

Alcune situazioni comuni

Complessita'	Esempio
$O(1)$	Leggere il primo elemento di un insieme di dati.
$O(\log n)$	Dividere un insieme di n elementi in 2 meta', poi dividere ciascuna delle 2 meta' di nuovo a meta', etc.
$O(n)$	"Attraversare" un insieme di n dati
$O(n \log n)$	Dividere un insieme di n dati a meta' ed "attraversare" ciascuna meta' ricorsivamente.
$O(n^2)$	"Attraversare" n volte un insieme di n dati
$O(n!)$	Generare tutte le possibili permutazioni di un insieme di dati

Merge Sort + Insertion Sort

```
Merge+Insertion-Sort(k, A, p, r)
If ((r-p) > k)
    then  q = ⌊(p+r)/2⌋
           Merge+Insertion-Sort(k, A, p,q)
           Merge+Insertion-Sort(k, A, q+1, r)
           Merge(A, p, q, r)
    else  Insertion-Sort(A,p,r)  →  Applica l' Insertion-Sort
                                   alla sottosequenza A[p...r]
```

Analisi dell'algoritmo

$$T(n) = \begin{cases} T_{IS}(n) = O(n^2) & n \leq k \\ 2 * T(n/2) + \Theta(n) & n > k \end{cases}$$

$$\text{Ipotizziamo } \begin{cases} T_{IS}(k) = Ak^2 \\ \Theta(n) = bn \\ n = k * 2^h \end{cases}$$

Siamo nella situazione precedente. Otteniamo:

$$T(n) = (n/k) T_{IS}(k) + (b n) \log_2 (n/k) = \Theta(n \log n)$$

Notare - Il comportamento asintotico non cambia.
Miglioriamo se $T_{IS}(k) < T_{ms}(k)$.

Merge Sort non ricorsivo

Ipotizziamo $n = \text{length}(A) = 2^h$ $h = \text{intero}$

Merge-Sort-NR(A)

salto $\leftarrow 2$

While (salto $\leq n$)

do { For $i \leftarrow 1$ step salto to n
do { $p = 1 + (i-1) * \text{salto}$
 $r = i * \text{salto}$
 $q = \lfloor (p+r)/2 \rfloor$
 Merge(A, p, q, r)
 salto $\leftarrow \text{salto} * 2$

Analisi computazionale

$$\begin{aligned} T(n) &= \sum_{j=1}^{\log_2(n)} \left[\frac{n}{2^j} T_{\text{merge}}(2^j) \right] = \\ &= \sum_{j=1}^{\log_2(n)} [T_{\text{merge}}(n)] = \log_2(n) [T_{\text{merge}}(n)] = \Theta(n \log(n)) \end{aligned}$$