Lezione 4

Sommario della Lezione

Notazioni Asintotiche 2

Sommario della Lezione

Notazioni Asintotiche 2

- Introdurremo le notazioni asintotiche Ω e Θ per il confronto di funzioni

Sommario della Lezione

Notazioni Asintotiche 2

- Introdurremo le notazioni asintotiche Ω e Θ per il confronto di funzioni
- Vedremo il loro utilizzo nell'analisi di algoritmi

Sommario della Lezione

Notazioni Asintotiche 2

- Introdurremo le notazioni asintotiche Ω e Θ per il confronto di funzioni
- Vedremo il loro utilizzo nell'analisi di algoritmi
- Faremo tanti esercizi

Ricordiamo la notazione O

Ricordiamo la notazione O

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

Ricordiamo la notazione O

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

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

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

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

Ricordiamo la notazione O

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

Ricordiamo la notazione O

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

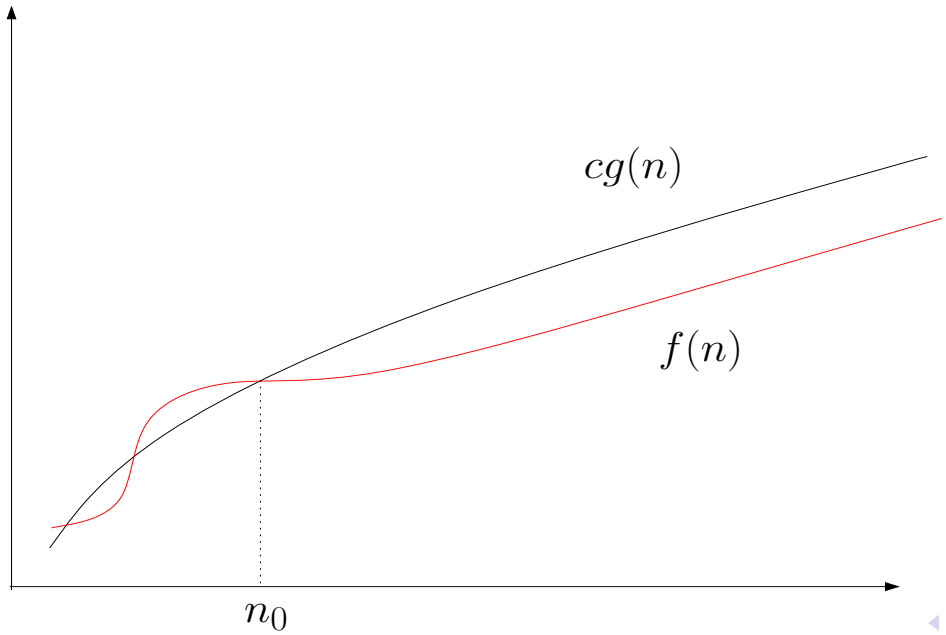
$$f(n) = O(g(n))$$

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:



Notazione Ω

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

$$f(n) = \Omega(g(n))$$

Notazione Ω

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

$$f(n) = \Omega(g(n))$$

se e solo se

$$\exists c > 0, n_0 \in \mathbb{N} \text{ tali che } f(n) \geq cg(n), \text{ per ogni } n \geq n_0$$

Notazione Ω

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

$$f(n) = \Omega(g(n))$$

se e solo se

$$\exists c > 0, n_0 \in \mathbb{N} \text{ tali che } f(n) \geq cg(n), \text{ per ogni } n \geq n_0$$

Informalmente, $f(n) = \Omega(g(n))$ se la funzione $f(n)$ cresce almeno tanto velocemente quanto cresce la funzione $g(n)$.

Notazione Ω

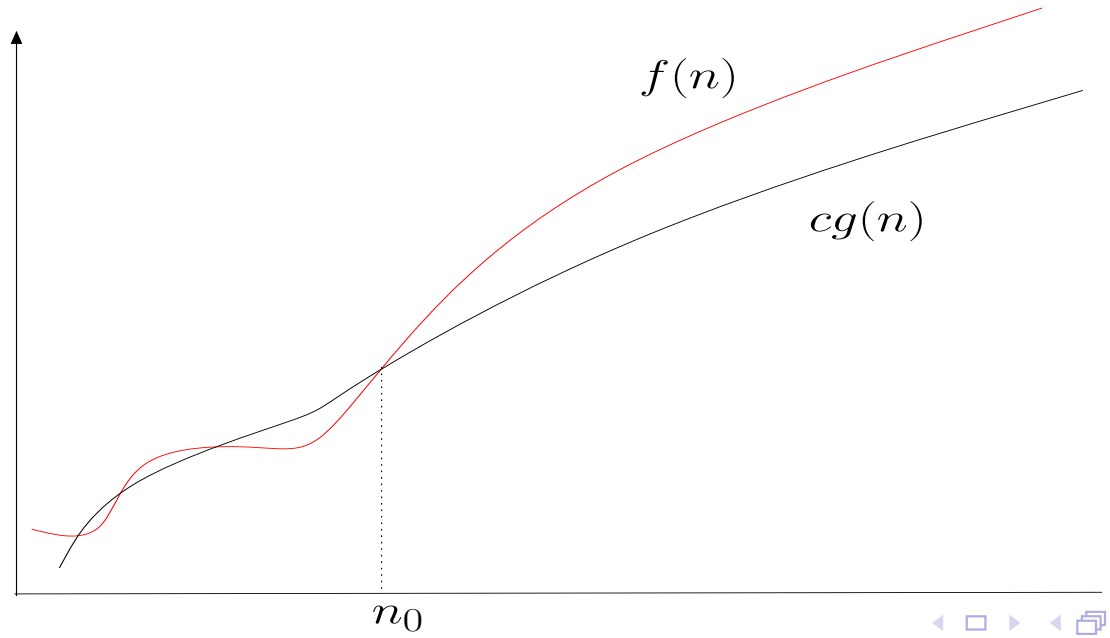
Date funzioni $f : n \in \mathbb{N} \to f(n) \in \mathbb{R}_+$, $g : n \in \mathbb{N} \to g(n) \in \mathbb{R}_+$, diremo che

$$f(n) = \Omega(g(n))$$

se e solo se

$$\exists c > 0, n_0 \in \mathbb{N} \text{ tali che } f(n) \geq cg(n), \text{ per ogni } n \geq n_0$$

Informalmente, $f(n) = \Omega(g(n))$ se la funzione $f(n)$ cresce almeno tanto velocemente quanto cresce la funzione $g(n)$.
Graficamente



Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$n^2 - 2n \geq cn^2$

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$n^2 - 2n \geq cn^2 \Leftrightarrow n^2 - cn^2 \geq 2n$

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$$n^2 - 2n \geq cn^2 \Leftrightarrow n^2 - cn^2 \geq 2n \Leftrightarrow n^2(1-c) \geq 2n$$

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$$n^2 - 2n \geq cn^2 \Leftrightarrow n^2 - cn^2 \geq 2n \Leftrightarrow n^2(1-c) \geq 2n \Leftrightarrow n(1-c) \geq 2$$

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$$n^2 - 2n \geq cn^2 \Leftrightarrow n^2 - cn^2 \geq 2n \Leftrightarrow n^2(1-c) \geq 2n \Leftrightarrow n(1-c) \geq 2 \Leftrightarrow n \geq 2/(1-c),$$

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$$n^2 - 2n \geq cn^2 \Leftrightarrow n^2 - cn^2 \geq 2n \Leftrightarrow n^2(1-c) \geq 2n \Leftrightarrow n(1-c) \geq 2 \Leftrightarrow n \geq 2/(1-c),$$

per cui basterà scegliere c come un qualsiasi valore < 1.

Esempio

Sia $f(n) = n^2 - 2n$, $g(n) = n^2$ e vogliamo provare che

$$n^2 - 2n = \Omega(n^2)$$

Occorre provare che esiste una costante $c > 0$ ed un numero intero n_0 tale che $n^2 - 2n \geq cn^2$, per ogni $n \geq n_0$.
Osserviamo che

$$n^2 - 2n \geq cn^2 \Leftrightarrow n^2 - cn^2 \geq 2n \Leftrightarrow n^2(1-c) \geq 2n \Leftrightarrow n(1-c) \geq 2 \Leftrightarrow n \geq 2/(1-c),$$

per cui basterà scegliere c come un qualsiasi valore < 1. Ad esempio, potremmo scegliere $c = 1/2$ ed avremmo quindi che $n^2 - 2n \geq cn^2$ per ogni valore di $n \geq 4$.

Osserviamo ora che valgono le seguenti relazioni

$$f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 : f(n) \geq cg(n),\ \forall n \geq n_0$$

Osserviamo ora che valgono le seguenti relazioni

$$f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 : f(n) \geq cg(n),\ \forall n \geq n_0$$
$$\Leftrightarrow \exists c, n_0 : g(n) \leq \frac{1}{c} f(n),\ \forall n \geq n_0$$

Osserviamo ora che valgono le seguenti relazioni

$$\begin{aligned} f(n) = \Omega(g(n)) &\Leftrightarrow \exists c, n_0 : f(n) \geq cg(n),\ \forall n \geq n_0 \\ &\Leftrightarrow \exists c, n_0 : g(n) \leq \frac{1}{c} f(n),\ \forall n \geq n_0 \\ &\Leftrightarrow g(n) = O(f(n)). \end{aligned}$$

Osserviamo ora che valgono le seguenti relazioni

$$\begin{aligned} f(n) = \Omega(g(n)) &\Leftrightarrow \exists c, n_0 : f(n) \geq cg(n),\ \forall n \geq n_0 \\ &\Leftrightarrow \exists c, n_0 : g(n) \leq \frac{1}{c} f(n),\ \forall n \geq n_0 \\ &\Leftrightarrow g(n) = O(f(n)). \end{aligned}$$

Pertanto, per provare che $f(n) = \Omega(g(n))$ basterà provare che

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

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

$$\sqrt{n} = \Omega(\log n)),$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

$$\sqrt{n} = \Omega(\log n)), \ n = \Omega(\sqrt{n}),$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

$$2^n = \Omega(n^k),$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

$$2^n = \Omega(n^k),\ n! = \Omega(2^n),$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

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

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

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

Esempio: per provare che $n^2 - \sqrt{n}\log n = \Omega(n^2)$, occorre provare che

$$\exists c, n_0 \ \text{ tale che } n^2 - \sqrt{n}\log n \geq cn^2, \forall n \geq n_0$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

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

Esempio: per provare che $n^2 - \sqrt{n}\log n = \Omega(n^2)$, occorre provare che

$$\exists c, n_0 \ \text{ tale che } n^2 - \sqrt{n}\log n \geq cn^2, \forall n \geq n_0$$

Osserviamo che

$$n^2 - \sqrt{n}\log n$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

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

Esempio: per provare che $n^2 - \sqrt{n}\log n = \Omega(n^2)$, occorre provare che

$$\exists c, n_0 \ \text{ tale che } n^2 - \sqrt{n}\log n \geq cn^2, \forall n \geq n_0$$

Osserviamo che

$$n^2 - \sqrt{n}\log n \geq n^2 - \sqrt{n}\sqrt{n}$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

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

Esempio: per provare che $n^2 - \sqrt{n}\log n = \Omega(n^2)$, occorre provare che

$$\exists c, n_0 \ \text{ tale che } n^2 - \sqrt{n}\log n \geq cn^2, \forall n \geq n_0$$

Osserviamo che

$$n^2 - \sqrt{n}\log n \geq n^2 - \sqrt{n}\sqrt{n} = n^2 - n \geq n^2 - \frac{1}{2}n^2$$

Sulla basedi quanto già provato per la notazione asintotica O, possiamo dire che

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

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

Esempio: per provare che $n^2 - \sqrt{n}\log n = \Omega(n^2)$, occorre provare che

$$\exists c, n_0 \ \text{ tale che } n^2 - \sqrt{n}\log n \geq cn^2, \forall n \geq n_0$$

Osserviamo che

$$n^2 - \sqrt{n}\log n \geq n^2 - \sqrt{n}\sqrt{n} = n^2 - n \geq n^2 - \frac{1}{2}n^2 = \frac{1}{2}n^2,\ \forall n \geq 2.$$

Notazione Θ

Useremo infine la notazione asintotica Θ se $f(n)$ e $g(n)$ crescono alla **stessa** velocità.

Notazione Θ

Useremo infine la notazione asintotica Θ se $f(n)$ e $g(n)$ crescono alla **stessa** velocità. Più precisamente

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \& f(n) = \Omega(g(n))$$

Notazione Θ

Useremo infine la notazione asintotica Θ se $f(n)$ e $g(n)$ crescono alla **stessa** velocità. Più precisamente

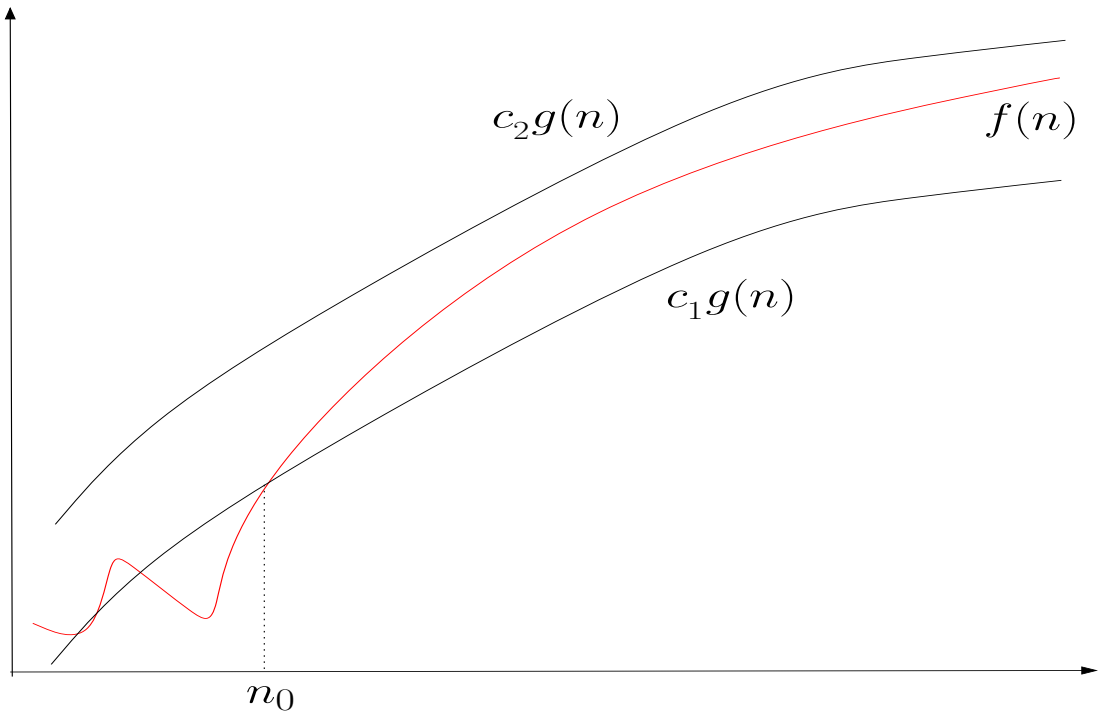
$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \& f(n) = \Omega(g(n))$$
$$\Leftrightarrow \exists c_1, c_2, n_0 : c_1 g(n) \leq f(n) \leq c_2 g(n),\ \forall n \geq n_0.$$

Notazione Θ

Useremo infine la notazione asintotica Θ se $f(n)$ e $g(n)$ crescono alla **stessa** velocità. Più precisamente

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \& f(n) = \Omega(g(n))$$
$$\Leftrightarrow \exists c_1, c_2, n_0 : c_1 g(n) \leq f(n) \leq c_2 g(n),\ \forall n \geq n_0.$$

Graficamente



Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Osserviamo che $4n^2 + \log n = \Omega(n^2)$ in quanto $4n^2 + \log n \geq n^2$.

Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Osserviamo che $4n^2 + \log n = \Omega(n^2)$ in quanto $4n^2 + \log n \geq n^2$.
Inoltre, $4n^2 + \log n = O(n^2)$ in quanto $4n^2 + \log n \leq 4n^2 + n \leq 5n^2$.

Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Osserviamo che $4n^2 + \log n = \Omega(n^2)$ in quanto $4n^2 + \log n \geq n^2$.
Inoltre, $4n^2 + \log n = O(n^2)$ in quanto $4n^2 + \log n \leq 4n^2 + n \leq 5n^2$.

Utilizzeremo la notazione asintotica Θ quando saremo in grado di dare una valutazione **precisa** della velocità di crescita di una funzione.

Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Osserviamo che $4n^2 + \log n = \Omega(n^2)$ in quanto $4n^2 + \log n \geq n^2$.
Inoltre, $4n^2 + \log n = O(n^2)$ in quanto $4n^2 + \log n \leq 4n^2 + n \leq 5n^2$.

Utilizzeremo la notazione asintotica Θ quando saremo in grado di dare una valutazione **precisa** della velocità di crescita di una funzione.

Quindi, è corretto dire che $n^2 + n = O(n^2)$,

Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Osserviamo che $4n^2 + \log n = \Omega(n^2)$ in quanto $4n^2 + \log n \geq n^2$.
Inoltre, $4n^2 + \log n = O(n^2)$ in quanto $4n^2 + \log n \leq 4n^2 + n \leq 5n^2$.

Utilizzeremo la notazione asintotica Θ quando saremo in grado di dare una valutazione **precisa** della velocità di crescita di una funzione.

Quindi, è corretto dire che $n^2 + n = O(n^2)$, ma sarà preferibile dire (in quanto è una affermazione più precisa) che $n^2 + n = \Theta(n^2)$,

Esempio

Proviamo che che

$$4n^2 + \log n = \Theta(n^2)$$

Osserviamo che $4n^2 + \log n = \Omega(n^2)$ in quanto $4n^2 + \log n \geq n^2$.
Inoltre, $4n^2 + \log n = O(n^2)$ in quanto $4n^2 + \log n \leq 4n^2 + n \leq 5n^2$.

Utilizzeremo la notazione asintotica Θ quando saremo in grado di dare una valutazione **precisa** della velocità di crescita di una funzione.

Quindi, è corretto dire che $n^2 + n = O(n^2)$, ma sarà preferibile dire (in quanto è una affermazione più precisa) che $n^2 + n = \Theta(n^2)$, dato che vale **sia** $n^2 + n = O(n^2)$ **che** $n^2 + n = \Omega(n^2)$.

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$ `Vero`

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$ `Vero`
2. $g(n) = \Theta(5n^6)$

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$ `Vero`
2. $g(n) = \Theta(5n^6)$ `Falso`

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$ `Vero`
2. $g(n) = \Theta(5n^6)$ `Falso`
3. $g(n) = O(n^{10})$

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$ `Vero`
2. $g(n) = \Theta(5n^6)$ `Falso`
3. $g(n) = O(n^{10})$ `Vero`

Esercizio

Sia $g(n) = n + 2n^3 - 3n^4 + 4n^5$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = \Omega(n \log n)$ `Vero`
2. $g(n) = \Theta(5n^6)$ `Falso`
3. $g(n) = O(n^{10})$ `Vero`
4. $g(n) = \Omega(n^5)$ `Vero`

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`
2. $g(n) = O(n^3)$

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`
2. $g(n) = O(n^3)$ `Vero`

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`
2. $g(n) = O(n^3)$ `Vero`
3. $g(n) = O(n^2)$

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`
2. $g(n) = O(n^3)$ `Vero`
3. $g(n) = O(n^2)$ `Falso`

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`
2. $g(n) = O(n^3)$ `Vero`
3. $g(n) = O(n^2)$ `Falso`
4. $g(n) = O(n^4)$

Esercizio

Sia $g(n) = n \log n + 2n^3 - 3n^2$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $g(n) = O(n \log n)$ `Falso`
2. $g(n) = O(n^3)$ `Vero`
3. $g(n) = O(n^2)$ `Falso`
4. $g(n) = O(n^4)$ `Vero`

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`
2. $f(n) = O(3n^2 + n + 3)$

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`
2. $f(n) = O(3n^2 + n + 3)$ `Vero`

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`
2. $f(n) = O(3n^2 + n + 3)$ `Vero`
3. $f(n) = \Omega(5n^2 + n + 3)$

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`
2. $f(n) = O(3n^2 + n + 3)$ `Vero`
3. $f(n) = \Omega(5n^2 + n + 3)$ `Vero`

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`
2. $f(n) = O(3n^2 + n + 3)$ `Vero`
3. $f(n) = \Omega(5n^2 + n + 3)$ `Vero`
4. $f(n) = \Omega(n^2)$

Esercizio

Sia $f(n) = 4n^2 + n + 3$. Dire quali delle seguenti affermazioni sono vere e quali sono false.

1. $f(n) = O(n^2)$ `Vero`
2. $f(n) = O(3n^2 + n + 3)$ `Vero`
3. $f(n) = \Omega(5n^2 + n + 3)$ `Vero`
4. $f(n) = \Omega(n^2)$ `Vero`

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4)$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$7n^4 - 8n^3 + 5 = O(n^4)$ `Vero`

$7n^4 - 8n^3 + 5 = O(n^3)$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5)$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4)$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4) \qquad \texttt{Vero}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$7n^4 - 8n^3 + 5 = O(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = O(n^3)$	`Falso`
$7n^4 - 8n^3 + 5 = O(n^5)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^3)$	

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^3) \qquad \texttt{Vero}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$7n^4 - 8n^3 + 5 = O(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = O(n^3)$	`Falso`
$7n^4 - 8n^3 + 5 = O(n^5)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^3)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^5)$	

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^3) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^5) \qquad \texttt{Falso}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$7n^4 - 8n^3 + 5 = O(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = O(n^3)$	`Falso`
$7n^4 - 8n^3 + 5 = O(n^5)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^3)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^5)$	`Falso`
$7n^4 - 8n^3 + 5 = \Theta(n^4)$	

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^3) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^5) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = \Theta(n^4) \qquad \texttt{Vero}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$7n^4 - 8n^3 + 5 = O(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = O(n^3)$	`Falso`
$7n^4 - 8n^3 + 5 = O(n^5)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^3)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^5)$	`Falso`
$7n^4 - 8n^3 + 5 = \Theta(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Theta(n^3)$	

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^3) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^5) \qquad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = \Theta(n^4) \qquad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Theta(n^3) \qquad \texttt{Falso}$$

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$7n^4 - 8n^3 + 5 = O(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = O(n^3)$	`Falso`
$7n^4 - 8n^3 + 5 = O(n^5)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^3)$	`Vero`
$7n^4 - 8n^3 + 5 = \Omega(n^5)$	`Falso`
$7n^4 - 8n^3 + 5 = \Theta(n^4)$	`Vero`
$7n^4 - 8n^3 + 5 = \Theta(n^3)$	`Falso`
$7n^4 - 8n^3 + 5 = \Theta(n^5)$	

Esercizio

Per ciascuna delle seguenti affermazioni, dire se essa è vera o falsa.

$$7n^4 - 8n^3 + 5 = O(n^4) \quad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = O(n^3) \quad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = O(n^5) \quad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^4) \quad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^3) \quad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Omega(n^5) \quad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = \Theta(n^4) \quad \texttt{Vero}$$
$$7n^4 - 8n^3 + 5 = \Theta(n^3) \quad \texttt{Falso}$$
$$7n^4 - 8n^3 + 5 = \Theta(n^5) \quad \texttt{Falso}$$

Esempi di analisi di algoritmi

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

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

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

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

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

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

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

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

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

Dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 \times 2 = 2^2$, 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 in cui $y = 2^i$, con $2^i \leq n < 2^{i+1}$,

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

Dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 \times 2 = 2^2$, 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 in cui $y = 2^i$, con $2^i \leq n < 2^{i+1}$, ovvero $i = \lfloor log\ n \rfloor$.

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

Dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 \times 2 = 2^2$, 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 in cui $y = 2^i$, con $2^i \leq n < 2^{i+1}$, ovvero $i = \lfloor log\ n \rfloor$. Poichè in ogni iterazione del ciclo `WHILE` eseguiamo un numero costante di operazioni, ne segue che la complessità $T(n)$ dell'algoritmo sarà $T(n) = O(\log n)$

Esempi di analisi di algoritmi

Tempo logaritmico: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per il logaritmo della dimensione dell'input.

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

Dopo la prima iterazione del ciclo `WHILE` vale che $y = 2^1$, dopo la seconda iterazione vale che $y = 2 \times 2 = 2^2$, 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 in cui $y = 2^i$, con $2^i \leq n < 2^{i+1}$, ovvero $i = \lfloor log\ n \rfloor$. Poichè in ogni iterazione del ciclo `WHILE` eseguiamo un numero costante di operazioni, ne segue che la complessità $T(n)$ dell'algoritmo sarà $T(n) = O(\log n)$ (Più precisamente, $T(n) = \Theta(\log n)$).

Esempi di analisi di algoritmi

Tempo Lineare: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per la dimensione dell'input.

Esempi di analisi di algoritmi

Tempo Lineare: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per la dimensione dell'input.

Esempio: Calcolo del massimo di n numeri $a_1, a_2, \ldots, a_n$

Esempi di analisi di algoritmi

Tempo Lineare: Il tempo di esecuzione dell'algoritmo è al più un fattore costante per la dimensione dell'input.

Esempio: Calcolo del massimo di n numeri $a_1, a_2, \ldots, a_n$

```
max = a_1
FOR(i = 2; i < n + 1; i = i + 1){
    IF(a_i > max){
      max = a_i
     }
   }
RETURN max
```

Altro esempio di tempo lineare.

Merge: Trasforma due liste **ordinate** $A = a_1, \ldots, a_n$ e $B = b_1, \ldots, b_n$, in cui $a_1 \leq \ldots \leq a_n$ e $b_1 \leq \ldots \leq b_n$

Altro esempio di tempo lineare.

Merge: Trasforma due liste **ordinate** $A = a_1, \ldots, a_n$ e $B = b_1, \ldots, b_n$, in cui $a_1 \leq \ldots \leq a_n$ e $b_1 \leq \ldots \leq b_n$ in un'unica lista **ordinata** L.

Altro esempio di tempo lineare.

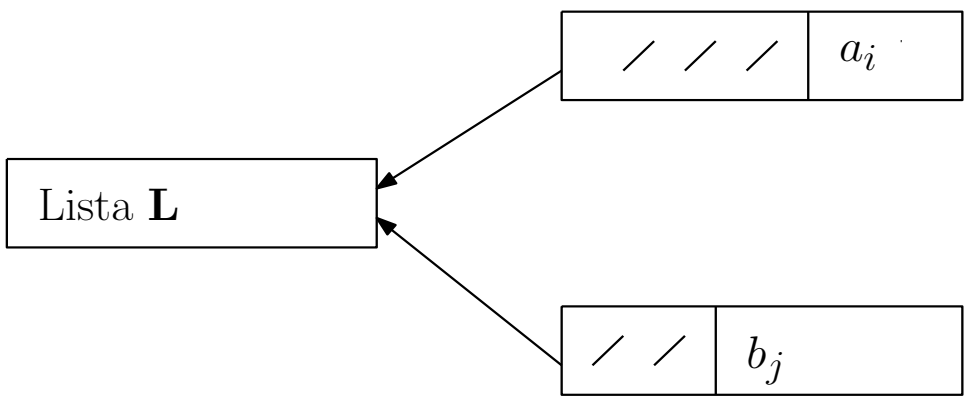
Merge: Trasforma due liste **ordinate** $A = a_1, \ldots, a_n$ e $B = b_1, \ldots, b_n$, in cui $a_1 \leq \ldots \leq a_n$ e $b_1 \leq \ldots \leq b_n$ in un'unica lista **ordinata** L.

```
Merge(A, B)
L = ∅
i = 1, j = 1
WHILE(entrambe le liste A, B non sono vuote){
    IF(a_i ≤ b_j){
      appendi a_i alla lista L, i = i + 1
      ELSE appendi b_j alla lista L, j = j + 1
   }
appendi il resto della lista non vuota, tra A e B, ad L
```

Altro esempio di tempo lineare.

Merge: Trasforma due liste **ordinate** $A = a_1, \ldots, a_n$ e $B = b_1, \ldots, b_n$, in cui $a_1 \leq \ldots \leq a_n$ e $b_1 \leq \ldots \leq b_n$ in un'unica lista **ordinata** L.

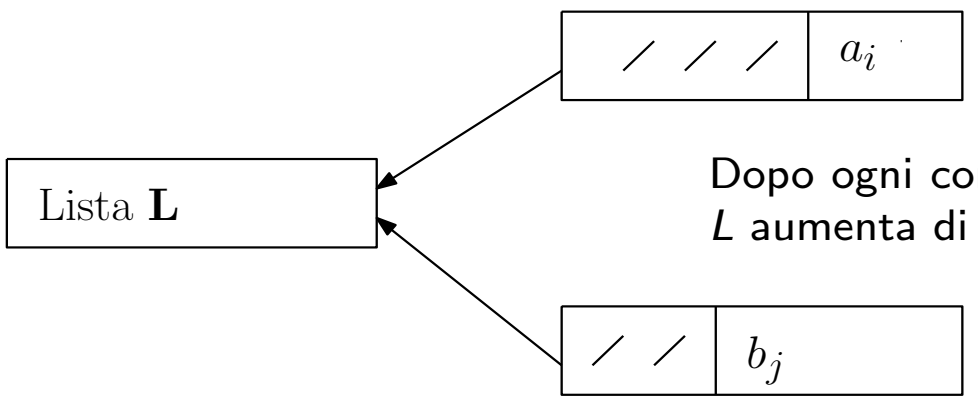
```
Merge(A, B)
L = ∅
i = 1, j = 1
WHILE(entrambe le liste A, B non sono vuote){
    IF(a_i ≤ b_j){
      appendi a_i alla lista L, i = i + 1
      ELSE appendi b_j alla lista L, j = j + 1
   }
appendi il resto della lista non vuota, tra A e B, ad L
```



Altro esempio di tempo lineare.

Merge: Trasforma due liste **ordinate** $A = a_1, \ldots, a_n$ e $B = b_1, \ldots, b_n$, in cui $a_1 \leq \ldots \leq a_n$ e $b_1 \leq \ldots \leq b_n$ in un'unica lista **ordinata** L.

```
Merge(A, B)
L = ∅
i = 1, j = 1
WHILE(entrambe le liste A, B non sono vuote){
    IF(a_i ≤ b_j){
      appendi a_i alla lista L, i = i + 1
      ELSE appendi b_j alla lista L, j = j + 1
   }
appendi il resto della lista non vuota, tra A e B, ad L
```

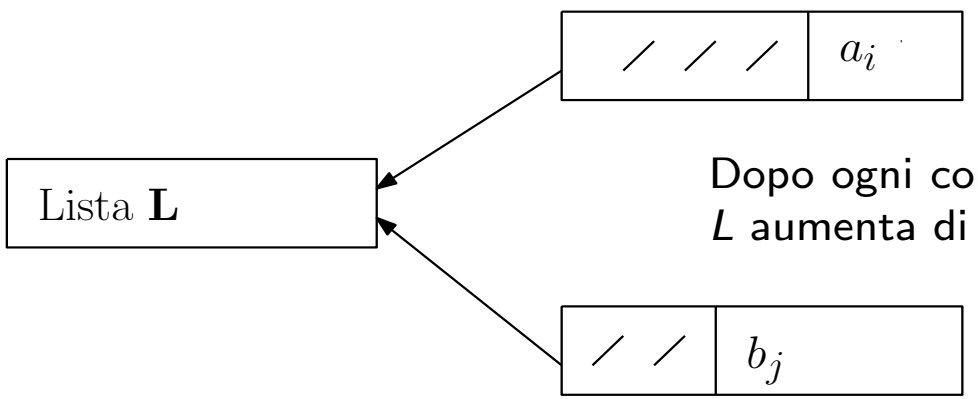


Dopo ogni confronto la lunghezza della lista output L aumenta di 1

Altro esempio di tempo lineare.

Merge: Trasforma due liste **ordinate** $A = a_1, \ldots, a_n$ e $B = b_1, \ldots, b_n$, in cui $a_1 \leq \ldots \leq a_n$ e $b_1 \leq \ldots \leq b_n$ in un'unica lista **ordinata** L.

```
Merge(A, B)
L = ∅
i = 1, j = 1
WHILE(entrambe le liste A, B non sono vuote){
    IF(a_i ≤ b_j){
      appendi a_i alla lista L, i = i + 1
      ELSE appendi b_j alla lista L, j = j + 1
   }
appendi il resto della lista non vuota, tra A e B, ad L
```



Dopo ogni confronto la lunghezza della lista output L aumenta di 1 $\Rightarrow$ il numero di operazioni $= O(n)$

Esempi di analisi di algoritmi: Tempo quadratico

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

1. $\min = (x_1 - x_2)^2 + (y_1 - y_2)^2$

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

1. $\min = (x_1 - x_2)^2 + (y_1 - y_2)^2$
2. $\texttt{FOR}(i = 1, i < n + 1; i = i + 1)\{$

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

1. $\min = (x_1 - x_2)^2 + (y_1 - y_2)^2$
2. $\texttt{FOR}(i = 1, i < n + 1; i = i + 1)\{$
3. $\qquad \texttt{FOR}(j = i + 1, j < n + 1; j = j + 1)\{$

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

1. $\min = (x_1 - x_2)^2 + (y_1 - y_2)^2$
2. $\texttt{FOR}(i = 1, i < n + 1; i = i + 1)\{$
3. $\quad \texttt{FOR}(j = i + 1, j < n + 1; j = j + 1)\{$
4. $\quad\quad d = (x_i - x_j)^2 + (y_i - y_j)^2$

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

1. $\min = (x_1 - x_2)^2 + (y_1 - y_2)^2$
2. $\texttt{FOR}(i = 1, i < n + 1; i = i + 1)\{$
3. $\quad \texttt{FOR}(j = i + 1, j < n + 1; j = j + 1)\{$
4. $\quad\quad d = (x_i - x_j)^2 + (y_i - y_j)^2$
5. $\quad\quad \texttt{IF}(d < \min)\{$
6. $\quad\quad \min = d$
   $\quad\quad \}$
   $\quad \}$
   $\}$

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

```
1. min = (x_1 − x_2)^2 + (y_1 − y_2)^2
2. FOR(i = 1, i < n + 1; i = i + 1){
3.       FOR(j = i + 1, j < n + 1; j = j + 1){
4.          d = (x_i − x_j)^2 + (y_i − y_j)^2
5.          IF(d < min){
6.          min = d
         }
      }
}
RETURN min
```

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

```
1. min = (x_1 − x_2)^2 + (y_1 − y_2)^2
2. FOR(i = 1, i < n + 1; i = i + 1){
3.       FOR(j = i + 1, j < n + 1; j = j + 1){
4.          d = (x_i − x_j)^2 + (y_i − y_j)^2
5.          IF(d < min){
6.          min = d
         }
      }
}
RETURN min
```

Analisi: il `FOR` delle linee 3.–5. esegue la prima volta $c(n-1)$ operazioni,

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

```
1. min = (x_1 − x_2)^2 + (y_1 − y_2)^2
2. FOR(i = 1, i < n + 1; i = i + 1){
3.       FOR(j = i + 1, j < n + 1; j = j + 1){
4.          d = (x_i − x_j)^2 + (y_i − y_j)^2
5.          IF(d < min){
6.          min = d
         }
      }
}
RETURN min
```

Analisi: il `FOR` delle linee 3.–5. esegue la prima volta $c(n-1)$ operazioni, la seconda volta $c(n-2)$ operazioni,

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

```
1. min = (x_1 − x_2)^2 + (y_1 − y_2)^2
2. FOR(i = 1, i < n + 1; i = i + 1){
3.       FOR(j = i + 1, j < n + 1; j = j + 1){
4.          d = (x_i − x_j)^2 + (y_i − y_j)^2
5.          IF(d < min){
6.          min = d
         }
      }
}
RETURN min
```

Analisi: il `FOR` delle linee 3.–5. esegue la prima volta $c(n-1)$ operazioni, la seconda volta $c(n-2)$ operazioni, la terza volta $c(n-3)$ operazioni,

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

```
1. min = (x_1 − x_2)^2 + (y_1 − y_2)^2
2. FOR(i = 1, i < n + 1; i = i + 1){
3.       FOR(j = i + 1, j < n + 1; j = j + 1){
4.          d = (x_i − x_j)^2 + (y_i − y_j)^2
5.          IF(d < min){
6.          min = d
         }
      }
}
RETURN min
```

Analisi: il `FOR` delle linee 3.–5. esegue la prima volta $c(n-1)$ operazioni, la seconda volta $c(n-2)$ operazioni, la terza volta $c(n-3)$ operazioni, ...

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

1. $\min = (x_1 - x_2)^2 + (y_1 - y_2)^2$
2. `FOR`$(i = 1, i < n + 1; i = i + 1)\{$
3. `FOR`$(j = i + 1, j < n + 1; j = j + 1)\{$
4. $d = (x_i - x_j)^2 + (y_i - y_j)^2$
5. `IF`$(d < \min)\{$
6. $\min = d$

```
    }
  }
}
RETURN min
```

Analisi: il `FOR` delle linee 3.–5. esegue la prima volta $c(n-1)$ operazioni, la seconda volta $c(n-2)$ operazioni, la terza volta $c(n-3)$ operazioni, ... In totale, l'algoritmo esegue
$c(n-1) + c(n-2) + c(n-3) + \ldots + c \cdot 1 = c \sum_{k=1}^{n-1} k$

Esempi di analisi di algoritmi: Tempo quadratico

Dati n punti, di coordinate $(x_1, y_1), \ldots, (x_n, y_n)$, si vuole determinare la coppia di punti più vicina

```
1. min = (x_1 − x_2)^2 + (y_1 − y_2)^2
2. FOR(i = 1, i < n + 1; i = i + 1){
3.       FOR(j = i + 1, j < n + 1; j = j + 1){
4.          d = (x_i − x_j)^2 + (y_i − y_j)^2
5.          IF(d < min){
6.          min = d
         }
      }
}
RETURN min
```

Analisi: il `FOR` delle linee 3.–5. esegue la prima volta $c(n-1)$ operazioni, la seconda volta $c(n-2)$ operazioni, la terza volta $c(n-3)$ operazioni, ... In totale, l'algoritmo esegue
$c(n-1) + c(n-2) + c(n-3) + \ldots + c \cdot 1 = c \sum_{k=1}^{n-1} k = O(n^2)$
operazioni

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. `FOR`$(i = 1, i < n + 1, i = i + 1)\{$

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. `FOR`$(i = 1, i < n + 1, i = i + 1)\{$
2. `FOR`$(j = i + 1, j < n + 1, j = j + 1)\{$

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. `FOR`$(i = 1, i < n + 1, i = i + 1)\{$
2. `FOR`$(j = i + 1, j < n + 1, j = j + 1)\{$
3. `FOR(ogni elemento` $x \in S_i)\{$
4. determina se x appartiene anche a S_j

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
}
5. IF(nessun elemento di S_i appartiene anche a S_j){

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
   }
5. IF(nessun elemento di S_i appartiene anche a S_j){
6. RETURN(S_i ed S_j sono disgiunti)

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
   }
5. IF(nessun elemento di S_i appartiene anche a S_j){
6. RETURN(S_i ed S_j sono disgiunti)
   }
   }
7. RETURN(non esistono insiemi disgiunti)

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
   }
5. IF(nessun elemento di S_i appartiene anche a S_j){
6. RETURN(S_i ed S_j sono disgiunti)
   }
   }
7. RETURN(non esistono insiemi disgiunti)

Analisi: il FOR delle linee 3.–6 esegue $O(n)$ operazioni,

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
}
5. IF(nessun elemento di S_i appartiene anche a S_j){
6. RETURN(S_i ed S_j sono disgiunti)
}
}
7. RETURN(non esistono insiemi disgiunti)

Analisi: il FOR delle linee 3.–6 esegue $O(n)$ operazioni, il FOR delle linee 2.–6 esegue $O(n^2)$ operazioni,

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
}
5. IF(nessun elemento di S_i appartiene anche a S_j){
6. RETURN(S_i ed S_j sono disgiunti)
}
}
7. RETURN(non esistono insiemi disgiunti)

Analisi: il FOR delle linee 3.–6 esegue $O(n)$ operazioni, il FOR delle linee 2.–6 esegue $O(n^2)$ operazioni,

Esempi di analisi di algoritmi

Tempo cubico : $O(n^3)$ L'algoritmo esamina tutte le triple di dati elementi

Esempio: Dati n insiemi $S_1, \ldots, S_n$, ciascuno di essi sottoinsieme di $\{1, \ldots, n\}$, esiste una coppia (S_i, S_j) tale che $S_i \cap S_j = \emptyset$?

1. FOR($i = 1, i < n + 1, i = i + 1$){
2. FOR($j = i + 1, j < n + 1, j = j + 1$){
3. FOR(ogni elemento $x \in S_i$){
4. determina se x appartiene anche a S_j
}
5. IF(nessun elemento di S_i appartiene anche a S_j){
6. RETURN(S_i ed S_j sono disgiunti)
}
}
7. RETURN(non esistono insiemi disgiunti)

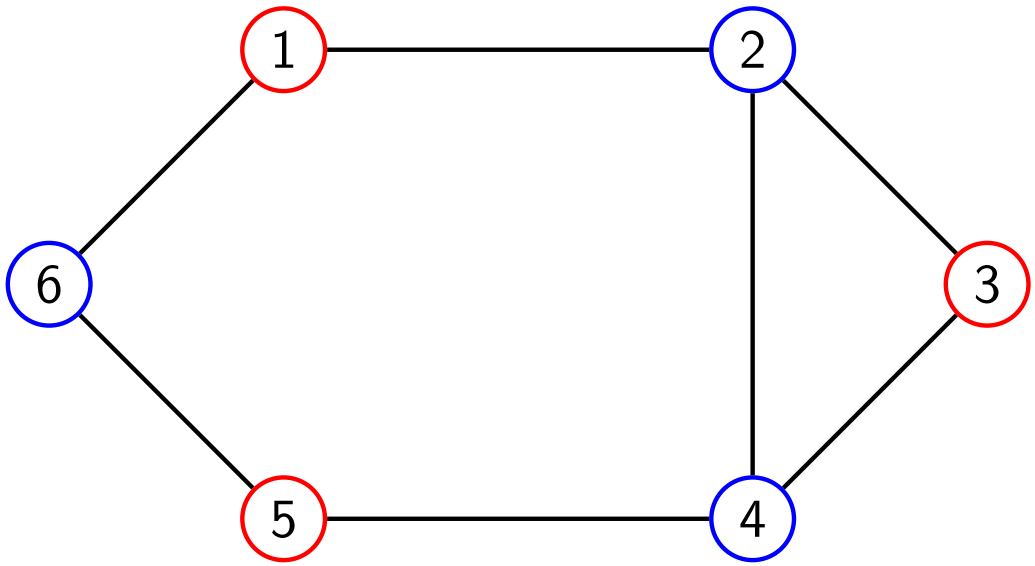
Analisi: il FOR delle linee 3.–6 esegue $O(n)$ operazioni, il FOR delle linee 2.–6 esegue $O(n^2)$ operazioni,
In totale, l'algoritmo esegue $O(n^3)$ operazioni

Esempi di analisi di algoritmi

Per introdurre l'ultimo esempio, diamo la seguente definizione. Un insieme di punti è detto **indipendente** se *nessuna* coppia di suoi elementi è unita da un segmento.

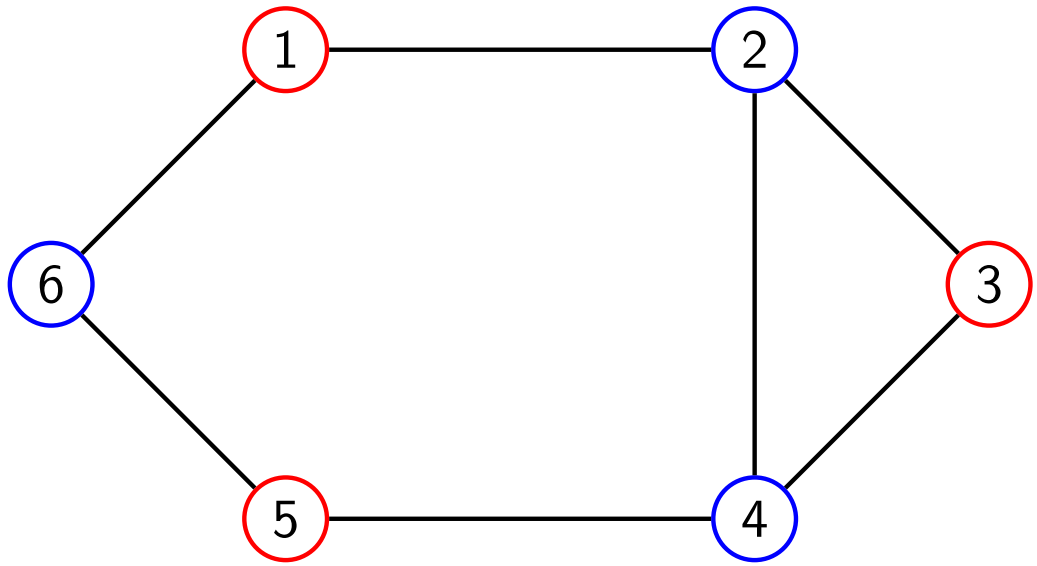
Esempi di analisi di algoritmi

Per introdurre l'ultimo esempio, diamo la seguente definizione. Un insieme di punti è detto **indipendente** se *nessuna* coppia di suoi elementi è unita da un segmento.



Esempi di analisi di algoritmi

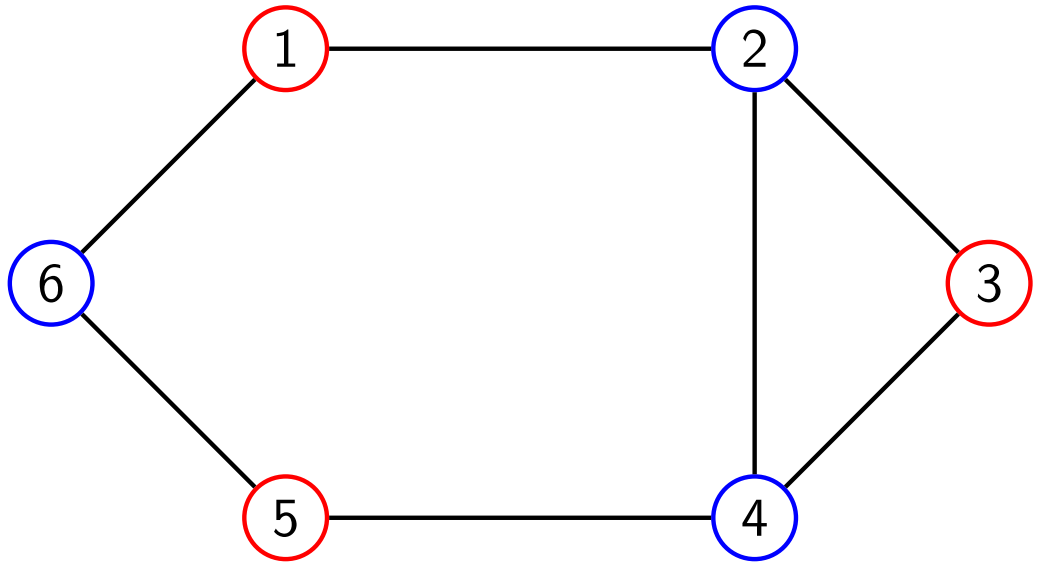
Per introdurre l'ultimo esempio, diamo la seguente definizione. Un insieme di punti è detto **indipendente** se *nessuna* coppia di suoi elementi è unita da un segmento.



L'insieme $\{1, 3, 5\}$ è indipendente,

Esempi di analisi di algoritmi

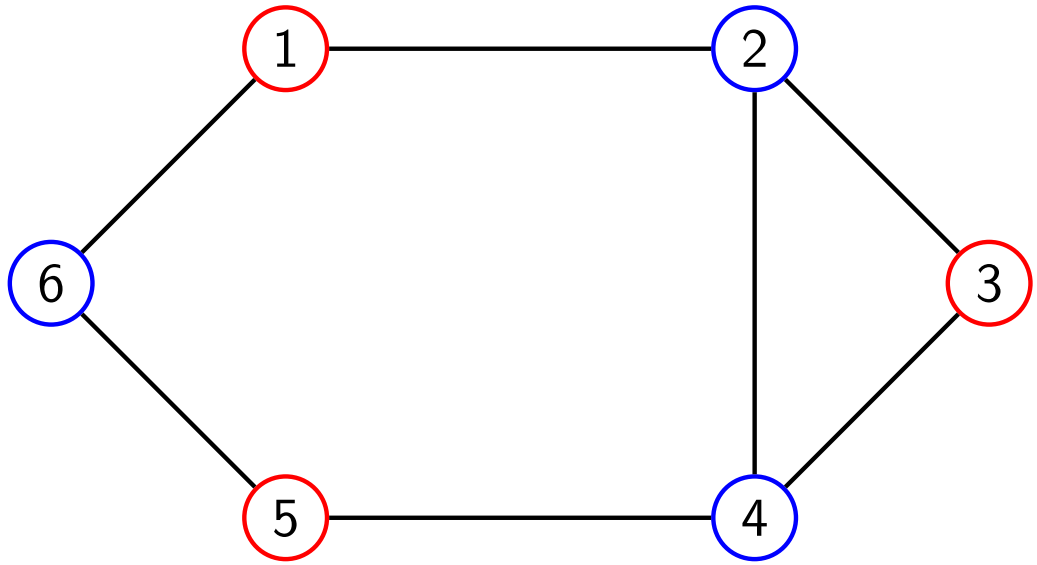
Per introdurre l'ultimo esempio, diamo la seguente definizione. Un insieme di punti è detto **indipendente** se *nessuna* coppia di suoi elementi è unita da un segmento.



L'insieme $\{1, 3, 5\}$ è indipendente, l'insieme $\{2, 4, 6\}$ **non** è indipendente,

Esempi di analisi di algoritmi

Per introdurre l'ultimo esempio, diamo la seguente definizione. Un insieme di punti è detto **indipendente** se *nessuna* coppia di suoi elementi è unita da un segmento.



L'insieme $\{1, 3, 5\}$ è indipendente, l'insieme $\{2, 4, 6\}$ **non** è indipendente,

Problema: trovare il più grande insieme indipendente in un grafo con n punti.

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `  controlla se` S `è indipendente`

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `controlla se` S `è indipendente`
4. **if** (S è il sottoinsieme indipendente più grande trovato finora)

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `controlla se` S `è indipendente`
4. **if** (S è il sottoinsieme indipendente più grande trovato finora)
5. `aggiorna` $S^* = S$

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `controlla se` S `è indipendente`
4. **if** (S è il sottoinsieme indipendente più grande trovato finora)
5. `aggiorna` $S^* = S$
6. **return**(S^*)

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `controlla se` S `è indipendente`
4. **if** (S è il sottoinsieme indipendente più grande trovato finora)
5. `aggiorna` $S^* = S$
6. **return**(S^*)

Analisi: il **for** delle linee 2.–5. viene eseguito 2^n volte (tanti sono tutti i sottoinsiemi di n punti).

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `controlla se` S `è indipendente`
4. **if** (S è il sottoinsieme indipendente più grande trovato finora)
5. `aggiorna` $S^* = S$
6. **return**(S^*)

Analisi: il **for** delle linee 2.–5. viene eseguito 2^n volte (tanti sono tutti i sottoinsiemi di n punti).
Controllare se un sottoinsieme S è indipendente richiede $O(n^2)$ operazioni (per ogni coppia di punti in S occorre controllare se vi è un segmento tra di loro).

Esempi di analisi di algoritmi

Tempo esponenziale : $O(c^n)$
L'algoritmo esamina tutte le possibili soluzioni per trovare la "migliore"

1. $S^* = \emptyset$
2. **for** `ogni sottoinsieme di punti` S
3. `controlla se` S `è indipendente`
4. **if** (S è il sottoinsieme indipendente più grande trovato finora)
5. `aggiorna` $S^* = S$
6. **return**(S^*)

Analisi: il **for** delle linee 2.–5. viene eseguito 2^n volte (tanti sono tutti i sottoinsiemi di n punti).
Controllare se un sottoinsieme S è indipendente richiede $O(n^2)$ operazioni (per ogni coppia di punti in S occorre controllare se vi è un segmento tra di loro). In totale, l'algoritmo esegue $O(n^2 2^n)$ operazioni.