# Algoritmi e Strutture Dati

**Foglio 1**
**21/02/2025**

**Esercizio 1.** Dimostrare le seguenti affermazioni usando le definizioni di $\Theta(\cdot)$ e $O(\cdot)$. In questo esercizio, così come nei seguenti, $\log$ denota il logaritmo in base $2$.

1. $(n^n + n2^n + 5^n)(n! + 5^n) = O(n^n n!)$
2. $(n! + 2^n)(n^3 + \log(n^2 + 1)) = O(n^3 n!)$
3. $n! \neq O(2^n)$
4. $\log n! = \Theta(n \log n)$

**Esercizio 2.** Dimostrare o confutare le seguenti affermazioni:

1. $n \log n = O(\log n!)$
2. $\log_{16} n = O(\log_4 n)$ e $\log_8 n = O(\log_{10} n)$
3. $\log(n^2 + 1) = O(\log n)$
4. Per risolvere un certo problema abbiamo a disposizione due algoritmi, $A_1$ e $A_2$. Se l'input ha dimensione $n$, $A_1$ impiega esattamente $n^2 2^n$ passi, mentre $A_2$ impiega esattamente $n!$ passi. Al crescere di $n$, $A_1$ impiega un numero minore di passi rispetto ad $A_2$.

**Esercizio 3.** Fornire un limite asintotico stretto per ognuna delle seguenti funzioni $f(n)$:

1. $f(n) = (n^2 + 1)^{10}$
2. $f(n) = 2n \log(n + 2)^2 + (n + 2)^2 \log \frac{n}{2}$
3. $f(n) = 2^{n+1} + 3^{n-1}$
4. $f(n)$ è la somma dei primi $n$ interi positivi

**Esercizio 4.** Quale delle seguenti sequenze di funzioni è tale che ogni funzione è $O$ della successiva?

1. $\log(\log n),\ n,\ \log n,\ n^n$
2. $\log n,\ 2^{2^n},\ n^n,\ n!$
3. $\log(\log n),\ \log n,\ n,\ 2^{2^n}$
4. nessuna delle precedenti

**Esercizio 5.** Siano $f(n)$ e $g(n)$ funzioni positive tali che $f(n) = O(g(n))$. Dimostrare che $g(n) = \Omega(f(n))$.

**Esercizio 6.** Sia $k$ un intero positivo. Dimostrare che un insieme di $n$ elementi ha $O(n^k)$ sottoinsiemi di cardinalità $k$.

**Esercizio 7.** Si consideri il seguente algoritmo, il cui input è un array $A[0..n-1]$ di interi di dimensione $n$. Quale problema risolve? Determinare un limite asintotico stretto per il tempo di esecuzione nel caso peggiore. Esiste un algoritmo più efficiente nel caso in cui l'array in input sia *ordinato*?

**Algorithm** $\textsc{UniqueElements}(A)$

**for** $i \leftarrow 0$ to $n-2$ **do**<br>
    **for** $j \leftarrow i+1$ to $n-1$ **do**<br>
        **if** $A[i] = A[j]$ **then return** false<br>
        **end if**<br>
    **end for**<br>
**end for**<br>
**return** true

**Esercizio 8.** Si consideri il seguente algoritmo, il cui input è un intero positivo $n$. Quale problema risolve? Determinare un limite asintotico stretto per il tempo di esecuzione nel caso peggiore.

**Algorithm** $\textsc{Binary}(n)$

$count \leftarrow 1$<br>
**while** $n > 1$ **do**<br>
    $count \leftarrow count + 1$<br>
    $n \leftarrow \lfloor n/2 \rfloor$<br>
**end while**<br>
**return** $count$