Strutture Dati e Algortimi

Pile e Code

Luca Di Gaspero, Università degli Studi di Udine

Pile

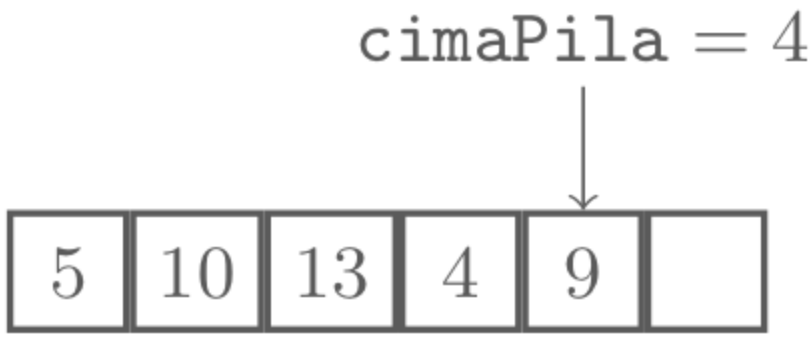
Pile

Collezioni di elementi in cui le operazioni disponibili, come l'estrazione di un elemento, sono ristrette unicamente a quello più recentemente inserito

- Politica di accesso **Last In First Out (LIFO)**: l'ultimo elemento inserito è il primo ad essere estratto
- Operazioni:
  - `Push(p, d)`: inserisce un nuovo elemento in cima alla pila
  - `Pop(p)`: estrae l'elemento in cima alla pila (opzionalmente restituendo l'informazione in esso contenuta, non per noi)
  - `Top(p)`: restituisce l'informazione contenuta nell'elemento in cima alla pila
  - `Empty(p)`: verifica se la pila è vuota

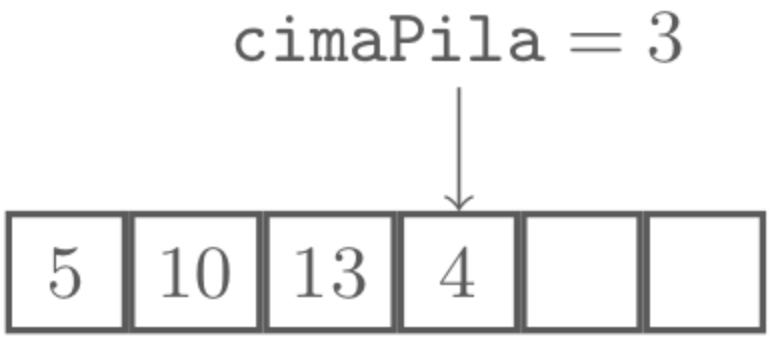
Pile: implementazione con array

- Elementi della pila memorizzati in un array di dimensione iniziale predefinita
- Array ridimensionato per garantire che la dimensione sia proporzionale al numero di elementi effettivamente nella pila
- Elementi memorizzati in sequenza nell'array a partire dalla locazione iniziale, inserendoli man mano nella prima locazione disponibile
- La cima della pila corrisponde all'ultimo elemento della sequenza

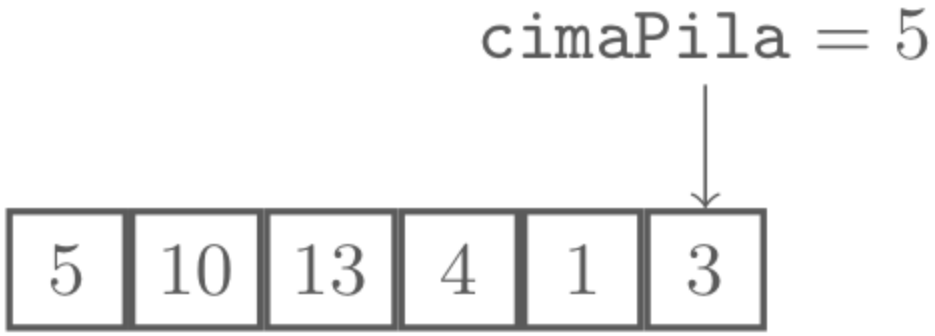


Pile con array: operazioni

- Dopo una `Pop(p)`

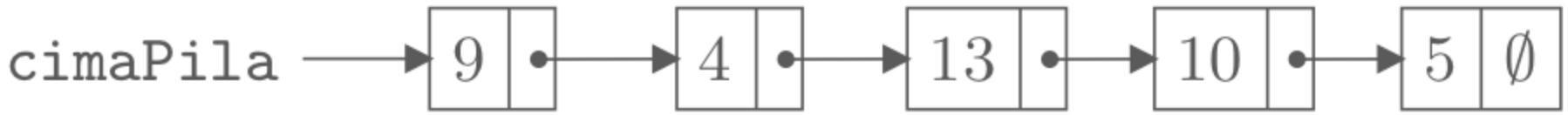


- Dopo `Push(p, 1)`, `Push(p, 3)`



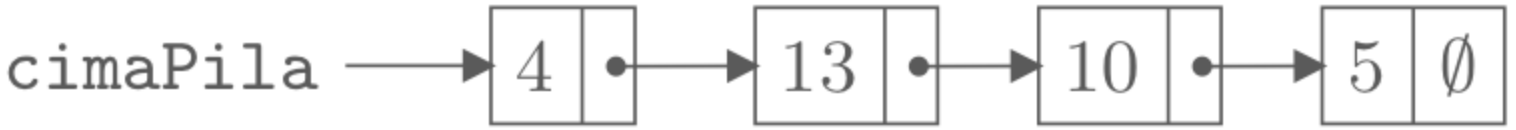
Pile: implementazione con lista

- Elementi della pila memorizzati in una lista ordinata per istante di inserimento decrescente
- La cima della pila corrisponde all'inizio della lista
- Le operazioni agiscono tutte sull'elemento iniziale della lista

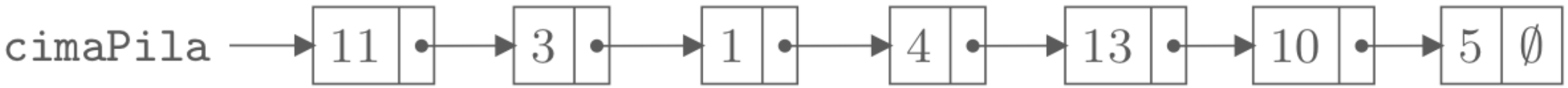


Pile con lista: operazioni

- Dopo una `Pop(p)`



- Dopo `Push(p, 1)`, `Push(p, 3)`, `Push(p, 11)`



Code

Code

Collezioni di elementi in cui le operazioni disponibili, come l'estrazione di un elemento, sono ristrette unicamente a quello inserito meno recentemente

Politica di accesso **First In First Out (FIFO)**: il primo elemento inserito è il primo ad essere estratto

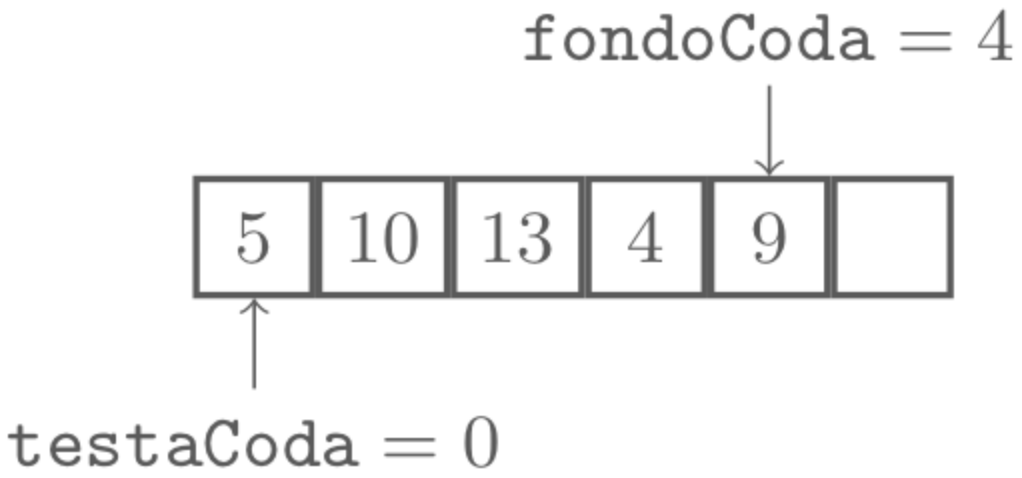
- Operazioni:
  - `Enqueue(q, d)`: inserisce un nuovo elemento in fondo alla coda
  - `Dequeue(q)`: estrae l'elemento in testa alla coda (opzionalmente restituendo l'informazione in esso contenuta, non per noi)
  - `First(q)`: restituisce l'informazione contenuta nell'elemento in testa alla coda senza estrarlo
  - `Empty(q)`: verifica se la coda è vuota o meno

Code con array: implementazione

- Elementi della coda memorizzati in un array di dimensione iniziale predefinita
- Array ridimensionato per garantire che la dimensione sia proporzionale al numero di elementi effettivamente nella coda
- Elementi memorizzati in sequenza nell’array a partire dalla locazione iniziale, inserendoli man mano nella prima locazione disponibile

Code con array: implementazione

- La testa della coda corrisponde al primo elemento della sequenza
- Il fondo della coda corrisponde all'ultimo elemento della sequenza
- Gestione "circolare" della coda



Code con liste: implementazione

- Nodi concatenati e ordinati in modo crescente secondo l'istante di inserimento
- Il primo nodo della sequenza corrisponde alla "testa" della coda ed è il nodo da estrarre nel caso di una `Dequeue`
- L'ultimo nodo corrisponde al "fondo" della coda ed è il nodo a cui concatenare un nuovo nodo, inserito mediante `Enqueue`

Code di Priorità

Code di priorità

- Collezioni di elementi in cui a ogni elemento è associato un valore (priorità) appartenente a un insieme totalmente ordinato (solitamente l'insieme degli interi positivi)
- Estensione della coda: le operazioni sono le stesse della coda: `Empty`, `Enqueue`, `First` e `Dequeue`
- `First` restituisce l'elemento di priorità massima (o minima)

Code di priorità con liste: implementazione

- Prima soluzione: *lista non ordinata*
  - La `Enqueue` richiede tempo costante, con i nuovi elementi inseriti a un estremo della lista
  - La `Dequeue` e la `First` richiedono tempo $O(n)$: perché è necessario individuare l'elemento di priorità massima all'interno della lista
- Seconda soluzione: *lista ordinata*
  - La `Dequeue` e la `First` richiedono tempo costante: l'elemento di massima priorità si trova in testa alla lista
  - La Enqueue richiede tempo $O(n)$: i nuovi elementi vanno inseriti alla posizione corretta rispetto all'ordinamento

Il costo delle operazioni è sbilanciato: vi è la necessità di un implementazione che lo renda più simile