Tecnica del Doubling

Una tecnica per ridimensionare lo spazio allocato quando non ci basta più

Un caso tipico:
L'utente immette dati in sequenza, senza specificare quanti. Quando l'input finisce vogliamo che tutti i valori siano memorizzati in un array.

Problema: non sappiamo quanto spazio allocare all'array contenente l'input, perchè ne ignoriamo la lunghezza

Possibile soluzione mediante la strategia del doubling...

Tecnica del Doubling

Si riempie un blocco di memoria (inizialmente di dimensione 1) con i dati che arrivano in input

Quando lo spazio nel blocco corrente è esaurito:

- Copiamo tutti i dati in un nuovo blocco di *dimensione doppia* (che diventa quello corrente)
- Liberiamo lo spazio allocato al blocco precedente

Esempio

INPUT: 0 3 4 5 .... EOF

2^0

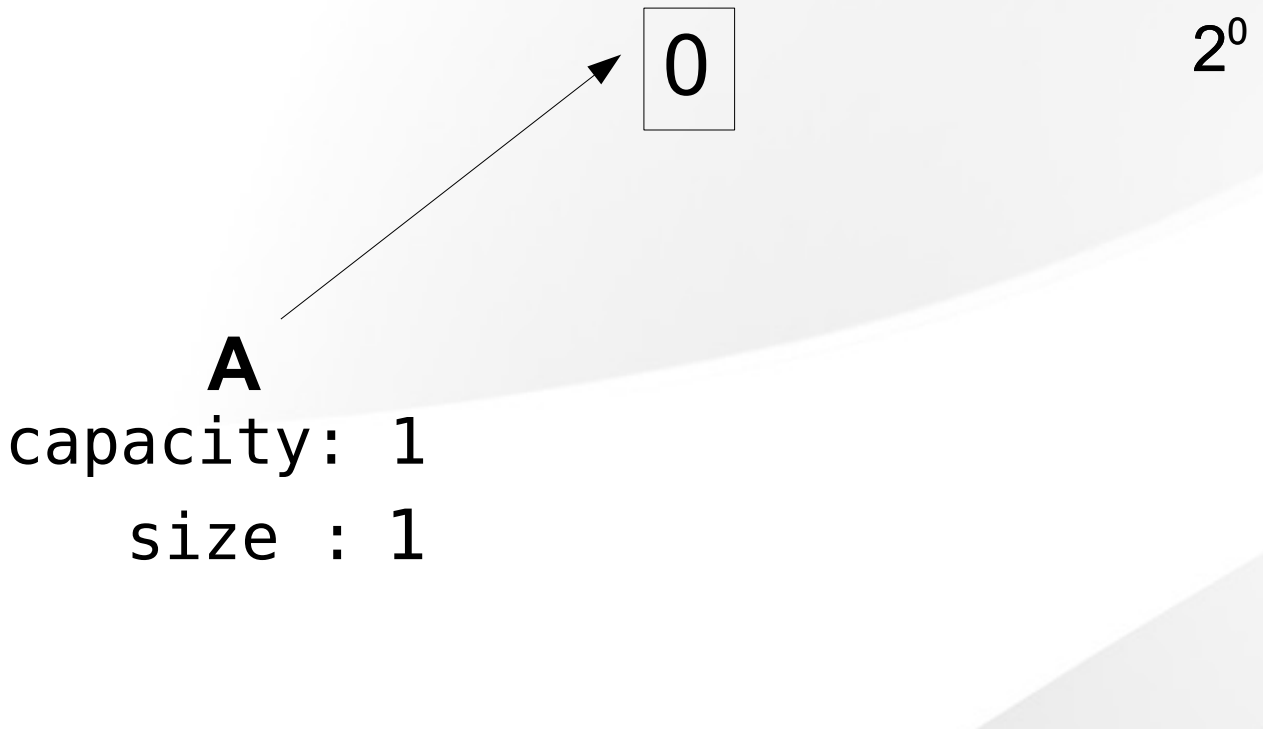
A

capacity: 1

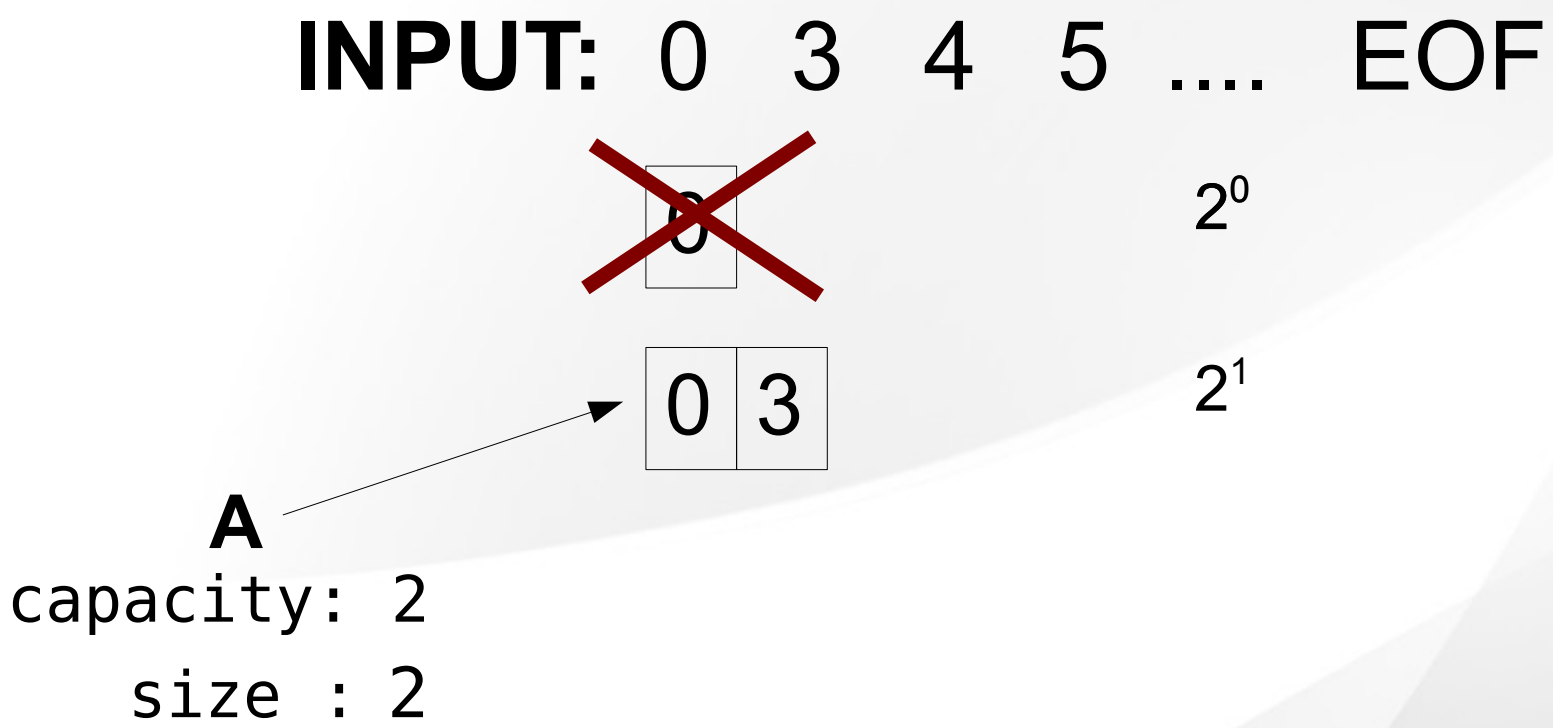
size : 0

Esempio

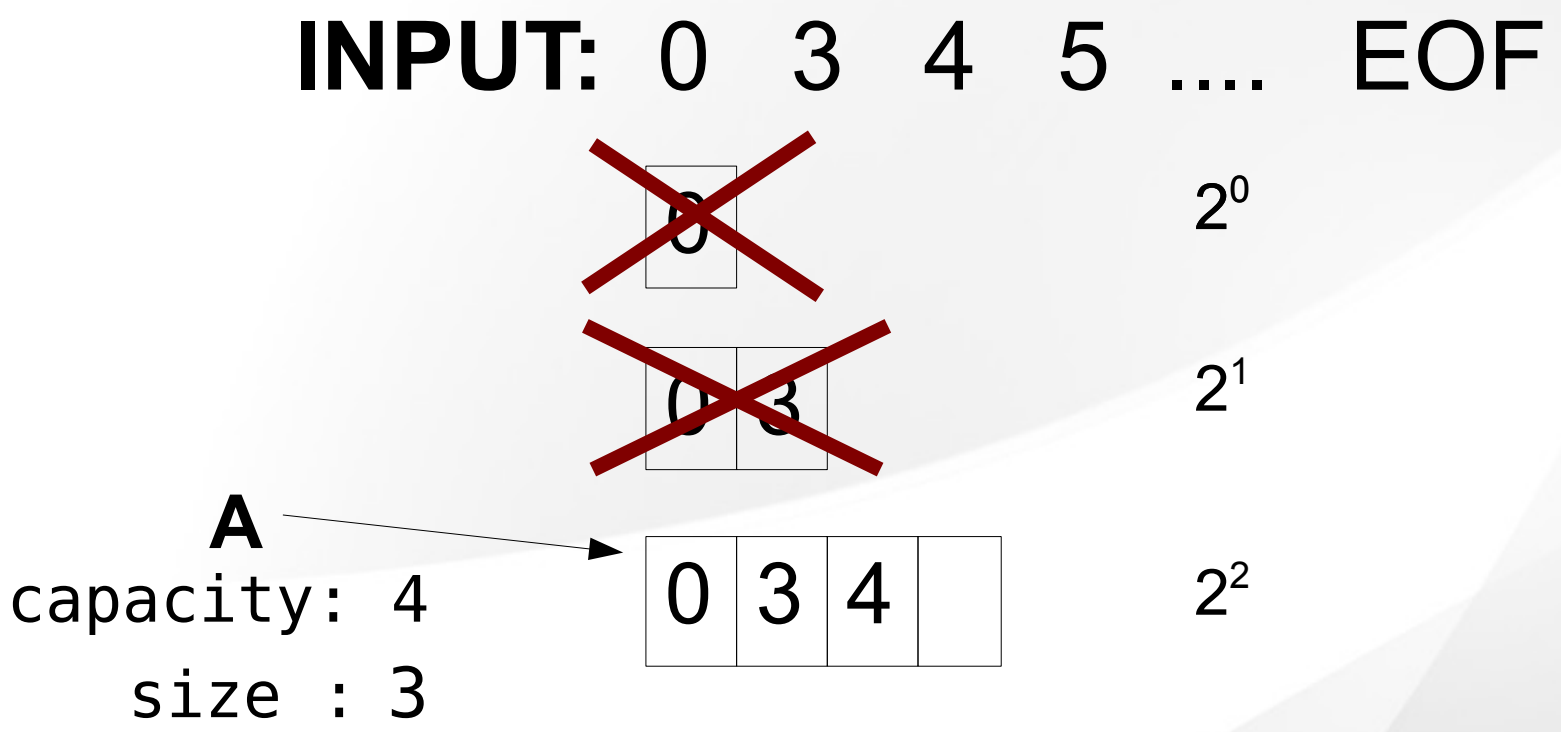
INPUT: 0 3 4 5 .... EOF



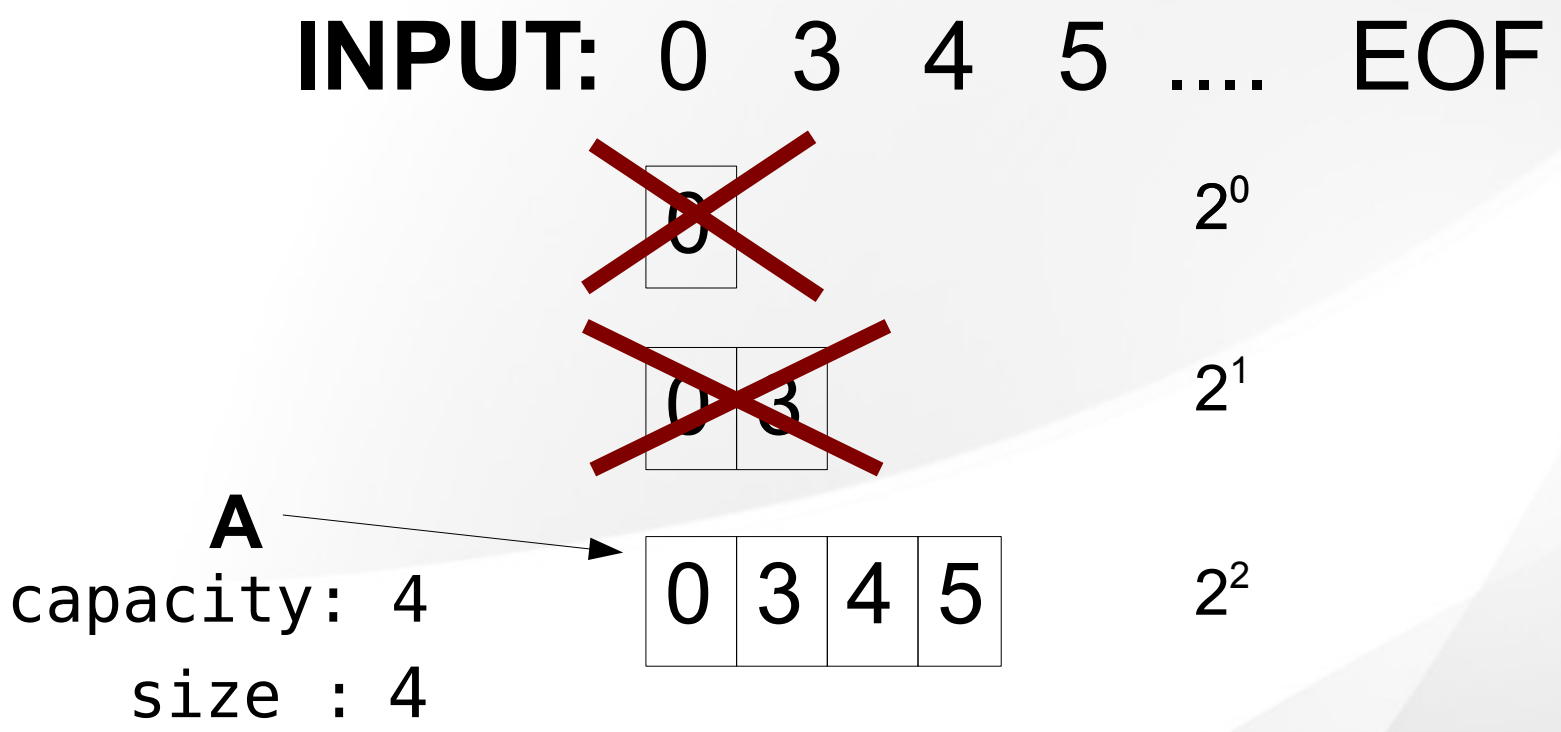
Esempio



Esempio



Esempio



Pseudo-codice

```
INSERT( e ):

  IF ( size == capacity )   {

   // copia gli elementi di A in un blocco di
      dimensione doppia

   // dealloca il blocco prima occupato da A

    capacity = 2*capacity;
  }

  A[size] = e;
  size = size + 1;
```

Pseudo-codice (alternativo)

```
INSERT( e ):

A[size] = e;
size = size + 1;

IF ( size == capacity )   {

  // copia gli elementi di A in un blocco di
     dimensione doppia

  // dealloca il blocco prima occupato da A

    capacity = 2*capacity;
 }
```

Analisi asintotica

Quanto ci costa effettuare N inserzioni:

- Scrittura di N elementi
- Copie degli elementi ad ogni riallocazione:

$$\sum_{i \leqslant w} 2^i = 2^{w+1} - 1 \leqslant 2\mathrm{N} \qquad \text{con} \quad 2^w < N \leqslant 2^{w+1}$$

Numero di elementi copiati nell' i-esima riallocazione

Analisi asintotica

Quanto ci costa effettuare N inserzioni:

- Scrittura di N elementi
- Copie degli elementi ad ogni riallocazione:

$$\sum_{i \leqslant w} 2^i = 2^{w+1} - 1 \leqslant 2\mathrm{N} \qquad \text{con} \quad 2^w < N \leqslant 2^{w+1}$$

Totale: 3N scritture = O(N)

Analisi asintotica

Quanto ci costa effettuare N inserzioni:

- Scrittura di N elementi
- Copie degli elementi ad ogni riallocazione:

$$\sum_{i \leqslant w} 2^i = 2^{w+1} - 1 \leqslant 2N \qquad \text{con} \quad 2^w < N \leqslant 2^{w+1}$$

Totale: 3N scritture = O(N)

Il costo *ammortizzato* per elemento è una costante, uguale a 3.

Analisi asintotica

Quanto ci costa effettuare N inserzioni:

- Scrittura di N elementi
- Copie degli elementi ad ogni riallocazione:

$$\sum_{i \leqslant w} 2^i = 2^{w+1} - 1 \leqslant 2\mathrm{N} \qquad \text{con} \quad 2^w < N \leqslant 2^{w+1}$$

Totale: 3N scritture = O(N)

Key Fact: blocchi esponenzialmente crescenti

- Quale complessità avremmo altrimenti?

Esercizio: lettura input

INPUT: array A di interi (lunghezza non specificata)
OUTPUT: se A è *ordinato in senso crescente* `SI x\n`, dove x è il numero di interi *distinti* in esso contenuti, altrimenti `NO\n`.

Per rilevare la fine dell'input si controlli il risultato di *scanf:*

```
while ( scanf("%d", &e) == 1 ) {

  // ciclo terminato da EOF o da
  // un carattere

  // inserzione di e con doubling...
}
```

Importante: stampare solo l'output nel formato specificato!

Esercizio: lettura input

INPUT: array A di interi (lunghezza non specificata)
OUTPUT: se A è *ordinato in senso crescente* SI x\n, dove x è il numero di interi *distinti* in esso contenuti, altrimenti NO\n.

```
while ( scanf("%d", &e) == 1 ) { // doubling
    IF ( size == capacity )   {

      // copia A in blocco di dimensione doppia
      // dealloca il blocco prima occupato da A

      capacity = 2*capacity;
  }

  A[size++] = e;
}
  // … verifica ordinato e conta distinti ...
```