

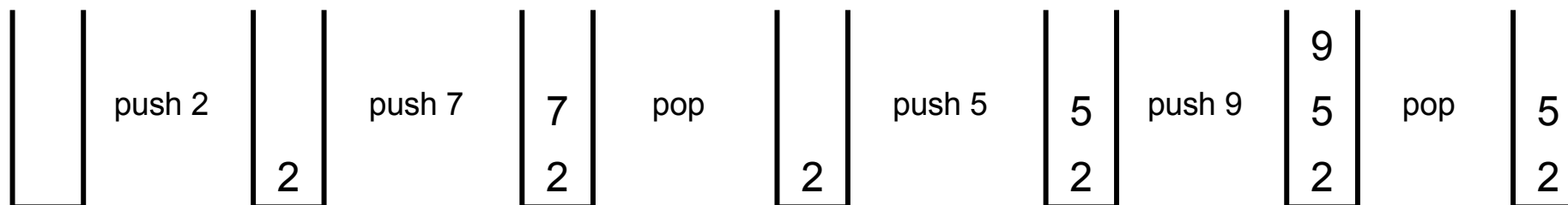
# Il tipo di dato astratto **Pila**

---

# Il tipo di dato Pila

---

- Una **pila (stack)** è una sequenza di elementi (tutti dello stesso tipo) in cui l'inserimento e l'eliminazione di elementi avvengono secondo la regola seguente:
  - L'elemento che viene eliminato tra quelli presenti nella pila deve essere quello che è stato inserito per ultimo.
- Si parla di gestione **LIFO** (per “**Last In, First Out**”).
- *Esempio*: rappresentazione grafica di una pila



- L'elemento in cima alla pila viene detto **elemento affiorante**.
- Una pila senza elementi viene detta **pila vuota**.

# Utilizzo delle pile

---

- Uno degli utilizzi delle pile si ha nella soluzione di problemi:
  - che richiedono di effettuare più scelte in successione, e
  - per cui può essere necessario ritrattare le scelte fatte per provare altre alternative
    - → si deve tornare indietro sull'ultima scelta fatta (backtracking)
- Si utilizza una pila per memorizzare la sequenza di scelte fatte:
  - Push di una nuova scelta;
  - Pop ed eventuale Push della scelta alternativa quando si deve fare backtracking.

# Specifica del tipo di dato Pila

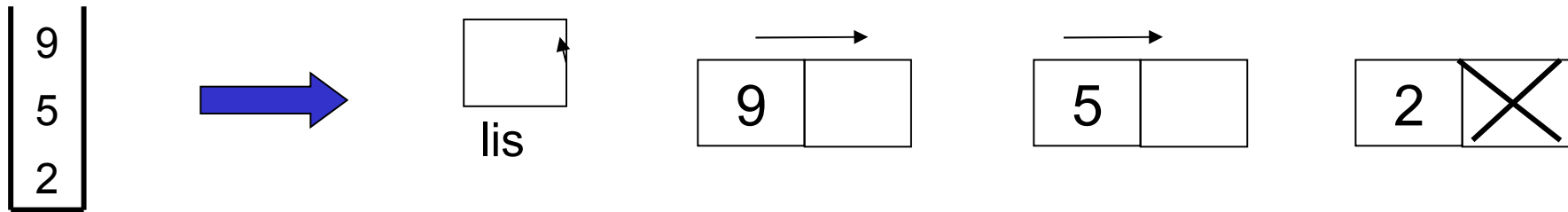
---

- *Domini di interesse:* Pila, Elemento, Booleano
- *Costanti:* **PilaVuota**
- *Operazioni:*
  - **TestPilaVuota** :  $Pila \rightarrow Booleano$ 
    - restituisce vero se la pila è vuota, falso altrimenti
  - **TopPila** :  $Pila \rightarrow Elemento$ 
    - restituisce l'elemento affiorante di una pila
  - **Push** :  $Pila \times Elemento \rightarrow Pila$ 
    - inserisce un elemento in cima ad una pila e restituisce la pila modificata
  - **Pop** :  $Pila \rightarrow Pila \times Elemento$ 
    - estrae l'elemento affiorante dalla cima della pila e lo restituisce (insieme alla pila modificata)

# Rappresentazione collegata

---

- Una pila è rappresentata mediante una **lista** nella quale il primo elemento è l'elemento affiorante della pila.



- Per rappresentare la pila vuota: *lista vuota (puntatore head = NULL)*
- Non è più necessario imporre un limite al numero massimo di elementi nella pila.
- Anche in questo caso le funzioni sono molto semplici e di costo costante:
  - **Push**: tramite un inserimento in testa ad una lista
  - **Pop**: tramite la cancellazione del primo elemento di una lista
- Perché non abbiamo usato una lista in cui l'elemento affiorante è l'ultimo della lista?

# Rappresentazione collegata

---

## InitPila

```
/* si inizializza la pila ponendo a NULL il puntatore all'elemento
   affiorante della pila */
   TipoPila * head = NULL;
```

## TestPilaVuota

```
/* restituisce TRUE se la pila p e' vuota, FALSE altrimenti */
int TestPilaVuota(Tipopila * p) {
    return (p == NULL);
}
```

## TopPila

```
/* restituisce l'elemento affiorante della pila p,
   senza modificare la pila */
TipoPila * TopPila(Tipopila *p) {
    return p;
}
```

# Rappresentazione collegata

---

## Push

```
/* inserisce l'elemento u in cima alla pila */
```

```
TipoPila * Push(TipoPila u, TipoPila *pila) {  
    add_to_head(u,pila);  
}
```

## Pop

```
/* elimina l'elemento affiorante della pila, restituendone il  
   valore in u */
```

```
TipoPila * Pop(TipoPila *pila, TipoPila * u) {  
    *u = *pila;  
    pila = remove_head(pila);  
    return pila;  
}
```