~ by Katsucurry 

# Algoritmi di ordinamento

## Selection Sort

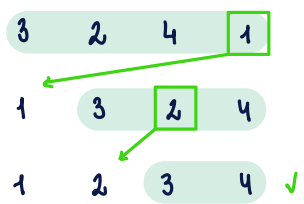
Come funziona?

I. Scorro tutto l'array per trovare l'elemento più piccolo.

II. Scambio l'elemento trovato con il primo elemento dell'array.

III. Ripeti l'operazione sull'array partendo dall'indice del primo elemento + 1 e facendo scorrere gli altri elementi verso dx.

**Esempio**




**Pseudocodice**

```
void selectionSort(v[])

    For j = 1 to length - 1  {

        pmin = j

        For i = j + 1 to length(v) {

            If v[i] < V[pmin]

                pmin = i

        }

        APP = V[j]

        V[j] = V[pmin]

        V[pmin] = APP

    }
```

**Tempo di esecuzione**

$T(n) = 5c(n-1) + \frac{1}{2}c\,\frac{n(n-1)}{2}$

<u>caso migliore</u> il vettore è già ordinato

$t_m(n) \approx 5cn + n^2 \approx n^2$

<u>caso peggiore</u> array ordinato al contrario

$T_p(n) \approx n^2$

# Insertion Sort

<u>Come funziona?</u>

I. Confronto i primi due numeri dell'array, scambiando la posizione in caso il 2° elemento è minore del primo.

II. Confronto il terzo e il secondo numero, scambiando eventualmente

III. In caso avvenga lo scambio, confrontare con il numero precedente. Ripetere questo passaggio fino a quando non arrivi alla prima coppia.

IV. Riprendere il confronto dall'ultimo elemento "scoperto".

**esempio**

72 > 29 [27 15] → sono ancora coperti

29 72 > 27 [15]

29 > 27 72 [15]

27 29 72 > 15

27 29 > 15 72

27 > 15 29 72

15 27 29 72 ✓

## Pseudocodice

```
void InsSort(A[])
  For i = 2 to length(A) {
      APP = A[i]
      j = i-1
      while APP < A[i] AND j > 0 {
          A[j+1] = A[j]
          j++
      }
      A[j+1] = APP
  }
```

## Tempo di esecuzione

$$T_{IS}(n) = 4c(n-1) + 3c\left(\sum_{i=2}^{n} t_{wi}\right)$$

$t_{wi}$ ↑ numero di volte in cui la condizione del while si verifica

<u>caso migliore</u> : array è già ordinato

$$t_m(n) = 4c(n-1) \approx 4c(n) \approx n$$

<u>caso peggiore</u> : array è ordinato al contrario e non scombinato

$$T_p(n) = 4c(n-1) + 3c\frac{(n-1)n}{2} \approx 4c(n) + 3c\frac{n^2}{2} \approx n^2$$

$$\sum_{i=1}^{n-1} i = \frac{(n-1)n}{2}$$

## SS vs IS

SS → indipendentemente dall'input, richiede sempre $n^2$

IS → nel caso migliore, può avere un tempo $n$