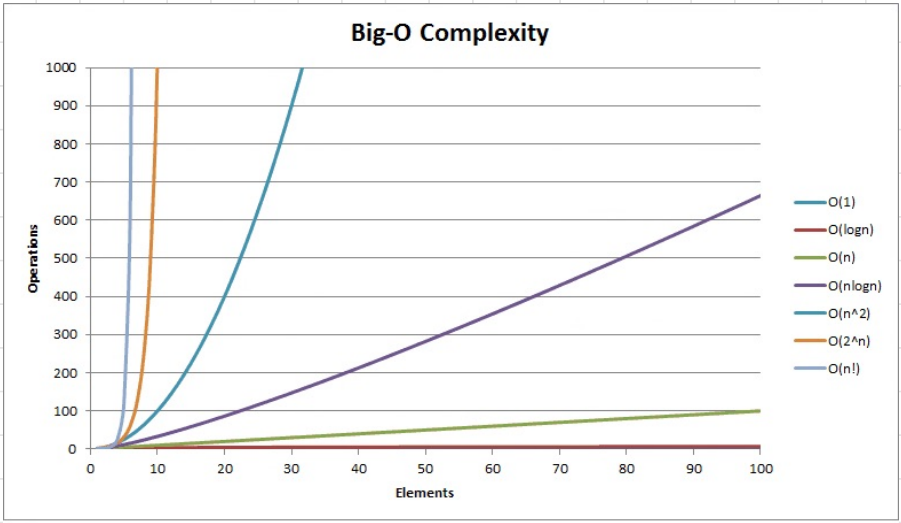
ANALISI COMPLESSITÀ

Analisi complessità

Complessità Utilizzo delle risorse
- tempo di esecuzione
- Spazio di memoria
- Banda di comunicazione

Correttezza fa quello per cui è progettato
- Dimostrazione formale
- Ispezione informale

Semplicità Facile da capire e mantenere
- Identificatori significativi
- algoritmo ben commentato
- Strutture dati adeguate
- rispetto degli standard



Caso migliore: Determiniamo un limite inferiore (Ω) del tempo di esecuzione per qualsiasi dimensione dell'input ($\forall n$)

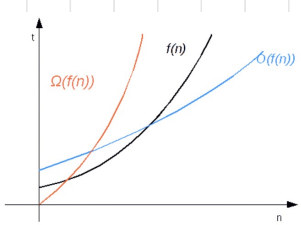
Caso medio: Determiniamo un limite sia superiore che inferiore (Θ) per qualsiasi dimensione dell'input ($\forall n$)

Caso peggiore: Determiniamo un limite superiore (O) del tempo di esecuzione per qualsiasi dimensione dell'input ($\forall n$)

- O: $T(n) = O(f(n)) \rightarrow \exists \quad c, n_0 > 0 \quad | \quad T(n) \leq c \cdot f(n) \forall n \geq n_0$
  - Il tasso di crescita della complessità è al più di $f(n)$.

- Ω: $T(n) = \Omega(f(n)) \rightarrow \exists \quad c, n_0 > 0 \quad | \quad T(n) \geq c \cdot f(n) \forall n \geq n_0$
  - Il tasso di crescita della complessità è al meno di $f(n)$.

- Θ: $T(n) = \Theta(f(n)) \rightarrow \exists \quad c_1, c_2, n_0 > 0 \quad | \quad c_1 \leq T(n) \leq c_2 \cdot f(n) \forall n \geq n_0$
  - É una delimitazione sia inferiore che superiore della complessità di un algoritmo.



Divide et Impera

Adatto da algoritmi ricorsivi, prevede 3 passi ad ogni livello di ricorsione

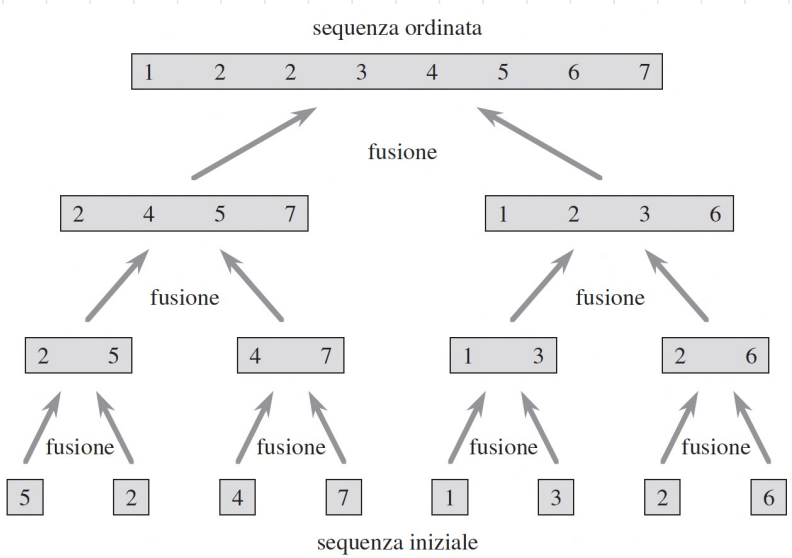
- **Divide**: il problema viene diviso in un certo numero di sottoproblemi, che sono istanze più piccole dello stesso problema
- **Impera**: i sottoproblemi vengono risolti in modo ricorsivo
- **Combine**: le soluzioni dei sottoproblemi vengono combinate per generare la soluzione del problema originale

```
MERGE(A, p, q, r)
1  n1 = q − p + 1
2  n2 = r − q
3  crea due nuovi array L[1..n1 + 1] e R[1..n2 + 1]
4  for i = 1 to n1
5       L[i] = A[p + i − 1]
6  for j = 1 to n2
7       R[j] = A[q + j]
8  L[n1 + 1] = ∞
9  R[n2 + 1] = ∞
10 i = 1
11 j = 1
12 for k = p to r
13      if L[i] ≤ R[j]
14           A[k] = L[i]
15           i = i + 1
16      else A[k] = R[j]
17           j = j + 1
```

```
MERGE-SORT(A, p, r)
1  if p < r
2       q = ⌊(p + r)/2⌋
3       MERGE-SORT(A, p, q)
4       MERGE-SORT(A, q + 1, r)
5       MERGE(A, p, q, r)
```

$3T(n) + 3\Theta n$

$\ln 3$



Equazione di ricorrenza

Esprime il tempo di esecuzione totale di un problema di dimensione n in funzione del tempo di esecuzione per input più piccoli

$$T(n) = \begin{cases} \Theta(1) & n \le c \\ D(n) + T(n_1) + \dots + T(n_a) + C(n) & \text{else} \end{cases}$$

- $\Theta(1)$: Costo costante su input piccoli
- $D(n)$: Costo divisione
- $T(n_1) + \dots + T(n_a)$: Costo chiamate ricorsive
- $C(n)$: Costo combinazione

Caso particolare

Se T è diviso in a sottoproblemi, tutti della stessa dimensione $\frac{n}{b}$ ($b > 1$, costante)

$$T(n) = \begin{cases} \Theta(1) & n \le c \\ aT\left(\frac{n}{b}\right) + D(n) + C(n) & \text{else} \end{cases}$$

- $\Theta(1)$: Costo costante su input piccoli
- $aT\left(\frac{n}{b}\right)$: Costo chiamate ricorsive
- $D(n)$: Costo divisione
- $C(n)$: Costo combinazione

Master theorem

Utile per risolvere le ricorrenze di forma $T(n) = aT(n/b) + f(n)$

- $a \ge 1 \wedge b > 1$ costanti
- $f(n)$ funzione asintoticamente positiva
- n/b rappresenta $\lfloor n/b \rfloor$ o $\lceil n/b \rceil$ $\Rightarrow$ $T(n)$ può essere asintoticamente limitata nei seguenti modi:
  1. se $f(n) = O(n^{\log_b a - \varepsilon})$ per qualche costante $\varepsilon > 0$ $\Rightarrow$ $T(n) = \Theta(n^{\log_b a})$
  2. se $f(n) = \Theta(n^{\log_b a})$ $\Rightarrow$ $T(n) = \Theta(n^{\log_b a} \lg n)$
  3. se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per qualche costante $\varepsilon > 0$ e se $af(n/b) \le cf(n)$ per qualche costante $c < 1$ e per ogni n sufficientemente grande $\Rightarrow$ $T(n) = \Theta(f(n))$

In linea generale si effettua il confronto tra $n^{\log_b a}$ e $f(n)$, il più grande asintoticamente determina il risultato

In ciascuno dei tre casi confrontiamo $f(n)$ con $n^{\log_b a}$ e determina le più grande

- Il master theorem non si applica alla ricorrenza $T(n) = 2T(n/2) + n \lg n$