# COMPLESSITÀ COMPUTAZIONALE

- COMPLESSITÀ = si definisce la complessità di un algoritmo come la funzione che associa alla dimensione di un problema il costo della sua risoluzione in termini di tempo e memoria.

- NB = nel calcolo della complessità si dà una misura in termini di tempo di un algoritmo svincolato dal tipo di macchina su cui sarà eseguito.

- TEMPO DI ESECUZIONE = si indica come $T_P(n)$ o $T(n)$ e rappresenta il numero di unità di tempo che occorrono per eseguire un programma P di dimensione n, (con n ingressi). $T(n)$ è una funzione che va da $\mathbb{N} \to \mathbb{N}$, dove con $\mathbb{N}$ indichiamo gli interi non negativi (quindi è possibile fare tutti i tempi; 0, 1, 2, 3, ...).

- ESEMPIO 1:

```
int max (int v[], int n){
    int m= v[0];
    for (int i=0; i<n; i++){
        if (m < v[i]) m= v[i];
    }
    return m;
}
```

Osserviamo per prima che l'assegnamento e il confronto del for impiegano un'unità di tempo, e che ad ogni ciclo viene eseguito il comando condizionale con il relativo assegnamento. Ad ogni ciclo, oltre alle 2 unità di tempo per l'if e il relativo assegnamento devono essere aggiunte altre due unità per la condizione del for e l'incremento, quindi ad ogni ciclo corrispondono 4 unità di tempo $\Rightarrow 4(n-1) = 4n-4$.

MASTER COPY
Tel. 388/9837745

Il primo numero di tempo di esecuzione va assegnato al primo assegnamento, l'assegnamento del for, l'ultimo incremento e il return, => (4n-4)+4 = 4n. Quindi il suo tempo di esecuzione è' 4n.

- NB = l'assunzione di addizione fatta quando si fa valere all'assegnamento una unità di tempo non è' del tutto corretta, poiché il suo tempo di esecuzione dipende:
  - DALLA COMPLESSITA' DELL'ESPRESSIONE
  - DAL TIPO DI COMPILATORE
  - DAL SET DI ISTRUZIONI MACCHINA

- Della complessità dell'espressione in quanto un assegnamento può avere come membro destro più espressioni, es: x = x + 4*y+3;
- Del tipo di compilatore, in quanto compilatori diversi possono tradurre con istruzioni di complessità diversa una stessa espressione.
- Del set di istruzioni, in quanto a seconda della macchina su cui viene eseguito l'espressione, le operazioni componenti possono essere più o meno complesse.

CLASSE DI COMPLESSITA' (ESPRESSIONE O-GRANDE): Date due funzioni f, g $\mathbb{N} \to \mathbb{N}$, si dice che g(n) è di ordine f(n) se esiste un intero $n_0$ e una costante c tale che $\forall n \geq n_0$ $g(n) \leq c \cdot f(n)$. Questo significa che g(n) è uguale al più a f(n) moltiplicato per una costante c, per certi valori di n (non tutti) ($n \geq n_0$).

$$g(n) \in O(f(n)) \Rightarrow \exists c>0 : \forall n \geq n_0 \quad g(n) \leq c \cdot f(n)$$

- CLASSE DI COMPLESSITÀ $O(f)$ = non classe di complessità individuata da una funzione $f$ che rappresenta la famiglia di funzioni che hanno lo stesso comportamento asintotico (lineare, quadratico, esponenziale, logaritmico, ecc.).

- CLASSE DI COMPLESSITÀ = una classe di complessità rappresenta un insieme di funzioni tutte del medesimo ordine di grandezza, aventi lo stesso comportamento asintotico (lineare, logaritmico, quadratico, ecc.)

- PROPRIETÀ COMPLESSITÀ COMPUTAZIONALE CONCRETA:
  - FATTORI COSTANTI: $c(f(n)) = K \cdot O(f(n))$
  - SOMMA: $f(n) \in O(g(n)) \Rightarrow f(n) + g(n) \in O(g(n))$
  - PRODOTTO: $f(n) \in O(f_1(n))$, $g(n) \in O(g_1(n)) \Rightarrow f(n) \cdot g(n) \in O(f_1(n) \cdot g_1(n))$
  - POLINOMI: $P(n) = a_m n^m + a_{m-1} n^{m-1} + a_1 n + a_0 \Rightarrow P(n) \in O(n^m)$
  - TRANSITIVA: $f(n) \in O(g(n))$, $g(n) \in O(h(n)) \Rightarrow f(n) \in O(h(n))$

- FUNZIONI INCOMMENSURABILI:

$$f(n) = \begin{cases} n & \text{se } n \text{ è pari} \\ n^2 & \text{se } n \text{ è dispari} \end{cases} \qquad g(n) = \begin{cases} n^2 & \text{se } n \text{ è pari} \\ n & \text{se } n \text{ è dispari} \end{cases}$$

Queste ultime funzioni sono incommensurabili perché appartengono a ordini di grandezza diversi.

- $f(n) = \begin{cases} n^2 & \text{se } n \text{ è dispari} \\ n^3 & \text{se } n \text{ è pari} \end{cases} \qquad g(n) = \begin{cases} n^4 & \text{se } n \text{ è primo} \\ n^3 & \text{se } n \text{ è composto} \end{cases}$

In questo caso sempre $f(n) \in O(g(n))$ ma non il contrario.

MASTER COPY

Tel. 388/9837745

- DIFFERENZA TRA COMPLESSITÀ COMPUTAZIONALE E COMPLESSITÀ COMP. CONCRETA: la differenza tra complessità computazionale e complessità computazionale concreta sta nel fatto che nella prima si stima la funzione che rappresenta il tempo di esecuzione di un algoritmo, in funzione della sua taglia, mentre nella complessità computazionale concreta si ~~[illegible]~~ individua la classe di complessità di un algoritmo, piuttosto che la funzione $T(n)$ rappresentante il suo tempo di esecuzione.

- COMPLESSITÀ COMPUTAZIONALE: la complessità computazionale ha ~~[illegible]~~ il compito di dare una stima del tempo di esecuzione di un algoritmo, indipendentemente dal tipo di macchina su cui viene eseguito.

- COMPLESSITÀ COMPUTAZIONALE CONCRETA: nella complessità computazionale concreta interessa sapere la relazione di crescita di una funzione $T(n)$ rispetto a quella della funzione stessa. Questo perché funzioni diverse, ma del solito ordine di grandezza, dal punto di vista asintotico hanno lo stesso comportamento. In definitiva ciò che la funzione $T(n)$ teniamo conto è il suo ordine di grandezza (o classe di complessità).

- CLASSI DI COMPLESSITÀ:
  - $O(1)$ COSTANTE
  - $O(\log n)$ LOGARITMICA *
  - $O(n)$ LINEARE
  - $O(n \log n)$ $n \log n$
  - $O(n^2)$ QUADRATICA
  - $O(n^3)$ CUBICA
  - $O(2^n)$ ESPONENZIALE

* nella complessità logaritmica non importa specificare la base poiché il suo comportamento è sempre uguale indipendentemente da questo: $\log_a(n) = \log_b(n) \cdot \log_a(b)$ dove $\log_a(b)$ è una costante $\log_a(n)$ e $\log_b(n)$

# COMPLESSITÀ PROGRAMMI ITERATIVI

- REGOLA CALCOLO COMPLESSITÀ = la regola è definita per induzione sulla sintassi dei costrutti del linguaggio. La complessità viene definita sui costrutti elementari, mentre per i costrutti composti la loro complessità è data dalla complessità dei costrutti componenti.

- GRAMMATICA DEFINIZIONE COMPLESSITÀ = con questa grammatica definiamo un sottoinsieme dei costrutti del C++, quali espressioni e comandi:
  - E: genera un espressione.
  - C: genera un comando.
  - V: genera un valore elementare.
  - I: genera un identificatore.
  - O: genera un operatore

mentre con i seguenti simboli definiamo la chiamata di funzione e la selezione di un elemento di un array.

  - I ($E_1$, ..., $E_n$): chiamata di funzione.
  - I [E]: selezione di un elemento di un array.

- DEFINIZIONE ESPRESSIONI E COMANDI =
  - E ::= V | I | I[E] | $E_1$ O $E_2$ | O E | I ($E_1$, ..., $E_n$)
  - C ::= I = E; | I[E] = E; | if (E) C | if (E) C else C | for (I = $E_1$; $E_2$; I = $E_3$) C | { $C_1$ ... $C_n$ } | while (E) C | do C while (E); | I ($E_1$, ..., $E_n$); | return E;

- SOMMA E PRODOTTO ERA CLASSI DI COMPLESSITÀ =
  - SOMMA: $O(f(n)) + O(g(n)) = O(f(n) + g(n))$
  - PRODOTTO: $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$