Insertion Sort

```
INSERTIONSORT(A)
n := A.lenght()
for j = 1 to n - 1
   x := A[j]
   i := j - 1
   while i ≥ 0 and A[i] > x
      A[i + 1] := A[i]
      i := i - 1
   A[i + x] = x
```

Caso peggiore se la sequenza è già ordinata in senso decrescente, $\in O(n^2)$.
L'ordinamento è **In Place**, non abbiamo bisogno di spazio extra e ricorsione ma pagheremo in termini di tempo.

Quicksort

Usa la tecnica DEI.

Divide: scelgo un indice in A (t):

- 1a parte di A $\longrightarrow$ tutti i valori più piccoli di A[t]
- 2a parte di A $\longrightarrow$ valori più grandi di A[t]

Impera: Ordina le due parti con la stessa tecnica usata prima

Combina: \

```
QUICKSORT(A, P, R)
   if P < R
      then
         q :=  PARTITION(A, P, R)
         QUICKSORT(A, P, q - 1)
         QUICKSORT(A, q + 1, R)
```

Partition

Il pivot sarà l'ultimo elemento della nostra sequenza

```
PARTITION(A, P, R)
   x := A[R]
   i := P - 1
   for j = P to R - 1
      if A[j] ≤ x
         then
            i := i + 1
            SCAMBIA A[i] con A[j]
   SCAMBIA A[R] con A[i + 1]
   return i + 1
```

ESEMPIO

Qui stiamo ordinando i numeri in-place, non ho mai bisogno di spazio extra, se non per lo scambio; nel Merge Sort invece avevo bisogno di uno spazio lineare in più.

Quicksort risulta essere in media molto efficace poichè difficilmente capita di avere partizioni troppo sbilanciate, in generale quindi funziona bene.

Pescando il pivot a caso capiterò probabilmente in un caso vantaggioso.