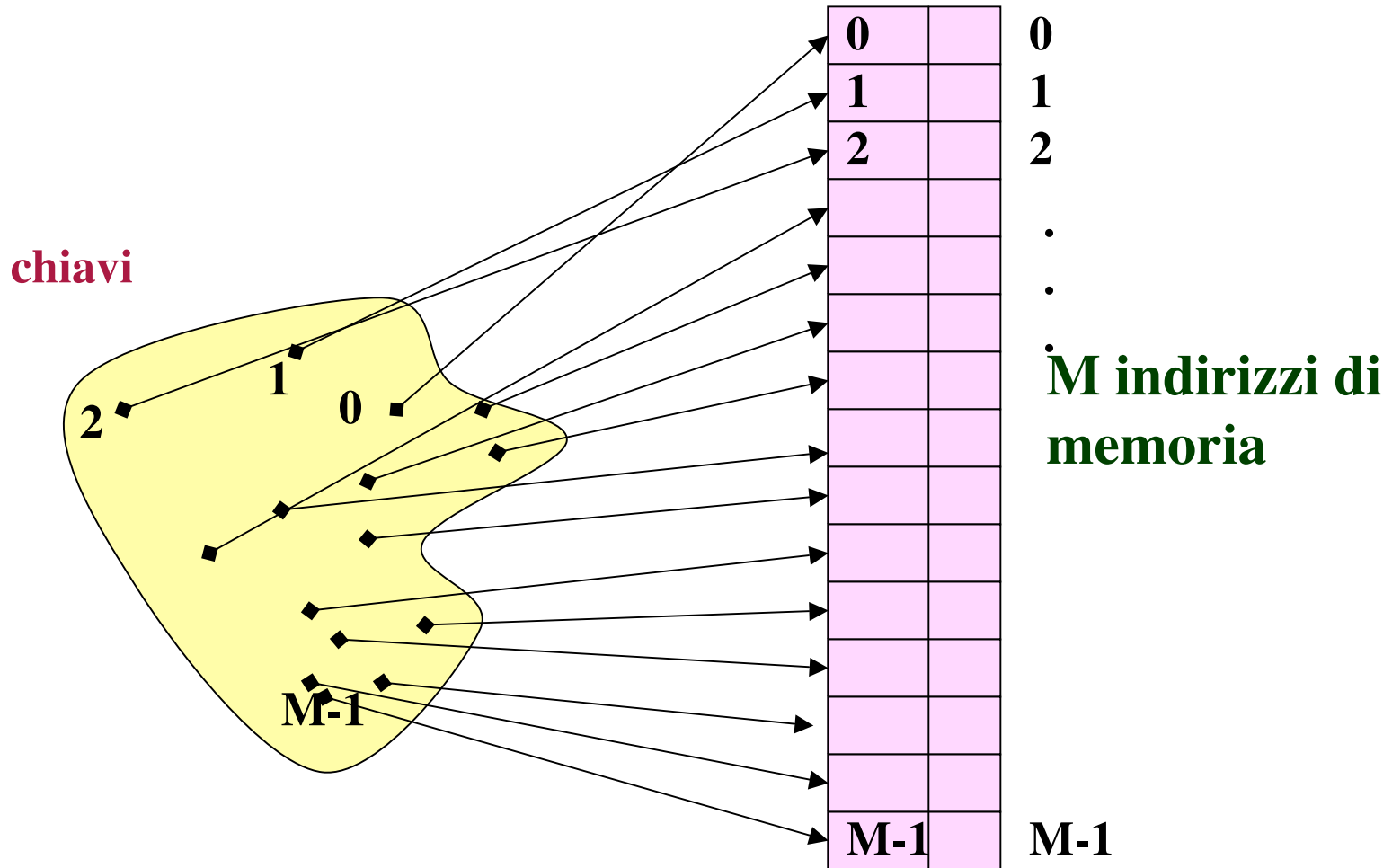


Il problema consiste nel memorizzare degli elementi per poi effettuare ricerche, inserimenti e cancellazioni (dizionari, tabelle dinamiche). Sia associata a ogni elemento un'unica "chiave". Se la chiave è un intero in un intervallo $0 \dots M-1$, si può semplicemente usare un vettore di M elementi!



Quando si può usare questa scelta?

1. Quando ogni chiave è un intero
2. Quando la chiave è unica
3. Quando questi interi variano in un insieme piuttosto piccolo
4. Quando non succede che su M elementi memorizzabili, con chiavi $0, \dots, M-1$, solo una frazione bassa viene inserita (spreco di memoria !!)

Proviamo a indebolire i vincoli di utilizzo del vettore

•1. Quando ogni chiave è un intero

Si potrebbe pensare di introdurre una funzione che associa un intero a una chiave, che può essere di qualunque tipo

•La funzione hash

$h(\text{chiave}) \mapsto \text{intero}$

•Così applicando h alla chiave si ottiene l'indirizzo nel vettore dove l'elemento di chiave h è memorizzato

• h deve essere biettiva, chiavi diverse devono essere associate a interi diversi nell'intervallo $0 \dots M-1$ e a ogni intero i nell'intervallo deve corrispondere una chiave possibile

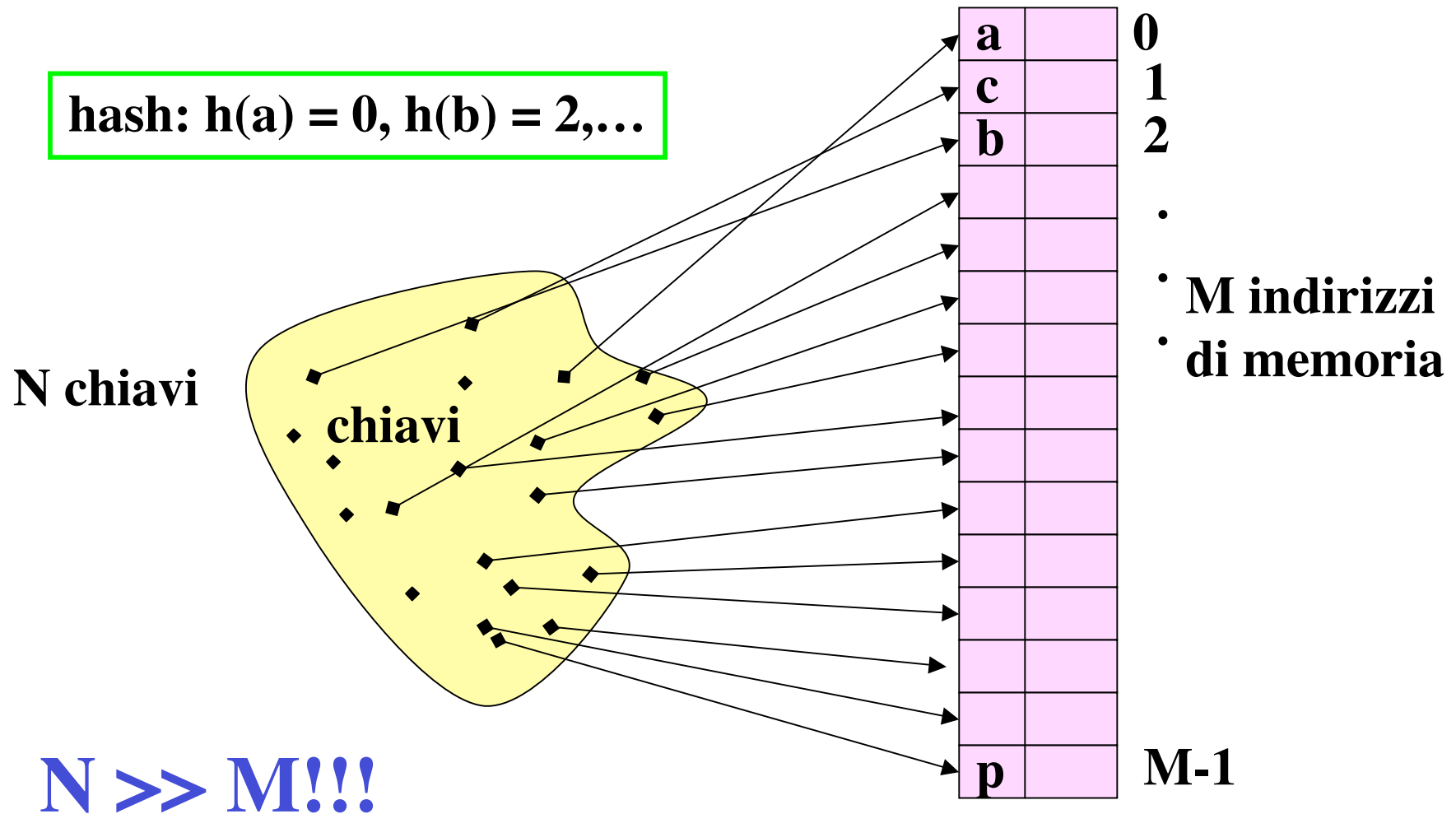
•se calcolare h è veloce, la ricerca, l'inserimento e la cancellazione sono ancora veloci!!

(Veloce = tempo indipendente dalla dimensione del vettore)

Proviamo a indebolire i vincoli di utilizzo del vettore

3. Quando questi interi variano in un insieme piuttosto piccolo

hash: $h(a) = 0, h(b) = 2, \dots$



Proviamo a indebolire i vincoli di utilizzo del vettore

2. Quando la chiave è unica

Normalmente l'universo delle chiavi è molto più grande dell'insieme che si vuole memorizzare. Per esempio se le chiavi sono stringhe di caratteri su un alfabeto di m elementi di lunghezza al più n ($n > 1$). Allora le possibili chiavi sono $m^{n+1}-1/m-1$.

Per $m = 21$ e $n = 6$ abbiamo 90.054.427 possibili chiavi.

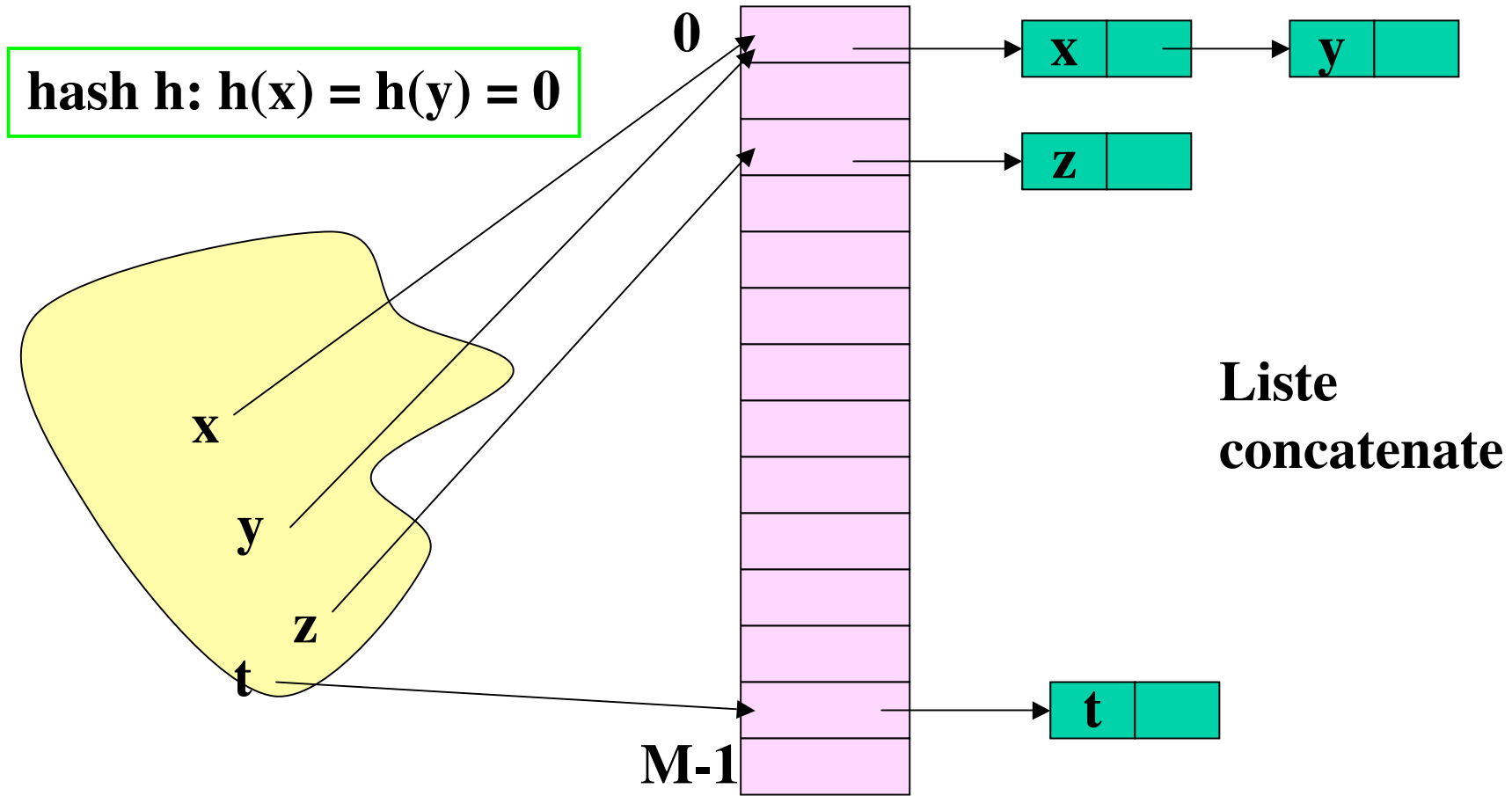
Questo rende difficile definire funzioni hash iniettive.

Se la funzione hash non è iniettiva, si verifica una **collisione**: due elementi con chiave diversa andrebbero nella stessa locazione di memoria.

Il metodo più semplice per ovviare a questo è utilizzare un vettore di **liste concatenate**.

Ora gli elementi sono memorizzati in un opportuno campo dei record della lista concatenata.

Metodo delle concatenazioni separate



Esempi di scelta per la funzione hash

Se k è un intero associato alla chiave, il metodo più comune per definire h è:

$$h(k) = k \bmod M, \text{ per } M \text{ primo}$$

Vantaggi:

- facilità di calcolo
- sufficiente uniformità di distribuzione

Svantaggio:

possibilità di collisioni: più chiavi associate allo stesso indirizzo