Progetto e analisi di algoritmi

Roberto Cordone

DTI - Università degli Studi di Milano

Polo Didattico e di Ricerca di Crema

Tel. 0373 / 898**089**

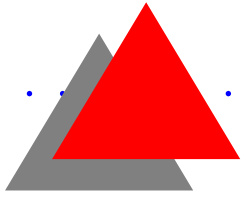
E-mail: **cordone@dti.unimi.it**

Ricevimento: **su appuntamento**

Web page: **http://www.dti.unimi.it/~cordone**

Lezioni: **Martedì dalle 14.00 alle 16.00**

Giovedì dalle 11.00 alle 13.00

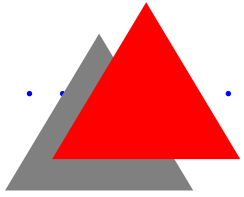


Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
147	102

Come fareste voi?



Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
147	102

Così fa l'algoritmo di Euclide:

- $|a - b| = 45$
- $\min(a, b) = 102$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
45	102

Così fa l'algoritmo di Euclide:

- $|a - b| = 57$
- $\min(a, b) = 45$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
45	57

Così fa l'algoritmo di Euclide:

- $|a - b| = 12$
- $\min(a, b) = 45$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
45	12

Così fa l'algoritmo di Euclide:

- $|a - b| = 33$
- $\min(a, b) = 12$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
33	12

Così fa l'algoritmo di Euclide:

- $|a - b| = 21$
- $\min(a, b) = 12$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

a	b
21	12

Così fa l'algoritmo di Euclide:

- $|a - b| = 9$
- $\min(a, b) = 12$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

$$a \quad b$$

$$9 \quad 12$$

Così fa l'algoritmo di Euclide:

- $|a - b| = 3$
- $\min(a, b) = 9$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

$$a \quad b$$

$$9 \quad 3$$

Così fa l'algoritmo di Euclide:

- $|a - b| = 6$
- $\min(a, b) = 3$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

$$a \quad b$$

$$6 \quad 3$$

Così fa l'algoritmo di Euclide:

- $|a - b| = 3$
- $\min(a, b) = 3$

Massimo Comun Divisore

Si voglia calcolare il MCD di due numeri interi a e b

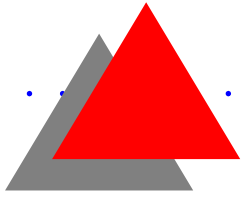
$$a \quad b$$

$$3 \quad 3$$

Così fa l'algoritmo di Euclide:

- $MCD = a = b = 3$!

Riuscite a scriverlo? E chi ci dice che funziona?

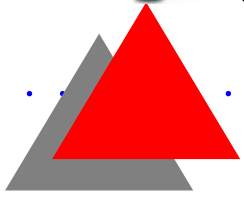


Algoritmo di Euclide

- Sostituire al maggiore fra a e b la loro differenza
- Fermarsi quando $a = b$
- Il risultato è il *MCD* dei due numeri originari

Ora stendetene lo pseudocodice
(istruzioni strutturate in modo formale)

- senza dichiarazioni di tipo (quelle complesse vanno nel testo di accompagnamento)
- senza la gestione della memoria
- con le sole inizializzazioni essenziali per la correttezza



Algoritmo di Euclide

Euclide(a, b)

While $a \neq b$ **do**

  If $a > b$

    then $a := a - b$;

    else $b := b - a$;

Return a;

È una semplice struttura iterativa

1. continuare o fermarsi?
2. che fare?

Algoritmo di Euclide

Euclide(a, b)

While $a \neq b$ **do**

If $a > b$

then $a := a - b$;

else $b := b - a$;

Return a;

È una semplice struttura iterativa

1. continuare o fermarsi? (condizione di arresto)
2. che fare?

Algoritmo di Euclide

Euclide(a, b)

While $a \neq b$ **do**

If $a > b$

then $a := a - b$;

else $b := b - a$;

Return a;

È una semplice struttura iterativa

1. continuare o fermarsi?
2. che fare? (transizione di stato)

Ricerca in un vettore ordinato

Si voglia determinare se un dato vettore A ordinato per valori crescenti contiene o no un dato numero x

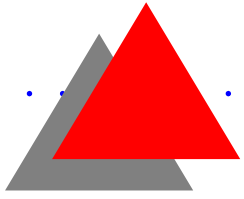
$$s = 1 \qquad\qquad d = 9$$

$$A = \left[\begin{array}{ccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$A' = \left[\begin{array}{ccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$x = 24$$

Come fareste voi?



Ricerca in un vettore ordinato

Si voglia determinare se un dato vettore A ordinato per valori crescenti contiene o no un dato numero x

$$s = 1 \qquad\qquad m = 5 \qquad\qquad d = 9$$

$$A = \left[\begin{array}{cccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$A' = \left[\begin{array}{cccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$x = 24$$

Così fa l'algoritmo di ricerca binaria:

- $m = \lfloor (s + d)/2 \rfloor = 5$ e $x \leq A[m]$
- $A' = A[s..m] = [2 \quad 4 \quad 21 \quad 24 \quad 28]$

Ricerca in un vettore ordinato

Si voglia determinare se un dato vettore A ordinato per valori crescenti contiene o no un dato numero x

$$s = 1 \quad m = 3 \qquad d = 5$$

$$A = \left[\begin{array}{cccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$A' = \left[\begin{array}{cccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$x = 24$$

Così fa l'algoritmo di ricerca binaria:

- $m = \lfloor (s + d)/2 \rfloor = 3$ e $x > A[m]$
- $A' = A[m + 1..d] = [24 \quad 28]$

Ricerca in un vettore ordinato

Si voglia determinare se un dato vettore A ordinato per valori crescenti contiene o no un dato numero x

$$s = m = 4 \; d = 5$$

$$A = \left[\begin{array}{ccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$A' = \left[\begin{array}{ccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$x = 24$$

Così fa l'algoritmo di ricerca binaria:

- $m = \lfloor (s + d)/2 \rfloor = 4$ e $x \leq A[m]$
- $A' = A[s..m] = [24]$

Ricerca in un vettore ordinato

Si voglia determinare se un dato vettore A ordinato per valori crescenti contiene o no un dato numero x

$$s = m = d = 4$$

$$A = \left[\begin{array}{cccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$A' = \left[\begin{array}{cccccccccc} 2 & 4 & 21 & 24 & 28 & 31 & 32 & 47 & 93 \end{array} \right]$$

$$x = 24$$

Così fa l'algoritmo di ricerca binaria:

- $s = d$ e $x = A[s]$
- Il numero x compare in A!

Ricerca binaria

RicercaBinaria(A, s, d)

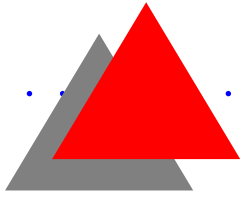
While $s < d$ **do**

$m := \lfloor \frac{s+d}{2} \rfloor$;

If $x \leq A[m]$ **then** $d := m$;

else $s := m + 1$;

If $s = d$ **and** $x = A[s]$ **then Return** *Sì* **else Return** *No*;

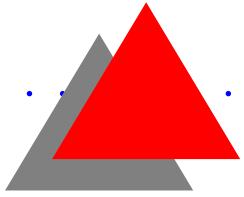


Insertion Sort

Si vogliano ordinare gli elementi di un dato vettore A per valori non decrescenti

$$A = \begin{bmatrix} 5 & 2 & 8 & 4 & 7 & 1 & 3 & 6 \end{bmatrix}$$

Vi ricordate la procedura *Insertion Sort*? L01-ISort.pps



Insertion Sort

InsertionSort(A, n)

$j := 2$;

While $j \leq n$ **do**

$x := A[j]$;

$i := j - 1$; { Scala tutti gli elementi maggiori di $A[j]$ }

While $i \geq 1$ and $A[i] > x$ do

$A[i+1] := A[i]$; $i := i - 1$;

$A[i+1] := x$ { Inserisce il vecchio $A[j]$ al posto giusto }

$j := j + 1$;

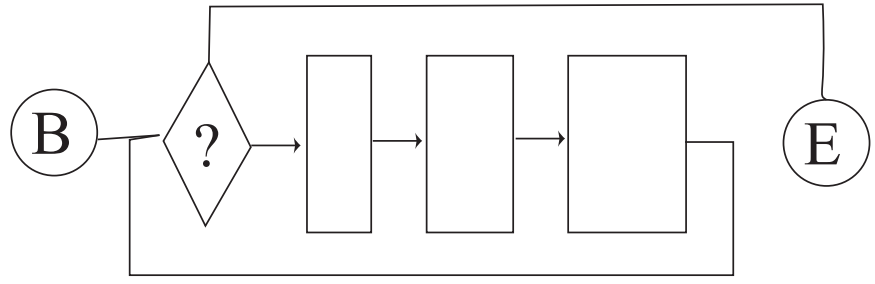
Return A;

Ancora una struttura iterativa!

Algoritmi iterativi

Gli algoritmi visti sinora hanno una struttura comune

- l'algoritmo iterativo più elementare consiste in un ciclo (sequenza di istruzioni eseguita ripetutamente, sinché non diviene vera una condizione di arresto)



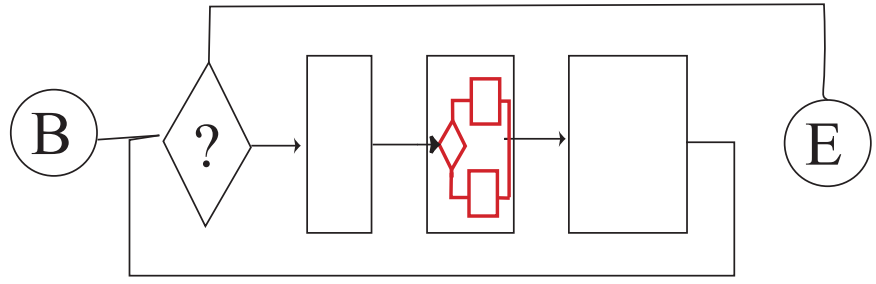
Generalizzando, il ciclo può

- contenere istruzioni condizionali e altri cicli
- essere preceduto e seguito da altre istruzioni

Algoritmi iterativi

Gli algoritmi visti sinora hanno una struttura comune

- l'algoritmo iterativo più elementare consiste in un ciclo (sequenza di istruzioni eseguita ripetutamente, sinché non diviene vera una condizione di arresto)



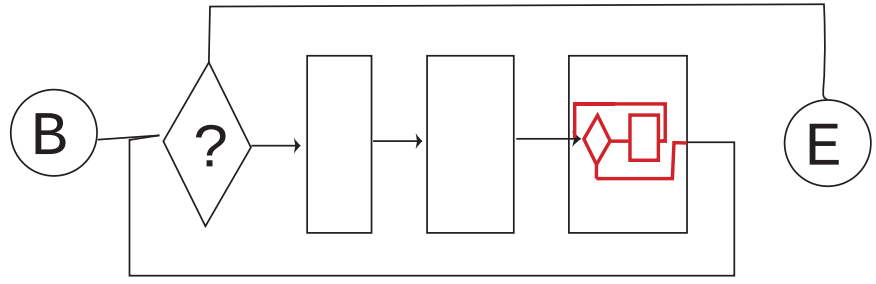
Generalizzando, il ciclo può

- contenere istruzioni condizionali e altri cicli
- essere preceduto e seguito da altre istruzioni

Algoritmi iterativi

Gli algoritmi visti sinora hanno una struttura comune

- l'algoritmo iterativo più elementare consiste in un ciclo (sequenza di istruzioni eseguita ripetutamente, sinché non diviene vera una condizione di arresto)



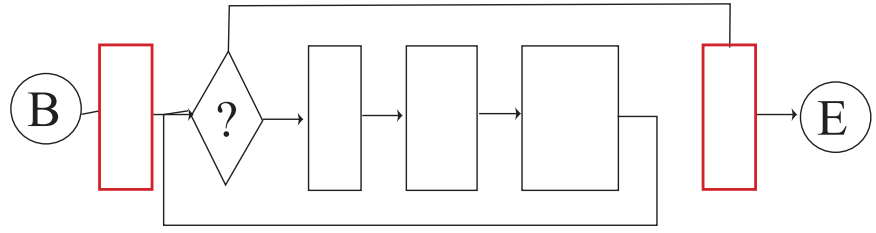
Generalizzando, il ciclo può

- contenere istruzioni condizionali e altri cicli
- essere preceduto e seguito da altre istruzioni

Algoritmi iterativi

Gli algoritmi visti sinora hanno una struttura comune

- l'algoritmo iterativo più elementare consiste in un ciclo (sequenza di istruzioni eseguita ripetutamente, sinché non diviene vera una condizione di arresto)



Generalizzando, il ciclo può

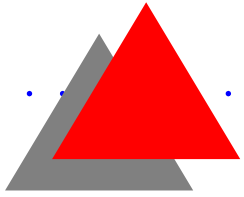
- contenere istruzioni condizionali e altri cicli
- essere preceduto e seguito da altre istruzioni

Obiettivi

Come si fa a. . .

1. . . . ricordare facilmente un algoritmo iterativo?
2. . . . descrivere (bene) un algoritmo iterativo?
3. . . . implementare (bene) un algoritmo iterativo?
4. . . . verificare la correttezza e valutare la complessità di un algoritmo iterativo?
5. . . . progettare un nuovo algoritmo iterativo

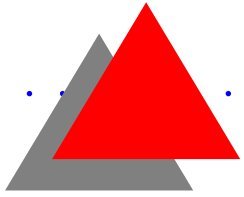
La risposta viene dal concetto di invariante di ciclo



La computazione

Eseguire un algoritmo (computazione) significa
far evolvere nel tempo lo stato della computazione, che è

- per noi, un insieme di oggetti matematici
  (numeri, simboli, grafi, relazioni. . .)
- per una Macchina di Turing, lo stato dell'automa più il contenuto del nastro e la posizione della testina
- per una Macchina RAM, il contenuto di tutti i registri più la posizione nel programma
- per un dispositivo fisico, lo stato dei suoi componenti
  (tensioni elettriche, posizioni di ingranaggi. . .)



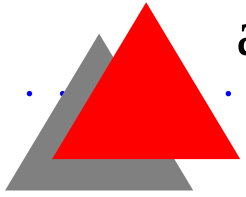
La computazione

Progettare un algoritmo deterministico significa definire regole in base a cui lo stato al prossimo passo dipende solo da

- stato corrente
- ingresso (dati)

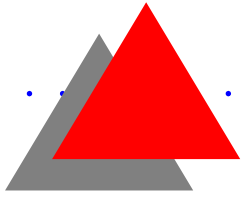
Negli algoritmi non deterministici lo stato futuro può dipendere anche da altro (caso, responsi di oracoli, ecc. . .)

- Alcuni stati godono di proprietà matematiche interessanti, altri no
- Alcuni passaggi da stato a stato richiedono regole semplici, altri regole molto complesse



Qualche concetto

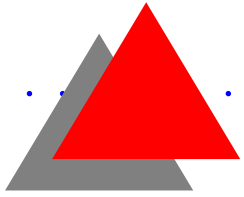
- uno stato iniziale che contiene solo i dati del problema
- uno stato finale che contiene la soluzione del problema
- un insieme di stati ammissibili che godono di alcune delle proprietà dello stato finale
- una distanza dello stato corrente dallo stato finale, misurata quantitativamente in base alle proprietà dello stato finale di cui lo stato corrente non gode



Algoritmi iterativi

- un passo iniziale che fa passare dallo stato iniziale a uno stato ammissibile
- un test di continuazione che verifica se la distanza dallo stato finale è positiva
- un passo iterativo (eseguito finché il test non fallisce) che
  1. conserva le proprietà istituite nello stato iniziale
  2. riduce di un fattore finito la distanza dallo stato finale
- un passo finale che estrae la soluzione dallo stato finale

Invariante di ciclo è l'insieme delle proprietà imposte nel passo iniziale e conservate nel passo iterativo



Algoritmi iterativi

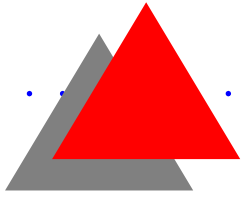
- un passo iniziale che istituisce l'invariante di ciclo
- un test di continuazione che verifica se la distanza è positiva
- un passo iterativo (eseguito finché il test non fallisce) che
  1. conserva l'invariante di ciclo
  2. riduce di un fattore finito la distanza
- un passo finale che estrae la soluzione dallo stato finale

L'invariante di ciclo è un insieme di affermazioni che

- diventano vere dopo il passo iniziale
- se sono vere all'inizio del passo iterativo,
  sono vere alla fine del passo iterativo

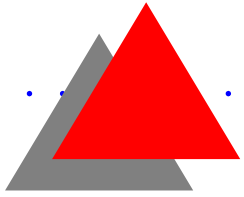
Invariante di Euclide

- *Stato della computazione*: due numeri a e b
- *Invariante di ciclo*: Il MCD di a e b coincide con quello dei due numeri dati
  - All'inizio: vale banalmente
  - Alla fine: i due numeri coincidono fra loro e con la soluzione
  - Si conserva perché: i divisori comuni fra a e b sono anche divisori di $|a - b|$ e $|a - b|$ non ha divisori più grandi...
- *Distanza*: la differenza tra i due numeri



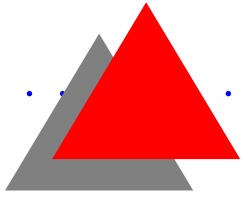
Invariante per la ricerca binaria

- *Stato della computazione*: un vettore di interi e una chiave
- *Invariante di ciclo*: Se la chiave compare nel vettore originale, compare anche in quello corrente
  - All'inizio, vale banalmente
  - Alla fine, il vettore contiene un solo elemento, da confrontare con la chiave
  - Si conserva perché si scartano solo elementi diversi dalla chiave
- *Distanza*: il numero di elementi del vettore corrente



Invariante per Insertion Sort

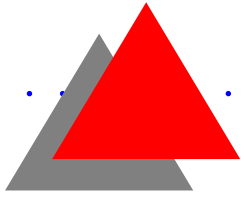
- *Stato della computazione*: un vettore e un contatore j
- *Invariante di ciclo*: I primi j elementi del vettore sono gli stessi del vettore originario, e sono ordinati
  - All'inizio: vale banalmente ($j = 1$)
  - Alla fine, la soluzione e' il vettore
  - Si conserva perché nessun elemento entra o esce dal sottovettore, e il nuovo elemento va in posizione ordinata
- *Distanza*: $n - j$



Ricordare l'algoritmo

Conoscere l'invariante di ciclo aiuta a ricordare lo pseudocodice perché

1. le istruzioni iniziali istituiscono l'invariante
2. la condizione di arresto confronta la distanza con zero
3. le operazioni del ciclo
   - conservano l'invariante
   - riducono la distanza dallo stato finale
4. le istruzioni finali raggiungono la soluzione dallo stato finale



Verificare la correttezza

La dimostrazione di correttezza di un algoritmo iterativo mostra che

1. le istruzioni iniziali istituiscono l'invariante
2. la condizione di arresto confronta la distanza con zero
3. le operazioni del ciclo
   - conservano l'invariante
   - riducono la distanza dallo stato finale
4. le istruzioni finali raggiungono la soluzione dallo stato finale

Spesso, la parte più complessa è la conservazione dell'invariante