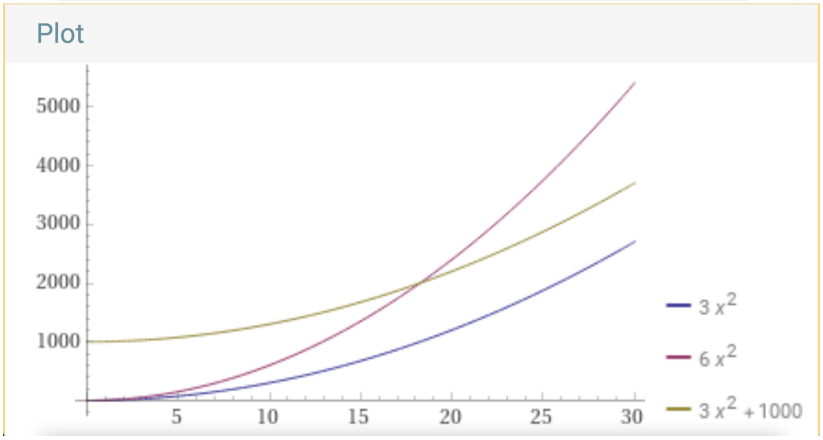
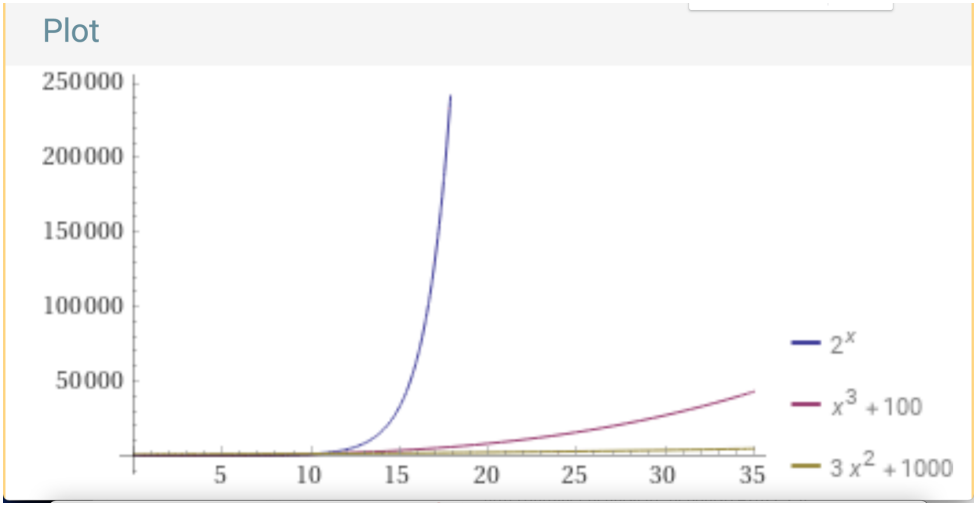
Funzioni quadratiche



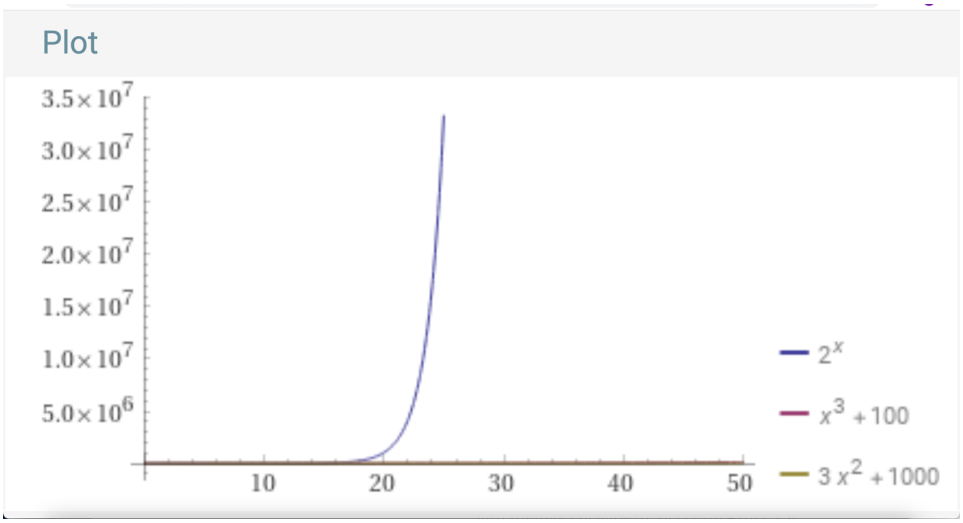
21

Una funzione quadratica, una funzione cubica ed una esponenziale



22

Una funzione quatratica, una funzione cubica ed una esponenziale viste in una range poco piu` grande…



23

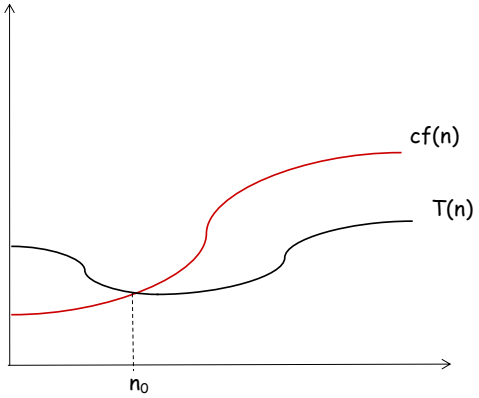
Efficienza asintotica degli algoritmi

- Per input piccoli può non essere corretto considerare solo l'ordine di grandezza ma per input "abbastanza" grandi è corretto farlo
- Esempio: $10n^2+100n+10$
  per $n<10$, il secondo termine è maggiore del primo
  man mano che n cresce il contributo dato dai termini meno significativi diminuisce

24

Ordine asintotico di grandezza

Limiti superiori. $0 \le T(n)$ è $O(f(n))$ se esistono due costanti $c > 0$ ed $n_0 \ge 0$ tali che per tutti gli $n \ge n_0$ si ha $T(n) \le c \cdot f(n)$.



25

Ordine asintotico di grandezza

Limiti superiori. $0 \le T(n)$ è $O(f(n))$ se esistono due costanti $c > 0$ ed $n_0 \ge 0$ tali che per tutti gli $n \ge n_0$ si ha $T(n) \le c \cdot f(n)$.

Esempio: dimostriamo che $3n^2 + 2n$ è $O(n^2)$.

Dobbiamo dimostrare l'esistenza delle costanti $c > 0$ ed $n_0 \ge 0$ tali che $3n^2 + 2n \le cn^2$ per ogni $n \ge n_0$.

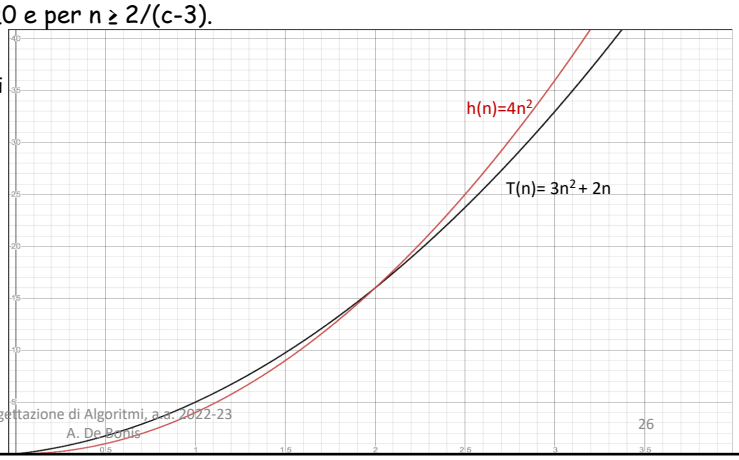
Risolviamo la disequazione $3n^2 + 2n \le cn^2$. Siccome consideriamo solo valori di $n \ge 0$ allora la disuguaglianza **non** puo` valere se $c < 3$ (escludiamo anche $c=3$ perche' la disuguaglianza varrebbe solo per $n=0$). Quindi se scriviamo la disuguaglianza come $(c-3)n^2 - 2n \ge 0$ il coefficiente $c-3$ è > 0 e risolvendo la disequazione di II grado abbiamo che essa è soddisfatta per $n \le 0$ e per $n \ge 2/(c-3)$.

Ci interessano solo i valori di $n \ge 2/(c-3)$.

Poniamo quindi $n_0 = 2/(c-3)$ dove c è una qualsiasi costante >3.

Se ad esempio prendiamo $c=4$ allora $n_0 = 2$.

In figura confrontiamo il grafico di $T(n) = 3n^2 + 2n$ con quello di $h(n)=n^2$



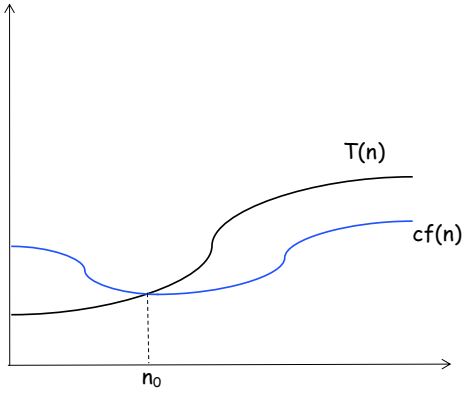
Se prendessimo $c=5$ allora potremmo prendere $n_0 = 1$.

Se prendessimo $c=3.1$ allora potremmo prendere $n_0 = 20$.

26

Ordine asintotico di grandezza

Limiti inferiori. T(n) è $\Omega(f(n))$ se esistono costanti $c > 0$ ed $n_0 \geq 0$ tali che per tutti gli $n \geq n_0$ si ha $T(n) \geq c \cdot f(n) \geq 0$.



27

Ordine asintotico di grandezza

Limiti inferiori. T(n) è $\Omega(f(n))$ se esistono costanti $c > 0$ ed $n_0 \geq 0$ tali che per tutti gli $n \geq n_0$ si ha $T(n) \geq c \cdot f(n) \geq 0$.

Esempio: dimostriamo che $3n^2 + 2n$ è $\Omega(n^2)$.

Dobbiamo dimostrare l'esistenza delle costanti $c > 0$ ed $n_0 \geq 0$ tali $3n^2 + 2n \geq cn^2$ per ogni $n \geq n_0$.

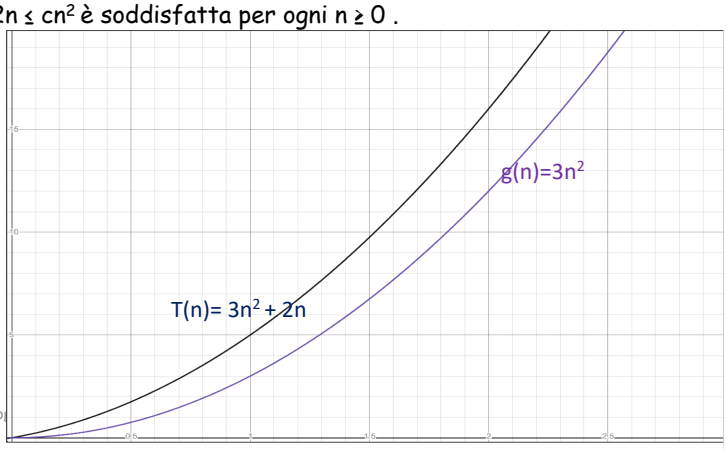
Risolviamo la disequazione $3n^2 + 2n \geq cn^2$. Questa è soddisfatta $\leftrightarrow$ $(3-c)n^2 + 2n \geq 0$.

Se consideriamo valori $c \leq 3$ è evidente che $(3 - c) \geq 0$ e quindi $(3-c)n^2 + 2n \geq 0$ è soddisfatta per ogni $n \geq 0$.

Scegliamo allora c=3 (meglio prendere la costante c quanto piu` grande).

Per c=3, la disequazione di partenza $3n^2 + 2n \leq cn^2$ è soddisfatta per ogni $n \geq 0$.

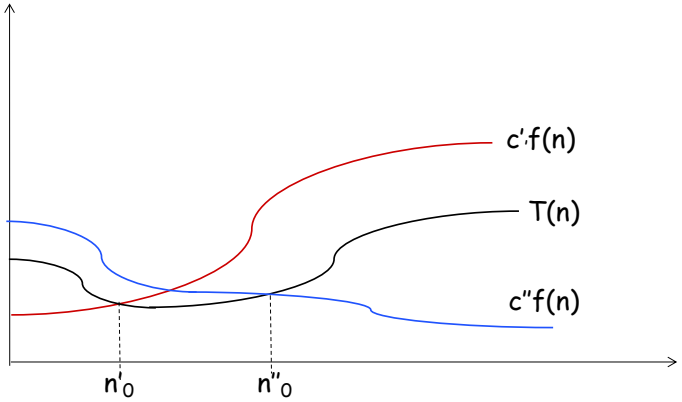
Abbiamo quindi trovato le costanti c ed n_0 (con c=3 ed n_0 =0) per cui si ha che $3n^2 + 2n \geq cn^2$ per ogni $n \geq n_0$.



28

Ordine asintotico di grandezza

Limiti esatti. T(n) è $\Theta(f(n))$ se T(n) sia $O(f(n))$ che $\Omega(f(n))$.

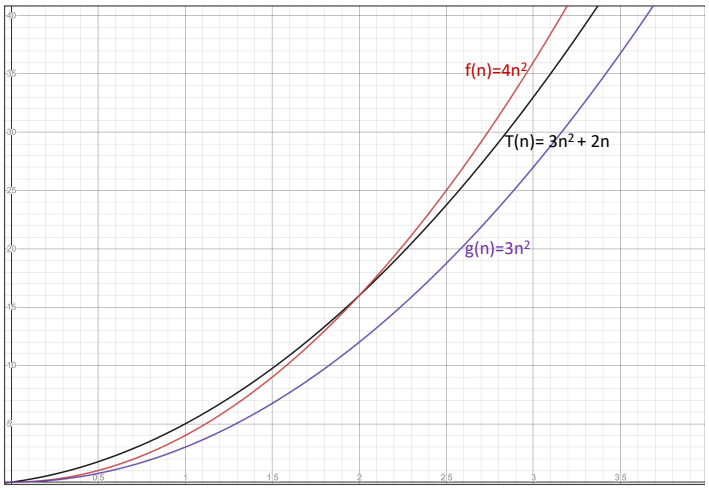


29

Ordine asintotico di grandezza

Limiti esaatti: T(n) è $\Theta(f(n))$ se T(n) sia $O(f(n))$ che $\Omega(f(n))$.

Esempio: abbiamo dimostrato che $3n^2 + 2n$ è sia $O(n^2)$ che $\Omega(n^2)$ → $3n^2 + 2n$ è $\Theta(n^2)$



30

Ordine asintotico di grandezza

- Quando analizziamo un algoritmo miriamo a trovare stime asintotiche quanto più "strette" è possibile
- Dire che InsertionSort ha tempo di esecuzione $O(n^3)$ non è errato ma $O(n^3)$ non è un limite "stretto" in quanto si può dimostrare che InsertionSort ha tempo di esecuzione $O(n^2)$
- $O(n^2)$ è un limite stretto?
  - Sì, perché il numero di passi eseguiti da InsertionSort è an^2+bn+c, con $a>0$, che non solo è $O(n^2)$ ma è anche $\Omega(n^2)$.
  - Si può dire quindi che il tempo di esecuzione di InsertionSort è $\Theta(n^2)$

31

Errore comune

Affermazione priva di senso. Ogni algoritmo basato sui confronti richiede almeno $O(n \log n)$ confronti.

- **Per i lower bound si usa Ω**

Affermazione corretta. Ogni algoritmo basato sui confronti richiede almeno $\Omega(n \log n)$ confronti.

32

Proprietà

Transitività .

- Se $f = O(g)$ e $g = O(h)$ allora $f = O(h)$.
- Se $f = \Omega(g)$ e $g = \Omega(h)$ allora $f = \Omega(h)$.
- Se $f = \Theta(g)$ e $g = \Theta(h)$ allora $f = \Theta(h)$.

Additività .

- Se $f = O(h)$ e $g = O(h)$ allora $f + g = O(h)$.
- Se $f = \Omega(h)$ e $g = \Omega(h)$ allora $f + g = \Omega(h)$.
- Se $f = \Theta(h)$ e $g = \Theta(h)$ allora $f + g = \Theta(h)$.

33

Bound asintotici per alcune funzioni di uso comune

Polinomi. $a_0 + a_1 n + \ldots + a_d n^d$, con $a_d > 0$, è $\Theta(n^d)$.

Dim. $O(n^d)$: dobbiamo trovare due costanti $c>0$ e $n_0 \geq 0$ tali che $a_0 + a_1 n + \ldots + a_d n^d \leq c\, n^d$ per ogni $n \geq n_0$

$a_0 + a_1 n + a_2 n^2 + \ldots + a_d n^d$

$\leq |a_0| + |a_1| n + |a_2| n^2 + \ldots + |a_d| n^d$

$\leq (|a_0| + |a_1| + |a_2| + \ldots + |a_d|)\, n^d$, per ogni $n \geq 1$.

Basta quindi prendere $n_0 = 1$ e $c = |a_0| + |a_1| + \ldots + |a_d|$ (è una costante)

34

Bound asintotici per alcune funzioni di uso comune

Dimostriamo come esercizio che $a_0 + a_1n + \dots + a_dn^d$ è anche $\Omega(n^d)$:

- ricordiamo che stiamo assumendo che a_d e` positivo
- $a_0 + a_1n + \dots + a_dn^d = a_dn^d + \dots + a_1n + a_0 \geq a_dn^d - (|a_0| + |a_1|n + \dots + |a_{d-1}|n^{d-1})$
- Abbiamo appena visto che un polinomio di grado d è $O(n^d)$
  - Ciò implica $|a_0| + |a_1|n + \dots + |a_{d-1}|n^{d-1} = O(n^{d-1})$

    e di conseguenza esistono due costanti $n'_0 \geq 0$ e $c' > 0$

    tali che $|a_0| + |a_1|n + \dots + |a_{d-1}|n^{d-1} \leq c'n^{d-1}$ per ogni $n \geq n'_0$
    - In realta` possiamo prendere $n_0' = 1$ come nella dim. precedente.
- Quindi $a_dn^d - (|a_0| + |a_1|n + \dots + |a_{d-1}|n^{d-1}) \geq a_dn^d - c'n^{d-1}$ **per ogni $n \geq n'_0$**
- da cui $a_0 + a_1n + \dots + a_dn^d \geq a_dn^d - c'n^{d-1}$ **per ogni $n \geq n'_0$**

35

Bound asintotici per alcune funzioni di uso comune

✸ Per dimostrare $a_0 + a_1n + \dots + a_dn^d = \Omega(n^d)$ dobbiamo trovare le costanti $n_0 \geq 0$ e $c > 0$ tali che $a_0 + a_1n + \dots + a_dn^d \geq cn^d$ per ogni $n \geq n_0$

🦆 Nella slide precedente abbiamo dimostrato che esistono due costanti $n'_0 \geq 0$ e $c' > 0$ tali che $a_0 + a_1n + \dots + a_dn^d \geq a_dn^d - c'n^{d-1}$ per ogni $n \geq n'_0 = 1$.

🌵 Da questo momento in poi siccome useremo la disuguaglianza in 🦆 assumeremo che $n \geq 1$.

Il fatto che vale la disuguaglianza in 🦆 ci dice che per trovare due costanti $n_0 \geq 0$ e $c > 0$ tali che $a_0 + a_1n + \dots + a_dn^d \geq cn^d$ per ogni $n \geq n_0$, è sufficiente trovare due costanti $n_0 \geq 0$ e $c > 0$ tali che $a_dn^d - c'n^{d-1} \geq cn^d$ per ogni $n \geq n_0$ (tenendo sempre presente che n_0 dovra` essere ≥ 1)

- Scriviamo la diseguaglianza $a_dn^d - c'n^{d-1} \geq cn^d$ come $(a_d - c)n^d - c'n^{d-1} \geq 0$.
- Stiamo considerando valori di n maggiori di 1 per cui se prendiamo $c < a_d$ (cioe` $a_d - c > 0$) esistono sicuramente valori di n per cui $(a_d - c)n^d - c'n^{d-1} \geq 0$.
- Dividiamo tutto per n^{d-1} e risolviamo la disequazione $(a_d - c)n - c' \geq 0$ che è soddisfatta per $n \geq c'/(a_d - c)$, dove c' è la costante in 🦆 e c è una costante positiva e minore di a_d a nostra scelta.
- Abbiamo quindi trovato le nostre costanti $n_0 = \max\{c'/(a_d - c), 1\}$ e c uguale ad una qualsiasi costante positiva minore di a_d

36

Ordine asintotico di grandezza

Esempio:

$T(n) = 32n^2 + 17n + 32.$

- $T(n)$ è $O(n^2)$, $O(n^3)$, $\Omega(n^2)$, $\Omega(n)$ e $\Theta(n^2)$.
- $T(n)$ non è $O(n)$, $\Omega(n^3)$, $\Theta(n)$ o $\Theta(n^3)$.

37

Tempo lineare: $O(n)$

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

Esempio:
Computazione del massimo. Computa il massimo di n numeri $a_1, \ldots, a_n$.

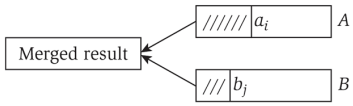
```
max ← a1
for i = 2 to n {
    Se (ai > max)
        max ← ai
}
```

Il problema dell'individuazione del max di n numeri e` Ω(n)
Dim. ogni numero diverso dal massimo deve partecipare ad almeno un confronto in cui risulta < dell'altro elemento → almeno un confronto per ciascuno degli n-1 elementi diversi dal massimo

38

Tempo lineare: $O(n)$

Merge. Combinare 2 sequenze ordinate $A = a_1, a_2, \ldots, a_n$ with $B = b_1, b_2, \ldots, b_m$ in una lista ordinata.




```
i = 1, j = 1
while (i≤n and j≤m) {
   if (a_i ≤ b_j) aggiungi a_i alla fine della lista output e incrementa i
   else aggiungi b_j alla fine della lista output e incrementa j
}
Ciclo per aggiungere alla lista output gli elementi non ancora
esaminati di una delle due liste input
```

Affermazione. Fondere due sequenze ordinate rispettivamente di dimensione n ed m richiede tempo $O(n+m)$.

Dim. Dopo ogni iterazione del while o del ciclo sottostante, la lunghezza dell'output aumenta di 1.

39

Tempo quadratico: $O(n^2)$

Tempo quadratico. Tipicamente si ha quando un algoritmo esamina tutte le coppie di elementi input

Coppia di punti più vicina. Data una lista di n punti del piano $(x_1, y_1), \ldots, (x_n, y_n)$, vogliamo trovare la coppia più vicina.

Soluzione $O(n^2)$. Calcola la distanza tra tutte le coppie di punti.

```
min ← (x_1 - x_2)^2 + (y_1 - y_2)^2
for i = 1 to n {
   for j = i+1 to n {
      d ← (x_i - x_j)^2 + (y_i - y_j)^2
      Se (d < min)
         min ← d
   }
}
```

← Per effettuare i confronti non c'è bisogno di estrarre la radice quadrata

40

Per esercizio, svolgiamo l'analisi dell'algoritmo nella slide precedente.

Il for esterno mi costa: tempo lineare in n + il tempo per eseguire tutte le iterazioni del for interno

Analizziamo il for interno:

Chiamiamo t_i il numero di iterazioni del for interno alla i-esima iterazione del for esterno

Quante volte viene iterato il for interno in totale? Risposta: $t_1+t_2+\ldots+t_n$.

Per ogni i si ha $t_i=(n-i)$

Quindi sommando i t_i per tutte le iterazioni del for esterno ho

$t_1+t_2+\ldots+t_n = (n-1)+(n-2)+\ldots+1+0 = n(n-1)/2$ iterazioni del for interno **IN TOTALE**

siccome la singola esecuzione del corpo del for interno richiede tempo pari ad una costante c
→ il tempo richiesto da tutte le iterazioni del for interno è $c(n(n-1)/2)=\Theta(n^2)$

Tempo totale: $\Theta(n)+\Theta(n^2)=\Theta(n^2)$

41

Tempo cubico: $O(n^3)$

Tempo cubico. Tipicamente si ha quando un algoritmo esamina tutte le triple di elementi.

Esempio:

Disgiunzione di insiemi. Dati n insiemi $S_1, \ldots, S_n$ ciascuno dei quali è un sottoinsieme di {1, 2, …, n}, c'è qualche coppia di insiemi che è disgiunta?

Soluzione $O(n^3)$. Per ogni coppia di insiemi, determinare se i due insiemi sono disgiunti. (Supponiamo di poter determinare in tempo costante se un elemento appartiene ad un insieme)

```
flag = true
for i = 1 to n{      //corpo iterato n volte
 for j = i+1 to n { //corpo iterato n-i volte ad ogni iterazione del for esterno
      foreach elemento p di Si { //corpo iterato al più n volte ad ogni iteraz. for su j
         if p appartiene anche a Sj //supponiamo test richiede ogni volta  O(1)
             flag = false; break;
      }
    If(flag = true)// nessun elemento di Si appartiene a Sj
          riporta che Si  e  Sj sono disgiunti
   }
}
```


42

Un utile richiamo

Alcune utili proprietà dei logaritmi:

1. $\log_a x = (\log_b x) / (\log_b a)$
2. $\log_a(xy) = \log_a x + \log_a y$
3. $\log_a x^k = k \log_a x$

Dalla 1. discende:

4. $\log_a x = 1/(\log_x a)$

Dalla 3. discende:

5. $\log_a(1/x) = -\log_a x$

Dalla 2. e dalla 5. discende:

6. $\log_a(x/y) = \log_a x - \log_a y$

43

Regole per la notazione asintotica

$d(n) = O(f(n)) \Rightarrow ad(n) = O(f(n)),\ \forall$ costante $a > 0$

Es.: $\log n = O(n) \Rightarrow 7 \log n = O(n)$

$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n)+e(n) = O(f(n)+g(n))$

Es.: $\log n = O(n), \sqrt{n} = O(n) \Rightarrow \log n + \sqrt{n} = O(n)$

$d(n) = O(f(n)), e(n) = O(g(n)) \Rightarrow d(n)e(n) = O(f(n)g(n))$

Es.: $\log n = O(\sqrt{n}), \sqrt{n} = O(\sqrt{n}) \Rightarrow \sqrt{n} \log n = O(n)$

$d(n) = O(f(n)), f(n) = O(g(n)) \Rightarrow d(n) = O(g(n))$

Es.: $\log n = O(\sqrt{n}), \sqrt{n} = O(n) \Rightarrow \log n = O(n)$

$f(n) = a_d n^d + \cdots a_1 n + a_0 \Rightarrow f(n) = O(n^d)$

Es.: $5n^7 + 6n^4 + 3n^3 + 100 = O(n^7)$

$n^x = O(a^n),\ \forall$ costanti $x > 0, a > 1$ Es.: $n^{100} = O(2^n)$

44

Regole per la notazione asintotica

- Le prime 5 regole nella slide precedente valgono anche se sostituiamo O con Ω o con Θ
- Dimostriamo per esercizio la 1.
- $d(n)=O(f(n)) \rightarrow ad(n)=O(f(n)$, per a costante positiva
- Dim.
- $d(n)=O(f(n)) \rightarrow$ esistono due costanti $c'>0$ ed $n'_0 \geq 0$ t.c. $d(n) \leq c'f(n)$ per ogni $n \geq n'_0$
- Moltiplicando entrambi i membri della disuguaglianza per a il verso della disuguaglianza rimane invariato perche' $a>0$.
- Quindi si ha $ad(n) \leq ac'f(n)$ per ogni $n \geq n'_0$.
- Abbiamo quindi trovato le costanti c ed n_0 per cui vale la definizione di $O(f(n))$
  - basta infatti porre $c=ac'$ ed $n_0 = n'_0$
  - NB: ac' e` una costante > 0 perche' sia a che c' sono costanti >0.

45

Dimostriamo l'additivita`

- $d(n)=O(f(n))$ ed $e(n)=O(g(n)) \rightarrow d(n)+e(n)=O(f(n)+g(n))$
- Dim.

1. $d(n)=O(f(n)) \rightarrow$esistono due costanti $c'>0$ ed $n'_0 \geq 0$ t.c. $d(n) \leq c'f(n)$ per ogni $n \geq n'_0$
2. $e(n)=O(g(n)) \rightarrow$esistono due costanti $c''>0$ ed $n''_0 \geq 0$ t.c. $e(n) \leq c''g(n)$ per ogni $n \geq n''_0$

- la 1 $\rightarrow d(n) \leq c'f(n)$ per ogni $n \geq n'_0$, la 2 $\rightarrow e(n) \leq c''g(n)$ per ogni $n \geq n''_0$
- e di conseguenza, $d(n)+e(n) \leq c'f(n) + c''g(n) \leq \max\{c',c''\}f(n) + \max\{c',c''\}g(n) = \max\{c',c''\} (f(n)+g(n))$ per ogni n maggiore di n'_0 e n''_0
- Ponendo $c= \max\{c',c''\}$ ed $n_0=\max\{n'_0, n''_0\}$, possiamo quindi affermare che
- $d(n)+e(n) \leq c(f(n)+g(n))$ per ogni $n \geq n_0$ e cio` implica $d(n)+e(n)=O(f(n)+g(n))$

46