L'algoritmo di Euclide: correttezza e complessità

Giovanni Righini

25 gennaio 2024

UNIVERSITÀ DEGLI STUDI DI MILANO

L'algoritmo di Euclide

// Algoritmo di Euclide. IN: a, b. OUT: $MCD(a, b)$.
if $a < b$ **then**
  Scambia(a, b)
end if
while $b > 0$ **do**
  $r \leftarrow a \bmod b$
  $a \leftarrow b$
  $b \leftarrow r$
end while
Return(a)

Esempio: 144   68

L'algoritmo di Euclide

// Algoritmo di Euclide. IN: a, b. OUT: $MCD(a, b)$.
if $a < b$ **then**
  Scambia(a, b)
end if
while $b > 0$ **do**
  $r \leftarrow a \bmod b$
  $a \leftarrow b$
  $b \leftarrow r$
end while
Return(a)

Esempio: 144 68 8

L'algoritmo di Euclide

```
// Algoritmo di Euclide. IN: a, b. OUT: MCD(a, b).
if a < b then
   Scambia(a, b)
end if
while b > 0 do
   r ← a mod b
   a ← b
   b ← r
end while
Return(a)
```

Esempio: 144 68 8 4

L'algoritmo di Euclide

// Algoritmo di Euclide. IN: a, b. OUT: $MCD(a, b)$.
if $a < b$ **then**
  Scambia(a, b)
end if
while $b > 0$ **do**
  $r \leftarrow a \bmod b$
  $a \leftarrow b$
  $b \leftarrow r$
end while
Return(a)

Esempio: 144 68 8 4 0.

Correttezza

Correttezza: cosa garantisce l'algoritmo?

Per dimostrare la correttezza di un algoritmo: invariante di ciclo.

Per dimostrare la terminazione di un algoritmo: monotonicità (stretta) di una grandezza (discreta e finita).

Nel caso degli algoritmi di ottimizzazione, le garanzie di correttezza sono sostituite tipicamente da

- garanzie di ammissibilità e
- garanzie di ottimalità o approssimazione.

Proprietà della divisione.

Dati $a \in \mathbb{Z}_+$ e $b \in \mathbb{Z}_+$ con $a \geq b$, detti q e r il quoziente ed il resto di a/b, per la proprietà della divisione:

- $a = qb + r$;
- $q \in \mathbb{Z}_+$;
- $r \in \mathbb{Z}_+$;
- $qb \leq a$;
- $r < b$.

Proprietà del MCD.

Se $r = 0$ allora $MCD(a, b) = b$.

Usiamo un indice t per contare le iterazioni.

$a^{(t)}$, $b^{(t)}$ e $r^{(t)}$ sono i valori di a, di b e di r al termine dell'iterazione t.

Si ha:

$$a^{(t)} = b^{(t-1)} \quad b^{(t)} = r^{(t)} \quad r^{(t)} = a^{(t-1)} \bmod b^{(t-1)} \;\; \forall t \geq 1$$

a causa degli assegnamenti eseguiti nel ciclo.

Inizialmente, $a^{(0)}$ e $b^{(0)}$ sono i valori all'inizio della prima iterazione del ciclo.

La procedura *Scambia*() garantisce che

$$a^{(0)} \geq b^{(0)}.$$

Invariante di ciclo:

$$a^{(t)} \geq b^{(t)} \geq 0 \;\; \forall t \geq 0.$$

Dimostrazione (per induzione).

Base dell'induzione: $(a^{(0)} \geq 0) \wedge (b^{(0)} \geq 0)$.

Passo induttivo:
$(a^{(t-1)} \geq 0) \wedge (b^{(t-1)} \geq 0) \Longrightarrow (a^{(t)} \geq 0) \wedge (b^{(t)} \geq 0) \;\; \forall t \geq 1$.

Infatti,

- $a^{(t)} = b^{(t-1)} \geq 0$;
- $b^{(t)} = r^{(t)} \geq 0$.

Inoltre $a^{(t)} = b^{(t-1)}$ e $b^{(t)} = r^{(t)} < b^{(t-1)}$, da cui

$$a^{(t)} > b^{(t)}.$$

Invariante di ciclo:

$$MCD(a^{(t)}, b^{(t)}) = MCD(a^{(t-1)}, b^{(t-1)}) \;\; \forall t \geq 1.$$

Dimostrazione.

Lemma. Dati $a \geq b > 0$ interi, con q e r quoziente e resto di a/b,

$$MCD(a, b) = MCD(b, r).$$

Dimostrazione.

Siano $a, b \in \mathbb{Z}_+ : a \geq b$ e sia k un fattore comune ad a e b:
$\exists \alpha, \beta \in \mathbb{Z}_+ : (a = \alpha k) \wedge (b = \beta k) \Longrightarrow r = a - qb = (\alpha - q\beta)k \Longrightarrow$
$\Longrightarrow \exists \gamma : r = \gamma k$ e $\gamma = \alpha - q\beta$.
Poiché $(\alpha \in \mathbb{Z}) \wedge (\beta \in \mathbb{Z}) \wedge (q \in \mathbb{Z})$, anche $(\gamma \in \mathbb{Z})$.
Poiché $r = \gamma k$ e $(\gamma \in \mathbb{Z})$, k è un fattore di r.
Quindi, se k appartiene al $MCD(a, b)$, allora appartiene anche al $MCD(b, r)$.

Analogamente si dimostra che se un intero k è divisore di b e di r, lo è anche di a.
Se k appartiene al $MCD(b, r)$, allora appartiene anche al $MCD(a, b)$.

Quindi tra $MCD(a, b)$ e $MCD(b, r)$ nessuno dei due può essere più piccolo dell'altro, perché deve contenerne tutti i fattori.

Poiché $a^{(t)} = b^{(t-1)}$, $b^{(t)} = r^{(t)}$ e $r^{(t)} = a^{(t-1)} \bmod b^{(t-1)}$ se

$$MCD(a, b) = MCD(b, r)$$

allora

$$MCD(a^{(t-1)}, b^{(t-1)}) = MCD(b^{(t-1)}, r^{(t)}) = MCD(a^{(t)}, b^{(t)}) \ \ \forall t \geq 1.$$

In particolare, se l'algoritmo termina all'iterazione T, allora $b^{(T)} = r^{(T)} = 0$, e quindi $MCD(a^{(T-1)}, b^{(T-1)}) = b^{(T-1)}$.
Quindi

$$MCD(a^{(0)}, b^{(0)}) = MCD(a^{(T-1)}, b^{(T-1)}) = b^{(T-1)} = a^{(T)},$$

che è appunto il valore che l'algoritmo dà in output.

Terminazione dell'algoritmo

Monotonicità.

$b \in \mathbb{Z}_+$ è strettamente decrescente, perché $b^{(t)} = r^{(t)} < b^{(t-1)}$.

Quindi l'algoritmo termina necessariamente in un numero finito di iterazioni.

Complessità computazionale

Complessità computazionale: quanto velocemente aumenta il consumo di spazio e di tempo all'aumentare delle dimensioni dell'input?

La complessità degli algoritmi (in tempo o in spazio) si può valutare

- nel caso peggiore (*worst-case complexity*) e si indica con $O()$;
- nel caso medio (*average-case complexity*).

Per dimostrare la complessità in tempo di un algoritmo nel caso peggiore bisogna:

- trovare un limite superiore al numero di iterazioni necessarie;
- dimostrare la complessità di ogni iterazione.

Tale numero di operazioni va rapportato al numero di bit necessari per descrivere l'input.

Complessità dell'algoritmo di Euclide

La complessità dell'algoritmo è data dal numero di operazioni elementari che deve compiere in funzione della dimensione n dell'input.

Dimensione dell'input.

Il numero n di bit necessari per descrivere l'input è pari a

$$n = \lceil \log_2 a \rceil + \lceil \log_2 b \rceil.$$

Numero di operazioni elementari.

Scambio: 1 confronto e 3 assegnamenti: tempo costante (indipendente da n).

Ogni iterazione: 1 confronto, 1 divisione e 2 assegnamenti: tempo costante (indipendente da n).

Bisogna valutare il (massimo) numero T di iterazioni.

Consideriamo separatamente due casi:

1. Se $b^{(t-1)} > a^{(t-1)}/2$, allora $b^{(t)} = r^{(t)} = a^{(t-1)} - b^{(t-1)} < a^{(t-1)}/2$ e quindi $a^{(t+1)} = b^{(t)} = r^{(t-1)} < a^{(t-1)}/2$.
2. Se $b^{(t-1)} \leq a^{(t-1)}/2$, allora $b^{(t)} = r^{(t)} < b^{(t-1)} \leq a^{(t-1)}/2$ e quindi $a^{(t+1)} = b^{(t)} = r^{(t)} < a^{(t-1)}/2$.

Quindi ogni due iterazioni a viene almeno dimezzato.
Quindi ogni due iterazioni b viene almeno dimezzato.

I numeri a e b sono interi.
Un numero intero k non può essere dimezzato più di $\log_2 k$ volte.
Quindi per il numero T di iterazioni vale

$$T \leq 2 \lfloor \log_2 b \rfloor \leq \lfloor \log_2 a \rfloor + \lfloor \log_2 b \rfloor .$$

$$T \leq n.$$

Per l'algoritmo di Euclide la complessità nel caso peggiore è lineare.