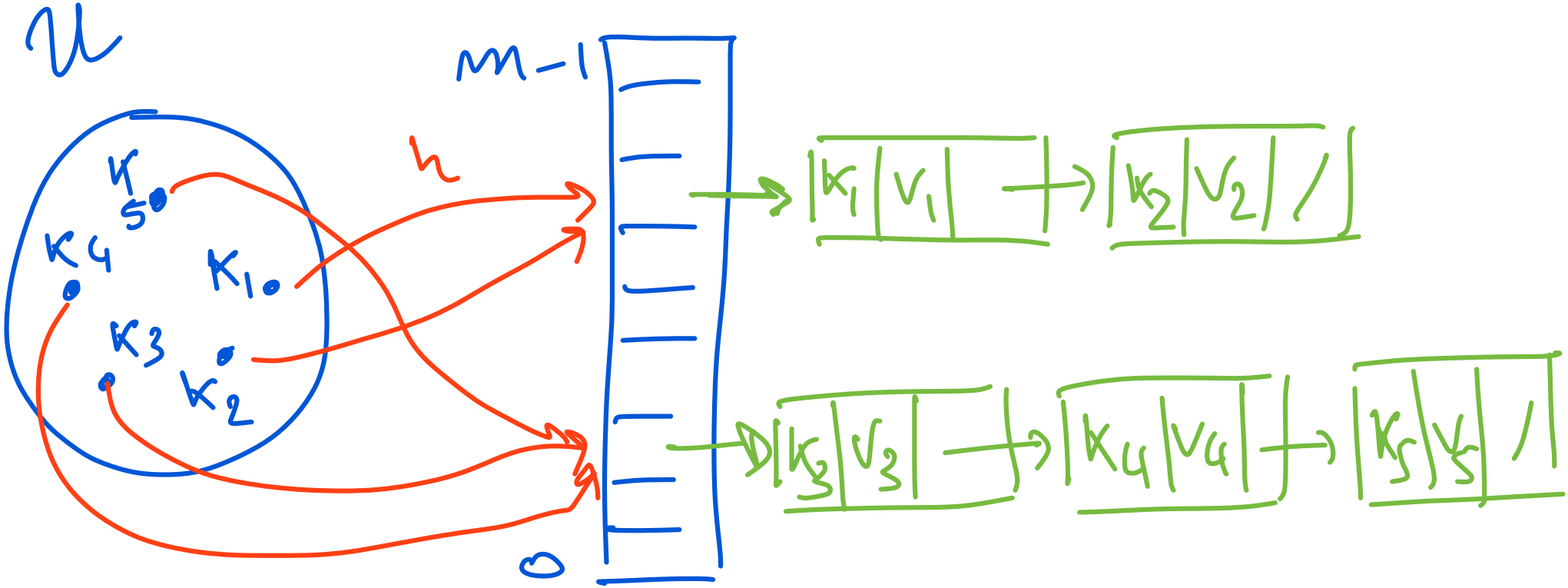
→ TENERE BASSI I COSTI

DUE ALTERNATIVE:

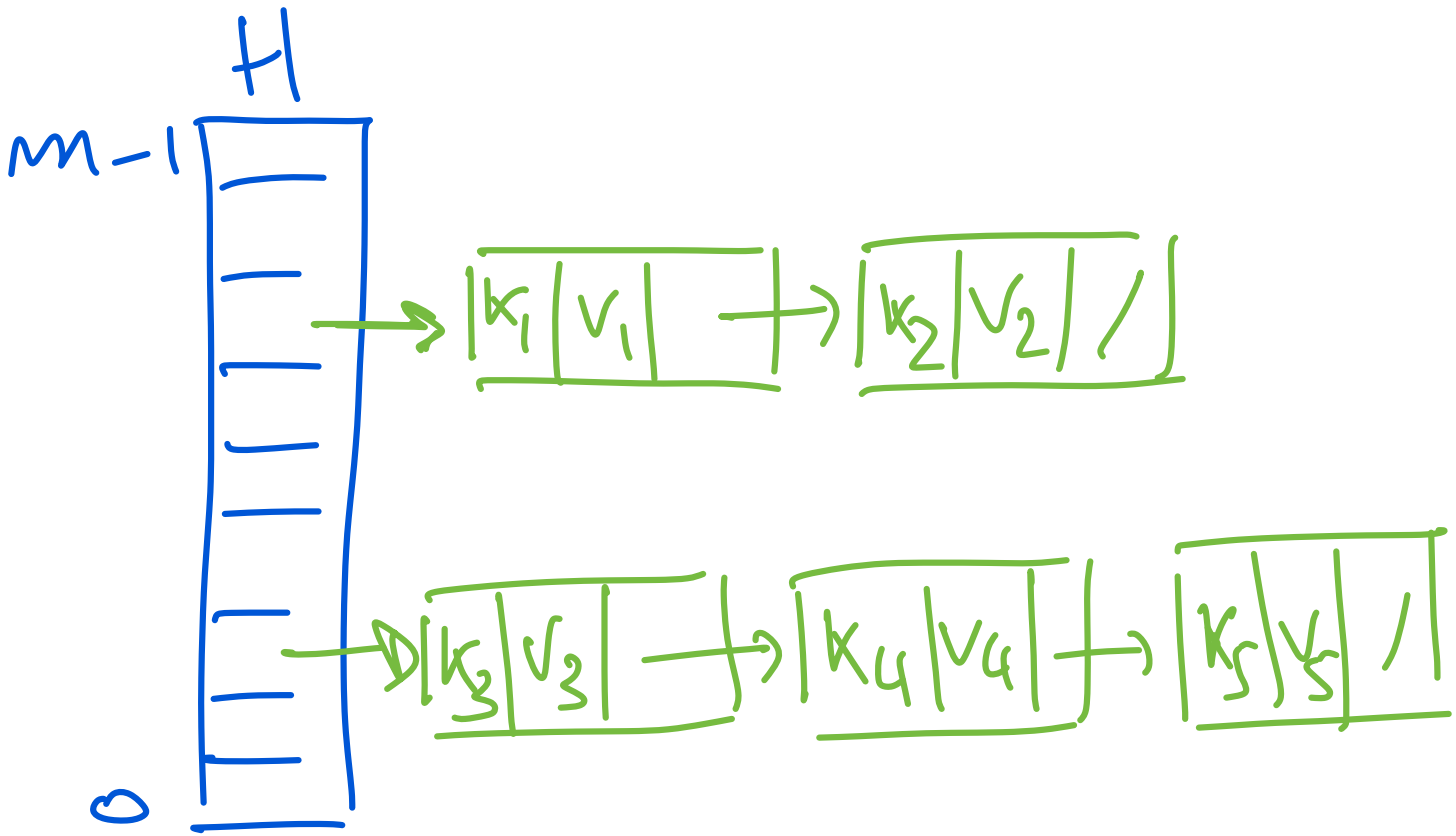
1) LISTE DI TRABOCCO
2) INDIRIZZAMENTO APERTO

Liste di trabocco

IDEA: USIAMO UNA LISTA PER MEMORIZZARE TUTTE LE COPPIE CHE "FINISCONO" NELLA STESSA CELLA DI H

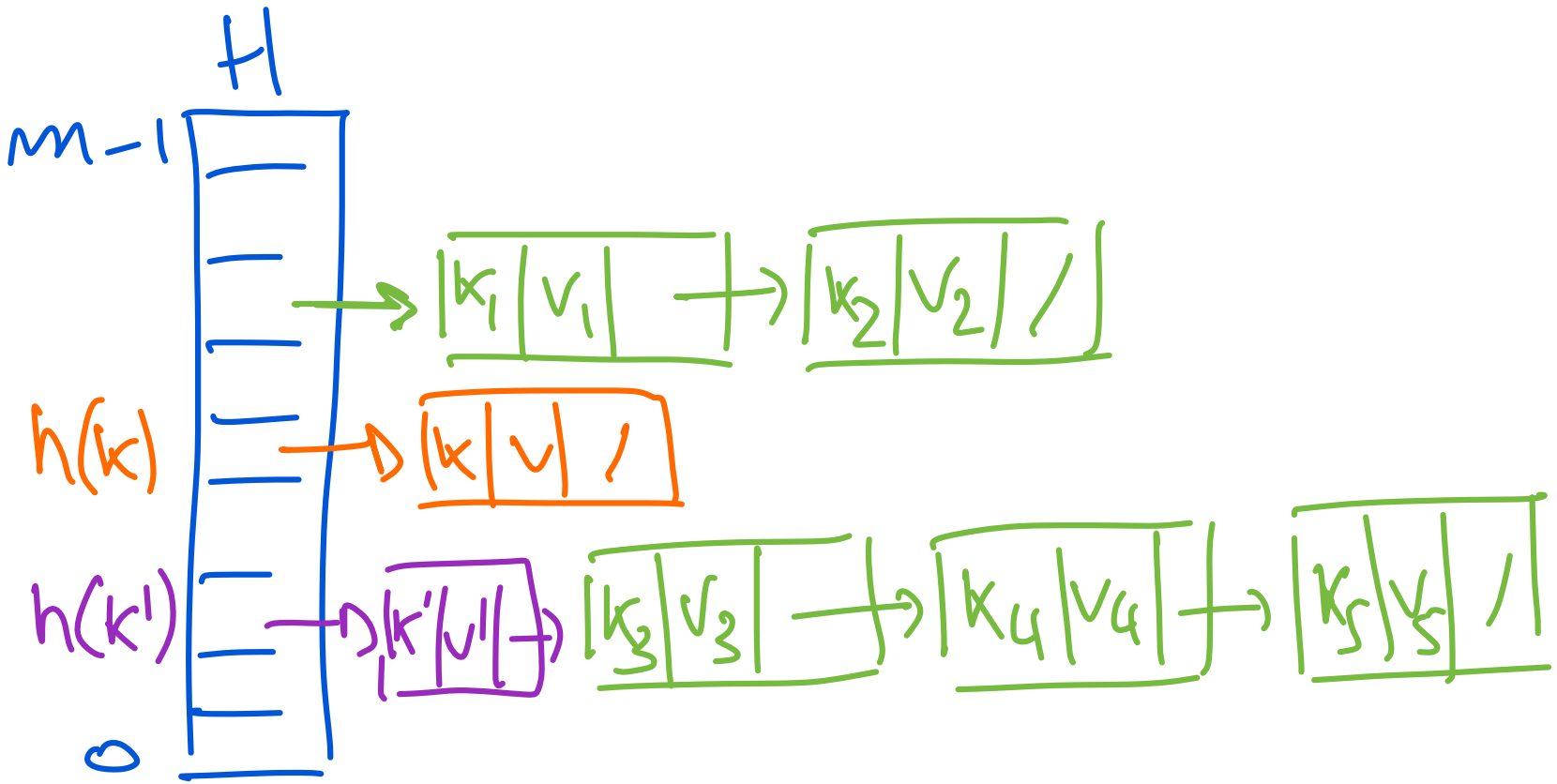


Liste di trabocco



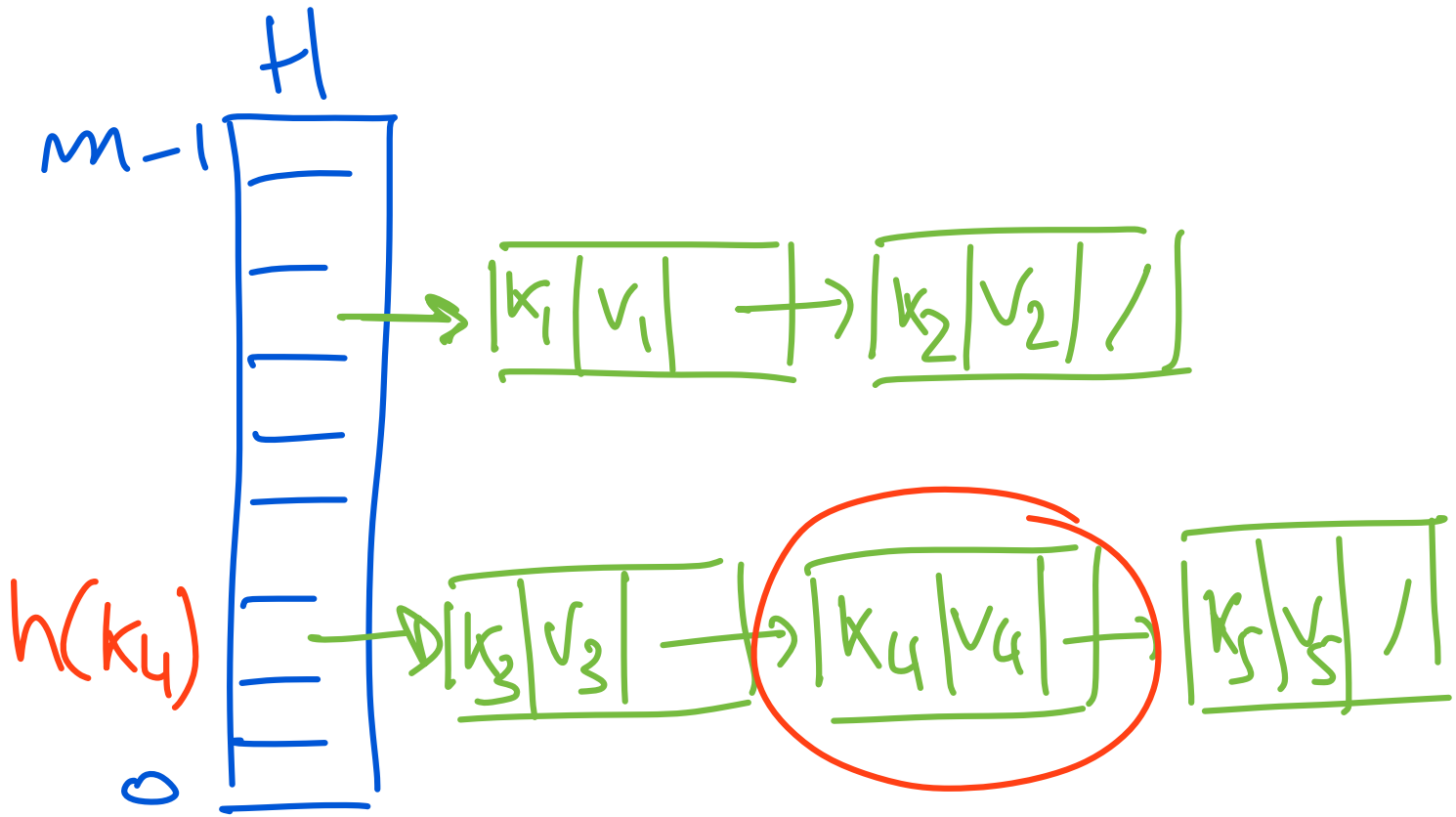
INSERZIONE NUOVA COPPIA (k,v) → INSERZIONE IN TESTA IN H[h(k)]

Liste di trabocco



INSERZIONE NUOVA COPPIA (k,v) → INSERZIONE IN TESTA IN H[h(k)]

Liste di trabocco



LOOKUP E REMOVE DI k_4 → RICERCA NELLA LISTA H[h(k)]

Liste di trabocco

COSTO COMPUTAZIONALE

CASO PEGGIORE: TUTTE LE COPPIE SONO IN UN'UNICA LISTA!

INSERT $O(1)$

LOOKUP - REMOVE $O(m)$

(ASSUMENDO IL COSTO DI h È $O(1)$)

M = # chiavi memorizzate

Liste di trabocco

COSTO COMPUTAZIONALE

FATTORE DI CARICO $\alpha = n/m$

SE $\alpha > 1$ =>

C'È SICURAMENTE UNA COLLISIONE

n = # chiavi memorizzate

m = capacità della Tabella hash

Liste di trabocco

COSTO COMPUTAZIONALE

FATTORE DI CARICO $\alpha = n/m$

SE h UNIFORME SEMPLICE

$\Rightarrow$ LA LUNGHEZZA ATTESA DELLE LISTE DI TRABOCCO È α

n = # chiavi memorizzate
m = capacità della Tabella hash

Liste di trabocco

COSTO COMPUTAZIONALE

SE h UNIFORME SEMPLICE E SI CALCOLA IN $\Theta(1)$, **IN MEDIA**

RICERCA CHIAVE (LOOKUP e REMOVE)

- SENZA SUCCESSO: $\Theta(1)+\alpha$
- CON SUCCESSO: $\Theta(1)+\alpha/2$

$\alpha = M/m$ FATTORE DI CARICO

M = # chiavi memorizzate

m = capacità della Tabella hash

Liste di trabocco

COSTO COMPUTAZIONALE

SE h UNIFORME SEMPLICE E SI CALCOLA IN $\Theta(1)$, IN MEDIA

RICERCA CHIAVE

- SENZA SUCCESSO: $\Theta(1) + \alpha \Rightarrow O(1)$
- CON SUCCESSO: $\Theta(1) + \alpha/2 \Rightarrow O(1)$

$\alpha = n/m$ FATTORE DI CARICO

n = # chiavi memorizzate

m = capacità della Tabella hash

Se $n \in O(m)$ $\Rightarrow$ $\alpha \in O(1)$

Liste di trabocco

COSTO COMPUTAZIONALE

CASO MEDIO: LE LISTE DI TRABOCCO SONO IN MEDIA LUNGHE α SE h UNIFORME SEMPLICE

INSERI $O(1)$

LOOKUP - REMOVE $O(\alpha) \rightarrow$ se $m \in O(M)$ $O(1)$

(ASSUMENDO IL COSTO DI h $\in \Theta(1)$)

$\alpha = M/m$ FATTORE DI CARICO

M = # chiavi memorizzate

m = capacità della tabella hash

Indirizzamento aperto

MEMORIZZIAMO TUTTE LE COPPIE IN H
SENZA UTILIZZARE LE LISTE

→ INSERIMENTO:
SE LA CELLA E' OCCUPATA
→ CERCARE UN'ALTERNATIVA

→ RICERCA:
PARTENDO DALL'INDICE h(k),
CERCARE ANCHE IN TUTTE LE
POSIZIONI ALTERNATIVE

Indirizzamento aperto

ISPEZIONE: ESAME DI UNA CELLA DI H

FUNZIONE HASH: ESTESA CON IL NUMERO DI ISPEZIONE

$h(k,0), h(k,1), h(k,2), \ldots, h(k,M-1)$

$h(k,0)$: 1° INDICE DI H DA ISPEZIONARE

$h(k,1)$: 2° INDICE

$h(k,2)$: 3° INDICE

$h(k,M-1)$: ULTIMO INDICE

$h(k,i) \neq h(k,j) \quad \forall i \neq j$

Indirizzamento aperto

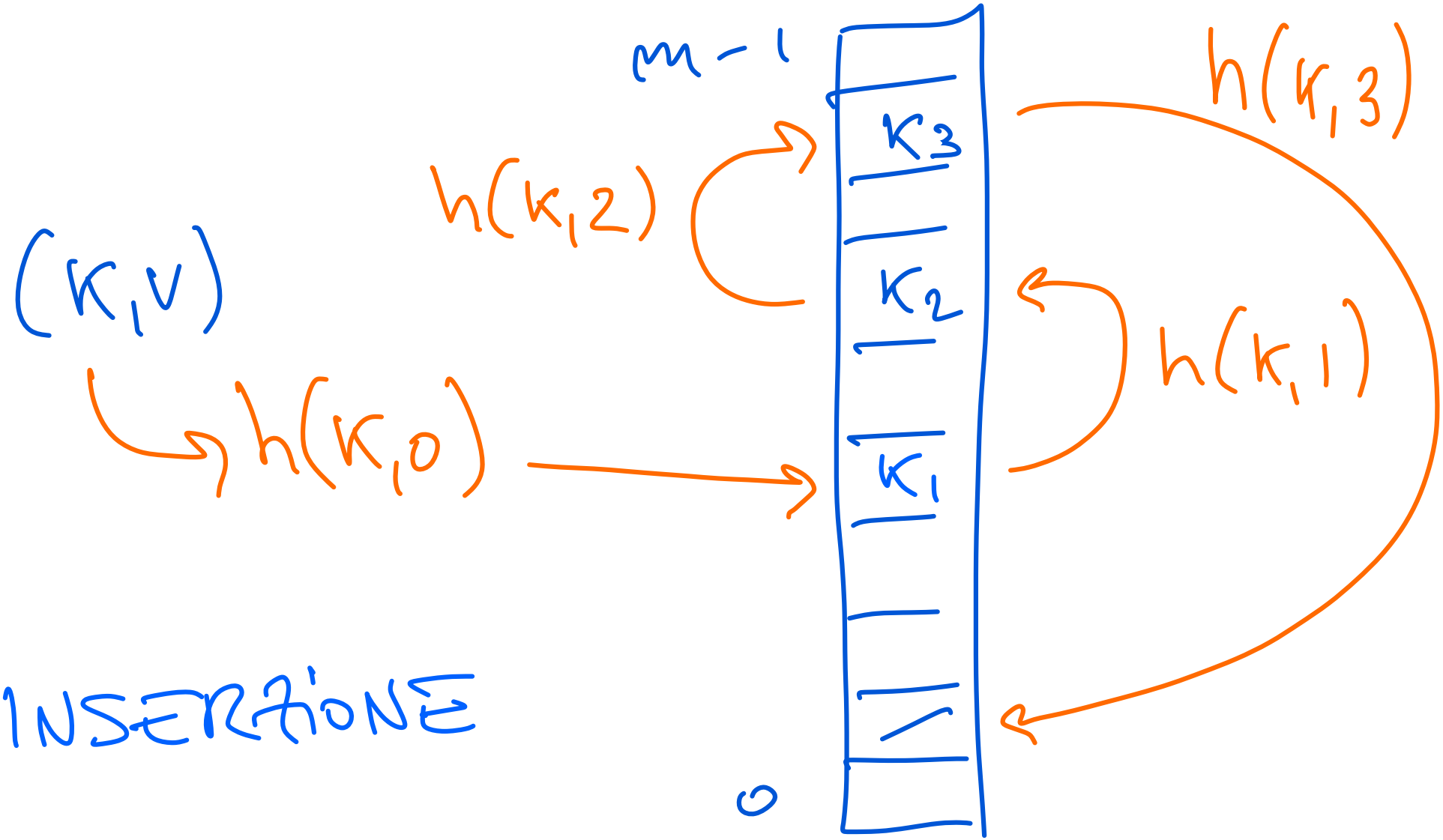
ISPEZIONE: ESAME DI UNA CELLA DI H

FUNZIONE HASH: ESTESA CON IL NUMERO DI ISPEZIONE

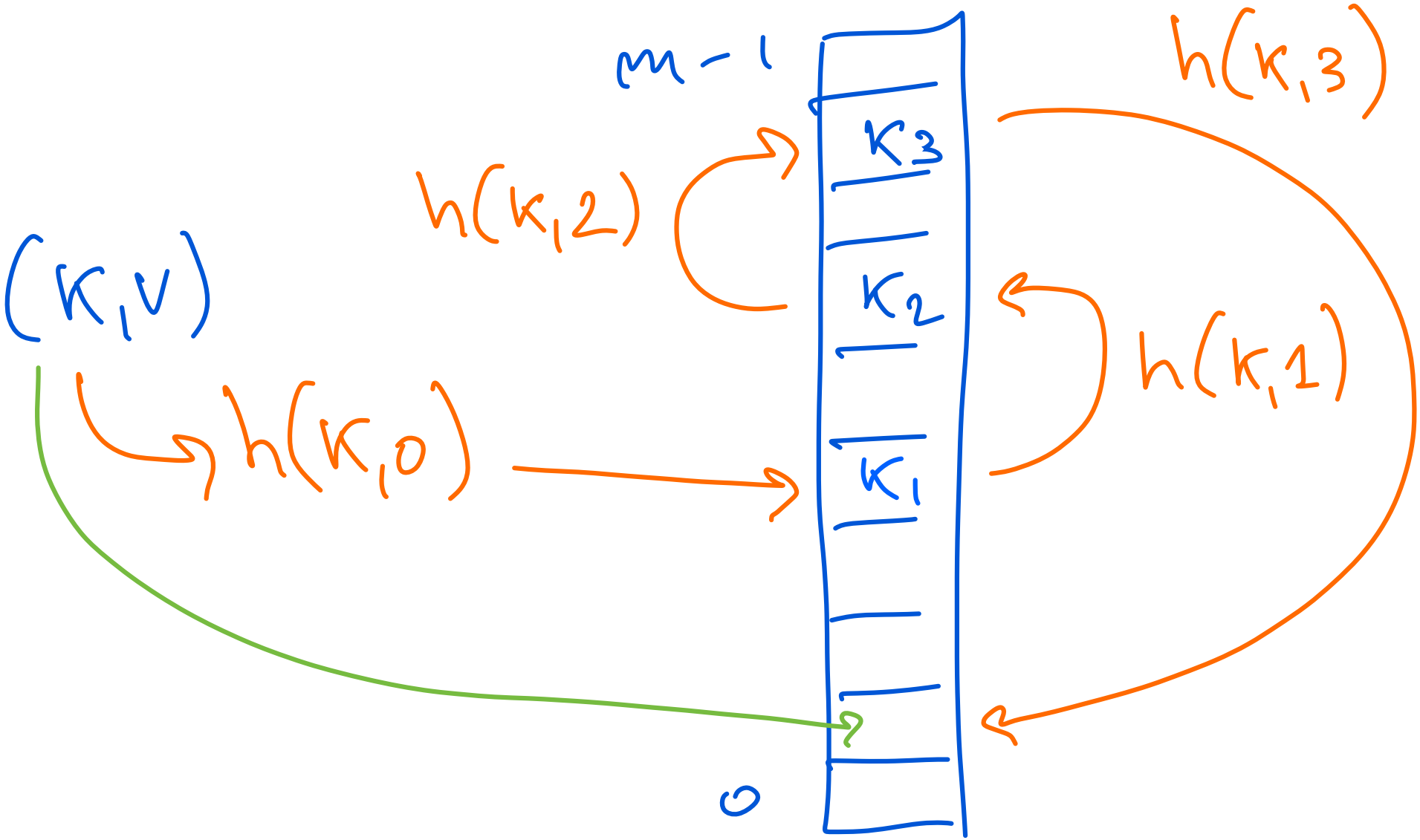
$h(k,0), h(k,1), h(k,2), \ldots, h(k,m-1)$

PERMUTAZIONE DEGLI INDICI DI H

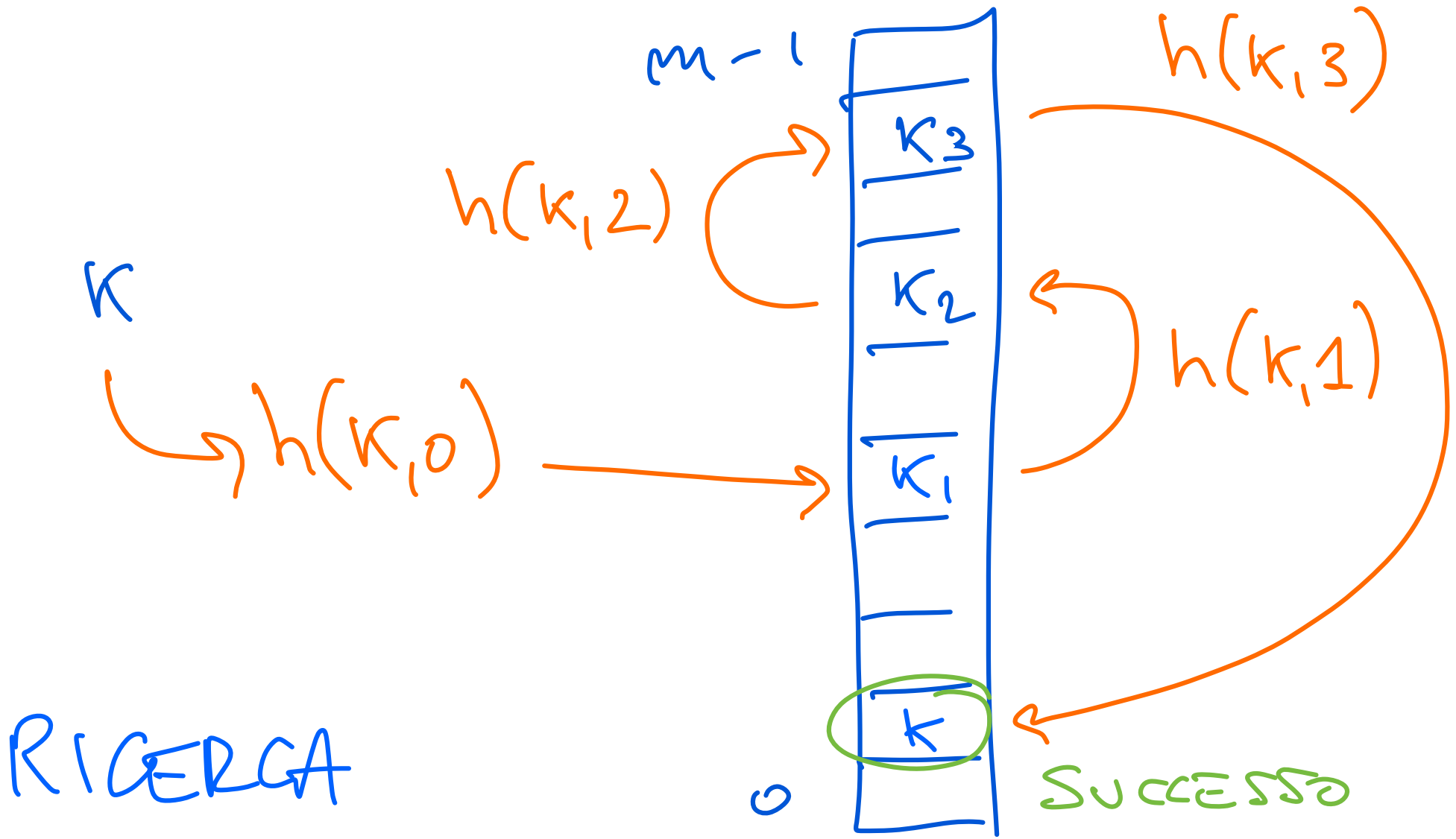
Indirizzamento aperto



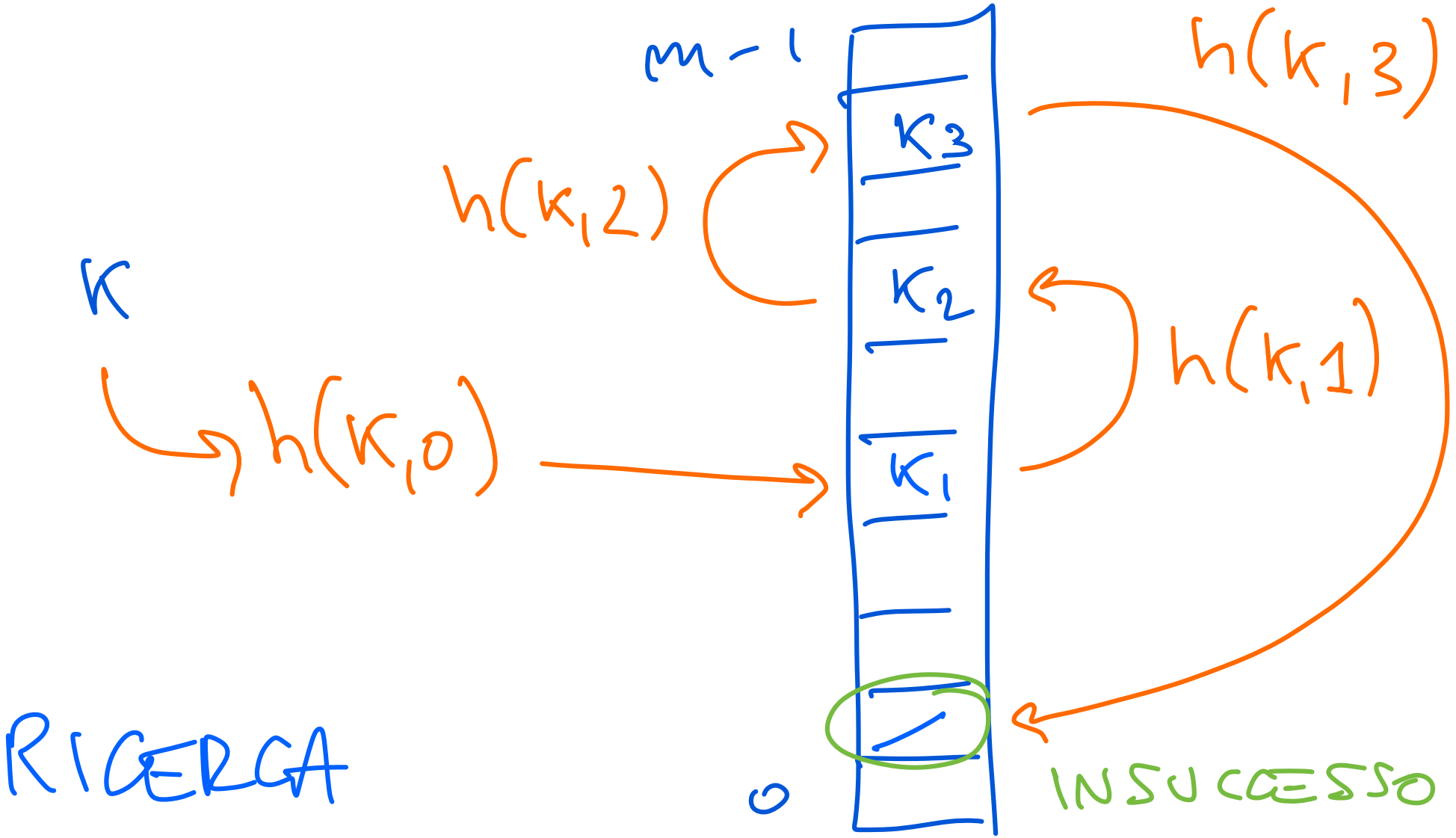
Indirizzamento aperto



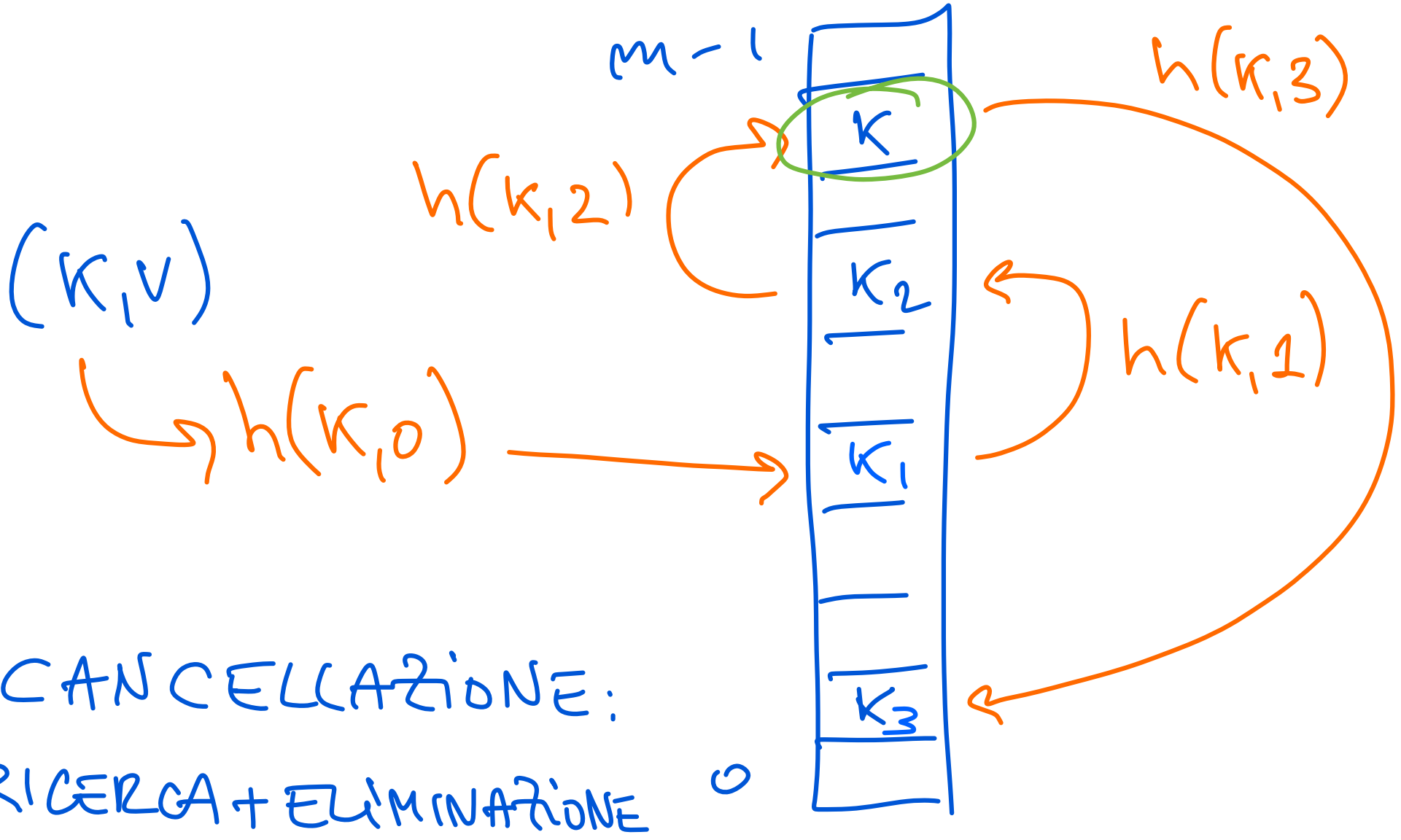
Indirizzamento aperto



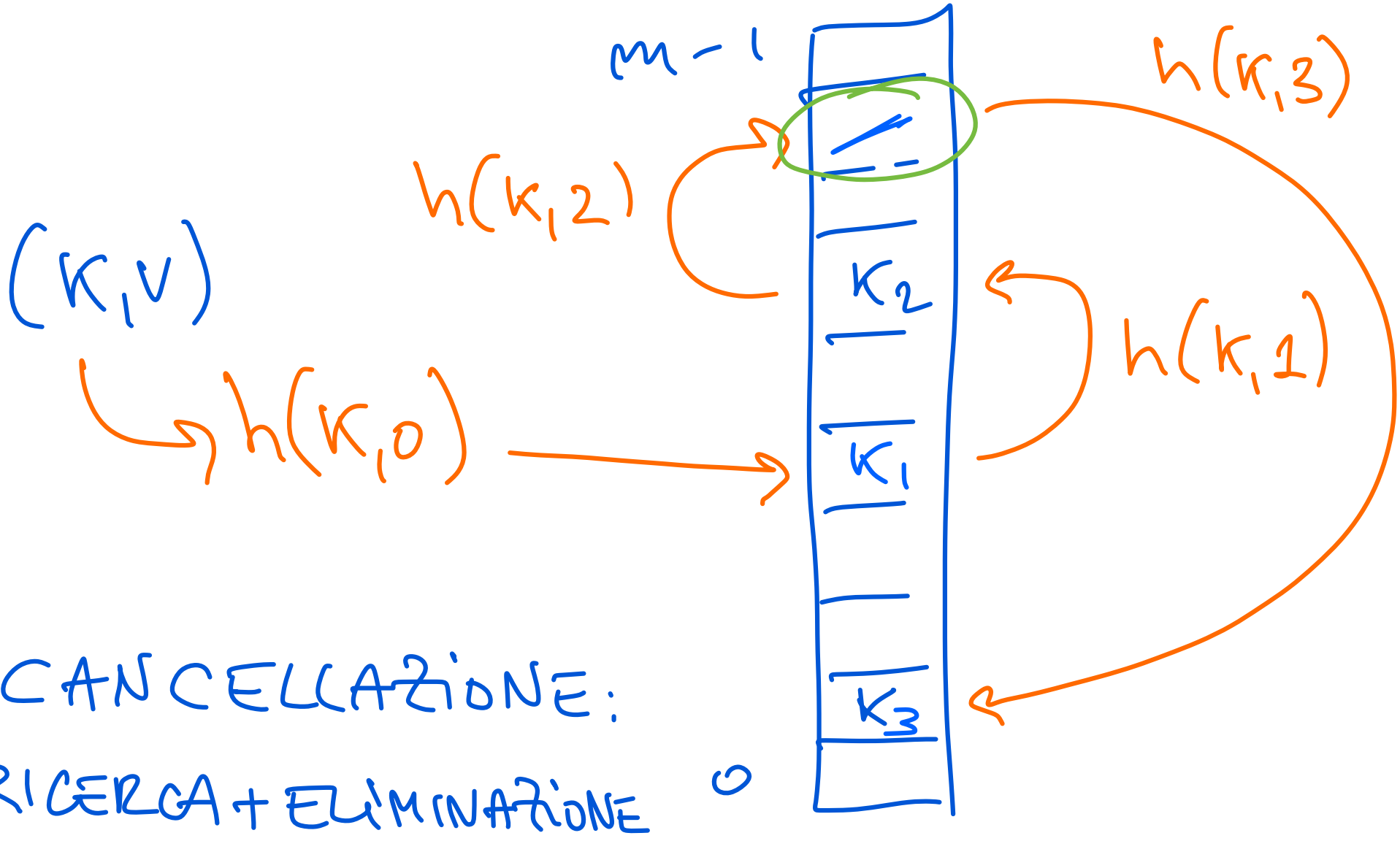
Indirizzamento aperto



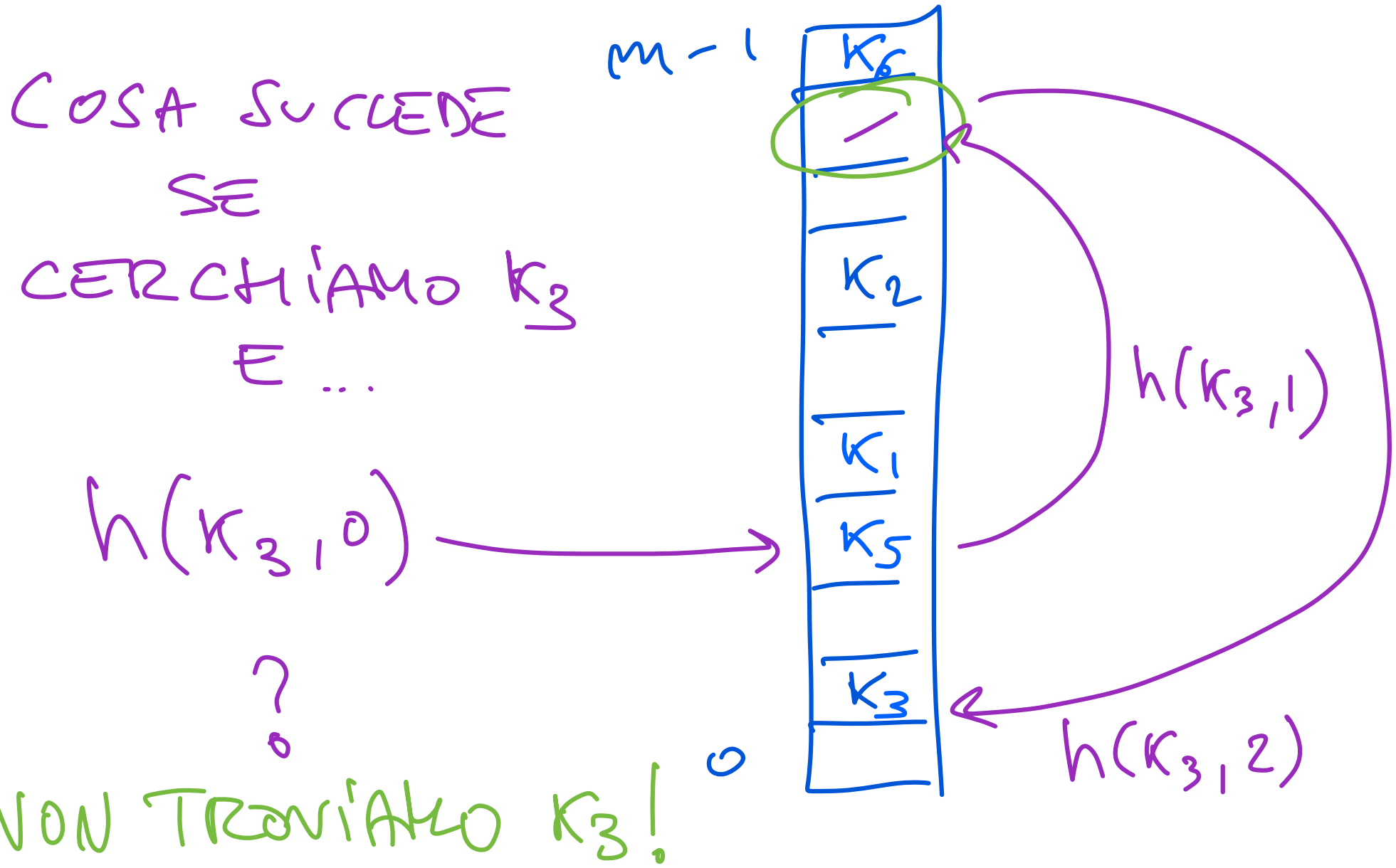
Indirizzamento aperto



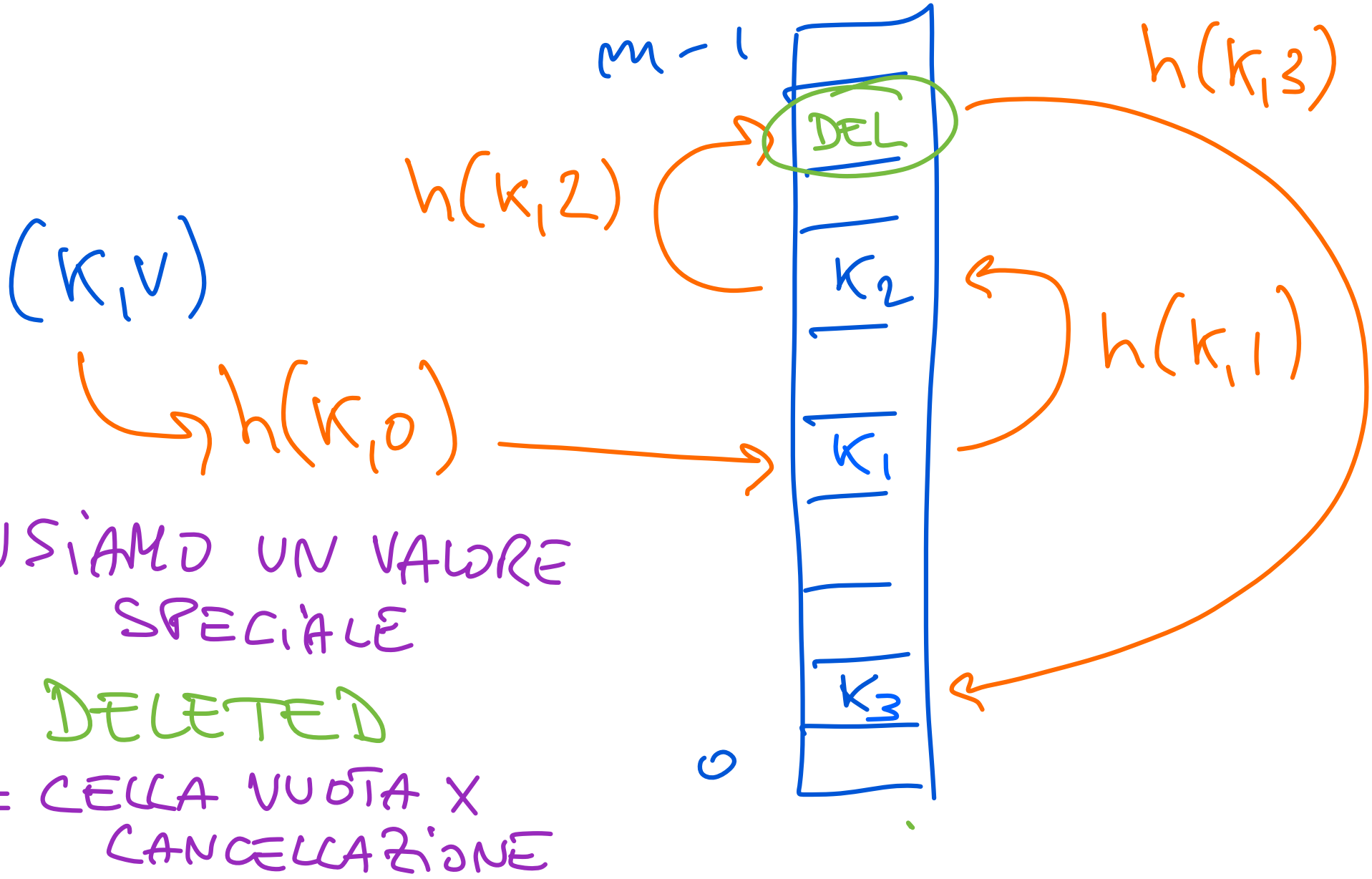
Indirizzamento aperto



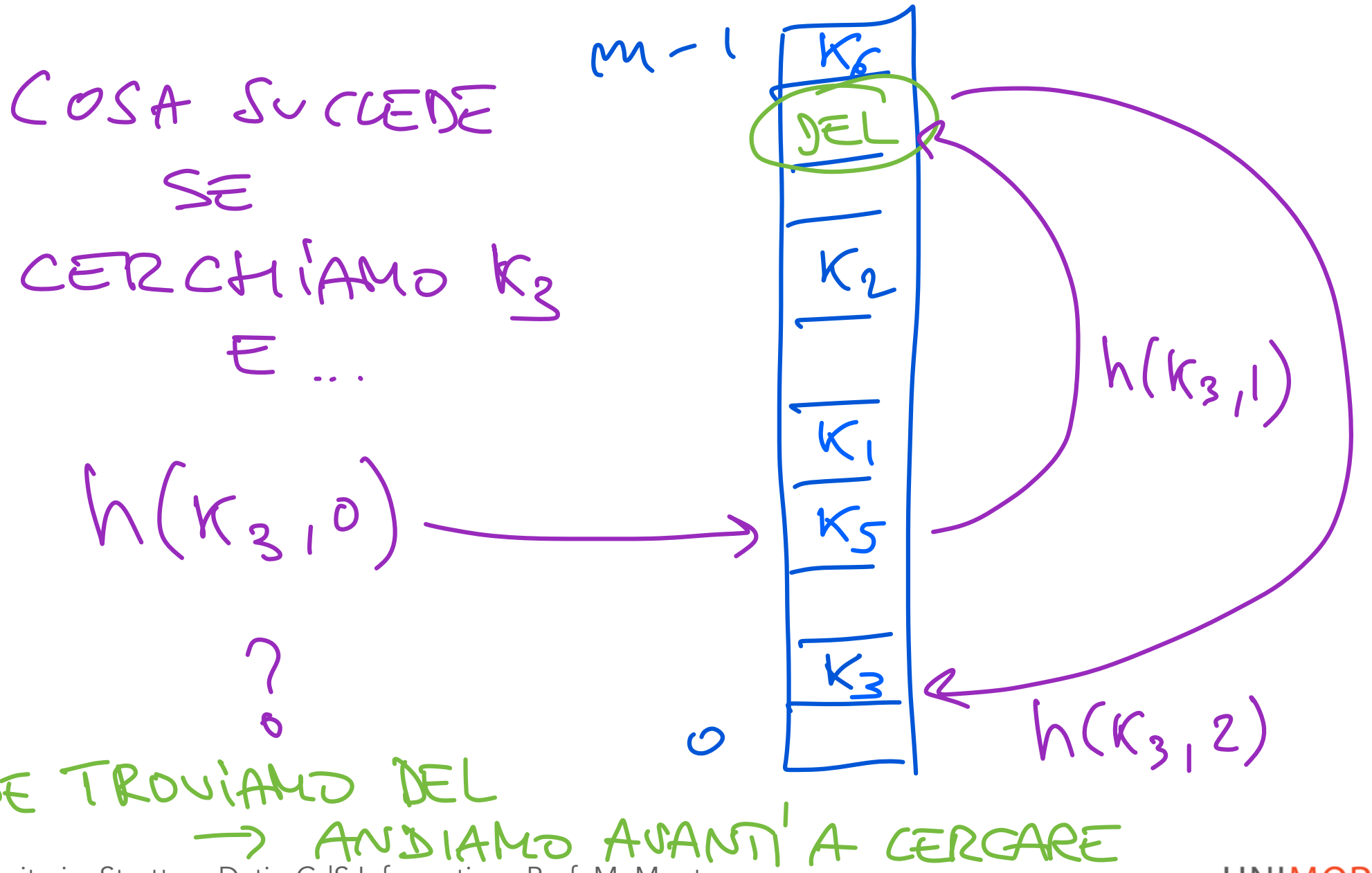
Indirizzamento aperto



Indirizzamento aperto



Indirizzamento aperto



Indirizzamento aperto

HASHING UNIFORME:

OGNI CHIAVE HA LA STESSA PROBABILITA' DI AVERE COME SEQUENZA DI ISPEZIONE UNA QUALUNQUE DELLE m! PERMUTAZIONI DI $[0..m-1]$

↳ DIFFICILE IMPLEMENTAZIONE

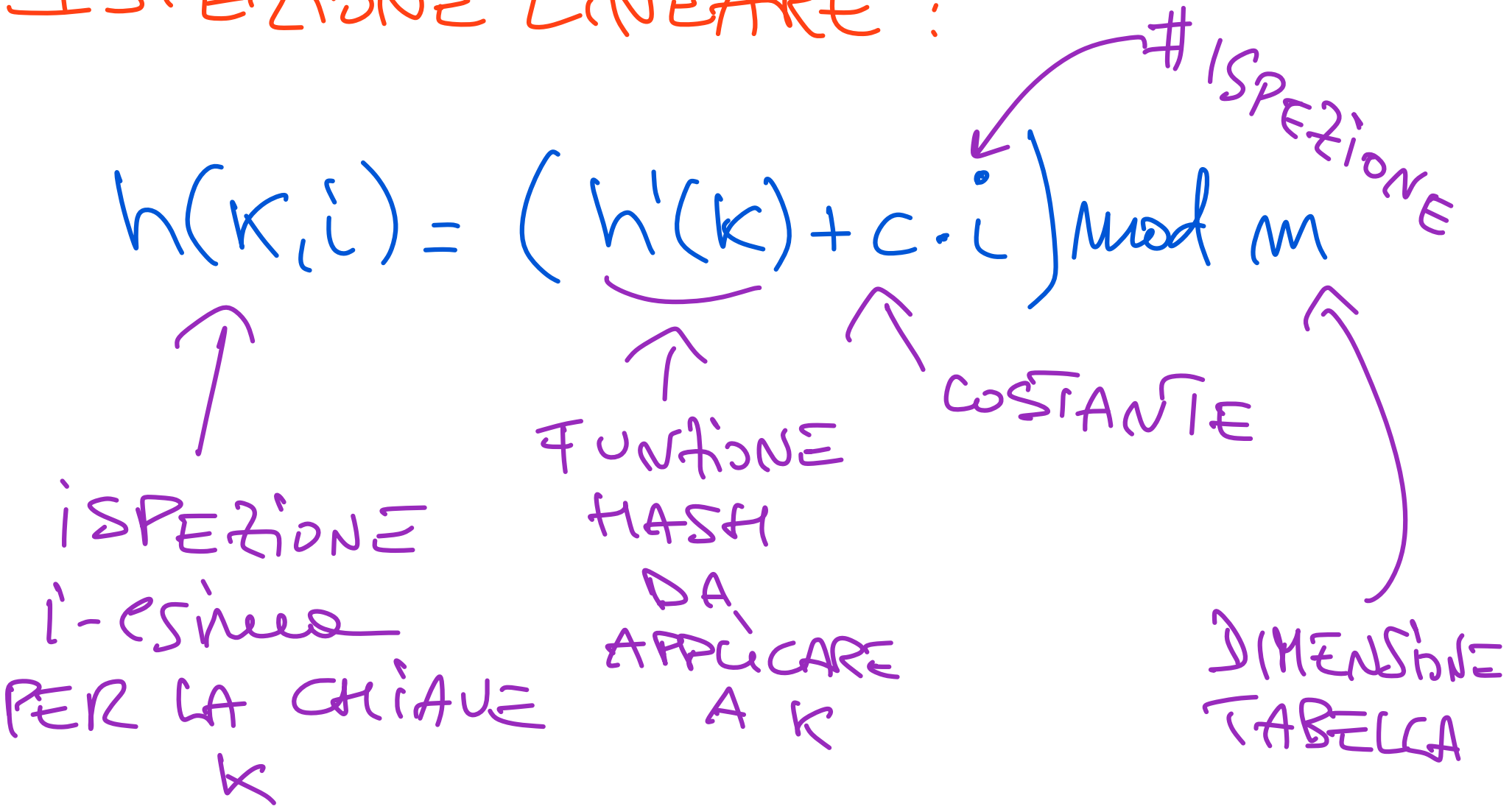
Indirizzamento aperto

TECNICHE UTILIZZATE:

- ISPEZIONE LINEARE
- ISPEZIONE QUADRATICA
- DOPPIO HASHING

Indirizzamento aperto

ISPEZIONE LINEARE :



Indirizzamento aperto

ISPEZIONE LINEARE:

$$h(k,i) = \left(h'(k) + c \cdot i \right) \bmod m$$

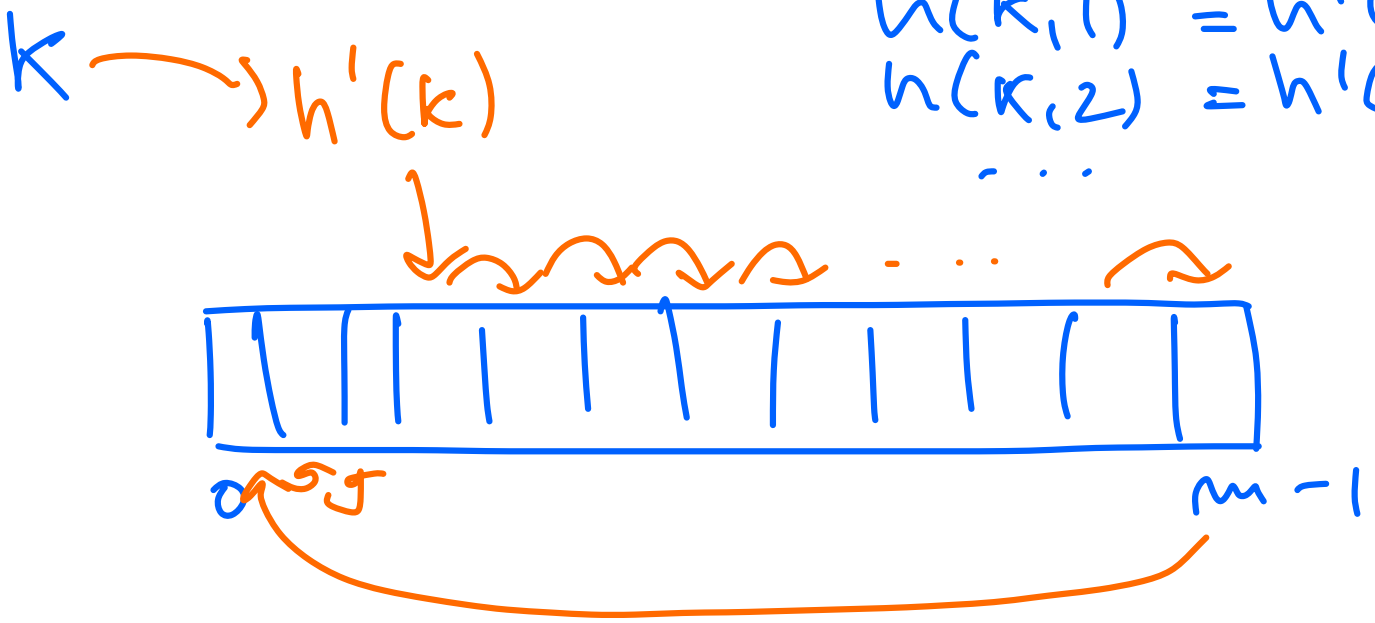
ESEMPIO: $c = 1$

$h(k,0) = h'(k) \bmod m$

$h(k,1) = h'(k) + 1 \bmod m$

$h(k,2) = h'(k) + 2 \bmod m$

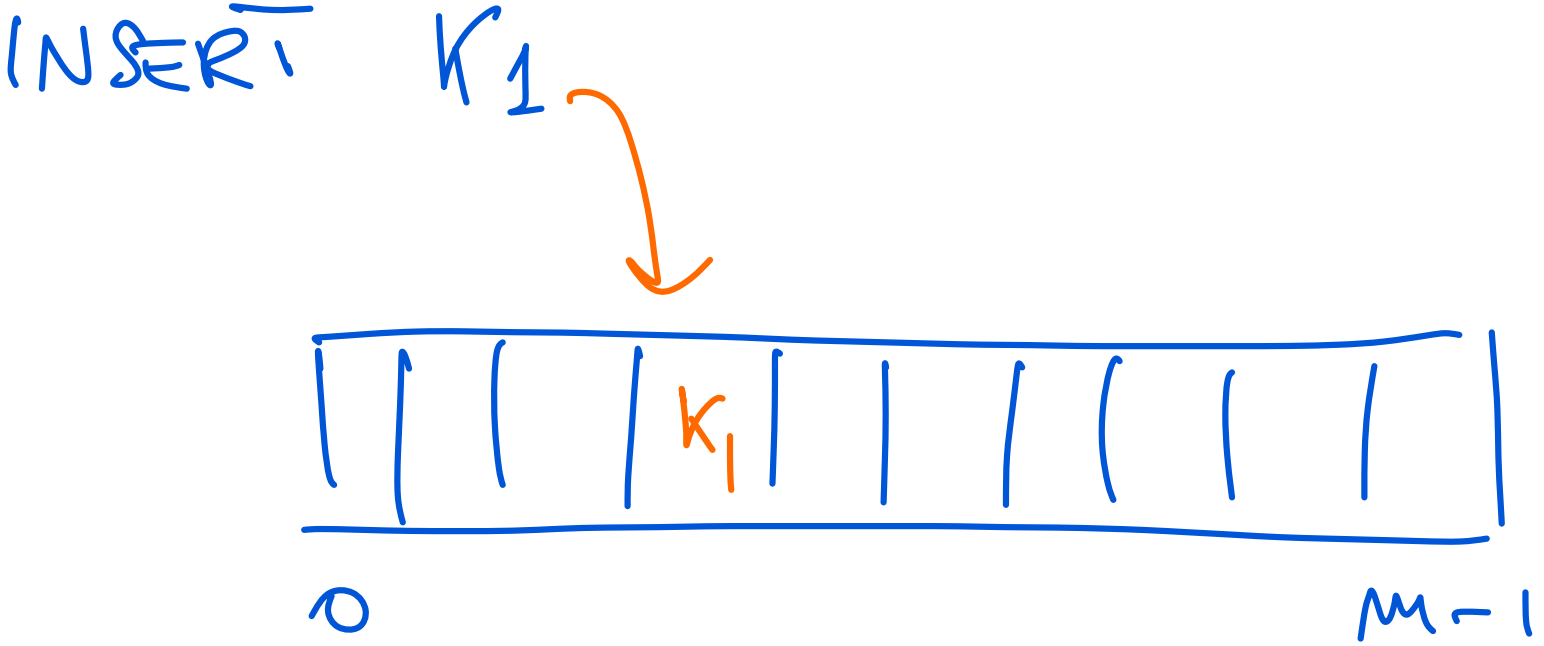
...



Indirizzamento aperto

ISPEZIONE LINEARE:

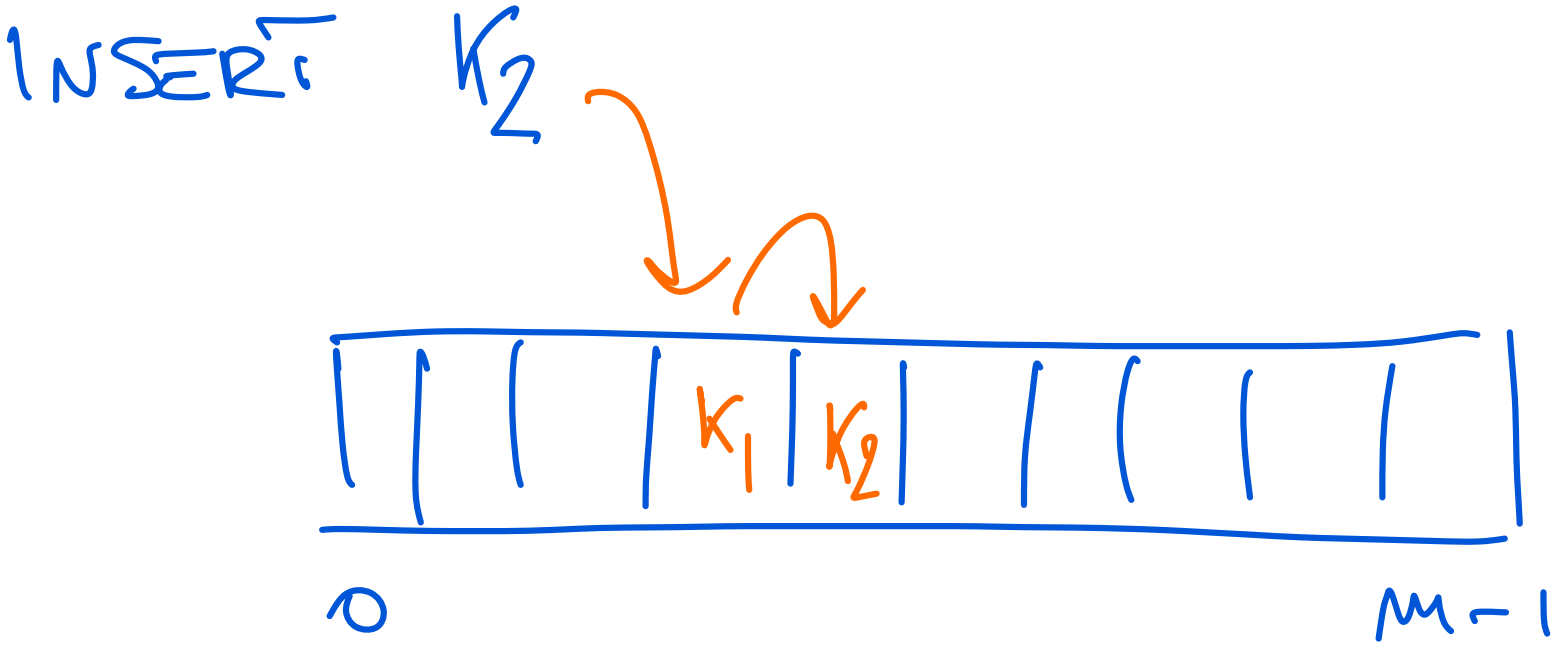
SI TENDE A CREARE AGGLOMERATI



Indirizzamento aperto

ISPEZIONE LINEARE:

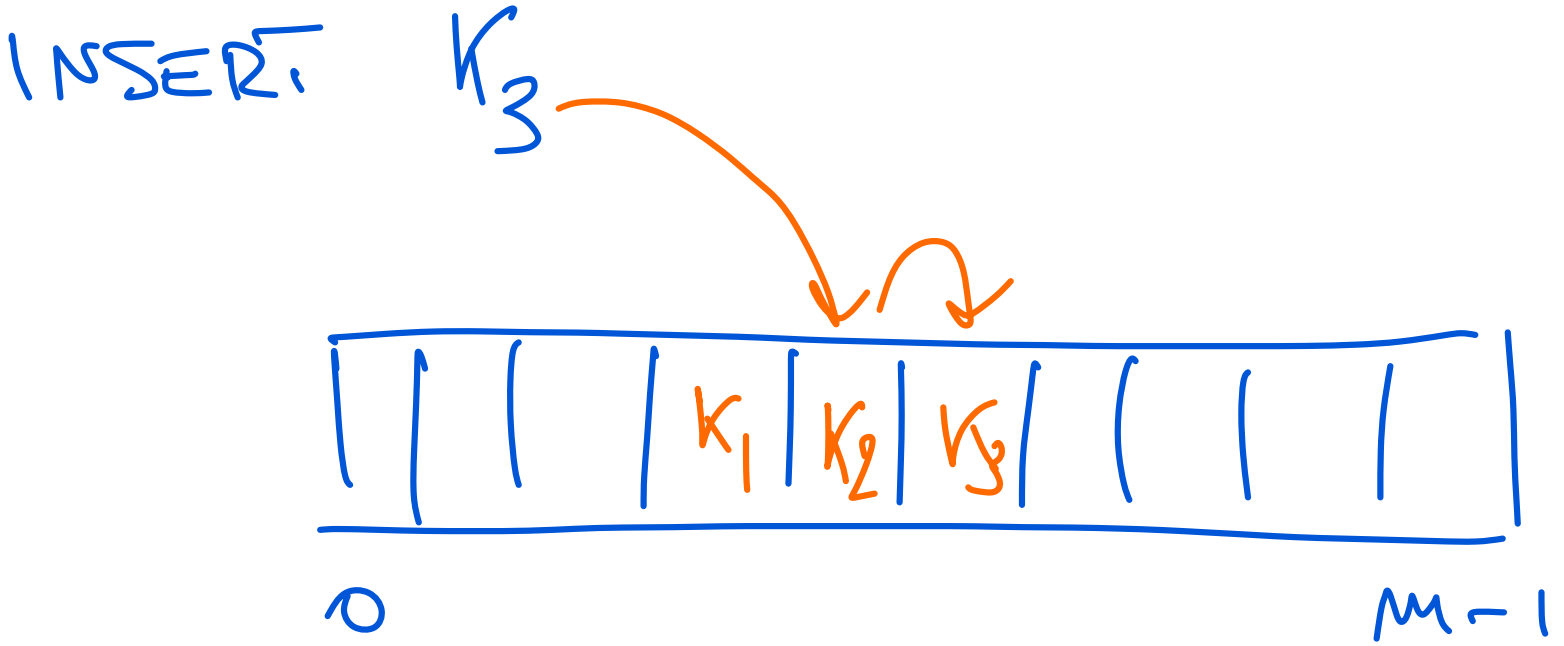
SI TENDE A CREARE AGGLOMERATI



Indirizzamento aperto

ISPEZIONE LINEARE:

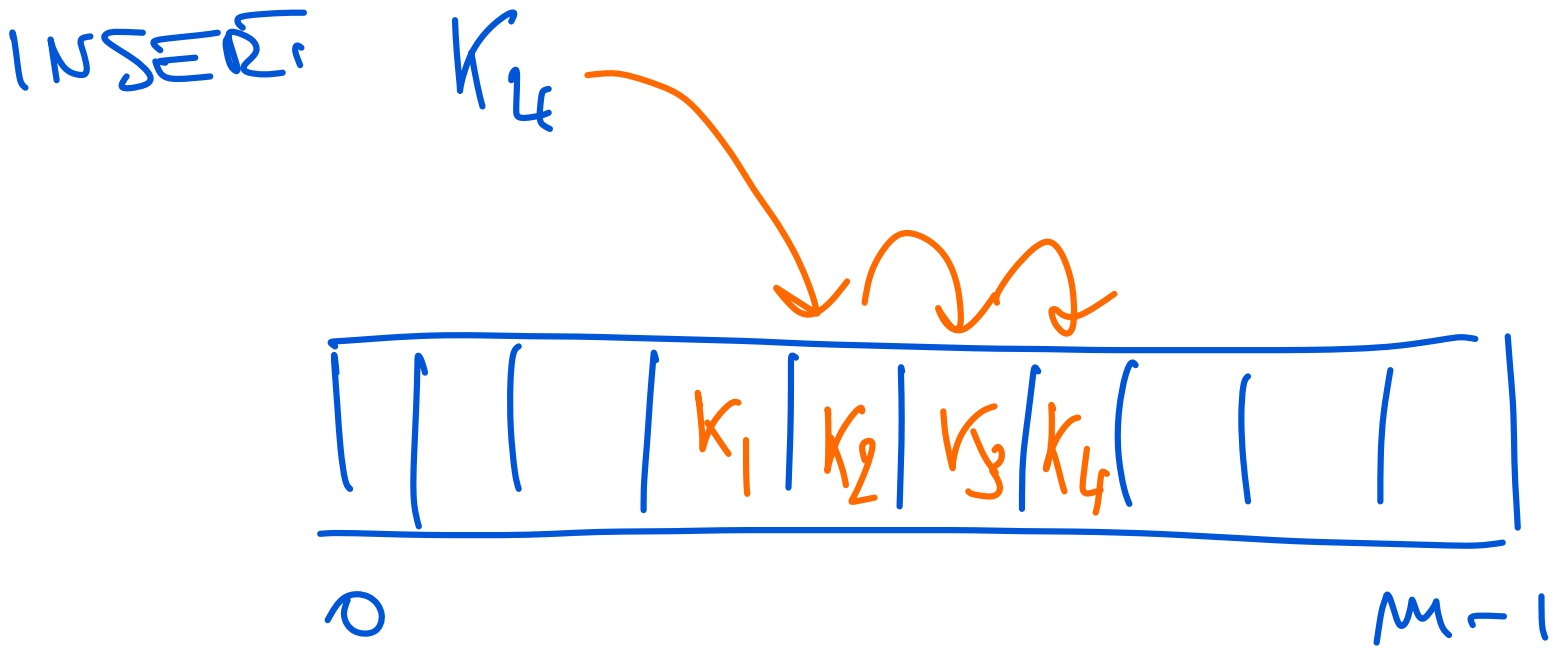
SI TENDE A CREARE AGGLOMERATI



Indirizzamento aperto

ISPEZIONE LINEARE:

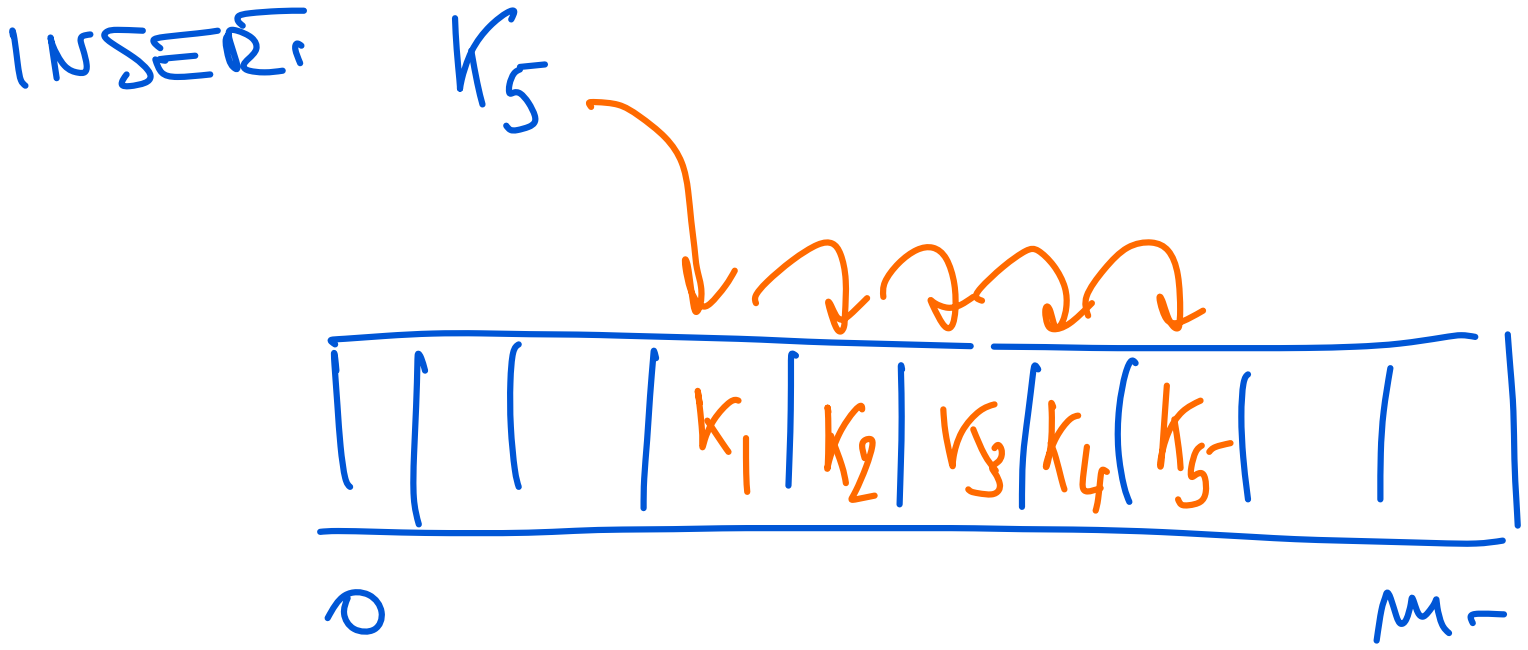
SI TENDE A CREARE AGGLOMERATI



Indirizzamento aperto

ISPEZIONE LINEARE:

SI TENDE A CREARE AGGLOMERATI



I TEMPI DELLE OPERAZIONI CRESCE!

Indirizzamento aperto

OSSERVAZIONI:

1) IL FATTORE DI CARICO È $0 \leq \alpha \leq 1$

2) LA TABELLA SI PUÒ RIEMPIRE E ANDARE IN OVERFLOW

Indirizzamento aperto

EVITARE L'OVERFLOW

QUANDO IL FATTORE DI CARICO α SALE SOPRA UNA CERTA SOGLIA (TIPICAMENTE 0,5 - 0,75)

- SI ALLOCA UNA NUOVA TABELLA DI DIMENSIONE DOPPIA $2m$
- SI INSERISCONO NUOVAMENTE LE CHIAVI NELLA NUOVA TABELLA

↙ ↘

× α DIMEZZATO
× NON CI SONO DELETED

COSTO $O(m)$ W.C.