

Insertion Sort Ricorsivo. La soluzione può comporsi di due funzioni:

```
InsSort(A, j)      # ordina A[1..j]
  if j > 1
    InsSort(A,j-1)      # ordina A[1..j-1]
    Insert(A,j)         # inserisce A[j] in A[1,j-1] ordinato

Insert(A, j)        # inserisce A[j] in A[1,j-1] ordinato

  # se j=1 oppure A[j-1] <= A[j],
  # A[j] e' gia' nel posto giusto
  # altrimenti ...

  if (j > 1) and (A[j] < A[j-1])
    A[j] <-> A[j-1]
  Insert(A, j-1)
```

La correttezza di **InsSort**, ovvero il fatto **InsSort**(A, j) ordina il sottoarray A[1..j] si dimostra per induzione su j

- ($j = 1$) In questo caso A[1..j] contiene un solo elemento e quindi è già ordinato.
- ($j > 1$) In questo caso, per ipotesi induttiva, la chiamata **InsSort**(A, j-1) ordina A[1..j-1]. Quindi la preconditione per **Insert**(A, j) è soddisfatta, e, dato che **Insert** è corretto (vedi sotto), si conclude che dopo la chiamata di **Insert** l'intero sottoarray A[1..j] è ordinato, come desiderato.

Anche la correttezza di **Insert** si dimostra per induzione. Più precisamente dimostriamo che, se A[1..j-1] è ordinato, allora **Insert**(A, j) inserisce A[j] in A[j-1], ovvero restituisce A[1..j] ordinato.

- ($j = 1$) In questo caso A[1..j-1] è vuoto, quindi $A[j] = A[1]$ è già nel posto giusto, A[1..1] è ordinato, come desiderato.
- ($j > 1$) Distinguiamo due sottocasi
 - Se $A[j-1] \leq A[j]$, allora A[1..j] è già ordinato, ovvero, anche in questo caso A[j] è già al posto giusto.
 - Se $A[j-1] > A[j]$, ricordando che A[1..j-1] è ordinato e quindi A[j-1] massimo in A[1..j-1], si ha che A[j-1] è il massimo di A[1..j]. Lo scambio con A[j-1] lo porta quindi correttamente in posizione j. A questo punto la chiamata **Insert**(A, j-1), per ipotesi induttiva ordina anche A[1..j-1], per cui alla fine A[1..j] è ordinato.