

# Cheatsheet di Algoritmi e Strutture Dati

Giacomo Scampini

3 luglio 2025

# Complessità degli Algoritmi

## Notazioni Asintotiche

Descrivono il comportamento di una funzione al crescere dell'input  $n$ .

- **Big O (Limite Superiore):**  $f(n) = \mathcal{O}(g(n))$  se esistono costanti  $c > 0$  e  $n_0 \geq 0$  tali che  $0 \leq f(n) \leq c \cdot g(n)$  per ogni  $n \geq n_0$ . Rappresenta il caso peggiore.
- **Big Omega (Limite Inferiore):**  $f(n) = \Omega(g(n))$  se esistono costanti  $c > 0$  e  $n_0 \geq 0$  tali che  $0 \leq c \cdot g(n) \leq f(n)$  per ogni  $n \geq n_0$ . Rappresenta il caso migliore.
- **Big Theta (Limite Stretto):**  $f(n) = \Theta(g(n))$  se  $f(n) = \mathcal{O}(g(n))$  e  $f(n) = \Omega(g(n))$ . Indica che  $g(n)$  è una stima precisa per  $f(n)$ .

## Classi di Complessità Comuni

- $\mathcal{O}(1)$ : Costante (es. accesso a un elemento di un array)
- $\mathcal{O}(\log n)$ : Logaritmica (es. ricerca binaria)
- $\mathcal{O}(n)$ : Lineare (es. scansione di una lista)
- $\mathcal{O}(n \log n)$ : Lineare-logaritmica (es. merge sort, heapsort)
- $\mathcal{O}(n^2)$ : Quadratica (es. bubble sort, selection sort)
- $\mathcal{O}(2^n)$ : Esponenziale (es. problemi risolti con la forza bruta)
- $\mathcal{O}(n!)$ : Fattoriale (es. problema del commesso viaggiatore con forza bruta)

## Analisi di Algoritmi Ricorsivi

Per risolvere ricorrenze della forma  $T(n) = aT(n/b) + f(n)$ .

**Master Theorem:** Date le costanti  $a \geq 1$ ,  $b > 1$  e una funzione  $f(n)$ :

1. Se  $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$  per qualche  $\epsilon > 0$ , allora  $T(n) = \Theta(n^{\log_b a})$ .
2. Se  $f(n) = \Theta(n^{\log_b a})$ , allora  $T(n) = \Theta(n^{\log_b a} \log n)$ .
3. Se  $f(n) = \Omega(n^{\log_b a + \epsilon})$  per qualche  $\epsilon > 0$  e se  $af(n/b) \leq cf(n)$  per qualche  $c < 1$  e  $n$  sufficientemente grande, allora  $T(n) = \Theta(f(n))$ .

## Strutture Dati

### Array

Blocco di memoria contiguo.

- **Accesso (lettura/scrittura):**  $\mathcal{O}(1)$
- **Ricerca (lineare):**  $\mathcal{O}(n)$
- **Inserimento/Cancellazione (in coda):**  $\mathcal{O}(1)$  (ammortizzato se dinamico)
- **Inserimento/Cancellazione (in mezzo):**  $\mathcal{O}(n)$

### Lista Collegata (Linked List)

Serie di nodi, ognuno con un puntatore al successivo.

- **Accesso/Ricerca:**  $\mathcal{O}(n)$
- **Inserimento/Cancellazione (in testa):**  $\mathcal{O}(1)$
- **Inserimento/Cancellazione (in coda/mezzo):**  $\mathcal{O}(n)$  (se non si ha un puntatore diretto)

### Stack (LIFO)

Operazioni principali: 'push', 'pop'.

- **Tutte le operazioni:**  $\mathcal{O}(1)$  (se implementato con array o lista)

## Coda (FIFO)

Operazioni principali: 'enqueue', 'dequeue'.

- **Tutte le operazioni:**  $\mathcal{O}(1)$  (se implementato con lista doppiamente collegata)

## Tabella Hash (Hash Table)

Mappa chiavi a valori usando una funzione hash.

- **Accesso/Ricerca/Inserimento/Cancellazione (caso medio):**  $\mathcal{O}(1)$
- **Caso peggiore (collisioni):**  $\mathcal{O}(n)$

## Albero Binario di Ricerca (BST)

Albero in cui il sottoalbero sinistro di un nodo contiene solo valori minori del nodo, e il destro solo valori maggiori.

- **Accesso/Ricerca/Inserimento/Cancellazione (medio/bilanciato):**  $\mathcal{O}(\log n)$
- **Caso peggiore (sbilanciato):**  $\mathcal{O}(n)$

## Heap Binario

Albero completo usato per implementare code di priorità.

- **Trova Min/Max:**  $\mathcal{O}(1)$
- **Inserimento:**  $\mathcal{O}(\log n)$
- **Estrai Min/Max:**  $\mathcal{O}(\log n)$