**8° lezione prog II**

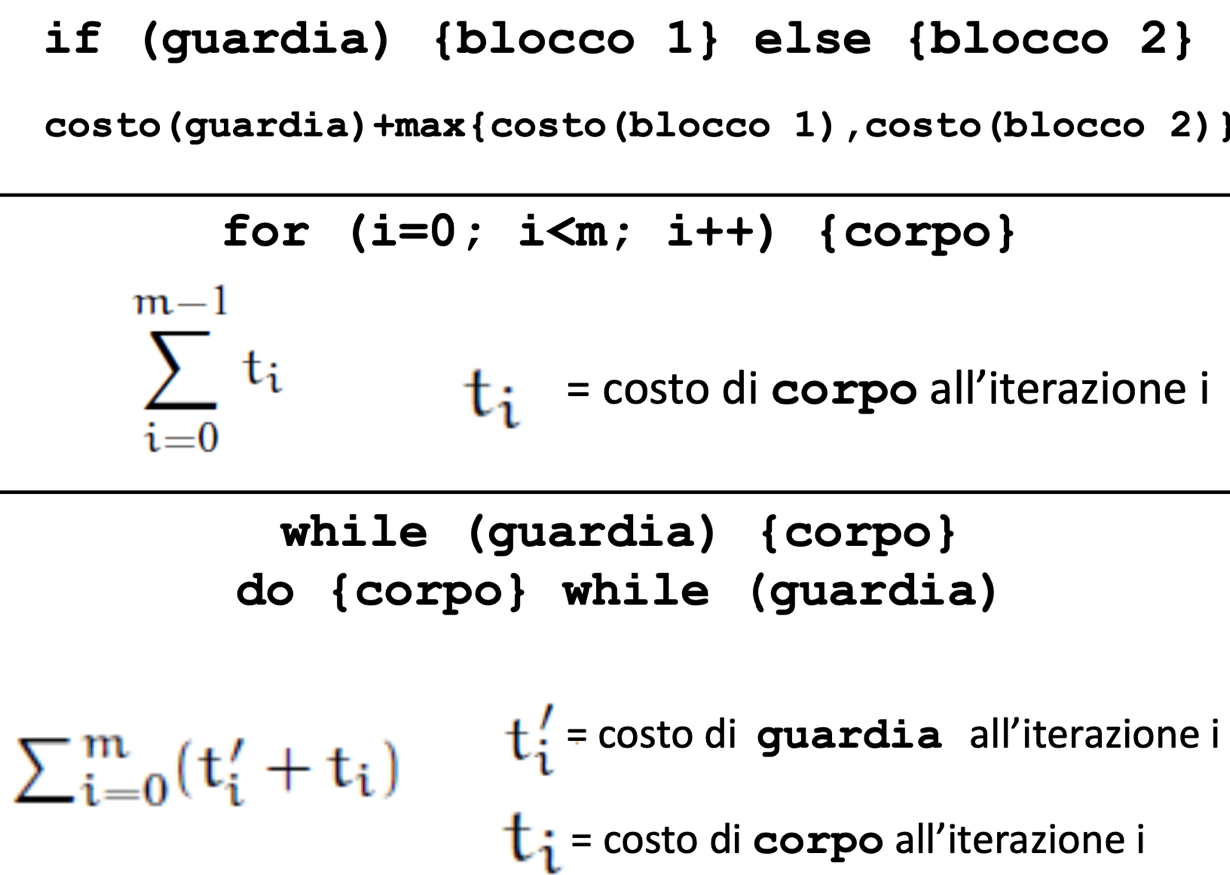
**Complessità (1° parte)**

**Analisi di complessità**

Costo di un algoritmo è in funzione di n (dimensione dei dati in input):

– tempo = numero di operazioni RAM eseguite

– spazio = numero di celle di memoria occupate (escluse quelle per contenere l’input)



**Da cosa è dato il costo di una funzione?**

Il costo di una funzione è dato dal costo del suo corpo (più il passaggio dei parametri)

**Da cosa è dato il costo di una sequenza?**

Il costo di una sequenza di istruzioni è la somma dei costi delle istruzioni nella sequenza

**Caso medio e caso pessimo**

Complessità o costo computazionale f(n) in tempo e in spazio di un problema Π.

* Caso pessimo o peggiore: è il costo massimo tra tutte le istanze di Π aventi dimensione dei dati pari a n.
* Caso medio: costo mediano tra tutte le istanze di Π aventi dimensioni pari a n.

**Complessità temporale**

La **complessità temporale** di un algoritmo è una misura del tempo di esecuzione in funzione del numero di operazioni aritmetiche, logiche, di accesso ai file, letture e scritture in memoria, ecc.

**I^ ipotesi semplificativa:**

* **Tempo proporzionale al numero di operazioni**: Si assume che il tempo impiegato da un algoritmo sia proporzionale al numero di operazioni eseguite, con ogni operazione avente un costo unitario.
* **Indipendenza dalla specifica macchina**: L'analisi della complessità non considera caratteristiche specifiche della macchina su cui l'algoritmo viene eseguito.

**Complessità temporale:**

* Il tempo impiegato per risolvere un problema dipende sia dall’algoritmo utilizzato sia dalla dimensione dei dati a cui si applica l’algoritmo.
* Trovare una funzione precisa che esprima il tempo impiegato in funzione della dimensione dell'input n è spesso molto difficile.

**II^ ipotesi semplificativa:**

* **Comportamento asintotico**: Per analizzare la complessità temporale, è sufficiente considerare il comportamento della funzione quando la dimensione dell'ingresso tende all'infinito.
* **Notazione asintotica**: Si utilizza per descrivere il comportamento della complessità temporale senza calcolare esplicitamente il tempo (o lo spazio) impiegato da un algoritmo, ma piuttosto come questi parametri crescono al variare della dimensione dell'input.

**Notazione Asintotica:**

* **Notazione O-grande (Big O): ​**
* **Notazione Ω-grande**:
* **Notazione Θ-grande**:

**Notazione O-grande (Big O)**

Denotiamo con O(g(n)) l’insieme delle funzioni

O(g(n)) = { f(n) : esistono delle costanti positive c, n0 t.c. 0 ≤ f(n) ≤ cg(n) per ogni n > n0 }

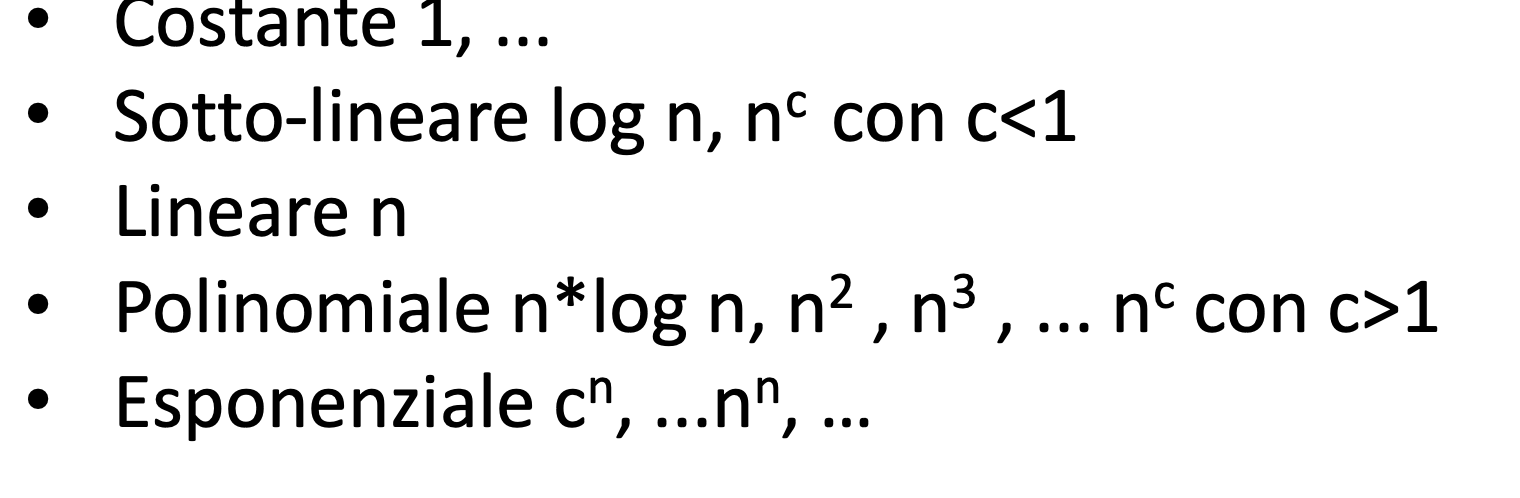
Poiché O(g(n)) è un insieme potremmo scrivere f(n) ∈ O(g(n)) ma di solito scriveremo

Immagine che contiene testo, linea, diagramma, Diagramma

Descrizione generata automaticamentef(n) = O(g(n))

Attraverso la notazione O( ), gli algoritmi vengono divisi in classi di equivalenza, ponendo nella medesima classe tutti quelli la cui complessità asintotica è dello stesso ordine di grandezza.

Si hanno così algoritmi (funzioni) di complessità asintotica di ordine:



**Funzione costante**

* f(n)=c con c costante

Funzione logaritmica

* f(n)=logb n (b>1)
* x=logb (n) 🡨> bx=n
* logb(1)=0

Funzione Lineare

* **f(n)=cn c costante (c non nulla)**

Funzione nlogn

* **f(n)=n log n**

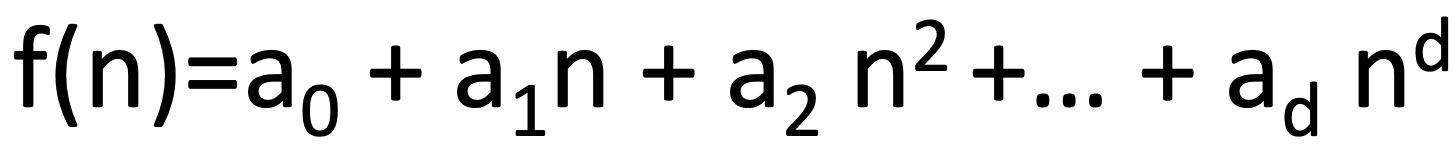
**Funzione esponensiale**

* **f(n)=bn**

Regole

* 1. (ba)c=bac
  2. ba bc=ba+c
  3. ba/bc=ba-c

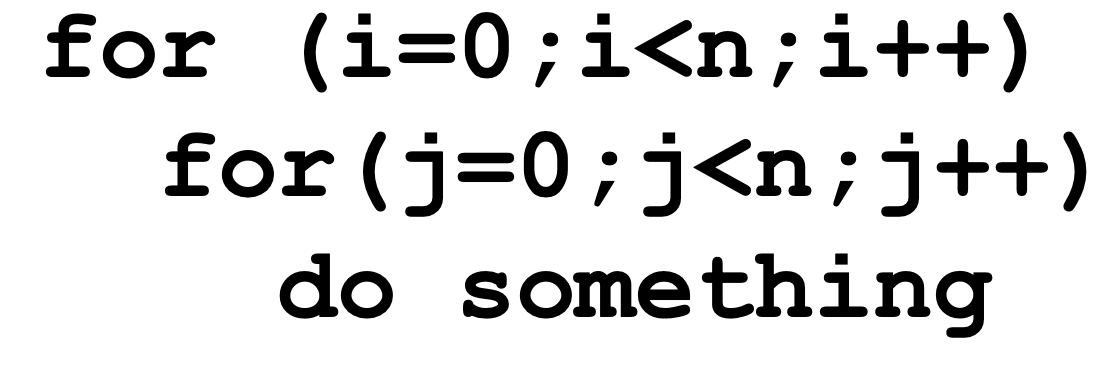
Funzione polinomiale



* Il grado è il valore della potenza più grande con ad diverso da 0.

**Funzione quadratica**

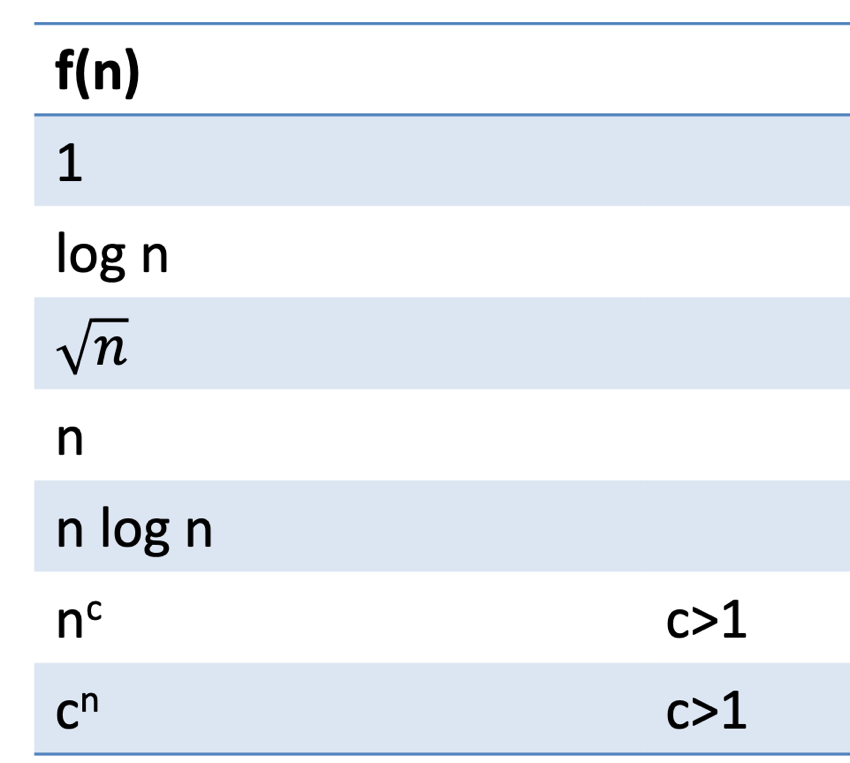
* f(n)=cn2



**Funzione cubica**

Immagine che contiene testo, Carattere, bianco, algebra

Descrizione generata automaticamente

Immagine che contiene linea, diagramma, Diagramma, Parallelo

Descrizione generata automaticamente

La notazione O-grande rappresenta una delimitazione asintotica superiore alla complessità dell'algoritmo, e non la delimitazione asintotica

superiore. Infatti, se T(n) = O(n4), è anche vero che:

T(n) = O(n7) T(n) = O(n4 log n) ecc.

Se per una funzione T(n) sono note più delimitazioni

asintotiche superiori, allora è da preferire quella più piccola.

Esempio

Immagine che contiene testo, Carattere, ricevuta, bianco

Descrizione generata automaticamente

**Notazione Asintotica “Ω-grande”**

Immagine che contiene testo, Carattere, bianco, calligrafia

Descrizione generata automaticamente

Immagine che contiene linea, Diagramma, diagramma, pendio

Descrizione generata automaticamente

La notazione **Ω**-grande definisce un limite inferiore per f(n). Per tutti i valori di n maggiori di n0, il valore di f(n) coincide o sta sopra c g(n)

**Notazione Asintotica ‘’ Θ-grande’’**

Immagine che contiene testo, Carattere, bianco, tipografia

Descrizione generata automaticamente

Immagine che contiene Carattere, tipografia, calligrafia, testo

Descrizione generata automaticamente

**Riassumendo**

Immagine che contiene testo, Carattere, bianco, algebra

Descrizione generata automaticamente

Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente

Delimitazione alla complessità di P

Per un dato problema P consideriamo un algoritmo A che lo risolve:

* Se A prende tempo t(n) diremo che O(t(n)) è un limite superiore
* Se riusciamo a provare che nessun algoritmo può far meglio di t(n) diremo che Ω(t(n)) è limite inferiorte
* A È ottimo se i due limiti coincidono in tal caso la complessità convenzionale del problema è **Θ(t(n)).** O anche se

Immagine che contiene Carattere, testo, bianco, calligrafia

Descrizione generata automaticamente