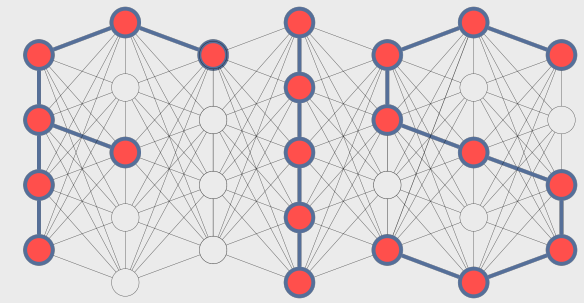


1222 • 2022  
800  
ANNI



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA



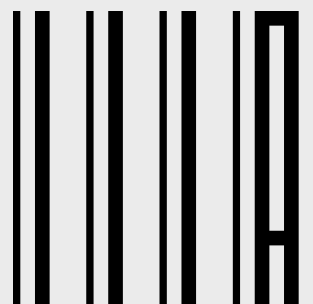
# Insertion Sort

Gianmaria Silvello

Department of Information Engineering  
University of Padua

[gianmaria.silvello@unipd.it](mailto:gianmaria.silvello@unipd.it)

<http://www.dei.unipd.it/~silvello/>



# Insertion Sort

---

Index      1, 2, 3, 4, 5, 6

A = <5, 2, 4, 6, 1, 3>

A = <5, **2**, 4, 6, 1, 3>

A = <2, 5, **4**, 6, 1, 3>

A = <2, 4, 5, **6**, 1, 3>

A = <2, 4, 5, 6, **1**, 3>

A = <1, 2, 4, 5, 6, **3**>

A = <1, 2, 3, 4, 5, 6>

Note that:

1. The leftmost element w.r.t. the current index is always ordered.
2. When we move an element from  $j$  to  $i < j$  we have to make room for it by shifting all the elements from  $i$  to  $j-1$

# Pseudo-code

---

```
INSERTION-SORT(A)
  for j = 2 to length(A) do
    key = A[j]
    // insert A[j] into the
    // sorted sequence A[1..j-1]
    i = j - 1
    while i > 0 and A[i] > key do
      A[i+1] = A[i]
      i = i - 1
    A[i+1] = key
```

# Pseudo-code

---

INSERTION-SORT(A)

```
for j = 2 to length(A) do  
    key = A[j]  
    // insert A[j] into the  
    // sorted sequence A[1..j-1]  
    i = j - 1  
    while i > 0 and A[i] > key do  
        A[i+1] = A[i]  
        i = i - 1  
    A[i+1] = key
```

Index     1, 2, 3, 4, 5, 6  
A = <5, 2, 4, 6, 1, 3>

## First iteration

$j = 2, A[2] = 2, i = 1, \text{key} = 2$

$\text{while } i > 0 \text{ and } A[i] = 5 > 2 \rightarrow \text{true}$

$A[i+1] = 5 \rightarrow A = \langle 5, 5, 4, 6, 1, 3 \rangle$

$i = 1 - 1 = 0$  then we exit the loop

$A[i+1] \rightarrow A[1] = 2 \rightarrow A = \langle 2, 5, 4, 6, 1, 3 \rangle$

# From pseudo-code to Python

---

<https://tinyurl.com/vvylgxa>

```
INSERTION-SORT(A)
  for j = 2 to length(A) do
    key = A[j]
    i = j - 1
    while i > 0 and A[i] > key do
      A[i+1] = A[i]
      i = i - 1
    A[i+1] = key
```

```
def insertionSort(A):
    for j in range(1, len(A)):
        key = A[j]
        i = j
        while i > 0 and A[i-1] > key:
            A[i] = A[i-1]
            i = i - 1
        A[i] = key
```



# Insertion Sort: Pseudo-code and costs

---

- The running time of an algorithm depends on its input size
- Input size can be define by the *number of items in the input*

INSERTION-SORT( $A, n$ )

**for**  $j = 2$  **to**  $n$

$key = A[j]$

    // Insert  $A[j]$  into the sorted sequence  $A[1 \dots j - 1]$ .

$i = j - 1$

**while**  $i > 0$  and  $A[i] > key$

$A[i + 1] = A[i]$

$i = i - 1$

$A[i + 1] = key$

*cost*    *times*

$c_1$      $n$

$c_2$      $n - 1$

0     $n - 1$

$c_4$      $n - 1$

$c_5$      $\sum_{j=2}^n t_j$

$c_6$      $\sum_{j=2}^n (t_j - 1)$

$c_7$      $\sum_{j=2}^n (t_j - 1)$

$c_8$      $n - 1$

From CLRS 3rd ed.

# Running time analysis

---

- If we assume that all the constants are equal to 1 and  $c_3 = 0$
- Best case: the input is an ordered sequence
  - $T(n) = 5n - 4$
  - $T(n) = bn + c$
- **Worst case:** the input is a sequence in inverse order
  - $T(n) = a n^2 + b n + c$

# Best, Worst and Average case

