Gli algoritmi

La soluzione algoritmica di problemi

- Algoritmo, il concetto fondamentale e centrale dell'informatica
  - da AL-KHOWARIZMI (825 dc). Una procedura per risolvere un problema matematico in un numero finito di passi che implicano frequenti ripetizioni di un'operazione.
  - In senso lato, una procedura che, eseguita passo a passo, risolve un problema.
    - Dal dizionario Webster

Esempi

- Algoritmi (o procedure):
  - per calcolare il Massimo Comun Divisore
  - per costruire modellini di aerei (espressi nei fogli di istruzioni)
  - per azionare la lavatrice
  - per suonare una melodia al piano (espressa in un insieme di simboli negli spartiti)

Algoritmo di Euclide

- Dati due numeri interi e positivi *m* e *n*, calcola il più grande intero che li divide entrambi

1. dividere *m* per *n*, e sia *r* il resto della divisione (con $0 \le r < n$)
2. se r = 0 allora la risposta è *n* e STOP
3. porre $m \leftarrow n$ e $n \leftarrow r$ e ripetere il passo 1

Verifica empirica

- **Funziona davvero? Il ciclo dei 3 passi termina sempre?**
- **Proviamo con $m \leftarrow 119$ e $n \leftarrow 544$:**

1. 119 / 544 dà resto $r \leftarrow 119$
2. se $r \neq 0$ allora
3. poniamo $m \leftarrow 544$ e $n \leftarrow 119$. Torniamo al passo 1.

1. 544 / 119 dà resto $r \leftarrow 68$
2. se $r \neq 0$ allora
3. poniamo $m \leftarrow 119$ e $n \leftarrow 68$ Torniamo al passo 1.

1. 119 / 68 dà resto $r \leftarrow 51$
2. se $r \neq 0$ allora
3. poniamo $m \leftarrow 68$ e $n \leftarrow 51$ Torniamo al passo 1.

1. 68 / 51 dà resto $r \leftarrow 17$
2. se $r \neq 0$ allora
3. poniamo $m \leftarrow 51$ e $n \leftarrow 17$ Torniamo al passo 1.

1. 51 / 17 dà resto $r \leftarrow 0$
2. se $r = 0$ allora il MCD = $n = 17$

Criteri per essere algoritmo

- Un insieme ***finito*** di istruzioni che dà luogo a una sequenza ***finita*** di operazioni
  1. Deve terminare dopo un numero finito di passi
  2. Ogni passo deve essere definito precisamente
  3. Deve operare su dati di ingresso in un insieme ben specificato
  4. Deve produrre un output che abbia la relazione specificata con i dati di ingresso
  5. Tutte le operazioni dell'algoritmo devono essere ***di base*** e poter essere eseguite in un tempo ***finito***

Il criterio di finitezza per MCD

1. è soddisfatto
   - La sequenza dei resti è una successione di numeri interi decrescenti che termina con 0
   - poiché `0 <= r < n`

Algoritmo vs procedura di calcolo

- Il processo di ripetizione di cicli deve terminare
- Non tutte le procedure soddisfano questo requisito
  - Si parla allora di *procedure di calcolo*
  - Trovare l'intero positivo `x` → `x + 7 = q`
  - Dati p e q interi positivi trovare un intero positivo `x` → `qx + p = 100`
  - Dati `p` e `q` interi positivi trovare un intero `x` → $x^2 = p^2 + q^2$

Gli altri criteri per MCD

2. Occorre definire precisamente divisione intera e resto per numeri positivi
   - stabilito che i numeri siano positivi all'inizio rimangono tali e quindi il criterio 2. è soddisfatto
3. I dati d'ingresso sono i due numeri positivi
4. Il dato calcolato è il MCD per la coppia
5. Si usano solo divisioni intere, test su numeri positivi e assegnamenti a variabili
   - Conosciamo procedure per eseguire queste operazioni e quindi il criterio 5. è soddisfatto

Trovare algoritmi

- La ricerca di algoritmi è stata una grande parte del lavoro dei matematici nei secoli
- Gli algoritmi rendono il lavoro più semplice
- La loro esecuzione non richiede la comprensione dei principi su cui si fonda
- Il computer è un esecutore di algoritmi

Criteri di bontà degli algoritmi

- Lunghezza del tempo di esecuzione espresso in termini del numero di passi da eseguire
- Occupazione di spazio di memoria
- Adattabilità dell'algoritmo a situazioni diverse
- Semplicità
- Modularità
- Eleganza

Analisi dell'algoritmo di Euclide

- E' possibile fissato n stabilire il numero di passi necessari per l'esecuzione al variare di m?
- E' possibile dare una risposta alla domanda
  - Si, il problema è ben posto e poiché fissato n il resto r < n, basta calcolare il numero di passi per `m = 1` … `m = n` e calcolare la media T_n
- Ora si studia come varia T_n al variare di n
  - Si può dimostrare che $T_n \sim k \log n$
    - dove $k = 12 \log 2 / \pi^2$

Tempi d'esecuzione

n	n	log n	n^2	n^3	n^4	n^{10}	2^n
10	0.01 µs	0.03 µs	0.1 µs	1 µs	10 µs	10 s	1 µs
20	0.02 µs	0.9 µs	0.4 µs	8 µs	160 µs	2.84 h	1 ms
30	0.03 µs	0.03 µs	0.9 µs	27 µs	810 µs	6.83 g	1 s
40	0.04 µs	0.21 µs	1.6 µs	64 µs	2.56 ms	121.36 g	18.3 m
50	0.05 µs	0.28 µs	2.5 µs	125 µs	6.25 ms	3.1 a	13 g
10^2	0.1 µs	0.66 µs	10 µs	1 ms	100 ms	3171 a	$4\ 10^{13}$ a
10^3	1 µs	9.96 µs	1 ms	1 s	16.67 m	$3.17\ 10^{13}$ a	$32\ 10^{283}$ a
10^4	10 µs	130.03 µs	100 ms	16.67 m	115.7 g	$3.17\ 10^{23}$ a	
10^6	1 ms	19.92 ms	16.67 m	31.71 a	$3.17\ 10^7$ a	$3.17\ 10^{43}$ a	

Tempi calcolati su un computer che esegue 10^9 istruzioni al secondo

Massima istanza

- Dato un certo computer consideriamo la **massima istanza** del problema che può essere risolto in **1 h**
- Quindi supponiamo di aver un computer 100 o 1000 volte più veloce
- Cerchiamo il miglioramento dovuto alla tecnologia sulla massima istanza computabile in 1 h

Il miglioramento tecnologico

f(n)	computer di riferimento	computer 100 volte + veloce	computer 1000 volte + veloce
n	N_1	100 N_1	1000 N_1
n^2	N_2	10 N_2	31.6 N_2
n^3	N_3	4.64 N_3	10 N_3
n^5	N_4	2.5 N_4	3.9 N_4
2^n	N_5	N_5 + 6.64	N_5 + 9.97
3^n	N_6	N_6 + 4.19	N_6 + 6.29

Considerazioni

- Non bastano i miglioramenti tecnologici
- Occorrono ***buoni*** algoritmi
- Gli algoritmi ***polinomiali*** sono ovviamente molto più apprezzabili di quelli ***esponenziali***

Teoria degli algoritmi

- Gli algoritmi possibilmente sono corretti ...
- Dimostrazioni di **correttezza** degli algoritmi vs **verifica empirica**
- **Complessità** degli algoritmi
  - Se gli algoritmi sono corretti, sono *buoni*?
  - Criteri di bontà: efficienza nell'uso delle risorse sia di tempo di calcolo, sia di occupazione di memoria
- Problemi intrinsecamente difficili
- **Decidibilità**. Dato un problema esiste un algoritmo per risolverlo?