L’efficienza degli algoritmi

Introduzione

CORRETTEZZA e COMPLESSITA'

algoritmo e struttura dati

validità semantica (correttezza)

Risolve il problema dato?

È un "buon" algoritmo?

efficienza (complessita')

codifica

UN ESEMPIO: il problema del calcolo del quoziente

Definizione "matematica": *Il quoziente* ***intero*** *di un numero* ***m*** *diviso un numero* ***n*** *(notazione* ***m/n****) e' il piu' grande numero* ***q*** *tale che* $\boldsymbol{q \cdot n \leq m}$.

Per esempio :

$$83/5 = 16$$

perchè $16 \cdot 5 = 80$ e $17 \cdot 5 = 85$.

Quindi 16 e' il *piu' grande* numero q tale che $q \cdot 5 \leq 83$.

La definizione ci dice che cosa si vuole, ma non come lo si trova.

Un algoritmo per calcolare il quoziente

$$
\begin{array}{r|l}
83 & 5 \\ \cline{2-2}
5 & 16 \\ \cline{1-1}
33 & \\
30 & \\ \cline{1-1}
3 &
\end{array}
$$

Non ci sono più riporti. Ho finito. Il risultato è 16

Un altro algoritmo per calcolare il quoziente

3 > 5 ?

Alla fine otteniamo

Segno: X X X X X X X X

X X X X X X X X

Contando i segni ricavo: 16.

Confrontiamo i due algoritmi

Supponiamo di dover dividere

12.235.917.612 : 3

- Col primo algoritmo ci vogliono 18 operazioni "elementari" (sottrazioni o moltiplicazioni di un numero)
- Col secondo ci vogliono 4.078.639.204 sottrazioni.

Un *essere umano* (1 operazione al secondo) impiega 18 secondi nel primo caso, 129 anni (circa) nel secondo

Un *elaboratore* (10.000.000 operazioni al secondo) impiega 1.8 μ-secondi nel primo caso e circa 20 minuti nel secondo.

Che conclusioni ne possiamo trarre?

1. **Certi algoritmi possono richiedere tempo eccessivo anche per un computer.**

2. **Algoritmi diversi per eseguire lo stesso compito possono avere prestazioni molto diverse.**

RICAPITOLANDO

Per svolgere un certo compito (per esempio la divisione) su un computer bisogna avere:

- un algoritmo per farlo
- che richieda un tempo di esecuzione ragionevole

Ma. . .

Non sempre gli algoritmi per eseguire un dato compito esistono e, anche se esistono, non sempre richiedono un tempo ragionevole.

DOMANDE

1. **Che cosa significa “tempo di esecuzione ragionevole”?**

2. **Come possiamo misurare la “bonta’” (“efficienza”) di un algoritmo?**

CONSIDERIAMO UN ALTRO ESEMPIO

Problema: dati due numeri naturali, x e y, calcolare x^y

Prima soluzione

$$x^y = \underbrace{x * x * \ . \ . \ . \ * x}_{\text{y volte}}$$

{ y > 0 }

```
Esp-prod (x, y)
    z ← 1
    ty ← y
    while ty > 0 do
        z ← z * x
        ty ← ty - 1
    return z
```

{ $z = x^y$ }

Quanto è “buono” questo algoritmo?

Le operazioni che vengono eseguite il numero maggiore di volte sono le operazioni effettuate nel corpo del ciclo:

$z \leftarrow z * x$ e $ty \leftarrow ty - 1$.

Contiamo il numero di moltiplicazioni.

Quanto maggiore è il valore di y, tanto maggiore è il numero di moltiplicazioni da effettuare.

Raddoppiando y, il numero di moltiplicazioni raddoppierà

⬇

Il numero di prodotti aumenta *linearmente* con y

Tentiamo di migliorare l'algoritmo

Per cercare di capire, consideriamo
un problema più semplice: $y = 2^n$, cioè y è una potenza di 2

Per calcolare x elevato 2^n potremmo successivamente elevare x al quadrato

Esempio: $y = 16 = 2^4$

$$x^0 = 1$$
$$x^1 * x^1 = x^2$$
$$x^2 * x^2 = x^4$$
$$x^4 * x^4 = x^8$$
$$x^8 * x^8 = x^{16}$$

n.b. $4 = \log_2 16$
$6 = \log_2 64$

Quando $y = 16 = 2^4$, si eseguono 4 moltiplicazioni
Quando $y = 64 = 2^6$, si eseguono 6 moltiplicazioni

Un algoritmo migliore per il problema piu' semplice

$\{ y = 2^n \}$

```
Esp-Pdue (x, y)
    z ← x
    ty ← 1
    while ty < y do
        z ← z * z
        ty ← 2 * ty
    return z
```

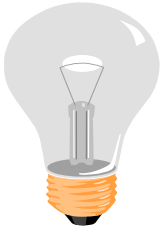
$\{ z = x^y \}$

Il numero di moltiplicazioni aumenta linearmente con l'esponente n

Il numero di moltiplicazioni aumenta *logaritmicamente* con y

Torniamo al problema originale

L'esponente y e` un numero naturale qualunque.



Ma qualunque numero naturale puo` essere espresso come somma di potenze di due !

$16/_{10} = 10000/_2$

$13/_{10} = 8 + 4 + 1 = 2^3 + 2^2 + 2^0 = 1101/_2$

$x^{13} = x^{8+4+1} = x^8 * x^4 * x^1 = x^{1*2^3} * x^{1*2^2} * x^{0*2^1} * x^{1*2^0}$

$y /_{10} = b_n\, b_{n-1} \ldots b_1\, b_0 /_2$

$\boxed{n = \log_2 y}$

$y = b_n * 2^n + \ldots + b_1 * 2^1 + b_0 * 2^0$

$x^y = x^{b_n*2^n + \ldots + b_1*2^1 + b_0*2^0} = x^{b_n*2^n} * \ldots * x^{b_1*2^1} * x^{b_0*2^0}$

Un algoritmo migliore per il problema originale

$$x^y = x^{b_n * 2^n} * \ldots * x^{b_1 * 2^1} * x^{b_0 * 2^0}$$

Seconda soluzione

$\{ y > 0 \}$

```
Esp-prodPdue (x, y)
    ty ← y
    i ← 1
    z ← 1
    while ty > 0 do
        if ty mod 2 = 1
            then z ← z * Esp-Pdue (x, i)
        i ← 2*i
        ty ← ty div 2
    return z
```

⇐ (log i) +

$\{ z = x^y \}$

Contiamo le moltiplicazioni fatte dal secondo algoritmo

Per ogni cifra binaria $b_j = 1$, $j + 1$ moltiplicazioni

caso migliore: solo $b_n = 1$

vengono eseguite $2*(n + 1)$ moltiplicazioni

$n = \log_2 y$ →

Il numero di moltiplicazioni aumenta *logaritmicamente* con y

caso peggiore: tutte le cifre binarie sono 1

le moltiplicazioni eseguite sono: $(n+1)+p$ con $p =$

$$\sum_{j=0}^{n} j + 1 = \sum_{j=1}^{n+1} j = (n+1)(n+2)/2$$

↓ $n = \log_2 y$

Il numero di prodotti aumenta in modo *quadratico nel logaritmo* di y

E' ancora giusto contare le moltiplicazioni?

- Si, perche' all'interno di ogni ciclo e' eseguita almeno una moltiplicazione
- Se il ciclo interno (quello all'interno dell'algoritmo **Esp-Pdue**) non contenesse moltiplicazioni, allora non sarebbe giusto…

NOTA: quando si confrontano due (o piu') algoritmi, le operazioni da contare potrebbero non essere le stesse (potrebbe cioe' essere necessario contare operazioni diverse per ciascun algoritmo).

Un algoritmo ancora migliore

Possiamo pero` evitare di calcolare tutte le volte x elevato alla potenza di due.

Terza soluzione

$\{ y > 0 \}$

```
Esp (x, y)
    ty ← y
    z ← 1
    PR ← x
    while ty > 0 do
        if ty mod 2 = 1
            then z ← z * PR
        ty ← ty div 2
        PR ← PR * PR
    return z
```

$\{ z = x^y \}$

Contiamo le moltiplicazioni fatte dal terzo algoritmo

$$x^y = x^{b_n * 2^n} * \ldots * x^{b_1 * 2^1} * x^{b_0 * 2^0}$$

Quante moltiplicazioni?

- Per ogni b_i (0 o 1) una moltiplicazione: PR ← PR * PR
- Per ogni $b_i = 1$ una moltiplicazione in più: z ← z * PR

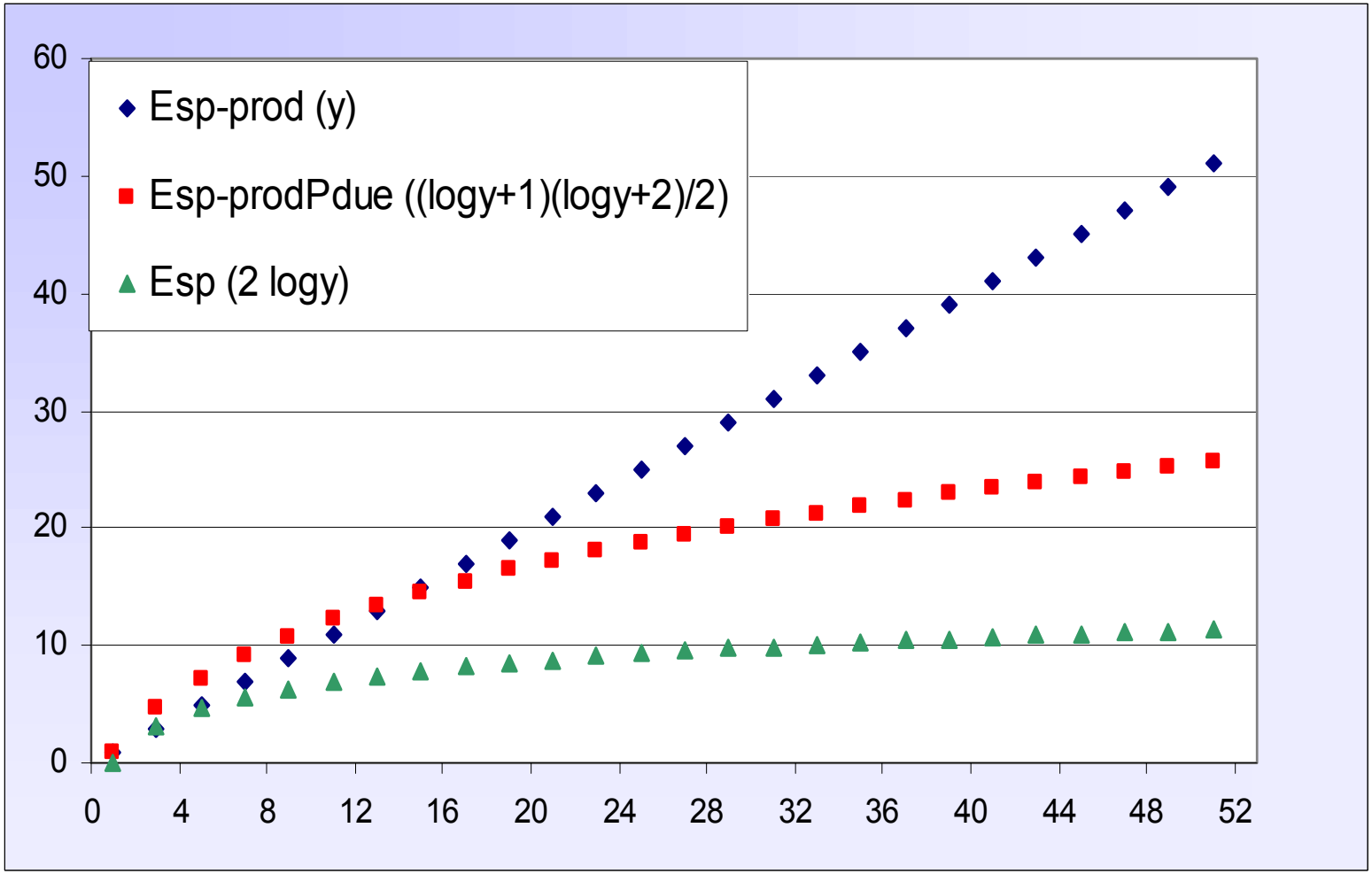
caso migliore: solo $b_n = 1$
vengono eseguite (log y) + 1 moltiplicazioni

Il numero di prodotti aumenta *logaritmicamente* con y

caso peggiore: tutte le cifre binarie sono 1
vengono eseguite 2 log y moltiplicazioni

Confrontiamo i tre algoritmi

Confronto tra il numero di moltiplicazioni dei tre algoritmi nel caso peggiore



ESERCIZIO DA FARE

- Dimostrare (utilizzando le asserzioni) la corretezza degli algoritmi:
  1. Esp-prod
  2. Esp-Pdue
  3. Esp-prodPdue
  4. Esp

ANNOTATE (SCRIVENDO SULLA STAMPA DEI LUCIDI) IL CODICE CON LE ASSERZIONI CHE NE DIMOSTRANO LA CORRETTEZZA. *In questo modo vi sara' piu' facile seguire il ragionamento sulla loro complessita'!!!*

INTERMEZZO 1: una favola araba

Un giorno un califfo chiese ad un giovane che gli aveva reso un servizio che ricompensa volesse. Il giovane rispose:

" Dammi tanti chicchi di riso quanti se ne possono porre sulle caselle di una scacchiera ideale, mettendo un chicco sulla prima casella, due sulla la seconda, quattro sulla terza e così via raddoppiando ogni volta in numero di chicchi"

Il califfo accettò con entusiamo, sembrandogli quella richiesta poca cosa. Ma se ne pentì ben presto….

Si può vedere con un facile calcolo che i chicchi di riso richiesto dal giovane (2^{64}-1) sarebbero sufficienti a tappezzare una strada larga circa 500 Km e lunga quanto la distanza dalla terra alla luna…

INTERMEZZO 2: Pinocchio in banca

Il signor Pinocchio ha messo da parte uno zecchino d'oro, che vuole saggiamente investire in buoni del Tesoro.

Individui come lui incontrano inevitabilmente i consulenti finanziari:

Il gatto e la volpe gli propongono un investimento presso la Tree Bank, che assicura interessi incomparabilmente più alti.

"Con la nostra opzione l'Istituto aggiunge al suo versamento la somma di cinque zecchini l'anno.
Questa formula è enormemente vantaggiosa in vista della consistenza, diciamo così, non proprio elevatissima della cifra a sua disposizione".

"D'altra parte se i Buoni del Tesoro fruttassero anche il 20%, il suo zecchino si moltiplicherebbe solo per 1,2 ogni anno. Sta a lei decidere"

Pinocchio trascorre la notte facendo conti:

	Versamento	1anno	2anni	3anni	. . .	10 anni
Tree Bank	1	6	11	16	. . .	51
Buoni del Tesoro	1	1,2	1,44	1,73	. . .	6,19

E ha scritto anche le due equazioni che regolano la crescita degli zecchini:

$1 + 5k$, se investe nella Tree Bank

$(1,20)^k$ se investe in Buoni del Tesoro

k = numero di anni

Pinocchio accetta pertanto il suggerimento dei due consulenti

Pinocchio ha scelto bene?

	10 anni	20 anni	25 anni	30 anni	40 anni	50 anni
Tree Bank	51	101	126	151	201	251
Buoni del Tesoro	6,19	38,34	95,40	237,37	1469,77	9100,44

La legge esponenziale finisce inesorabilmente per vincere.

Quando il suo valore supera quello di un'altra funzione non esponenziale, se ne stacca con rapidità strabiliante.

Tutto però è relativo: se l'investimento è per meno di 25/26 anni conviene la Tree Bank!

Una tabella rivelatrice

Se un algoritmo è eseguito in tempo T(n), dove n è la dimensione dell'istanza in input, quanto tempo richiede risolvere un'istanza del problema di dimensione n se un'istanza di dimensione 1 richiede 1 μs (10^{-6} s)?

T(n)	n=10	n=20	n=50	n=100	n=1000
n lgn	33 μs	86 μs	282 μs	664 μs	10 ms
n^2	100 μs	400 μs	3 ms	10 ms	1 s
n^5	100 ms	3 s	5 min	3 hr	32 yr
2^n	1ms	1 s	36 yr	$4x10^{16}$yr	10^{287} yr
n!	4 s	77147yr	10^{51} yr	10^{144} yr	10^{2554} yr

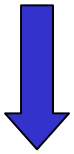
Un'altra tabella rivelatrice

Se un algoritmo è eseguito in tempo T(n), qual è la massima istanza del problema che può essere risolta in un minuto ?

T(n)	Il computer più veloce del mondo nel 1990	Un computer 1000 volte piu' veloce
n lgn	1.5×10^3 miliardi	1×10^6 miliardi
n^2	8 milioni	260 milioni
n^5	570	2300
2^n	46	56
n!	16	18

RICAPITOLANDO

Classificare gli algoritmi a seconda della loro efficienza



avere modi precisi per analizzarli

Misure di efficienza:

- spazio utilizzato durante l'esecuzione degli algoritmi
- tempo di calcolo degli algoritmi (e delle operazioni sulle strutture dati)

Siamo interessati a determinare la dipendenza dello spazio utilizzato e del tempo di calcolo dalla dimensione dell'input.

In generale il tempo aumenta con la dimensione dell'input.

DOMANDE

1. Qual è il modo corretto per misurare il tempo di calcolo ?
2. Qual è il modo corretto per misurare lo spazio utilizzato ?

ANCORA UN ESERCIZIO DA FARE

- Scrivete in pseudo-codice i due algoritmi (iterativi) illustrati all'inizio di questi lucidi e dimostratene la correttezza con il metodo asserzioni.
  Suggerimento: *sviluppate ciascun algoritmo assieme alla prova di correttezza (ovvero: prima di iniziare a scrivere lo pseudo-codice formulate precisamente l'idea sulla quale si basa l'algoritmo, questa idea sara' proprio l'invariante).*