COMPLESSITA' COMPUTAZIONALE

STUDIO SENZA ESEGUIRE IL CODICE DEL CONSUMO DI RISORSE DI UN ALGORITMO / PROGRAMMA.

- TEMPO DI ESECUZIONE
- MEMORIA
- QUANTITA' DI BIT TRASMESSI IN RETE
- ...

IL TEMPO DI ESECUZIONE DIPENDE DA TROPPI FATTORI

- VELOCITA' COMPUTER
- GRADO DI UTILIZZO DEL COMPUTER
- LINGUAGGIO DI PROGRAMMAZIONE
- COMPILATORE

?

MISURIAMO IL NUMERO DI "OPERAZIONI ELEMENTARI" CHE VENGONO SVOLTE DURANTE L'ESECUZ. DEL PROGRAMMA

OPERAZIONI LA CUI ESECUZIONE RICHIEDE UN TEMPO COSTANTE, INDIPENDENTE DAI DATI SU CUI OPERA (PUÒ DIPENDERE DAL LINGUAGGIO DI PROGRAMMAZIONE)

ESEMPI OP. ELEMENTARI IN PYTHON

`X = y` (ASSEGNAMENTO)

`3+2` , `7*4` , ... (OPERAZIONI ARITMETICHE)

`LISTA[i]` (ACCESSO ALLE LISTE)

`LISTA[1] = 5` (MODIFICA ELEMENTO DI UNA LISTA)

LISTA.APPEND(v)

ESEMPI OP. NON ELEMENTARI

LISTA1 + LISTA2 (CONCATENAZ. TRA LISTE)

[0] * n

STRINGA1 + STRINGA2

S.UPPER()

ESEMPIO: FUNZIONE CHE RESTITUISCE LA SOMMA DI TUTTI GLI ELEMENTI DI UNA LISTA

```
def somma_lista_iterativa(l):
    somma = 0                    (1)
    for x in l:                  (1)   } VIENE ESEGUITA len(l) VOLTE
        somma += x               (1)
    return somma                 (1)
```

$1 + (1+1) \cdot len(l) + 1 = 2 + 2\,len(l)$

$= 2 + 2n$ dove n è la lunghezza di l

DIMENSIONE DELL'INPUT

QUANDO SI CALCOLA LA COMPLESSITÀ DI UN ALGORITMO, IL RISULTATO È DI SOLITO UNA FORMULA CHE CONTIENE LE DIMENSIONI DELL'INPUT.

- LISTE: lunghezza
- STRINGHE: " "
- MATRICE: numero righe, numero colonne
  oppure
  numero di celle

ESEMPIO: RICERCA LINEARE

```
def linear_search(l, x):
    """
    Restituisce la posizione
    """
    for i in range(len(l)):
        if l[i] == x:
            return i
    return -1
```

linear_search([10, 20, 5, 17], 10) (0, 1, 2, 3)

n. volte

2

1 } dipende dall'input (da 1 a len(l))

1 * 1 } sono in alternativa

1 * 1

→ COMPLESSITÀ 1. CASO OTTIMO:

SUPPONIAMO CHE GLI INPUT CI SIANO FAVOREVOLI.

$C = 1 + 1 + 1 = 3$

⇒ COMPLESSITÀ CASO PESSIMO: (n è la lunghezza di l)

SUPPONIAMO CHE GLI INPUT CI SIANO SFAVOREVOLI

$C = 1 * n + 1 * n + 1 = 2n + 1$

COMPLESSITÀ CASO MEDIO:

LA COMPLESSITÀ MEDIA PER TUTTI I POSSIBILI INPUT DI UNA DATA LUNGHEZZA.

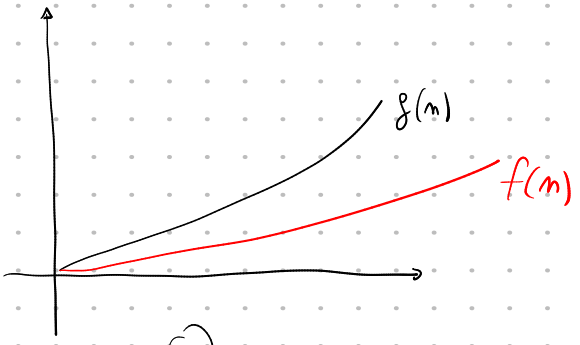
ORDINI DI GRANDEZZA (O-grande)

INFORMALMENTE

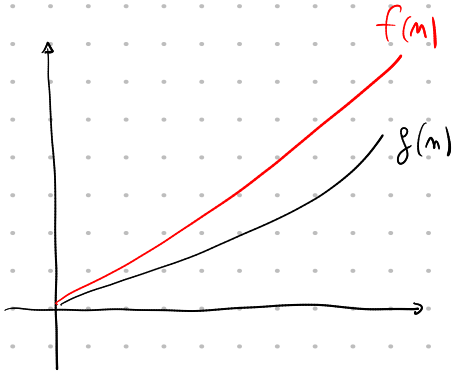
Se $f: \mathbb{N} \to \mathbb{N}$, $g: \mathbb{N} \to \mathbb{N}$, diciamo che $f(n)$ è $O(g(n))$ quando

- $f(n)$ cresce al più come $g(n)$
- $f(n)$ ha un ordine di infinito minore o uguale a $g(n)$

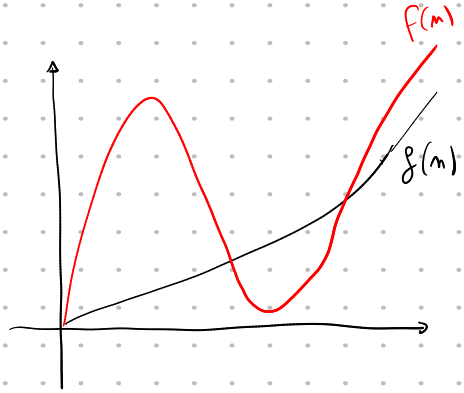
Esempi



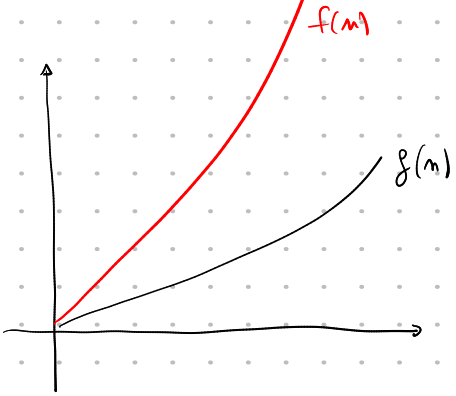
$f(n)$ è $O(g(n))$



$f(n)$ è $O(g(n))$



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



$f(n)$ non è $O(g(n))$

Definizione (formale)

Date $f: \mathbb{N} \to \mathbb{N}$ e $g: \mathbb{N} \to \mathbb{N}$, si dice che $f(n)$ è $O(g(n))$ ("O-grande di $g(n)$") quando

$$\exists n_0 > 0 \quad \exists c > 0 \quad \text{t.c.} \quad \forall n > n_0 \quad f(n) \leq c \cdot g(n)$$

Teorema

Se $\lim_{n \to +\infty} \frac{f(n)}{g(n)} = k < +\infty$, allora $f(n) = O(g(n))$.

$$O(\log n) < O(\sqrt{n}) = O(n^{1/2}) < O(n) < O(n^2) < O(n^3) < \dots < O(2^n) < O(3^n) < \dots$$

$n^2 \in O(2^n)$

base > 1

ESEMPIO: DI NUOVO SOMMA LISTA

$f(n) = 2n + 2$ complessità calcolata precedentemente

$f(n) = O(n)$ Ho preso il termine di ordine di infinito massimo e ho tolto le costanti

$$\lim_{n\to+\infty} \frac{2n+2}{n} = \lim_{n\to+\infty} \frac{2n}{n} + \lim_{n\to+\infty} \frac{2}{n} = 2 + \frac{2}{+\infty} = 2 \leq +\infty$$

Notare che

$$2n+2 = O(n^2)$$
$$= O(n^{100.000})$$
$$= O(2^n)$$

SONO TUTTI VERI, MA OVVIAMENTE QUANDO SI DÀ LA COMPLESSITÀ SI SCEGLIE L'ORDINE PIÙ PRECISO CHE IN QUESTO CASO È $O(n)$.

ESEMPIO: CALCOLO TRACCIA DI UNA MATRICE QUADRATA

↳ SOMMA ELEMENTI SULLA DIAGONALE PRINCIPALE

$$\begin{pmatrix} 3 & 2 & 0 \\ 1 & 5 & 4 \\ 7 & 0 & 3 \end{pmatrix}$$

traccia ~> 11

$$O(f(n)) + O(g(n)) = O(f(n)\,g(n))$$

$O(f(n)) + O(g(n)) = O($ quella di ordine massimo$)$

```
DEF TRACCIA1 (M):
    SOMMA = 0                          (1)
    FOR I IN RANGE (LEN(M)):           (1)
        SOMMA += M[I][I]               (1)   * len(M)
    RETURN SOMMA                       (1)
```

M è QUADRATA.
N. RIGHE = N. COLONNE

n = numero righe delle matrice

$$C(n) = 2 + 2n = O(n)$$

n = n. righe e colonne di M

```
DEF TRACCIA2 (M):
    SOMMA=0
    FOR I IN RANGE (LEN(M)):
        FOR J IN RANGE (LEN(M[0])):
            IF i==J:
                SOMMA += M[I][J]
    RETURN SOMMA
```

$O(1) * 1$

$O(1) * n$ — numero di volte che estraggo J

$O(1) * n * n$ — numero di volte che eseguo il for J da capo

$O(1) * n^2$

$O(1) * n$

$O(1) * 1$

Non n^2 perché le somme le faccio solo se $i = J$, quindi solo se sono nella diagonale, e la diagonale ha solo n elementi

$$O(1) + O(n) + O(n^2) + O(n^2) + O(n) + O(1)$$

$$= O(n^2)$$