



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

AA 2006/07 © Alberti 2 Programmazione 2. Algoritmi

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)

AA 2006/07 © Alberti 3 Programmazione 2. Algoritmi

Algoritmo di Euclide

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

1. dividere m per n , e sia r il resto della divisione (con $0 \leq r < n$)
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

AA 2006/07 © Alberti 4 Programmazione 2. Algoritmi

Algoritmo di Euclide - 2

- L' *intelligenza* necessaria per trovare la soluzione del problema è tutta codificata nell'algoritmo.
- Chiunque sappia comprendere ed eseguire le operazioni che costituiscono l'algoritmo di Euclide, può calcolare il MCD tra 2 numeri.

AA 2006/07 © Alberti 5 Programmazione 2. Algoritmi

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 = $n = 17$**

AA 2006/07 © Alberti 6 Programmazione 2. Algoritmi

Definizione di algoritmo

Insieme **finito** e ordinato di passi **eseguibili e non ambigui**, che definiscono un processo che **termina**.

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**

AA 2006/07
© Alberti

7

Programmazione
2. Algoritmi

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 $0 \leq r < n$

AA 2006/07
© Alberti

8

Programmazione
2. Algoritmi

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 $x \rightarrow x + 7 = q$
 - Generare la sequenza dei numeri primi e ordinarla in modo crescente
 - Dati p e q interi positivi trovare un intero $x \rightarrow x^2 = p^2 + q^2$

AA 2006/07
© Alberti

9

Programmazione
2. Algoritmi

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 numeri interi positivi
4. Il dato calcolato è il MCD per la coppia data
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

AA 2006/07
© Alberti

10

Programmazione
2. Algoritmi

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

AA 2006/07
© Alberti

11

Programmazione
2. Algoritmi

Programma

- È l'espressione di un algoritmo in un linguaggio che l'esecutore è in grado di comprendere senza bisogno di ulteriori spiegazioni

AA 2006/07
© Alberti

12

Programmazione
2. Algoritmi

Algoritmi e programmi

- Un algoritmo è un oggetto astratto, concettuale. Un programma è un'espressione concreta dell'algoritmo
- Lo stesso algoritmo può essere espresso in differenti linguaggi, in base agli esecutori ai quali è destinato
- La scrittura del programma è una fase successiva all'individuazione dell'algoritmo per risolvere un determinato problema

AA 2006/07
© Alberti

13

Programmazione
2. 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

AA 2006/07
© Alberti

14

Programmazione
2. Algoritmi

Analisi dell'algoritmo di Euclide

- E' possibile fissato n stabilire il numero di passi necessari per l'esecuzione al variare di m ?
- Il problema è ben posto e la risposta è affermativa
 - Fissato n , per il resto r vale $r < n$, quindi basta calcolare il numero di passi per la sequenza finita $m = 1 \dots m = n$ 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 $k = 12 \log 2 / \pi^2$

AA 2006/07
© Alberti

15

Programmazione
2. Algoritmi

Tempi d'esecuzione

| n | n | n log ₂ 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 | 0.09 μs | 0.4 μs | 8 μs | 160 μs | 2.84 h | 1 ms |
| 30 | 0.03 μs | 0.15 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 ²³ 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 10⁹ istruzioni al secondo

AA 2006/07
© Alberti

16

Programmazione
2. Algoritmi

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

AA 2006/07
© Alberti

17

Programmazione
2. Algoritmi

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 |

AA 2006/07
© Alberti

18

Programmazione
2. Algoritmi

Considerazioni

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

AA 2006/07
© Alberti

19

Programmazione
2. Algoritmi

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?

AA 2006/07
© Alberti

20

Programmazione
2. Algoritmi