

complessità

Programmazione 2

A. A. 2021/22

costo di un algoritmo

- Dipende dall'input
 - Tempo
 - Spazio

(N)

GUARDIA $\left[\begin{array}{l} \text{while } (x < N) \{ \\ \quad \text{if } x \% 2 == 0 \{ \\ \quad \quad \dots \\ \quad \} \\ \} \end{array} \right.$

CORPO $\left[\right.$

complessità

$\sum_{i=0}^N$

C_i

costo del corpo
della funzione

ARRAY : N elementi

Tempo \rightarrow
Tempo

0	1	2	3	4	5	6
3	7	0	5	6	2	4
6	5	4	3	2	1	0

CASO
MIGLIORE
CASO
MEDIO
CASO
PEGGIORE

ord.
crescente

complessità temporale

ASSUNZIONE 1

complessità Temporale

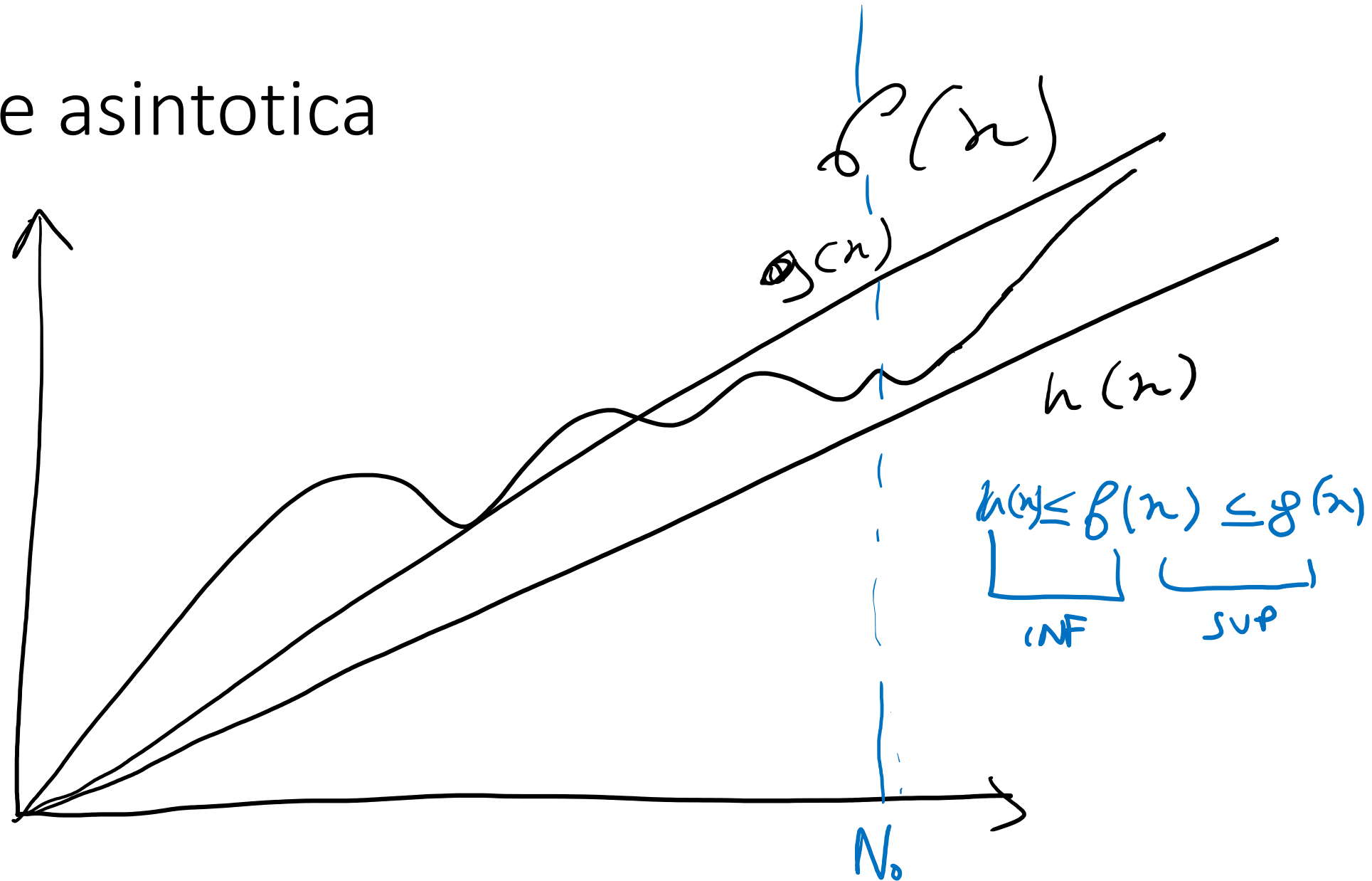
- PROPORZIONALE AL NUMERO DI OPERAZIONI ESEGUITE (ISTRUZIONI)
- INDIPENDENTE DALL'HW CONSIDERATO

complessità temporale

ASSUNZIONE 2

- DIPENDENZA DALLA
DIMENSIONE DELL'INPUT
- COMPORTAMENTO
ASINTOTICO $n \rightarrow \infty$

notazione asintotica



notazione O

\mathcal{O} limite asintotico superiore
 $g(n)$

$$\mathcal{O}(g(n)) = \left\{ f(n) : \exists c, n_0 \in \mathbb{N} \text{ c.c.} \right. \\ \left. 0 \leq f(n) \leq c g(n) \quad \forall n \geq n_0 \right\}$$

notazione O

$O(g(n))$

un possibile
limite superiore

$$f(n) = \begin{bmatrix} O(n \log n) \\ O(n^2), O(n^2 \log n), O(n^4) \end{bmatrix}$$

funzioni notevoli

$\log(m)$ for ($\text{int } i=1; i \leq m; i *= 2$) {
 n \rightarrow for ($j=1; j \leq \frac{m}{2}; j++$) {
 $\log(m)$ \rightarrow for ($k=1; k \leq m; k *= 2$) {
 $O(n \log^2 m)$ $O(1)$
 $\}$
 $\}$

$\frac{1}{2} \log m$
 $\frac{1}{2} \log m$
 $\frac{1}{2} \log m$
 $\frac{1}{2} \log m$

0	1	$\sqrt{3}$
1	1	$\sqrt{3}$
2	1	$\sqrt{3}$
3	1	$\sqrt{3}$

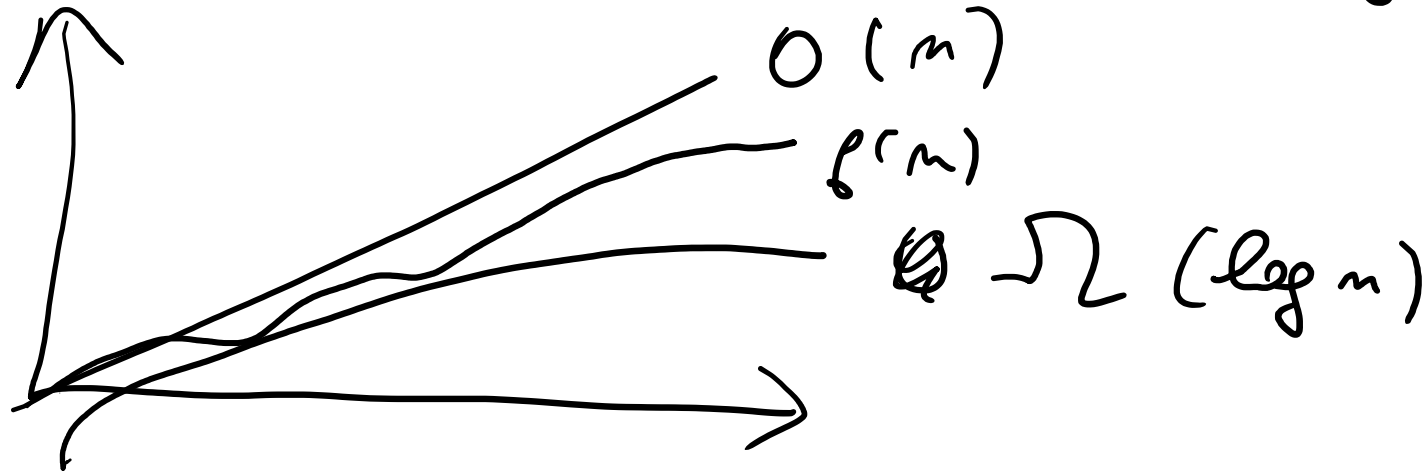
notazione Ω

$$\Omega(g(n)) = \{f(n) : \exists c, n_0 \mid 0 \leq c g(n) \leq f(n)$$

$$\forall n \geq n_0\}$$

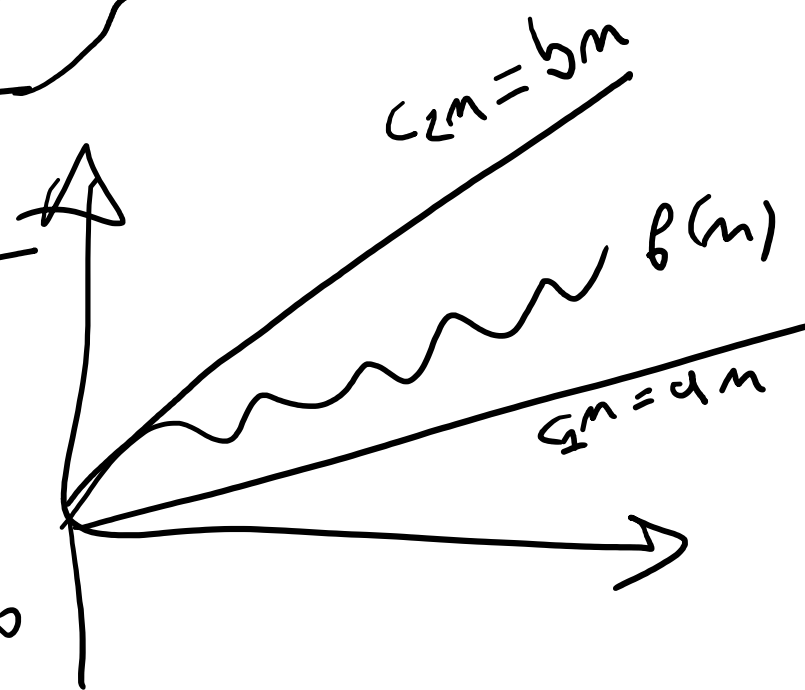
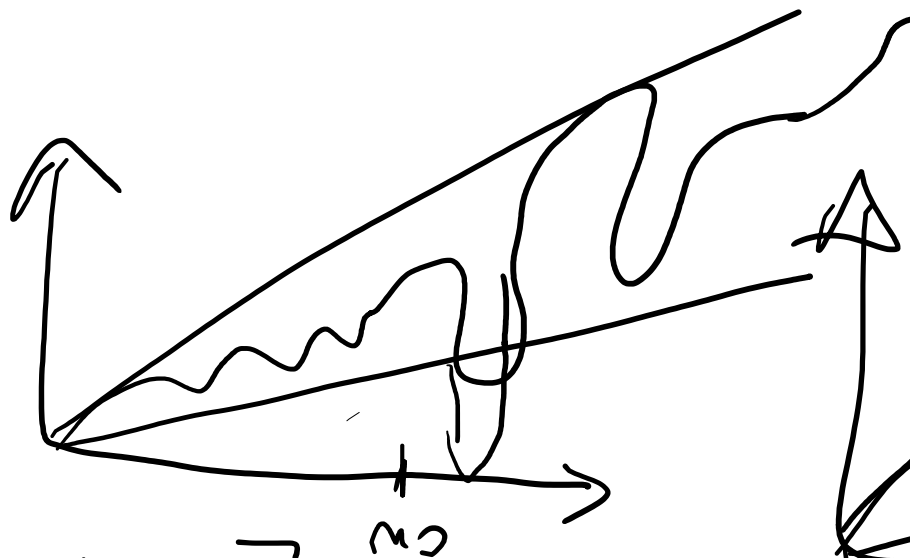
\in

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



notazione Θ

①



$$\Theta(g(n)) = \{ f(n) : \exists c_1, c_2, n_0$$

$$0 < c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \forall n \geq n_0 \}$$

$$f(n) = \Theta(g(n)) \iff f(n) = \Omega(g(n)) \text{ e } f(n) = O(g(n))$$

$$O(g(n)) = \{f(n) : \exists c, n_1 \in \mathbb{N}$$

$$0 \leq f(n) \leq c g(n) \quad \forall n \geq n_1\}$$

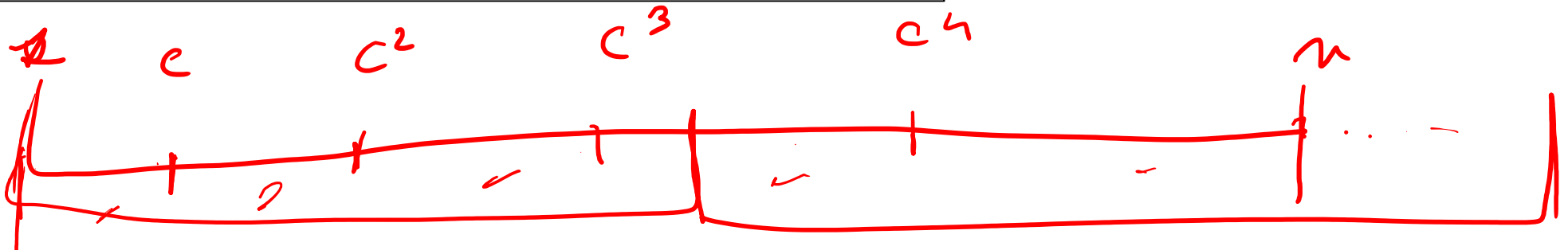
Analisi di Algoritmi: esempi pratici

```
// c è una costante positiva
for (int i = 1; i <= n; i *= c) {
    //espressioni con costo O(1)
}
```

→ **$O(\log n)$**

```
// c è una costante positiva > 1
for(int i = 2; i <= n; i = pow(i, c)) {
    //espressioni con costo O(1)
}
```

$O(\log \log n)$



Esempio

Qual è la complessità del seguente algoritmo?

```
void func(int n) {  
    int count=0;  
    for(i=n/2; i<=n; i++) {  
        for(j=1; j<=n; j=2*j) {  
            for(k=1; k<=n; k=k*2) {  
                count++;  
            }  
        }  
    }  
}
```

$\Theta(n)$
 ~~$\frac{n}{2}$~~ $\Rightarrow O(n)$



Esempio

Qual è la complessità del seguente algoritmo?

```
void func(int n) {  
    int count=0;  
    for(i=n/2; i<=n; i++) {  
        for(j=1; j<=n; j=2*j) {  
            for(k=1; k<=n; k=k*2) {  
                count++;  
            }  
        }  
    }  
}
```

$f(n) : \exists c, m_0$
 $f(n) \leq c g(n)$
 $O(f(n))$
 $\frac{\log}{2}$
 $6 \log_2 n < 1287$
 \log_2



Esempio

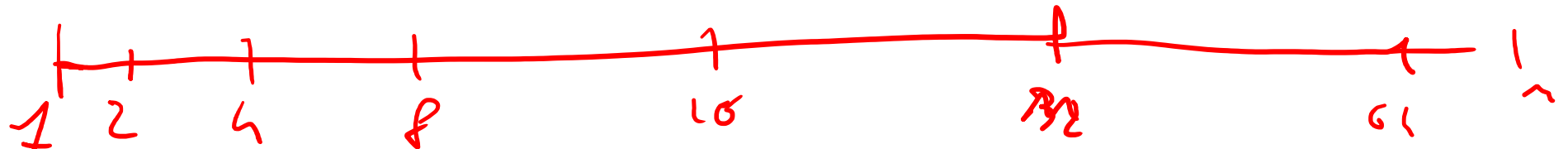
Qual è la complessità del seguente algoritmo?

```
void func(int n) {  
    int count=0;  
    for(i=n/2; i<=n; i++) {  
        for(j=1; j<=n; j=2*j) {  
            for(k=1; k<=n; k=k*2, {  
                count++; }  
        }  
    }  
}
```

$O(n)$

$O(\log n)$

$O(\log n)$



Esempio

Qual è la complessità del seguente algoritmo?

```
void func(int n) {  
    int count=0;  
    for(i=n/2; i<=n; i++) {  
        for(j=1; j<=n; j=2*j) {  
            for(k=1; k<=n; k=k*2) {  
                count++;  
            }  
        }  
    }  
}
```

$O(n)$

$O(\log n)$

$O(\log n)$

Esempio

Qual è la complessità del seguente algoritmo?

```
void func(int n) {  
    int count=0;  
    for(i=n/2; i<=n; i++) {  
        for(j=1; j<=n; j=2*j) {  
            for(k=1; k<=n; k=k*2) {  
                count++;  
            }  
        }  
    }  
}
```

$O(n)$

$O(\log n)$

$O(\log n)$

$O(n \log^2 n)$

$n \times \log n \times \log n$
 $O(n \log^2 n)$

Esempio

Qual'è l'ordine di grandezza (notazione O-grande) della funzione G

$$T1 = 5n^2 + 2n - 10$$

$$T2 = 5\sqrt{n} + 22$$

$$G = T1 + T2$$


Esempio

Qual'è l'ordine di grandezza (notazione O-grande) della funzione G

$$T1 = 5n^2 + 2n - 10$$

$$T2 = 5\sqrt{n} + 22$$

$$G = T1 + T2$$


$$O(n^2)$$

Esempio

Ricerca di un elemento:



```
int ricercaLineare (int vettore[], int dim, int chiave)  
{  
    ➔ for (int i = 0; i < dim; i++)  
        if (vettore[i] == chiave) return i;  
    return -1;  
}
```

Caso migliore: 1
Caso peggiore: n
Caso medio: ?

42

Esempio

Ricerca di un elemento:

v[]	2	5	3	0	12	45	55	12	4
	v[0]	v[1]	...						
	v[n-1]								

```
int ricercaLineare (int vettore[], int dim, int chiave)
{
    for (int i = 0; i < dim; i++)
        if (vettore[i] == chiave) return i;
    return -1;
}
```

Caso migliore: 1

Caso peggiore: n

Caso medio: ?

$$\sum_{(i=1..N)} \text{Prob}(\text{el}(i)) * i$$

Esempio

Ricerca di un elemento:

v[]	2	5	3	0	12	45	55	12	4
	v[0]	v[1]	...						
	v[n-1]								

```
int ricercaLineare (int vettore[], int dim, int chiave)
{
    for (int i = 0; i < dim; i++)
        if (vettore[i] == chiave) return i;
    return -1;
}
```

Caso migliore: 1
Caso peggiore: n
Caso medio: ?

$$\sum_{(i=1..N)} \text{Prob}(\text{el}(i)) * i = \sum_{(i=1..N)} (1/N) * i :$$

Esempio

Ricerca di un elemento:

v[]	2	5	3	0	12	45	55	12	4
	v[0]		v[1]	...					
									v[n-1]

Handwritten notes showing the derivation of the average case for linear search:

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

```
int ricercaLineare (int vettore[], int dim, int chiave)
{
    for (int i = 0; i < dim; i++)
        if (vettore[i] == chiave) return i;
    return -1;
}
```

Caso migliore: 1

Caso peggiore: n

Caso medio: (n+1)/2

$$\sum_{(i=1..N)} \text{Prob}(\text{el}(i)) \cdot i = \sum_{(i=1..N)} (1/N) \cdot i$$