Tabelle: sparse e non

- Tabella == array bidimensionale
  - Naturale
  - Intuitiva
  - Efficiente?
- PROBLEMA: occupazione memoria per tabelle sparse

Es. Tabella sparsa: Voti per studenti

- Dati:
  - 1000 studenti
  - esami * ridondanza * CCS * mutuabilita’
  - 28 * 1.5 * 10 * 0.6 = 252 ~ 1/10
- La tabella (struttura dati “naturale”) spreca ~ 80% dello spazio allocato
- Quantifichiamo
  - 252 * 1000 * 4 = ~1MB

Problemi di indirizzamento

- Soluzione pessima
  - Array bidimensionale di stringhe
  - Uso prima riga e prima colonna come chiave
- Spreco enorme di memoria
- Tabelle di associazione corso numero e studente numero
- Non ordinate? Basta una colonna
- Ordinate? Due colonne

Spreco di memoria

- 1 MB? WHO CARES!
- Non sempre vero…
  - Il nostro e' solo un esempio: ci possono essere molti piu' dati
    - E.g. grandi popolazioni su cui fare statistiche
  - E se invece di un PC state progettando un sistema Embedded?
- Sempre FARE ATTENZIONE

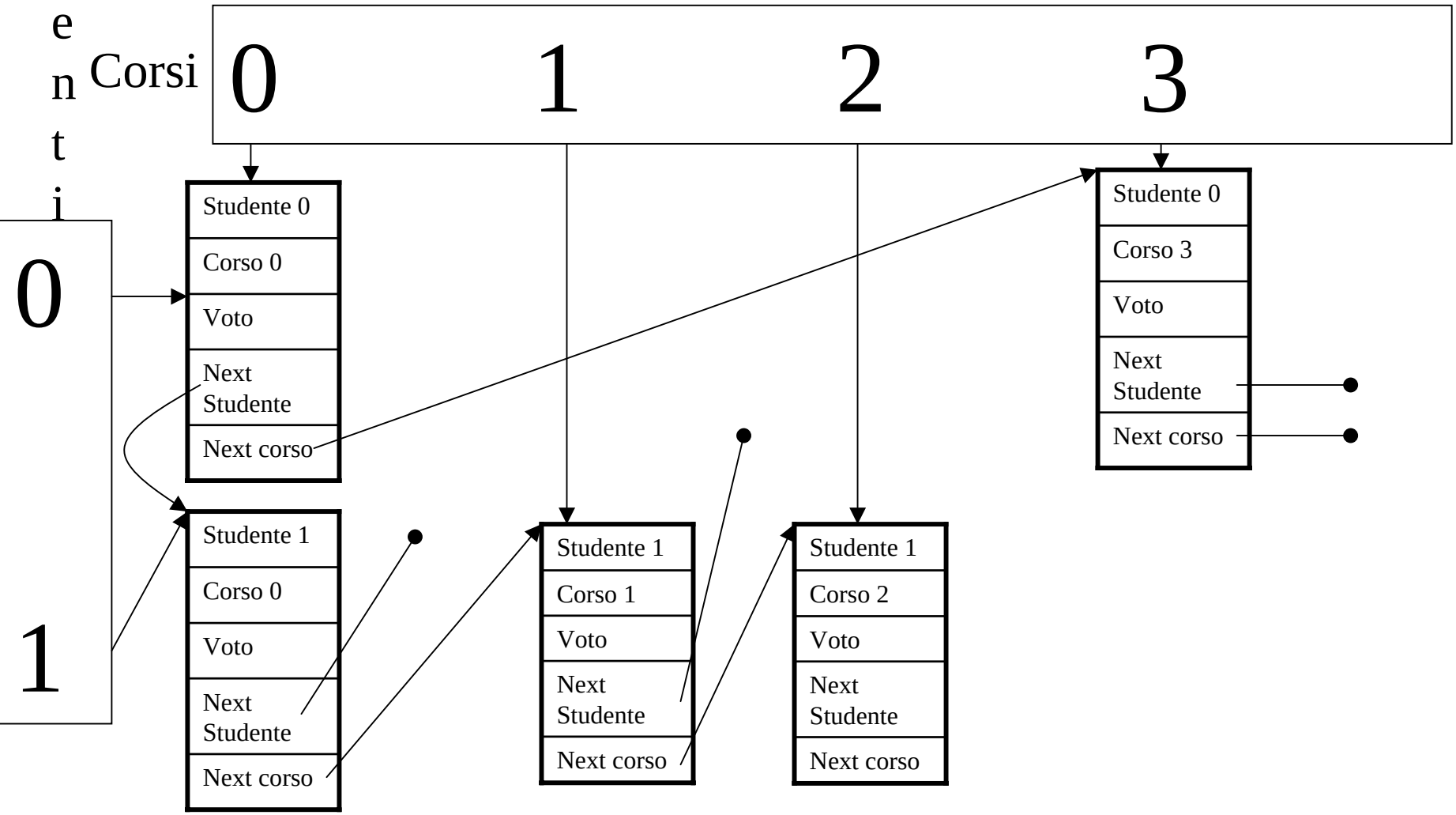
Risparmiare memoria

- Soluzione 1: lookup table
- Soluzione 2: implementazione meno naturale ma piu’ efficiente
  - Tabelle di liste

Implementazione

- 1 Array monodimensionale di riferimenti a nodi numerata secondo i corsi
- 1 Array monodimensionaledi riferimenti a nodi numerata secondo gli studenti
- Nodi contenenti:
  - Numero studente
  - Numero corso
  - Voto
  - Prossimo studente
  - Prossimo corso

Visualizzazione della struttura



Array vs Liste

- ARRAY
- Pro
  - Array consente accesso diretto
  - Array non usa memoria per riferimenti
- Contro
  - Inserimenti e rimozioni richiedono riorganizzazione
  - Dimensione fissata

Array vs Liste

- LISTE
- Pro
  - Inserimenti e rimozioni non richiedono riorganizzazione
  - Dimensione variabile
- Contro
  - Accesso sequenziale
  - Richiede memoria per i riferimenti