Lezione 3

Sommario della Lezione

Notazioni Asintotiche

Sommario della Lezione

Notazioni Asintotiche

- Introdurremo la notazione asintotica O per il confronto di funzioni

Sommario della Lezione

Notazioni Asintotiche

- Introdurremo la notazione asintotica O per il confronto di funzioni
- Vedremo il suo utilizzo nell'analisi di algoritmi

Sommario della Lezione

Notazioni Asintotiche

- Introdurremo la notazione asintotica O per il confronto di funzioni
- Vedremo il suo utilizzo nell'analisi di algoritmi
- Faremo tanti esercizi

Date funzioni $f : n \in \mathbb{N} \to f(n) \in \mathbb{R}_+$, $g : n \in \mathbb{N} \to g(n) \in \mathbb{R}_+$,

Date funzioni $f : n \in \mathbb{N} \to f(n) \in \mathbb{R}_+$, $g : n \in \mathbb{N} \to g(n) \in \mathbb{R}_+$, diremo che

$$f(n) = O(g(n))$$

se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che

$$f(n) \leq cg(n), \quad \text{per ogni } n \geq n_0.$$

Date funzioni $f : n \in \mathbb{N} \to f(n) \in \mathbb{R}_+$, $g : n \in \mathbb{N} \to g(n) \in \mathbb{R}_+$, diremo che

$$f(n) = O(g(n))$$

se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che

$$f(n) \leq cg(n), \quad \text{per ogni } n \geq n_0.$$

Informalmente, $f(n) = O(g(n))$ se la funzione $f(n)$ non cresce più velocemente della funzione $g(n)$.

Date funzioni $f : n \in \mathbb{N} \to f(n) \in \mathbb{R}_+$, $g : n \in \mathbb{N} \to g(n) \in \mathbb{R}_+$, diremo che

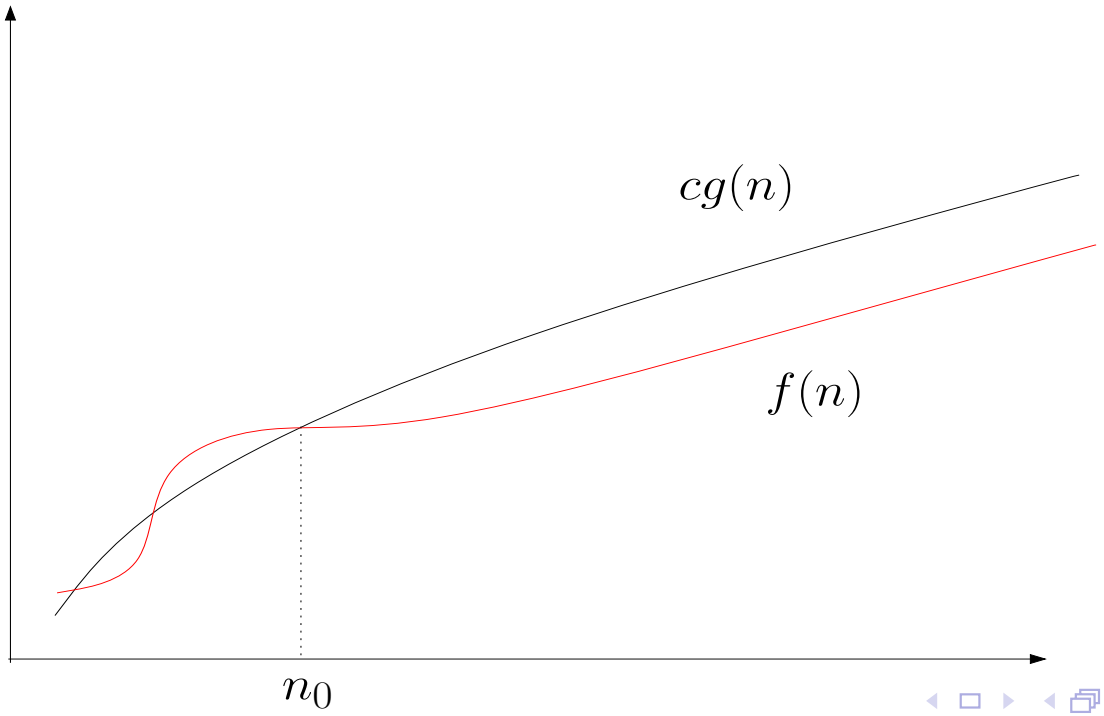
$$f(n) = O(g(n))$$

se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che

$$f(n) \leq cg(n), \quad \text{per ogni } n \geq n_0.$$

Informalmente, $f(n) = O(g(n))$ se la funzione $f(n)$ non cresce più velocemente della funzione $g(n)$.

Graficamente:



Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$3n^2 + 2n + 10 \leq cn^2, \quad \forall n \geq n_0 \tag{1}$$

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$3n^2 + 2n + 10 \leq cn^2, \quad \forall n \geq n_0 \tag{1}$$

Facciamolo:

$3n^2 + 2n + 10$

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$3n^2 + 2n + 10 \leq cn^2, \quad \forall n \geq n_0 \tag{1}$$

Facciamolo:

$$3n^2 + 2n + 10 \leq 3n^2 + 2n^2 + 10$$

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$3n^2 + 2n + 10 \leq cn^2, \quad \forall n \geq n_0 \tag{1}$$

Facciamolo:

$$3n^2 + 2n + 10 \leq 3n^2 + 2n^2 + 10 \leq 3n^2 + 2n^2 + n^2$$

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$3n^2 + 2n + 10 \leq cn^2, \quad \forall n \geq n_0 \tag{1}$$

Facciamolo:

$$3n^2 + 2n + 10 \leq 3n^2 + 2n^2 + 10 \leq 3n^2 + 2n^2 + n^2 = 6n^2 \ \forall n \geq 4$$

Esempio: proviamo che $3n^2 + 2n + 10 = O(n^2)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $3n^2 + 2n + 10 = O(n^2)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$3n^2 + 2n + 10 \leq cn^2, \quad \forall n \geq n_0 \tag{1}$$

Facciamolo:

$$3n^2 + 2n + 10 \leq 3n^2 + 2n^2 + 10 \leq 3n^2 + 2n^2 + n^2 = 6n^2 \ \forall n \geq 4$$

Quindi abbiamo trovato le costante $c = 6$ e $n_0 = 4$ per cui la (1) è vera.

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$2n^3 + 7n^2\sqrt{n} + 4 \leq cn^3, \quad \forall n \geq n_0 \tag{2}$$

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$2n^3 + 7n^2\sqrt{n} + 4 \leq cn^3, \quad \forall n \geq n_0 \tag{2}$$

Facciamolo:

$2n^3 + 7n^2\sqrt{n} + 4$

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$2n^3 + 7n^2\sqrt{n} + 4 \leq cn^3, \quad \forall n \geq n_0 \tag{2}$$

Facciamolo:

$2n^3 + 7n^2\sqrt{n} + 4 \leq 2n^3 + 7n^2 \cdot n + 4$

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$2n^3 + 7n^2\sqrt{n} + 4 \leq cn^3, \quad \forall n \geq n_0 \tag{2}$$

Facciamolo:

$$2n^3 + 7n^2\sqrt{n} + 4 \leq 2n^3 + 7n^2 \cdot n + 4 \leq 2n^3 + 7n^3 + n^3$$

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$2n^3 + 7n^2\sqrt{n} + 4 \leq cn^3, \quad \forall n \geq n_0 \tag{2}$$

Facciamolo:

$$2n^3 + 7n^2\sqrt{n} + 4 \leq 2n^3 + 7n^2 \cdot n + 4 \leq 2n^3 + 7n^3 + n^3 = 10n^3 \quad \forall n \geq 2$$

Esempio: proviamo che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$

Ricordiamo che $f(n) = O(g(n))$ se e solo se possiamo trovare delle costanti $c > 0$ e $n_0 \in \mathbb{N}$ tale che $f(n) \leq cg(n)$, per ogni $n \geq n_0$.

Come provare che $2n^3 + 7n^2\sqrt{n} + 4 = O(n^3)$?

Dalla definizione, occorre trovare costanti c e intero n_0 tale che

$$2n^3 + 7n^2\sqrt{n} + 4 \leq cn^3, \quad \forall n \geq n_0 \tag{2}$$

Facciamolo:

$$2n^3 + 7n^2\sqrt{n} + 4 \leq 2n^3 + 7n^2 \cdot n + 4 \leq 2n^3 + 7n^3 + n^3 = 10n^3 \quad \forall n \geq 2$$

Quindi abbiamo trovato le costante $c = 10$ e $n_0 = 2$ per cui la (2) è vera.

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Cosa vuol dire ciò *in pratica*?

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Cosa vuol dire ciò *in pratica*?
Vuol dire che la velocità di crescita della funzione $7n^{10} + 8n^5 + 5$ non è maggiore di quella della più semplice funzione n^{10},

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Cosa vuol dire ciò *in pratica*?
Vuol dire che la velocità di crescita della funzione $7n^{10} + 8n^5 + 5$ non è maggiore di quella della più semplice funzione n^{10}, infatti se consideriamo il rapporto tra le due funzioni $7n^{10} + 8n^5 + 5$ ed n^{10},

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Cosa vuol dire ciò *in pratica*?
Vuol dire che la velocità di crescita della funzione $7n^{10} + 8n^5 + 5$ non è maggiore di quella della più semplice funzione n^{10}, infatti se consideriamo il rapporto tra le due funzioni $7n^{10} + 8n^5 + 5$ ed n^{10}, otteniamo

$$\frac{7n^{10} + 8n^5 + 5}{n^{10}} = 7 + \frac{8}{n^5} + \frac{5}{n^{10}},$$

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Cosa vuol dire ciò *in pratica*?
Vuol dire che la velocità di crescita della funzione $7n^{10} + 8n^5 + 5$ non è maggiore di quella della più semplice funzione n^{10}, infatti se consideriamo il rapporto tra le due funzioni $7n^{10} + 8n^5 + 5$ ed n^{10}, otteniamo

$$\frac{7n^{10} + 8n^5 + 5}{n^{10}} = 7 + \frac{8}{n^5} + \frac{5}{n^{10}},$$

ed entrambi i termini $8/n^5$ e $5/n^{10}$ vanno a zero al crescere di n,

Gli esempi visti ammettono una semplice generalizzazione. Ovvero, possiamo provare che per ogni k costante vale che

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k). \tag{3}$$

Quindi, ad esempio, la (3) ci dice che $7n^{10} + 8n^5 + 5 = O(n^{10})$.

Cosa vuol dire ciò *in pratica*?
Vuol dire che la velocità di crescita della funzione $7n^{10} + 8n^5 + 5$ non è maggiore di quella della più semplice funzione n^{10}, infatti se consideriamo il rapporto tra le due funzioni $7n^{10} + 8n^5 + 5$ ed n^{10}, otteniamo

$$\frac{7n^{10} + 8n^5 + 5}{n^{10}} = 7 + \frac{8}{n^5} + \frac{5}{n^{10}},$$

ed entrambi i termini $8/n^5$ e $5/n^{10}$ vanno a zero al crescere di n, ovvero i termini $8n^5$ e 5 sono entrambi **trascurabili**, rispetto a n^{10}, quando n è molto grande.

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

$$a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 \leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0|$$

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

$$\begin{aligned} a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 &\leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\ &\leq |a_k| n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k \end{aligned}$$

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

$$\begin{aligned} a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 &\leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\ &\leq |a_k| n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k \\ &\leq (|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|) n^k \end{aligned}$$

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

$$\begin{aligned} a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 &\leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\ &\leq |a_k| n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k \\ &\leq (|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|) n^k \\ &= c n^k. \end{aligned}$$

ponendo $c = |a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|$.

Proviamo che $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 = O(n^k)$

$$\begin{aligned} a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0 &\leq |a_k| n^k + |a_{k-1}| n^{k-1} + \ldots + |a_1| n + |a_0| \\ &\leq |a_k| n^k + |a_{k-1}| n^k + \ldots + |a_1| n^k + |a_0| n^k \\ &\leq (|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|) n^k \\ &= c n^k. \end{aligned}$$

ponendo $c = |a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|$. Quindi, se vogliamo descrivere la velocità di crescita di un polinomio, basterà concentrarci solo sul suo termine di grado massimo.

Si inizia quindi a comprendere la utilità della notazione asintotica, che permette in maniera compatta di esprimere come una funzione cresce, al crescere del suo argomento.

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n,

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$,

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via.

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via. In totale, il numero di operazioni effettuate sarà quindi

$$n$$

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via. In totale, il numero di operazioni effettuate sarà quindi

$$n + (n - 1)$$

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via. In totale, il numero di operazioni effettuate sarà quindi

$$n + (n-1) + (n-2)$$

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
  y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via. In totale, il numero di operazioni effettuate sarà quindi

$$n + (n-1) + (n-2) + \ldots + 2 + 1$$

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1] \dots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via. In totale, il numero di operazioni effettuate sarà quindi

$$n + (n-1) + (n-2) + \ldots + 2 + 1 = \sum_{i=1}^{n} i =$$

Esercizio: analizziamo la complessità del seguente algoritmo, che prende in input una sequenza $a = a[0]a[1]\ldots a[n-1]$ di n numeri.

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
   y = y + i }
RETURN x + y
```

Quando $i = 0$, l'algoritmo esegue l'istruzione $x = x + a[j]$ all'interno del secondo `FOR` un numero di volte pari ad n, quando $i = 1$, l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 1$, quando $i = 2$ l'algoritmo esegue l'istruzione $x = x + a[j]$ un numero di volte pari ad $n - 2$ e così via. In totale, il numero di operazioni effettuate sarà quindi

$$n + (n-1) + (n-2) + \ldots + 2 + 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}. \qquad (4)$$

Proviamo che

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Proviamo che

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

É più facile calcolare $2\sum_{i=1}^{n} i$.

Proviamo che

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

É più facile calcolare $2\sum_{i=1}^{n} i$. Infatti, essa sarà

$$2\sum_{i=1}^{n} i = 1 + \quad 2 + \quad 3 + \ldots \quad + \quad n +$$

Proviamo che

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

É più facile calcolare $2\sum_{i=1}^{n} i$. Infatti, essa sarà

$$2\sum_{i=1}^{n} i = 1 + \quad 2 + \quad 3 + \ldots \quad + \quad n +$$
$$n + (n-1) + (n-2) + \ldots + 2 + 1$$

Proviamo che

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

É più facile calcolare $2\sum_{i=1}^{n} i$. Infatti, essa sarà

$$\begin{aligned} 2\sum_{i=1}^{n} i =& 1 + \quad 2 + \quad 3 + \ldots \quad + \quad n + \\ & n + (n-1) + (n-2) + \ldots + 2 + 1 \\ =& (n+1)n \end{aligned}$$

Proviamo che

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

É più facile calcolare $2\sum_{i=1}^{n} i$. Infatti, essa sarà

$$\begin{aligned} 2\sum_{i=1}^{n} i =& 1 + \quad 2 + \quad 3 + \ldots \quad + \quad n + \\ & n + (n-1) + (n-2) + \ldots + 2 + 1 \\ =& (n+1)n \end{aligned}$$

da cui

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Ritorniamo all'algoritmo

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
  y = y + i }
RETURN x + y
```

Ritorniamo all'algoritmo

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
  y = y + i }
RETURN x + y
```

Il numero di operazioni $y = y + i$ al di fuori del secondo `FOR` dell'algoritmo `Algoritmo`(a) è pari ad n,

Ritorniamo all'algoritmo

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
  y = y + i }
RETURN x + y
```

Il numero di operazioni $y = y + i$ al di fuori del secondo `FOR` dell'algoritmo `Algoritmo`(a) è pari ad n, per cui il numero totale $T(n)$ di operazioni elementari che l'algoritmo esegue è pari a $n(n+1)/2 + n$,

Ritorniamo all'algoritmo

```
Algoritmo(a)
x = 0; y = 0
FOR(i = 0; i < n; i = i + 1){
    FOR(j = i; j < n; j = j + 1){
        x = x + a[j] }
  y = y + i }
RETURN x + y
```

Il numero di operazioni $y = y + i$ al di fuori del secondo `FOR` dell'algoritmo `Algoritmo`(a) è pari ad n, per cui il numero totale $T(n)$ di operazioni elementari che l'algoritmo esegue è pari a $n(n+1)/2 + n$, per cui, applicando la definizione di O, possiamo dire che $T(n) = O(n^2)$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$,

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n + 1) \leq n + 1$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n + 1) \leq n + 1$. Avremo

$\log(n+1) \leq \log(n+n)$

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n + 1) \leq n + 1$. Avremo

$$\log(n+1) \leq \log(n+n) = \log(2n)$$

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n+1) \leq n+1$. Avremo

$$\log(n+1) \leq \log(n+n) = \log(2n) = \log 2 + \log n$$

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n + 1) \leq n + 1$. Avremo

$\log(n+1) \leq \log(n+n) = \log(2n) = \log 2 + \log n \leq 1 + n$ (dall'ipotesi induttiva)

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n+1) \leq n+1$. Avremo

$$\log(n+1) \leq \log(n+n) = \log(2n) = \log 2 + \log n \leq 1 + n \text{ (dall'ipotesi induttiva)}$$

Osserviamo inoltre che, poichè vale $\log n^k = k \log n$, per ogni costante k abbiamo anche che $\log n^k = k \log n \leq kn$

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n+1) \leq n+1$. Avremo

$$\log(n+1) \leq \log(n+n) = \log(2n) = \log 2 + \log n \leq 1 + n \text{ (dall'ipotesi induttiva)}$$

Osserviamo inoltre che, poichè vale $\log n^k = k \log n$, per ogni costante k abbiamo anche che $\log n^k = k \log n \leq kn$ e quindi $\log^k n = O(n)$.

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n+1) \leq n+1$. Avremo

$$\log(n+1) \leq \log(n+n) = \log(2n) = \log 2 + \log n \leq 1 + n \text{ (dall'ipotesi induttiva)}$$

Osserviamo inoltre che, poichè vale $\log n^k = k \log n$, per ogni costante k abbiamo anche che $\log n^k = k \log n \leq kn$ e quindi $\log^k n = O(n)$.
Come esempio, possiamo quindi osservare che

$$7n + 8 \log n \leq 7n + 8n = O(n)$$

Ricordiamo che $\forall n$ il valore $x = \log_2 n$ è tale che $2^x = n$.
Proviamo che $\log n = O(n)$, cioè proviamo che esiste $c > 0$, $n_0 \geq 0$ tale che $\log n \leq cn$, per ogni $n \geq n_0$.

Lo proveremo per induzione su n, con costanti $c = 1 = n_0$. Per $n = 1$ abbiamo che $\log 1 = 0 \leq 1$. Supposto vero $\log n \leq n$, proviamo che $\log(n+1) \leq n+1$. Avremo

$$\log(n+1) \leq \log(n+n) = \log(2n) = \log 2 + \log n \leq 1 + n \text{ (dall'ipotesi induttiva)}$$

Osserviamo inoltre che, poichè vale $\log n^k = k \log n$, per ogni costante k abbiamo anche che $\log n^k = k \log n \leq kn$ e quindi $\log^k n = O(n)$.
Come esempio, possiamo quindi osservare che

$$7n + 8 \log n \leq 7n + 8n = O(n) \qquad 7n + 8 \log n^3 \leq 7n + 24n = O(n)$$

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$,

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 = 2^2$,

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 = 2^2$, dopo la terza iterazione vale che $y = 2 \times 2 \times 2 = 2^3$,

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 = 2^2$, dopo la terza iterazione vale che $y = 2 \times 2 \times 2 = 2^3$, ..., e così via.

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 = 2^2$, dopo la terza iterazione vale che $y = 2 \times 2 \times 2 = 2^3$, ..., e così via.

Pertanto, la iterazione i-esima in cui terminiamo sarà quella per cui vale che $y = 2^i \leq n < 2^{i+1}$,

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 = 2^2$, dopo la terza iterazione vale che $y = 2 \times 2 \times 2 = 2^3$, ..., e così via.

Pertanto, la iterazione i-esima in cui terminiamo sarà quella per cui vale che $y = 2^i \leq n < 2^{i+1}$, ovvero $i \leq \log n$.

Analizziamo la complessità dell'algoritmo:

```
Algoritmo(n)
x = 0; y = 1
WHILE(y < n + 1){
    x = x + 1
    y = 2 × y
}
RETURN x
```

Osserviamo che dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 = 2^2$, dopo la terza iterazione vale che $y = 2 \times 2 \times 2 = 2^3$, ..., e così via.

Pertanto, la iterazione i-esima in cui terminiamo sarà quella per cui vale che $y = 2^i \leq n < 2^{i+1}$, ovvero $i \leq \log n$.

Poichè in ogni iterazione del ciclo `WHILE` eseguiamo un numero costante di operazioni, ne segue che la complessità $T(n)$ dell'algoritmo sarà $T(n) = O(\log n)$.

Abbiamo visto che $\log n \leq n$.

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Vale che $\log \sqrt{n} \leq \log n$,

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Vale che $\log \sqrt{n} \leq \log n$, da cui otteniamo che

$$(1/2) \log n = \log n^{1/2}$$

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Vale che $\log \sqrt{n} \leq \log n$, da cui otteniamo che

$$(1/2) \log n = \log n^{1/2} = \log \sqrt{n}$$

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Vale che $\log \sqrt{n} \leq \log n$, da cui otteniamo che

$$(1/2) \log n = \log n^{1/2} = \log \sqrt{n} \leq \sqrt{n}$$

(in quanto sappiamo che $\log x \leq x$),

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Vale che $\log \sqrt{n} \leq \log n$, da cui otteniamo che

$$(1/2) \log n = \log n^{1/2} = \log \sqrt{n} \leq \sqrt{n}$$

(in quanto sappiamo che $\log x \leq x$), ovvero $\log n \leq 2\sqrt{n}$,

Abbiamo visto che $\log n \leq n$. Cosa possiamo dire su $\log n$ e $\sqrt{n}$?

Vale che $\log \sqrt{n} \leq \log n$, da cui otteniamo che

$$(1/2) \log n = \log n^{1/2} = \log \sqrt{n} \leq \sqrt{n}$$

(in quanto sappiamo che $\log x \leq x$), ovvero $\log n \leq 2\sqrt{n}$, per cui $\log n = O(\sqrt{n})$.

Vale che

$$\log n = O(\sqrt{n}),\ \sqrt{n} = O(n),\ n^k = O(n^{k+1})$$

Vale che

$$\log n = O(\sqrt{n}),\ \sqrt{n} = O(n),\ n^k = O(n^{k+1})$$

e

$$n^k = O(2^n),\ 2^n = O(n!),\ n! = O(n^n).$$

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Data la **costante** k, scegliamo n maggiore o uguale al più piccolo valore (sia esso n_0) per cui vale che $n/(\log n) \geq k$.

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Data la **costante** k, scegliamo n maggiore o uguale al più piccolo valore (sia esso n_0) per cui vale che $n/(\log n) \geq k$.
Un tale valore esiste sicuramente in quanto il rapporto $n/(\log n)$ è crescente in n e quindi prima o poi supererà il valore costante k.

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Data la **costante** k, scegliamo n maggiore o uguale al più piccolo valore (sia esso n_0) per cui vale che $n/(\log n) \geq k$.
Un tale valore esiste sicuramente in quanto il rapporto $n/(\log n)$ è crescente in n e quindi prima o poi supererà il valore costante k.
Ne segue che, per ogni $n \geq n_0$ vale che

$$n^k \leq n^{\frac{n}{\log n}}$$

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Data la **costante** k, scegliamo n maggiore o uguale al più piccolo valore (sia esso n_0) per cui vale che $n/(\log n) \geq k$.
Un tale valore esiste sicuramente in quanto il rapporto $n/(\log n)$ è crescente in n e quindi prima o poi supererà il valore costante k.
Ne segue che, per ogni $n \geq n_0$ vale che

$$n^k \leq n^{\frac{n}{\log n}} = (2^{\log n})^{\frac{n}{\log n}}$$

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Data la **costante** k, scegliamo n maggiore o uguale al più piccolo valore (sia esso n_0) per cui vale che $n/(\log n) \geq k$.
Un tale valore esiste sicuramente in quanto il rapporto $n/(\log n)$ è crescente in n e quindi prima o poi supererà il valore costante k.
Ne segue che, per ogni $n \geq n_0$ vale che

$$n^k \leq n^{\frac{n}{\log n}} = (2^{\log n})^{\frac{n}{\log n}} = 2^n,$$

Proviamo che

$$n^k = O(2^n) \qquad \text{per ogni costante } k$$

Data la **costante** k, scegliamo n maggiore o uguale al più piccolo valore (sia esso n_0) per cui vale che $n/(\log n) \geq k$.
Un tale valore esiste sicuramente in quanto il rapporto $n/(\log n)$ è crescente in n e quindi prima o poi supererà il valore costante k.
Ne segue che, per ogni $n \geq n_0$ vale che

$$n^k \leq n^{\frac{n}{\log n}} = (2^{\log n})^{\frac{n}{\log n}} = 2^n,$$

il che prova che

$$n^k = O(2^n)$$

Per provare che $2^n = O(n!)$, osserviamo che

$$2^n = \underbrace{2 \times 2 \times \ldots \times 2}_{n \text{ volte}}$$

Per provare che $2^n = O(n!)$, osserviamo che

$$2^n = \underbrace{2 \times 2 \times \ldots \times 2}_{n \text{ volte}} \leq \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}}$$

Per provare che $2^n = O(n!)$, osserviamo che

$$2^n = \underbrace{2 \times 2 \times \ldots \times 2}_{n \text{ volte}} \leq \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}} = n!, \qquad \forall n > 3.$$

Per provare che $2^n = O(n!)$, osserviamo che

$$2^n = \underbrace{2 \times 2 \times \ldots \times 2}_{n \text{ volte}} \leq \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}} = n!, \qquad \forall n > 3.$$

Infine, per provare che $n! = O(n^n)$ osserviamo che

$$n! = \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}}$$

Per provare che $2^n = O(n!)$, osserviamo che

$$2^n = \underbrace{2 \times 2 \times \ldots \times 2}_{n \text{ volte}} \leq \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}} = n!, \qquad \forall n > 3.$$

Infine, per provare che $n! = O(n^n)$ osserviamo che

$$n! = \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}} \leq \underbrace{n \times n \times n \times \ldots \times n}_{n \text{ volte}}$$

Per provare che $2^n = O(n!)$, osserviamo che

$$2^n = \underbrace{2 \times 2 \times \ldots \times 2}_{n \text{ volte}} \leq \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}} = n!, \qquad \forall n > 3.$$

Infine, per provare che $n! = O(n^n)$ osserviamo che

$$n! = \underbrace{1 \times 2 \times 3 \times \ldots \times n}_{n \text{ volte}} \leq \underbrace{n \times n \times n \times \ldots \times n}_{n \text{ volte}} = n^n, \qquad \forall n \geq 1.$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \tag{5}$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \tag{5}$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \tag{5}$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$.

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$,

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$.

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \tag{5}$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \tag{6}$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n)$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n)$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \tag{5}$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \tag{6}$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k)$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k) \leq b\frac{1}{k} d n^k,$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k) \leq b\frac{1}{k} d n^k,$$

provando la (6) per $a = 1$, (ma $\forall k$), con $c = (bd)/k$.

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k) \leq b\frac{1}{k} d n^k,$$

provando la (6) per $a = 1$, (ma $\forall k$), con $c = (bd)/k$. Nel caso generale, usando la (6) con parametri $a = 1, b$ arbitrario, e k/a avremo

$$\log^a n^b = (\log n^b)^a$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k) \leq b\frac{1}{k} d n^k,$$

provando la (6) per $a = 1$, (ma $\forall k$), con $c = (bd)/k$. Nel caso generale, usando la (6) con parametri $a = 1, b$ arbitrario, e k/a avremo

$$\log^a n^b = (\log n^b)^a \leq (cn^{(k/a)})^a$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k) \leq b\frac{1}{k} d n^k,$$

provando la (6) per $a = 1$, (ma $\forall k$), con $c = (bd)/k$. Nel caso generale, usando la (6) con parametri $a = 1, b$ arbitrario, e k/a avremo

$$\log^a n^b = (\log n^b)^a \leq (cn^{(k/a)})^a = c^a n^k$$

Proviamo ora che

$$\forall \text{ costanti } a > 0, b > 0, k > 0 \text{ vale } \log^a n^b = O(n^k). \qquad (5)$$

Ad esempio, $\log^5 n^6 = O(\sqrt[3]{n})$, infatti basta infatti porre $a = 5, b = 6, k = 1/3$ nella (5).
Altro esempio: $\log^2 n^{10} = O(\sqrt[6]{n})$, basta porre $a = 2, b = 10, k = 1/6$ nella (5).
Occorre provare che per ogni $a, b, k > 0$

$$\exists c, n_0 \ : \ (\log n^b)^a \leq cn^k, \ \forall n \geq n_0 \qquad (6)$$

Proviamolo innanzitutto nel caso $a = 1$. Ricordiamo che $\log x^y = y \log x$, e che $\log x \leq dx$. Pertanto

$$(\log n^b) = (b \log n) = (b\frac{1}{k} \cdot k \log n) = (b\frac{1}{k} \cdot \log n^k) \leq b\frac{1}{k} d n^k,$$

provando la (6) per $a = 1$, (ma $\forall k$), con $c = (bd)/k$. Nel caso generale, usando la (6) con parametri $a = 1, b$ arbitrario, e k/a avremo

$$\log^a n^b = (\log n^b)^a \leq (cn^{(k/a)})^a = c^a n^k \quad \text{da cui } \log^a n^b = O(n^k)$$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log \log n)$	log log

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log \log n)$	log log
$O(\log n)$	logaritmico

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log \log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$
- $\sqrt[2]{n} \log n = O(n)$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$
- $\sqrt[2]{n} \log n = O(n)$
  $\Longrightarrow \sqrt[2]{n} = O(n/\log n)$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$
- $\sqrt[2]{n} \log n = O(n)$
  $\Longrightarrow \sqrt[2]{n} = O(n/\log n)$
- $n^{10} = O(2^n)$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$
- $\sqrt[2]{n} \log n = O(n)$
  $\Longrightarrow \sqrt[2]{n} = O(n/\log n)$
- $n^{10} = O(2^n)$
- $n^{10} = O(2^n/n^7)$

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$
- $\sqrt[2]{n} \log n = O(n)$
  $\Longrightarrow \sqrt[2]{n} = O(n/\log n)$
- $n^{10} = O(2^n)$
- $n^{10} = O(2^n/n^7)$ (in quanto equivale a dire che $n^{10} \cdot n^7 = n^{17} = O(2^n)$)

La seguente tabella è molto utile per risolvere esercizi. Essa và interpretata nel modo seguente: leggendo dall'alto in basso, una funzione $f(n)$ in "alto" è O di una qualsiasi funzione $g(n)$ più in "basso".

Espressione O	nome
$O(1)$	costante
$O(\log\log n)$	log log
$O(\log n)$	logaritmico
$O(\sqrt[c]{n}),\ c > 1$	sublineare
$O(n)$	lineare
$O(n \log n)$	$n \log n$
$O(n^2)$	quadratico
$O(n^3)$	cubico
$O(n^k)\ (k \geq 1)$	polinomiale
$O(a^n)\ (a > 1)$	esponenziale
$O(n!)$	fattoriale

Per cui, ad esempio

- $3 \log^3 n^2 = O(n)$
- $7 \log^2 n^3 = O(\sqrt[3]{n})$
- $6n \log^5 n^{10} = O(n^{5/4})$
- $\sqrt[2]{n} = O(n)$
- $\sqrt[2]{n} \log n = O(n)$
  $\Longrightarrow \sqrt[2]{n} = O(n/\log n)$
- $n^{10} = O(2^n)$
- $n^{10} = O(2^n/n^7)$ (in quanto equivale a dire che $n^{10} \cdot n^7 = n^{17} = O(2^n)$)
- $7^n = O(n!)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)) \Rightarrow ad(n) = O(f(n)),\ \forall \text{ costante } a > 0$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)) \Rightarrow ad(n) = O(f(n)),\ \forall \text{ costante } a > 0$$

Ad esempio: $\log n = O(n)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)) \Rightarrow ad(n) = O(f(n)),\ \forall \text{ costante } a > 0$$

Ad esempio: $\log n = O(n) \Rightarrow 7 \log n = O(n)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n) + e(n) = O(f(n) + g(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n) + e(n) = O(f(n) + g(n))$$

Ad esempio: $\log n = O(n), \sqrt{n} = O(n)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n) + e(n) = O(f(n) + g(n))$$

Ad esempio: $\log n = O(n), \sqrt{n} = O(n) \Rightarrow \log n + \sqrt{n} = O(n)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n)e(n) = O(f(n)g(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n)e(n) = O(f(n)g(n))$$

Ad esempio: $\log n = O(\sqrt{n}), \sqrt{n} = O(\sqrt{n})$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n)e(n) = O(f(n)g(n))$$

Ad esempio: $\log n = O(\sqrt{n}), \sqrt{n} = O(\sqrt{n}) \Rightarrow \sqrt{n}\log n = O(n)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), f(n) = O(g(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), f(n) = O(g(n)) \Rightarrow d(n) = O(g(n))$$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), f(n) = O(g(n)) \Rightarrow d(n) = O(g(n))$$

Ad esempio: $\log n = O(\sqrt{n}), \sqrt{n} = O(n)$

Utili regole per il confronto tra funzioni

$$d(n) = O(f(n)), f(n) = O(g(n)) \Rightarrow d(n) = O(g(n))$$

Ad esempio: $\log n = O(\sqrt{n}), \sqrt{n} = O(n) \Rightarrow \log n = O(n)$

Utili regole per il confronto tra funzioni

$$f(n) = a_d n^d + \cdots a_1 n + a_0$$

Utili regole per il confronto tra funzioni

$$f(n) = a_d n^d + \cdots a_1 n + a_0 \Rightarrow f(n) = O(n^d)$$

Utili regole per il confronto tra funzioni

$$f(n) = a_d n^d + \cdots a_1 n + a_0 \Rightarrow f(n) = O(n^d)$$

Ad esempio: $5n^7 + 6n^4 + 3n^3 + 100 = O(n^7)$

Utili regole per il confronto tra funzioni

$$n^x = O(a^n),\ \forall \text{ costanti } x > 0, a > 1$$

Utili regole per il confronto tra funzioni

$$n^x = O(a^n),\ \forall \text{ costanti } x > 0, a > 1$$

Ad esempio: $n^{100} = O(2^n)$