Insertion sort

Pseudocodice

```
for i=2 to lenght[A] do
        key = A[i]
        # Inserisce A[i] nella sequenza ordinata A[1 ... i-1]
        j = j-1
        while j>0 and A[j]>key do
                        A[j+1] = A[j]
                        j = j - 1
        A[j+1] = key
```

Analisi computazionale

Nella soluzione non teniamo traccia del costo degli assegnamenti ma solo del numero di **letture/scritture** in memoria e di **operazioni** effettuate (confronti, somme, decrementi, . . .).

Contiamo il numero di *swap* e di *confronti/incrementi* che vengono eseguiti nel codice tramite due variabili globali: `ct_op` e `ct_swap`, inizializzate a `0`.

Caso peggiore $O(n^2)$

L'array A è ordinato in senso decrescente.
`ct_op` e `ct_swap` a meno di una costante sono proporzionali a n^2 , infatti vengono eseguite

$$\sum_{i=0}^{n-1} i = \sum_{i=1}^{n-1} i = \frac{n \cdot (n-1)}{2} = O(n^2)$$

chiamate a `swap()` e

$$\sum_{i=0}^{n-1} (2+7i) = \sum_{i=0}^{n-1} 2 + \sum_{i=1}^{n-1} 7i = 2n + \frac{7n \cdot (n-1)}{2} = \frac{n \cdot (7n-3)}{2} = O(n^2)$$

operazioni aritmetico-logiche.

Caso migliore $O(n)$

L'array A è già ordinato in senso crescente.
`ct_op` e `ct_swap` a meno di una costante sono proporzionali a n (stesso ragionamento del precedente paragrafo).