

Insertion sort

Leonardo Bizzoni

January 1, 2024

Input: sequenza di n numeri

Output: una permutazione della sequenza ricevuta in input dove i numeri sono in ordine crescente

L'algoritmo ordina i numeri in input *sul posto* ovvero i numeri sono sistemati all'interno dell'array.

Al termine della **procedura** (*void function*) l'array contiene la sequenza di output ordinata.

```
def insertion_sort(A):
    for j in range(1, len(A)):
        current = A[j]
        i = j - 1

        while i >= 0 and A[i] > current:
            A[i+1] = A[i]
            i = i - 1

        A[i+1] = current
```

Gli elementi $A[0..j]$ sono quelli che originariamente occupavano le posizione da 0 a j ma che adesso sono ordinati. Questa proprietà si chiama invariante di ciclo:

- **inizializzazione:** $j = 1$ è la prima istruzione che esegue il ciclo for, in $A[0..j - 1]$ c'è solo un elemento quindi il sottoarray è ordinato e contiene gli elementi originali di $A[0..j - 1]$

- **conservazione:** il corpo del ciclo for opera spostando gli elementi verso destra finchè non trova la posizione appropriata per inserirli l'elemento corrente quindi in $A[0..j]$ ci sono sempre gli elementi originali e sono ancora ordinati correttamente
- **conclusione:** la condizione che determina la conclusione del ciclo for è che $j > \text{len}(A)$. Sostituendo j con $\text{len}(A) + 1$ nella formulazione dell'invariante di ciclo si ha che:
 - gli elementi contenuti in $A[0..\text{len}(A)]$ sono gli elementi originariamente contenuti in $A[0..\text{len}(A)]$
 - gli elementi contenuti in $A[0..\text{len}(A)]$ sono ordinati

1 Analisi tempo di esecuzione

$\text{def insertion_sort}(V[]) \{$
 $\text{for } (j = 2 \text{ to } \text{len}(V)) \{$
 $\text{current} = V[j]$
 $i = j - 1$
 $\text{while } (i > 0 \wedge V[i] > \text{current}) \{$
 $V[i+1] = V[i]$
 $i = i - 1$
 $\}$
 $V[i+1] = \text{current}$
 }

$C_1 \cdot (n-1)$
 $C_2 \cdot (n-1)$
 $C_3 \cdot (n-1)$
 $C_4 \cdot \frac{n(n-1)}{2} = C_4 \cdot \frac{n^2 - n}{2} \approx n^2$
 $C_5 \cdot (n-1)$

2 Esempio di esecuzione

$A =$	¹ 5	² 2	³ 4	⁴ 6	⁵ 7	⁶ 3	
	5 > 2? ✓						
$A =$	¹ 2	² 5	³ 4	⁴ 6	⁵ 7	⁶ 3	
	5 > 4? ✓						
$A =$	¹ 2	² 5	³ 5	⁴ 6	⁵ 7	⁶ 3	
	2 > 4? ✗						
$A =$	¹ 2	² 4	³ 5	⁴ 6	⁵ 7	⁶ 3	
	5 > 6? ✗						
$A =$	¹ 2	² 4	³ 5	⁴ 6	⁵ 7	⁶ 3	
	6 > 7? ✓						
$A =$	¹ 2	² 4	³ 5	⁴ 6	⁵ 6	⁶ 3	
	5 > 7? ✓						
$A =$	¹ 2	² 4	³ 5	⁴ 5	⁵ 6	⁶ 3	
	4 > 7? ✓						
$A =$	¹ 2	² 4	³ 4	⁴ 5	⁵ 6	⁶ 3	
	2 > 7? ✓						
$A =$	¹ 1	² 2	³ 4	⁴ 5	⁵ 6	⁶ 3	
	6 > 3? ✓						
$A =$	¹ 1	² 2	³ 4	⁴ 5	⁵ 6	⁶ 6	
	5 > 3? ✓						
$A =$	¹ 1	² 2	³ 4	⁴ 5	⁵ 5	⁶ 6	
	4 > 3? ✓						
$A =$	¹ 1	² 2	³ 4	⁴ 4	⁵ 5	⁶ 6	
	2 > 3? ✗						
$A =$	¹ 1	² 2	³ 3	⁴ 4	⁵ 5	⁶ 6	