# Capitolo 4: Algoritmi fondamentali di teoria dei numeri

### *§.19 L'algoritmo Euclideo*

Uno dei più antichi e importanti algoritmi di teoria dei numeri descritto da Euclide nei suoi Elementi (nel 300 A.C. circa) individua il massimo comun divisore tra due numeri interi.
Rimasto sostanzialmente inalterato per 2300 anni, è tuttora di estrema utilità e noi lo ritroveremo, quindi, in molti tra gli algoritmi qui presentati.
Ad esempio gran parte degli algoritmi di fattorizzazione di numeri interi si basano proprio sulla ricerca di fattori che non siano coprimi col numero $n$ da fattorizzare per poi ricavare un fattore di $n$ proprio con l'ausilio dell'algoritmo euclideo.
Oppure, più banalmente, ritroveremo l'algoritmo euclideo ogni volta che dovremo ridurre una frazione ai minimi termini.

Nel seguito useremo la consueta notazione. Nonostante molti dei risultati che presenteremo si applichino agli interi relativi $\mathbb{Z}$, per gli scopi della fattorizzazione trascureremo gli interi negativi e quindi, parlando di interi ci riferiremo solo a quelli positivi ovvero a $\mathbb{N}$.

Diremo che un intero $n$ è ***divisibile*** per un intero $d$ o che $d$ ***divide*** $n$ se esiste un intero $m$ tale che $n = m \cdot d$.

Se $d$ divide $n$ scriveremo $d \mid n$ e diremo che $d$ è un ***divisore*** di $n$ e che $n$ è un ***multiplo*** di $d$.

Viceversa se $d$ non divide $n$ scriveremo $d \nmid n$

Ogni intero $n$ è divisibile per i suoi ***divisori banali*** 1 e $n$. I divisori non banali sono detti ***fattori***.

Un intero maggiore di 1 che ammette solo divisori banali si dirà ***numero primo***. Viceversa verrà chiamato ***numero composito***.

E' già evidente che per la funzionalità dell'algoritmo euclideo è necessario disporre di un algoritmo che effettui la divisione tra interi ovvero che, dati due numeri interi $a \geq b > 0$ restituisca la coppia di interi $q, r$ tali che $a = q \cdot b + r$ con $0 \leq r < b$.

Chiameremo $q$ il ***quoziente*** e $r$ il ***resto*** della divisione di $a$ per $b$.

Ovviamente $b \mid a \Leftrightarrow r = 0$ cioè $b$ divide $a$ se e solo se il resto della divisione è $r = 0$.

Un modo banale per implementare la divisione tra interi è fornito nella seguente procedura.

**Procedura** $\mathrm{Div}(a,b)$
**if** $b=0$ **then return** $(0,a)$
$q=\left\lfloor \frac{a}{b} \right\rfloor$
$r=a-qb$
**return** $(q,r)$

Poichè l'algoritmo di divisione non funziona se $r \notin \mathbb{Z}$ (cade l'unicità di quoziente e resto) verifichiamone la validità in $\mathbb{Z}$.

**Teorema**: $\forall a \in \mathbb{Z}, b \in \mathbb{N} \quad \exists! \, q,r \in \mathbb{Z} : a = qb + r, 0 \le r < b$
Il punto cruciale è la proprietà che ogni sottoinsieme di $\mathbb{N}$ ammette minimo.
Osserviamo che $\forall a,b \in \mathbb{N}$ il resto della divisione può essere definito come il più piccolo elemento dell'insieme $\{a-qb : q \in \mathbb{N}, a-qb \ge 0\} \subseteq \mathbb{N}$.
Da qui è facile dimostrare che $0 \le r < b$ e che la scelta della coppia $q,r$ è univoca.

La ***classe di equivalenza modulo n*** contentente un intero $a$ verrà indicata con:

(66) $$[a]_n = \{a + kn : k \in \mathbb{Z}\}$$

e diremo che

(67) $$b \in [a]_n \Leftrightarrow b \equiv a \bmod n \Leftrightarrow \exists k : b = a + kn$$

L'insieme delle classi di equivalenza modulo n viene usualmente indicata con:

(68) $$\mathbb{Z}_n = \{[a]_n : 0 \le a \le n-1\} = \{0,1,\ldots,n-1\}$$

Come rappresentante di ogni classe viene preso il più piccolo intero positivo che vi appartiene per cui, ad esempio: $-1 \in [n-1]_n$, $n \in [0]_n$ e $n+1 \in [1]_n$

Per le proprietà delle classi di resto modulo $n$ è evidente che $r \equiv a \bmod b$ da cui:

(69) $$b \mid a \Leftrightarrow a \bmod b = 0$$

ovvero

(70) $$b \mid a \Leftrightarrow a \equiv 0 \bmod b$$

Se $d \mid a$ e $d \mid b$ allora $d$ viene chiamato **divisore comune** di $a$ e $b$.

Inoltre, importante conseguenza è che

(71) $\text{se } d \mid a \wedge d \mid b \Rightarrow d \mid xa + yb \quad \forall x, y \in \mathbb{Z}$

e, in particolare, $d \mid a+b$ e $d \mid a-b$. Inoltre:

(72) $a \mid b \wedge b \mid a \Rightarrow a = \pm b$

Dati due interi $a, b \neq 0$, il più grande divisore comune, indicato con $\gcd(a,b)$ oppure con $\text{mcd}(a,b)$, viene detto massimo comun divisore.
Per convenzione $\text{mcd}(0,0) = 0$. Inoltre:

(73) $\text{mcd}(a,b) = \text{mcd}(b,a)$

(74) $\text{mcd}(a,b) = \text{mcd}(-a,b)$

(75) $\text{mcd}(a,b) = \text{mcd}(|a|,|b|)$

(76) $\text{mcd}(a,0) = |a|$

(77) $\text{mcd}(a,ka) = |a| \quad \forall k \in \mathbb{Z}$

Il seguente teorema fornisce una diversa caratterizzazione del massimo comun divisore

**Teorema**: dati due numeri interi $a, b : ab \neq 0$ allora il massimo comun divisore $\text{mcd}(a,b)$ è il più piccolo elemento positivo dell'insieme $S = \{xa + yb : x, y \in \mathbb{Z}\}$ delle combinazioni lineari di $a$ e $b$

Dim.: sia $s = xa + yb$ il più piccolo elemento positivo di $S$ e, inoltre, sia $q = \left\lfloor \frac{a}{s} \right\rfloor$.
Allora $a \bmod s = a - qs = a - q(xa + yb) = a(1 - qx) + b(-qy)$ ovvero $a \bmod b \in S$.
Ma poichè $0 \leq a \bmod s < s$ ed $s$ è il più piccolo elemento di $S$ ne deduciamo che $a \bmod s = 0$ ovvero $s \mid a$. Analogamente ricaviamo anche $s \mid b$. Quindi $s$ è un divisore comune di $a$ e $b$ e pertanto $\text{mcd}(a,b) \geq s$.
D'altro canto $\text{mcd}(a,b) \mid xa + yb = s$ ed essendo $s > 0$ ricaviamo $\text{mcd}(a,b) \leq s$.
Quindi, complessivamente, $\text{mcd}(a,b) = s$

**Corollario**: se $d \mid a \wedge d \mid b \Rightarrow d \mid \text{mcd}(a,b)$

**Corollario**: $\forall a, b \in \mathbb{Z}, n > 0 \in \mathbb{N} \Rightarrow \text{mcd}(na, nb) = n\,\text{mcd}(a,b)$

**Corollario** $\forall a, b, n > 0 \in \mathbb{N}$ $\text{se } n \mid ab \wedge \text{mcd}(a,n) = 1 \Rightarrow n \mid b$

Due interi $a, b$ per i quali $\text{mcd}(a,b) = 1$ si dicono **co-primi**

**Teorema**: $\forall a, b, p \in \mathbb{Z} : \text{mcd}(a, p) = \text{mcd}(b, p) = 1 \Rightarrow \text{mcd}(ab, p) = 1$

La ***fattorizzazione primaria*** (o più semplicemente ***fattorizzazione***) di un intero consiste nella sua rappresentazione come prodotto di numeri esclusivamente primi.

**Teorema**: $\forall a, b, p \in \mathbb{Z}$ con $p$ primo: $p \mid ab \Leftrightarrow p \mid a \vee p \mid b$

Da quest'ultimo discende l'importante

**Teorema sulla fattorizzazione unica**: ogni intero ha una unica fattorizzazione in primi. Più precisamente, un intero composto $a$ può essere espresso in un unico modo come prodotto della forma

(78) $$a = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$$

dove i $p_j$ sono primi ordinati ovvero $p_i < p_j \quad \forall 1 \le i < j \le r$ e gli $e_j$ sono interi positivi.

Grazie alla proprietà (75) la trattazione successiva può essere limitata agli interi positivi.

Un primo algoritmo per la ricerca del minimo comune multiplo tra due numeri è dato dall'applicazione della nota regola:
Dati due numeri interi positivi $a = p_1^{e_1} p_2^{e_2} \cdots p_r^{e_r}$ e $b = p_1^{f_1} p_2^{f_2} \cdots p_r^{f_r}$, scrittura in cui per mantenere la stessa base di primi potranno comparire anche esponenti nulli, avremo che:

(79) $$\mathrm{mcd}(a,b) = p_1^{\min(e_1, f_1)} p_2^{\min(e_2, f_2)} \cdots p_r^{\min(e_r, f_r)}$$

Poichè questo algoritmo richiede la scomposizione in fattori primi di $a$ e $b$ e, come vedremo, l'algoritmo più efficiente di fattorizzazione ad oggi conosciuto non elabora in tempi polinomiali, il metodo più efficiente di calcolo del minimo comun divisore resta l'algoritmo euclideo che si basa sul seguente

**Teorema**: per ogni coppia $a, b \in \mathbb{N}$ con $b \neq 0$ vale la relazione:

(80) $$\mathrm{mcd}(a,b) = \mathrm{mcd}(b, a \bmod b)$$

**Dim.**: sia $d = \mathrm{mcd}(a,b)$ allora $d \mid a, d \mid b$ e, ricordando che $a \bmod b = xa + yb$, la (71) ci garantisce che $d \mid a \bmod b$. Dunque $d = \mathrm{mcd}(a,b) \mid \mathrm{mcd}(b, a \bmod b)$.
Sia ora $d = \mathrm{mcd}(b, a \bmod b)$ dunque $d \mid b$ e $d \mid a \bmod b$ e pertanto $d \mid xb + ya \bmod b$. In particolare $d \mid a$ essendo $a = qb + a \bmod b$ e quindi $d = \mathrm{mcd}(b, a \bmod b) \mid \mathrm{mcd}(a,b)$.

Per la (72) abbiamo $\mathrm{mcd}(a,b) = \pm \mathrm{mcd}(b, a \bmod b)$ da cui la tesi essendo il minimo comune multiplo una quantità positiva per definizione.

Questo teorema ci fornisce gli elementi necessari alla scrittura della procedura che realizza l'algoritmo euclideo:

> **Procedura** mcd($a$,$b$)
> **if** $b = 0$ **return** $a$
> **return** $\mathrm{mcd}(b, a \bmod b)$

Poichè nelle chiamate ricorsive il secondo argomento è sempre non negativo e decresce strettamente (tranne al più alla prima chiamata) l'algoritmo termina sempre in tempi finiti (e fornendo sempre il risultato corretto).

Analizziamo ora i tempi di elaborazione nel caso peggiore. I tempi sarenno funzione della dimensione di $a$ e $b$.

Senza perdere in generalità ipotizzeremo $a > b$ (ormai sappiamo che, in caso contrario, a questa condizione ci si riconduce al secondo passaggio).

**Teorema**: se $a > b \geq 1$ e l'algoritmo esegue $k \geq 1$ cicli ricorsivi allora $a \geq F_{k+2}$ e $b \geq F_{k+1}$

**Dim**.: Si procede per induzione. Per $k = 1$ il teorema è vero infatti $b \geq 1 = F_2$ e $a > b \geq 1 \Rightarrow a \geq 2 = F_3$. Ora verifichiamo che se il teorema è vero per $k-1$ allora vale anche per $k$.
Supponiamo che l'algoritmo effettui $k$ cicli ricorsivi per elaborare l'istanza $(a,b)$ allora per l'istanza $(b, a \bmod b)$ effettuerà $k-1$ cicli e dunque $b \geq F_{k+1}$ e $a \bmod b \geq F_k$.
Sommando ricaviamo che $F_{k+2} = F_{k+1} + F_k \leq b + a \bmod b = b + \left( a - \left\lfloor \frac{a}{b} \right\rfloor b \right) \leq a$

**Teorema di Lamè**: per ogni $k \geq 1$, se $a > b$ e $b \leq F_{k+1}$ allora l'algoritmo richiede meno di $k$ cicli per completare il calcolo sull'istanza $(a,b)$.

Osserviamo che per $k > 2$, essendo $F_{k+1} = F_k + F_{k-1}$, vale la proprietà

(81) $$F_{k+1} \bmod F_k = F_{k-1}$$

Quindi

(82) $$\mathrm{mcd}(F_{k+1}, F_k) = \mathrm{mcd}(F_k, F_{k+1} \bmod F_k) = \mathrm{mcd}(F_k, F_{k-1})$$

da cui deduciamo che l'istanza costituita da due numeri di Fibonacci consecutivi realizza il caso peggiore richiedendo esattamente $k-1$ cicli elaborativi.

Ricordando che il caso peggiore fornisce un limite superiore ai tempi di elaborazione e che $b = F_k \simeq \frac{1}{\sqrt{5}} f^k$ da cui $k \simeq \lg f \left( \frac{\lg 5}{2} + \lg b \right)$, possiamo concludere che il numero di cicli elaborativi è $O(\lg b)$.

### *§.20 L'algoritmo euclideo esteso*

Abbiamo visto in precedenza che $d = \mathrm{mcd}(a,b)$ può essere espresso come combinazione lineare di $a$ e $b$ ovvero che $d = xa + yb$. L'obiettivo dell'algoritmo euclideo esteso è quello di individuare i coefficienti $x$ e $y$ della combinazione.

Tali coefficienti hanno estremo rilievo in molte questioni. Una tra tutte: rivestono un ruolo centrale nella dimostrazione del teorema della fattorizzazione unica enunciato nel precedente paragrafo.

Ovviamente un teorema ci garantisce l'esistenza della coppia di coefficienti della combinazione lineare.

La seguente procedura, a fronte di un'istanza costituita dalla solita coppia $(a,b)$, restituisce la terna $(d, s, t)$ definita da $d = \mathrm{mcd}(a,b) = sa + tb$.

**Procedura** xmcd$(a,b)$
**if** $b = 0$ **then return** $(a,1,0)$
$(d',s',t') = \mathrm{xmcd}(b, a \bmod b)$
$(d,s,t) = \left( d', t', s' - \left\lfloor \frac{a}{b} \right\rfloor t' \right)$
**return** $(d,s,t)$

Innanzi tutto quantifichiamo i tempi di elaborazione. Poichè il numero di cicli ricorsivi è lo stesso dell'algoritmo euclideo classico i tempi di elaborazione sono uguali a meno di un fattore costante. Dunque anche in questo caso il numero di cicli ricorsivi per un'istanza $(a,b)$ con $0 < b < a$ è $O(\lg b)$.

Per capire il funzionamento dell'algoritmo euclideo esteso (ricorsivo) costruiamone prima una versione iterativa.

Ricordiamo che al ciclo k-esimo l'algoritmo euclideo prende la coppia $(a_{k-1}, b_{k-1})$ e la sostituisce con la coppia $(a_k, b_k) = (b_{k-1}, a_{k-1} \bmod b_{k-1})$.

Definiamo la successione dei resti $r_{k+1} = a_{k-1} \bmod b_{k-1} = a_{k-1} - q_k b_{k-1}$ con le condizioni $r_0 = a$ e $r_1 = b$. Osserviamo che $b_{k-1} = r_k$ e, di conseguenza, $a_{k-1} = b_{k-2} = r_{k-1}$

Sostituendo tali valori nel termine $(k+1)$-esimo della successione dei resti ricaviamo la sua espressione ricorsiva:

(83) $$r_{k+1} = r_{k-1} - q_k r_k$$

che ci dice che ogni resto è combinazione lineare intera dei precedenti due resti.
In particolare, sostituendo a ritroso i vari resti a secondo membro della (83), potremo arrivare ad esprimere l'ultimo resto, ovvero $r_n = d$, come combinazione lineare intera dei primi due resti, ovvero $r_0 = a$ e $r_1 = b$.

Evidentemente è $r_0 = a = 1 \cdot a + 0 \cdot b$ e $r_1 = b = 0 \cdot a + 1 \cdot b$ e dunque, detta $(x_k, y_k)$ la coppia dei coefficienti della combinazione lineare di $a$ e $b$ che realizza proprio il resto k-esimo $r_k$, avremo che $(x_0, y_0) = (1,0)$, $(x_1, y_1) = (0,1)$ e $(x_n, y_n) = (x, y)$.

In generale avremo che $r_{k-1} = x_{k-1} \cdot a + y_{k-1} \cdot b$ e $r_k = x_k \cdot a + y_k \cdot b$. Pertanto sostituendo queste due espressioni nella (83) ricaviamo l'espressione:

(84) $$r_{k+1} = r_{k-1} - q_k r_k = x_{k-1} \cdot a + y_{k-1} \cdot b - q_k (x_k \cdot a + y_k \cdot b) = (x_{k-1} - q_k x_k) a + (y_{k-1} - q_k y_k) b$$

che ci consente di esprimere i coefficienti della combinazione lineare in modo ricorsivo dimostrandone, tra l'altro, l'esistenza per induzione:

(85) $$(x_{k+1}, y_{k+1}) = (x_{k-1}, y_{k-1}) - q_k (x_k, y_k)$$

Dunque, durante l'esecuzione dell'algoritmo euclideo esteso, verrà generata la seguente successione di valori:

| **k** | $\mathbf{a_{k-1} = r_{k-1}}$ | $\mathbf{b_{k-1} = r_k}$ | $\mathbf{q_k} = \left\lfloor \frac{\mathbf{a_{k-1}}}{\mathbf{b_{k-1}}} \right\rfloor = \left\lfloor \frac{\mathbf{r_{k-1}}}{\mathbf{r_k}} \right\rfloor$ | $\mathbf{r_{k+1} = a_{k-1} \bmod b_{k-1}}$ $\mathbf{= r_{k-1} - q_k r_k}$ | $(\mathbf{x_{k-1}, y_{k-1}}) =$ $(\mathbf{x_{k-3} - q_{k-2} x_{k-2}, y_{k-3} - q_{k-2} y_{k-2}})$ |
|---|---|---|---|---|---|
| 1 | $r_0 = a$ | $r_1 = b$ | $q_1 = \left\lfloor \frac{a}{b} \right\rfloor$ | $r_2 = a - q_1 b$ | $(1,0)$ |
| 2 | $r_1 = b$ | $r_2$ | $q_2 = \left\lfloor \frac{r_1}{r_2} \right\rfloor$ | $r_3 = r_1 - q_2 r_2$ | $(0,1)$ |
| ... | ... | ... | ... | ... | ... |
| n-1 | $r_{n-2}$ | $r_{n-1}$ | $q_{n-1} = \left\lfloor \frac{r_{n-2}}{r_{n-1}} \right\rfloor$ | $r_n = d$ | $(x_{n-2}, y_{n-2})$ |
| n | $r_{n-1}$ | $r_n = d$ | $q_n = \left\lfloor \frac{r_{n-1}}{r_n} \right\rfloor = \frac{r_{n-1}}{d}$ | $r_{n+1} = 0$ | $(x_{n-1}, y_{n-1})$ |
| n+1 | $d$ | 0 | | | $(x_{n-2}, y_{n-2}) - q_{n-1} (x_{n-1}, y_{n-1})$ |

Ad esempio, per la coppia $(97,18)$ otteniamo la seguente successione:

| **k** | $\mathbf{r_{k-1}}$ | $\mathbf{r_k}$ | $\mathbf{q_k}$ | $\mathbf{r_{k+1}}$ | $\mathbf{x_{k-1}}$ | $\mathbf{y_{k-1}}$ |
|---|---|---|---|---|---|---|
| 1 | 97 | 18 | $q_1 = 5$ | 7 | $x_0 = 1$ | $y_0 = 0$ |
| 2 | 18 | 7 | $q_2 = 2$ | 4 | $x_1 = 0$ | $y_1 = 1$ |
| 3 | 7 | 4 | $q_3 = 1$ | 3 | $x_2 = x_0 - q_1 x_1 = 1$ | $y_2 = y_0 - q_1 y_1 = -5$ |
| 4 | 4 | 3 | $q_4 = 1$ | 1 | $x_3 = x_1 - q_2 x_2 = -2$ | $y_3 = y_1 - q_2 y_2 = 11$ |
| 5 | 3 | 1 | $q_5 = 3$ | 0 | $x_4 = x_2 - q_3 x_3 = 3$ | $y_4 = y_2 + q_3 y_3 = -16$ |
| 6 | 1 | 0 | | | $x_5 = x_3 - q_4 x_4 = -5$ | $y_5 = y_3 - q_4 y_4 = 27$ |

Adesso possiamo dunque scrivere l'algoritmo in forma iterativa in cui vengono definiti un vettore $v_0 \equiv \left[k_0, (a_0, b_0), (q_0, r_0), (x_0, y_0)\right]$ atto a memorizzare i risultati del ciclo corrente e due vettori $v_{-1}$ e $v_{-2}$ in cui memorizzare i risultati dei due cicli precedenti.

**Procedura** $\mathrm{xmcd}\, I(a, b)$

$v_{-2} = \left[0, (0,0), (0,0), (0,0)\right]$ **,** $v_{-1} = \left[0, (0,0), (0,0), (0,1)\right]$

**if** $b = 0$ **then return** $v_0 = \left[k_0 = 0, (a_0, b_0) = (a, b), , , (x_0, y_0) = (1,0)\right]$

$$v_0 = \left[k_0 = 1, (a_0, b_0) = (a, b), (q_0, r_0) = \left(\left\lfloor \frac{a_0}{b_0} \right\rfloor, a_0 - q_0 b_0\right), (x_0, y_0) = (1,0)\right]$$

**while** $r_0 \neq 0$

$v_0 = F\left(v_{-2}, v_{-1}\right)$

**end while**

$v_{-2} \leftarrow v_{-1} \leftarrow v_0$

$v_0 = \left[, d, 0, , , (x_0, y_0) = (x_{-2}, y_{-2}) - q_{-2}(x_{-1}, y_{-1})\right]$

**return** $v_0$

**Procedura** $F\left(v_{-2}, v_{-1}\right)$

$v_{-2} \leftarrow v_{-1} \leftarrow v_0$

$$\begin{cases} k_0 = k_{-1} + 1 & (q_0, r_0) = \left(\left\lfloor \frac{b_{-1}}{r_{-1}} \right\rfloor, b_{-1} - \left\lfloor \frac{b_{-1}}{r_{-1}} \right\rfloor r_{-1}\right) \\ (a_0, b_0) = (b_{-1}, r_{-1}) & (x_0, y_0) = (x_{-2}, y_{-2}) - q_{-2}(x_{-1}, y_{-1}) \end{cases}$$

**return** $v_0 = \left[k_0, (a_0, b_0), (q_0, r_0), (x_0, y_0)\right]$

Per esercizio verificare la correttezza della procedura nei casi critici $(a, 0)$ e $(ab, b)$.

Ora ritorniamo all'analisi della procedura ricorsiva.

Osserviamo, innanzi tutto, una differenza fondamentale tra le due versioni dell'algoritmo.
Nel caso iterativo le coppie di coefficienti vengono calcolate dall'alto al basso e ad ogni passo $k$ restituiscono $a_{k-1} = x_{k-1}a + y_{k-1}b$ in funzione sempre dell'istanza iniziale $(a, b)$.
Nel caso ricorsivo, invece, le coppie di coefficienti vengono calcolate dal basso in alto (ovvero in fase di uscita dai cicli ricorsivi) e ad ogni passo $k$ restituiscono sempre $d = x_{k-1}a_{k-1} + y_{k-1}b_{k-1}$ ma in funzione dell'istanza $(a_{k-1}, b_{k-1})$ relativo al $k$-esimo ciclo ricorsivo.

La relazione ricorsiva si ricava immediatamente osservando che:

(86) $$xa + yb = d = d' = x'b + y'(a \bmod b) = x'b + y'\left(a - \left\lfloor \frac{a}{b} \right\rfloor b\right) = y'a + \left(x' - \left\lfloor \frac{a}{b} \right\rfloor y'\right)b$$

da cui uguagliando termine a termine primo e ultimo membro otteniamo:

(87) $$(x, y) = \left(y', x' - \left\lfloor \frac{a}{b} \right\rfloor y'\right)$$

Anche in questo caso, la precedente relazione ci consente di dimostrare per induzione l'esistenza della coppia di coefficienti della combinazione lineare.

Come esempio, confrontiamo le coppie $(s,t)$ generate (dal basso verso l'alto) dall'algoritmo ricorsivo con le coppie $(x, y)$ generate dalla versione iterativa durante il calcolo sulla stessa istanza $(97,18)$ dell'esempio precedente.

| **k** | **a** | **b** | **q** | **r** | **s** | **t** | **d=s*a+t*b** | **x** | **y** | **a=x*A+y*B** |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 97 | 18 | 5 | 7 | **-5** | **27** | **1** | 1 | 0 | 97 |
| 2 | 18 | 7 | 2 | 4 | 2 | -5 | 1 | 0 | 1 | 18 |
| 3 | 7 | 4 | 1 | 3 | -1 | 2 | 1 | 1 | -5 | 7 |
| 4 | 4 | 3 | 1 | 1 | 1 | -1 | 1 | -2 | 11 | 4 |
| 5 | 3 | 1 | 3 | 0 | 0 | 1 | 1 | 3 | -16 | 3 |
| 6 | 1 | 0 | | | 1 | 0 | 1 | **-5** | **27** | **1** |