Algoritmi

Preliminari

Introduzione

La nozione di algoritmo riguarda qualsiasi campo in cui si possano descrivere sequenze di operazioni finalizzate allo svolgimento di un compito. In tal senso una ricetta di cucina é un algoritmo eseguibile da un cuoco, uno spartito musicale é un algoritmo eseguibile da un musicista.

Esempio: pasta con le sarde

Ingredienti:
300g di bucatini, 350g di sarde fresche, 200g di finocchio selvatico, 30g di pinoli, 30g di uvetta sultanina, 4 filetti d'acciuga sottosale, farina, 1/2 cipolla, olio di oliva, sale.
Preparazione:
Fate lessare in acqua leggermente salata il finocchio selvatico. Dopo 10 minuti scolatelo e tritatelo. In un tegame fate rosolare, a fuoco basso, la cipolla tritata con due cucchiai di olio e le acciughe diliscate. Unite il finocchio tritato, i pinoli e l'uvetta ammollata e strizzata. Fate cuocere la salsa a fuoco basso e con il coperchio per 15 minuti. Preparate le sarde: pulitele, lavatele, asciugatele. Dividetele a metà lasciandole attaccate dalla parte del dorso, infarinatele e fatele friggere in olio bollente. Passatele in un piatto con della carta assorbente e salatele leggermente. Cuocete i bucatini al dente, scolatele e conditele con metà della salsa. Ungete una pirofila da forno, disponetevi uno strato di pasta e, sopra questa, uno strato di sarde fritte e qualche cucchiaiata di salsa. Procedete così e terminate mettendo qualche cucchiaio di salsa. Riscaldate il forno a 200C. Infornate la pasta per 10 minuti e servite.

Algoritmo di Euclide per il MCD

Consideriamo il problema del calcolo del massimo comun divisore di due numeri:

1. Siano x e y due numeri.
2. Calcola il resto della divisione di x e y.
3. Se il resto é diverso da zero, ricomincia dal passo 2 utilizzando come x il valore attuale di y e come y il valore del resto, altrimenti prosegui al passo successivo.
4. Il massimo comun divisore é uguale al valore attuale di y.

Osservazioni 1

- Chiunque sappia comprendere ed eseguire le operazioni può cucinare la pasta con le sarde o calcolare il massimo comun divisore.
- L'esecutore può non conoscere quello che l'algoritmo calcola.
- Un algoritmo é un insieme ordinato di passi, istruzioni, eseguibili e non ambigui, che definiscono la soluzione di un problema.
- Talune istruzioni manipolano le variabili che rappresentano delle quantità che nell'esecuzione dell'algoritmo possono cambiare valore. L'algoritmo precedente manipola 3 quantità contenute nelle variabili x, y, resto.
- I passi che costituiscono un algoritmo devono essere eseguibili: l'algoritmo deve essere un metodo effettivo.

```
1. Crea un elenco di tutti i numeri primi.
2. Ordina l'elenco in modo decrescente.
3. Preleva il secondo elemento dall'elenco risultante.
```

Osservazioni 2

- Ad ogni passo del procedimento di calcolo deve essere possibile applicare al più una istruzione. Non sono quindi ammesse scelte non deterministiche.
- Non si pone alcun limite al tempo necessario ad eseguire il calcolo, né allo spazio di memoria richiesto, basta che siano finiti.
- La nozione di algoritmo focalizza l'attenzione sul metodo di soluzione e non sul linguaggio o sul modo utilizzato per descrivere tale metodo.
- Il linguaggio naturale non é un buon linguaggio per descrivere algoritmi soprattutto a causa della sua ambiguità.

```
1. Siano x e y due numeri.
2. Calcola il modulo di x e y.
```

Esempio 1

Problema: stabilire se un dato anno è o meno bisestile

1. Sia A l'anno di cui si vuole stabilire la bisestilità.
2. Se A diviso 400 dà resto zero vai al passo 6.
3. Se A diviso 100 dà resto zero vai al passo 5.
4. Se A diviso 4 dà resto zero vai al passo 6.
5. A non è bisestile, vai al passo 7.
6. A è bisestile.
7. Fine.

Eseguiamo l'algoritmo per stabilire la bisestilità dell'anno 2001:

Esempio 2

Problema: dato un numero N, intero e maggiore di uno, calcolare tutti i suoi divisori.

1. Sia N il numero di cui si vogliono calcolare i divisori.
2. Poni I uguale a 1.
3. Se N diviso I dà resto zero, allora I è un divisore di N.
4. Incrementa I di uno.
5. Se I*2 minore o uguale a N allora vai al passo 3.
6. N è un divisore di N;
7. Fine.

Eseguiamo l'algoritmo per calcolare i divisori di 6

La programmazione strutturata

Programmazione strutturata

- Metodologia introdotta agli inizi degli anni settanta.
- L'esecutore é guidato alla sequenza di esecuzione opportuna, tra tutte quelle possibili, mediante tre strutture di controllo fondamentali:
  - **sequenza**
    permette di eseguire le istruzioni secondo l'ordine in cui sono scritte
  - **selezione**
    permette di scegliere l'esecuzione di un blocco di istruzioni tra due possibili in base al valore di una condizione
  - **iterazione**
    permette di ripetere l'esecuzione di una o più istruzioni in base al valore di una condizione.

Programmazione strutturata

- Tutti i programmi esprimibili tramite istruzioni di salto (*goto*) o diagrammi di flusso (*flow-chart*) possono essere riscritti utilizzando esclusivamente le tre strutture di controllo fondamentali.
- L'impiego di queste strutture migliora la leggibilità dei programmi prodotti:
  - ogni struttura di controllo ha un solo punto di ingresso e un solo punto d'uscita
  - il flusso di esecuzione é evidente dalla struttura del codice

Sequenza

- Le istruzioni sono eseguite nello stesso ordine in cui compaiono nel programma, cioé secondo la sequenza in cui sono scritte.

Esempio: somma di due numeri

```
leggi i numeri a, b
calcola a+b
scrivi il risultato
```

Selezione

- Schema (sintassi):

```
SE condizione
   ALLORA
      blocco1
   ALTRIMENTI
      blocco2
FINESE
```

- Esecuzione (semantica):
  - Viene valutata condizione:
    - se risulta vera, vengono eseguite le istruzioni del *blocco1*
    - se risulta falsa, vengono eseguite quelle del blocco2
  - L'esecuzione procede con l'istruzione che segue immediatamente la fine del costrutto di selezione (`FINESE`)

Selezione

- Nel caso in cui il *blocco2* non contenga alcuna istruzione, si può utilizzare il seguente schema semplificato:

```
SE condizione
   ALLORA
      blocco1
FINESE
```

- Esecuzione (semantica):
  - Viene valutata *condizione*:
    - se risulta *vera*, vengono eseguite le istruzioni del *blocco1*, quindi l'esecuzione prosegue direttamente dalla prima istruzione che segue il costrutto di selezione
    - se risulta *falsa*, l'esecuzione prosegue direttamente dalla prima istruzione che segue il costrutto di selezione

Esempio

Calcolo della divisione tra due numeri controllando che il divisore sia diverso da zero:

```
leggi il dividendo e il divisore
SE il divisore é diverso da zero
   ALLORA
      calcola dividendo/divisore
      scrivi il risultato
   ALTRIMENTI
      scrivi “errore: divisione per zero”
FINESE
```

Esempio: calcolo delle radici di $ax^2 + bx + c = 0$

leggi i valori dei parametri a, b, c
calcola il discriminante
SE il discriminante è minore di zero
  ALLORA
    scrivi “nessuna soluzione reale”
  ALTRIMENTI
    SE il discriminante è uguale a zero
      ALLORA
        calcola $\frac{-b}{2a}$
        scrivi “Due soluzioni coincidenti: ”, il risultato
      ALTRIMENTI
        scrivi “Due soluzioni: ”,
        calcola $\frac{-b-\sqrt{b^2-4ac}}{2a}$
        scrivi il risultato
        calcola $\frac{-b+\sqrt{b^2-4ac}}{2a}$
        scrivi il risultato
    FINESE
FINESE

Iterazione (schema 1)

- Schema (sintassi):

  ESEGUI
  blocco
  QUANDO *condizione*

- Esecuzione (semantica):
  - Anzitutto viene eseguito il *blocco* di istruzioni. Quindi si valuta la *condizione*:
    - se risulta vera, si ripete eseguendo nuovamente il *blocco* e valutando ancora la *condizione*
    - se risulta falsa, si prosegue con l'istruzione scritta dopo il costrutto iterativo
- Osservazioni
  - il *blocco* é eseguito *almeno una volta*
  - l'esecuzione del costrutto iterativo termina quando la *condizione* *diventa* falsa.

Esempio: somma dei primi 100 numeri interi

Senza utilizzare la formula di Gauss ($\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$).

Calcolo iterativo:

poni il valore della somma a zero
inizia a considerare il numero 1
ESEGUI
    aggiungi alla somma il numero che stai considerando
    considera il numero successivo
QUANDO il numero che stai considerando non supera 100
scrivi la somma

Iterazione (schema 2)

- Schema (sintassi):

  `QUANDO` *condizione* `ESEGUI`
  blocco
  `RIPETI`

- Esecuzione (semantica):
  - Prima di tutto viene valutata la *condizione*:
    - se risulta vera, allora si esegue il *blocco* e si valuta nuovamente la *condizione*.
    - se risulta falsa, si passa a eseguire l'istruzione che segue il costrutto iterativo, cioé l'istruzione che segue la parola `RIPETI`.
- Osservazioni
  - il *blocco* può essere eseguito anche zero volte
  - l'esecuzione del costrutto iterativo termina quando la *condizione* *diventa* falsa.

Simulazione

Il comportamento dello schema `QUANDO...RIPETI` può essere simulato combinando lo schema della selezione con lo schema `ESEGUI...QUANDO...`, come segue:

```
SE condizione
   ALLORA
      ESEGUI
         blocco
      QUANDO condizione
FINESE
```

Variabili, tipi e assegnamenti

Variabili e assegnamenti

- Una variabile è un *contenitore* preposto a contenere dei valori.
- Un'istruzione di assegnamento

  variabile ← espressione

  ha la seguente semantica

  (1) viene calcolato il valore dell'espressione scritta a destra del simbolo ←

  (2) il risultato ottenuto è assegnato alla variabile (quindi posto nel contenitore) il cui nome è scritto a sinistra del simbolo ←, eliminando l'eventuale valore precedentemente contenuto prima.

- Molti linguaggi, tra cui purtroppo anche C, utilizzano per l'assegnamento il simbolo =, usato comunemente per indicare l'uguaglianza.

Esempio

Per eseguire l'istruzione

$$x \leftarrow y + z$$

- si valuta l'espressione y+z, recuperando i valori presenti in y e z e facendone la somma
- si pone il risultato nel contenitore di cui x è il nome dopo avere eliminato il valore precedentemente presente

Esempio

Per eseguire l'istruzione

$$k \leftarrow k + 1$$

- si valuta l'espressione k+1, recuperando il valore di k (3) e sommandogli 1
- si pone il risultato nel contenitore di cui k è il nome dopo avere eliminato il valore precedentemente presente

Tipo

- Il tipo di una variabile specifica
  - la classe di valori che questa può assumere
  - l'insieme delle operazioni che possono essere effettuate su di essa.
- Ad esempio una variabile x di tipo intero
  - può assumere come valori solo numeri interi
  - su di essa possono essere effettuate soltanto le operazioni consentite per i numeri interi.

Dichiarazione delle variabili

- Molti linguaggi richiedono di dichiarare le variabili utilizzate nel programma indicandone il tipo.
  - Alcuni linguaggi (ad esempio Pascal) richiedono che le variabili siano dichiarate tutte all'inizio del programma.
  - Alcuni linguaggi richiedono che siano dichiarate prima del loro utilizzo (ad esempio C).

- Questo
  - accresce la leggibilità dei programmi
  - diminuisce la possibilità di errori

Successore di un numero

variabili a: numero intero

leggi a
a ← a+1
scrivi “Il successore è : ”, a

Valore assoluto

variabili a: numero intero

leggi a
```
SE a < 0
  ALLORA
    a ← -a
FINESE
```
scrivi “Valore assoluto : ”, a

Esempio: calcolo delle radici di $ax^2 + bx + c = 0$

variabili a, b, c, discriminante, x, x1, x2: numeri reali

leggi a, b, c
discriminante $\leftarrow$ b^2 - 4·a·c
SE discriminante < 0
  ALLORA
    scrivi “nessuna soluzione reale”
  ALTRIMENTI
    SE discriminante == 0
      ALLORA
        x $\leftarrow$ - b / (2·a)
        scrivi “Due soluzioni coincidenti: ”, x
      ALTRIMENTI
        x1 $\leftarrow$ (- b - $\sqrt{\text{discriminante}}$) / (2 · a)
        x2 $\leftarrow$ (- b + $\sqrt{\text{discriminante}}$) / (2 · a)
        scrivi “Due soluzioni: ”, x1, x2
    FINESE
FINESE

Problemi, Algoritmi ed Esecuzione

Problemi, Algoritmi ed Esecuzione

- É opportuno sottolineare la differenza tra problema, algoritmo che lo risolve ed esecuzione dell'algoritmo.
- Un problema non specifica la soluzione.
- Lo stesso problema può essere risolto da più algoritmi. Ad esempio, il problema del massimo comun divisore tra due numeri é risolto anche dal seguente algoritmo:

```
variabili x, y, z: numeri interi

leggi x, y
SE x > y
  ALLORA
    z ← y
  ALTRIMENTI
    z ← x
FINESE
QUANDO z non divide x o non divide y ESEGUI
  z ← z - 1
RIPETI
scrivi "Il MCD è : ", z
```

Bisestilità

```
variabili a: numero intero

leggi a
SE a%400 = 0
  ALLORA
    scrivi "L'anno ", a , " è bisestile"
  ALTRIMENTI
    SE a%100 = 0
      ALLORA
        scrivi "L'anno ", a , " non è bisestile"
      ALTRIMENTI
        SE a%4 = 0
          ALLORA
            scrivi "L'anno ", a , " è bisestile"
          ALTRIMENTI
            scrivi "L'anno ", a , " è bisestile"
        FINESE
    FINESE
FINESE
```

Osservazioni

- Chi esegue un algoritmo deve sapere in ogni momento qual é il valore delle variabili;
- come vedremo i calcolatori hanno una memoria in cui conservano il valore di ogni variabile;
- quando un algoritmo viene eseguito da un esecutore umano piuttosto che ricordare il valore delle variabili si usa una tabella, detta **trace table**, la quale ha tante colonne quante sono le variabili dell'algoritmo.
- dopo avere eseguito una istruzione si riporta su una nuova riga della trace table il valore di tutte le variabili.
- é buona norma introdurre nella trace table una colonna in cui si scrive l'istruzione che e' stata eseguita.
- **Esercizio:** eseguite l'algoritmo del lucido precedente con A=1996 compilando passo dopo passo la trace table.

Divisori di un numero

```
variabili n, i: numeri interi

leggi n
i ← 1
ESEGUI
  SE n/i = 1
    ALLORA
      scrivi i, " è un divisore di ", n
  FINESE
  i ← i + 1
QUANDO (i·2)≤n
scrivi n, " è un divisore di ", n
```

Minimo di 3 Numeri

variabili a, b, c, m: numeri interi

```
leggi a, b, c
SE a<b
  ALLORA
    m ← a
  ALTRIMENTI
    m ← b
FINESE SE c<m
  ALLORA
    m ← c
FINESE
scrivi "Il minimo tra ", a , b, c, " è ", m
```

Minimo di 3 Numeri (2)

```
variabili a, b, c, m: numeri interi

leggi a, b, c
m ← a
SE m>b
  ALLORA
    m ← b
  ALTRIMENTI
    SE m>c
      ALLORA
        m ← c
    FINESE
FINESE scrivi "Il minimo tra ", a , b, c, " è ", m
```

Calcolo degli Interessi

Per un buono postale di 1000 euro viene dato un interesse annuo del 3%. Calcolare il valore del buono dopo 5 anni.

```
variabili a: numero reale

a ← 1000
a ← a+a·0,03
a ← a+a·0,03
a ← a+a·0,03
a ← a+a·0,03
a ← a+a·0,03
scrivi "Il valore del buono dopo 5 anni é ", a
```

Questa soluzione é corretta. Si osservi però che lo stesso schema di operazione é ripetuta 5 volte. Se il problema avesse chiesto di calcolare il valore del buono dopo 50 anni seguendo questo schema avremmo scritto un algoritmo lunghissimo.

Calcolo degli Interessi (2)

Dato il valore V di un buono postale e l'interesse annuo I, calcolare il valore del buono dopo X anni.

variabili `i, v, k`: numeri interi, `x`: numero reale

```
leggi v, i, x
k ← v
QUANDO x>0 ESEGUI
  k ← k+k·i
  x ← x-1
RIPETI
scrivi "Il valore del buono è ", k
```

Alcuni Commenti (1)

- Un algoritmo é caratterizzato da un insieme di istruzioni che permettono di risolvere un problema;
- ha dei dati in ingresso;
- produce risultati in uscita;
- le istruzioni vengono eseguite una dietro l'altra a partire dalla prima;
- talvolta questa sequenza può essere alterata da due costrutti linguistici:
  - **Selezione**, che vincola l'esecuzione di un gruppo di istruzioni alla verità di una condizione
  - **Iterazione**, che mentre una condizione é vera ripete l'esecuzione di un gruppo di istruzioni.

Alcuni Commenti (2)

- Si osservi che nella pratica comune noi eseguiamo le operazioni una dopo l'altra. Talvolta una o più azioni devono essere eseguite solo in determinate circostanze (**selezione**) oppure una o più azioni devono essere eseguite un certo numero di volte (**iterazione**).
  Ad esempio:
- durante una telefonata ad un numero verde potremo sentire: "...se volete prenotare un posto per un volo nazionale premete il tasto 1 ed attendete che l'operatore vi risponda, se volete sapere il vostro punteggio 1000 miglia premete 2, attendete il segnale acustico poi inserite il vostro pin,...";
- una prescrizione medica potrebbe riportare "...fino a quando permane lo stato influenzale assumere 1 pastiglia, 1 cucchiaio di sciroppo e bere un bicchiere di acqua e zucchero ogni 8 ore...".

Calcolo della Retta dei Minimi Quadrati

- Supponiamo che dati i punti $(x_1, y_1), \ldots, (x_n, y_n)$ vogliamo determinare la retta $y = C_0 + C_1 x$ che minimizza la distanza dai punti dati.
- I coefficienti C_0 e C_1 possono essere determinati come segue:

$$C_0 = \frac{\sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$C_1 = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}$$

Calcolo della Retta dei Minimi Quadrati

- Le due formule costituiscono un algoritmo utilizzabile da chiunque conosca il linguaggio matematico ed in particolare il significato di $\sum$.

variabili x, y, a, b, c, d, c0, c1: numeri reali, i, n: numeri interi

leggi n
i ← 0, a ← 0, b ← 0, c ← 0, d ← 0
QUANDO i≤n ESEGUI
  leggi x, y
  a ← a+x
  b ← b+y
  c ← c+x·x
  d ← d+x·y
  i ← i+1
RIPETI
c0 ← (b·c-a·d)/(n·c-a·a)
c1 ← (n·d-a·b)/(n·c-a·a)
scrivi "I coefficienti della retta dei minimi quadrati sono: ", c0, c1

Calcolo del Fattoriale

variabili `ris, n`: numero reale

leggi `n`
`ris` $\leftarrow$ `1`
`QUANDO n`$\leq$`2 ESEGUI`
  `ris` $\leftarrow$ `ris`$\cdot$`n`
  `n` $\leftarrow$ `n-1`
`RIPETI`
scrivi "Il valore del fattoriale è: ", ris

Calcolo della Media Mobile a 3 Termini

Scrivere un algoritmo che dati n numeri $a_1, \ldots, a_n$, calcoli la media mobile a 3 termini.

variabili `x, y, z, t`: numeri reali, `i, n`: numeri interi

```
leggi n
i ← 1
QUANDO i≤(n-2) ESEGUI
  leggi x, y, z
  t ← (x+y+z)/3
  i ← i+1
  scrivi "Termine di posto", i " : ", t
RIPETI
```

Esercizi

- Scrivere un algoritmo che dati due numeri a e b calcoli la soluzione dell'equazione $ax + b = 0$;
- scrivere un algoritmo che dati tre numeri a, b e c calcoli la soluzione dell'equazione $ax^2 + bx + c = 0$, tenendo conto che l'esecutore non sa calcolare la radice quadrata di numeri negativi;
- scrivere un algoritmo che sommi due interi a e b nell'ipotesi che l'esecutore sappia fare solo incrementi e decrementi di una unità;
- scrivere un algoritmo che dato un intero N calcoli la somma dei primi N numeri dispari;
- scrivere un algoritmo che dati n numeri $a_1, \ldots, a_n$, calcoli la media mobile a 3 termini.

Complessità computazionale di un algoritmo (1)

Eseguire certi algoritmi sembra più *semplice* che eseguirne altri ad esempio:

- dividere due numeri interi é più *difficile* che sommarli;
- l'algoritmo dell'incremento é più *veloce* dell'algoritmo del fattoriale.

Inoltre il tempo per eseguire un algoritmo dipende dalla lunghezza dei dati. Ad esempio sommare due numeri interi di 5000 cifre é più *lento* che sommare due numeri interi di 5 cifre.

Complessità computazionale di un algoritmo (2)

Per valutare la bontà di un algoritmo di solito si considerano due fattori:

- il tempo necessario ad eseguire l'algoritmo;
- lo spazio necessario a contenere i dati per portare a termine l'esecuzione dell'algoritmo.

Entrambi devono essere valutati in funzione della dimensione dei dati

in ingresso.

Esempio

Supponendo di poter disporre di una macchina che esegue un passo elementare in un nanosecondo, ovvero un miliardo di passi elementari al secondo si ha:

Complessità	$n = 10$	$n = 100$	$n = 10^3$	$n = 10^6$
n	$10\eta s$	$0,1\mu s$	$1\mu s$	ms
$n \cdot log_2 n$	$33,2\eta s$	$0,6\mu s$	$9,9\mu s$	$19,9ms$
n^2	$0,1\mu s$	$10\mu s$	$1ms$	$\approx 16minuti$
n^3	$1\mu s$	$1ms$	$1s$	$\approx 32anni$
2^n	$1\mu s$	$\approx 10^{13}anni$	∞	∞