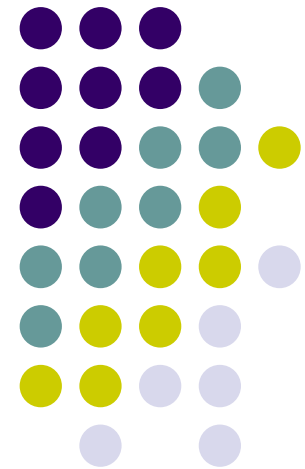


Euclidean Algorithm for GCD and Extended Euclidean Algorithm for the Inverse of an Element in \mathbb{Z}_n^*

Pei-yih Ting



Greatest Common Divisor



- Euclidean Algorithm: calculating GCD
 $\text{gcd}(1180, 482)$

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

1180

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

482 | 1180

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

$$\begin{array}{r|l|l} 482 & 1180 & 2 \end{array}$$

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

$$\begin{array}{r|l} 482 & 1180 \\ & 964 \\ & \hline & 2 \end{array}$$

Greatest Common Divisor



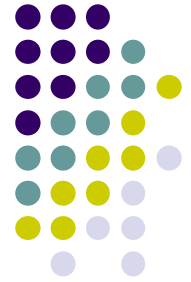
- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

482	1180	2
	964	
	216	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
		964	
		216	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
		216	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
<hr/>			
	50	216	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
<hr/>			
	50	216	4

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
<hr/>			
	50	216	4
		200	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
	50	216	4
		200	
		16	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
3	50	216	4
		200	
		16	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
3	50	216	4
	48	200	
		16	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
3	50	216	4
	48	200	
	2	16	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482	1180	2
	432	964	
3	50	216	4
	48	200	
	2	16	8

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482 432	1180 964	2
3	50 48	216 200	4
	2	16 16	8

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482 432	1180 964	2
3	50 48	216 200	4
	2	16 16	8
		0	

Greatest Common Divisor



- Euclidean Algorithm: calculating GCD

$\text{gcd}(1180, 482)$

(輾轉相除法)

2	482 432	1180 964	2
3	50 48	216 200	4
	2	16 16	8
		0	

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\gcd(a, b)$ or (a, b)
- ex. $\gcd(6, 4) = 2$, $\gcd(5, 7) = 1$
- Euclidean algorithm
 - ex. $\gcd(482, 1180)$

Greatest Common Divisor (cont'd)



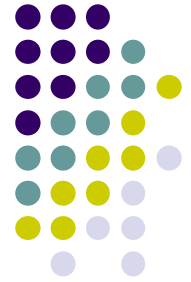
- GCD of a and b is the largest positive integer divides both a and b
- $\gcd(a, b)$ or (a, b)
- ex. $\gcd(6, 4) = 2$, $\gcd(5, 7) = 1$
- Euclidean algorithm
 - ex. $\gcd(482, 1180)$
 $1180 = 2 \cdot 482 + 216$

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\gcd(a, b)$ or (a, b)
- ex. $\gcd(6, 4) = 2$, $\gcd(5, 7) = 1$
- Euclidean algorithm
 - ex. $\gcd(482, 1180)$
 $1180 = 2 \cdot 482 + 216$
 $482 = 2 \cdot 216 + 50$

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\text{gcd}(a, b)$ or (a, b)
- ex. $\text{gcd}(6, 4) = 2$, $\text{gcd}(5, 7) = 1$
- Euclidean algorithm
 - ex. $\text{gcd}(482, 1180)$
$$1180 = 2 \cdot 482 + 216$$
$$482 = 2 \cdot 216 + 50$$
$$216 = 4 \cdot 50 + 16$$

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\text{gcd}(a, b)$ or (a, b)
- ex. $\text{gcd}(6, 4) = 2$, $\text{gcd}(5, 7) = 1$
- Euclidean algorithm
 - ex. $\text{gcd}(482, 1180)$
$$1180 = 2 \cdot 482 + 216$$
$$482 = 2 \cdot 216 + 50$$
$$216 = 4 \cdot 50 + 16$$
$$50 = 3 \cdot 16 + 2$$

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\gcd(a, b)$ or (a, b)
- ex. $\gcd(6, 4) = 2$, $\gcd(5, 7) = 1$
- Euclidean algorithm
 - ex. $\gcd(482, 1180)$
$$1180 = 2 \cdot 482 + 216$$
$$482 = 2 \cdot 216 + 50$$
$$216 = 4 \cdot 50 + 16$$
$$50 = 3 \cdot 16 + 2$$
$$16 = 8 \cdot 2 + 0$$

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\text{gcd}(a, b)$ or (a, b)
- ex. $\text{gcd}(6, 4) = 2$, $\text{gcd}(5, 7) = 1$
- Euclidean algorithm

- ex. $\text{gcd}(482, 1180)$

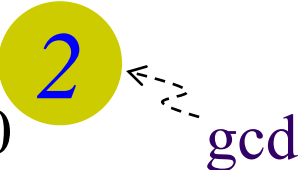
$$1180 = 2 \cdot 482 + 216$$

$$482 = 2 \cdot 216 + 50$$

$$216 = 4 \cdot 50 + 16$$

$$50 = 3 \cdot 16 + 2$$

$$16 = 8 \cdot 2 + 0$$



Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\text{gcd}(a, b)$ or (a, b)
- ex. $\text{gcd}(6, 4) = 2$, $\text{gcd}(5, 7) = 1$
- **Euclidean algorithm**

remainder → divisor → dividend → ignore

- ex. $\text{gcd}(482, 1180)$

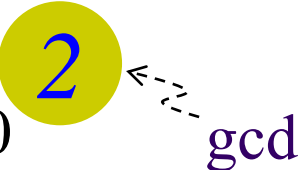
$$1180 = 2 \cdot 482 + 216$$

$$482 = 2 \cdot 216 + 50$$

$$216 = 4 \cdot 50 + 16$$

$$50 = 3 \cdot 16 + 2$$

$$16 = 8 \cdot 2 + 0$$



Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\text{gcd}(a, b)$ or (a, b)
- ex. $\text{gcd}(6, 4) = 2$, $\text{gcd}(5, 7) = 1$
- **Euclidean algorithm**

remainder → divisor → dividend → ignore

- ex. $\text{gcd}(482, 1180)$

$$\begin{aligned} 1180 &= 2 \cdot 482 + 216 \\ 482 &= 2 \cdot 216 + 50 \\ 216 &= 4 \cdot 50 + 16 \\ 50 &= 3 \cdot 16 + 2 \\ 16 &= 8 \cdot 2 + 0 \end{aligned}$$

← gcd

Greatest Common Divisor (cont'd)



- GCD of a and b is the largest positive integer divides both a and b
- $\text{gcd}(a, b)$ or (a, b)
- ex. $\text{gcd}(6, 4) = 2$, $\text{gcd}(5, 7) = 1$
- Euclidean algorithm

- ex. $\text{gcd}(482, 1180)$

$$\begin{aligned} 1180 &= 2 \cdot 482 + 216 \\ 482 &= 2 \cdot 216 + 50 \\ 216 &= 4 \cdot 50 + 16 \\ 50 &= 3 \cdot 16 + 2 \\ 16 &= 8 \cdot 2 + 0 \end{aligned}$$

← gcd

remainder → divisor → dividend → ignore

Why does it work?

Let $d = \text{gcd}(482, 1180)$

$d \mid 482$ and $d \mid 1180 \Rightarrow d \mid 216$

because $216 = 1180 - 2 \cdot 482$

$d \mid 216$ and $d \mid 482 \Rightarrow d \mid 50$

$d \mid 50$ and $d \mid 216 \Rightarrow d \mid 16$

$d \mid 16$ and $d \mid 50 \Rightarrow d \mid 2$

$2 \mid 16 \Rightarrow d = 2$

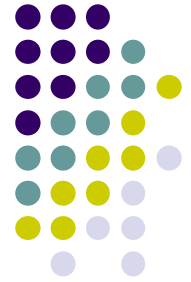
Iterated C Program Design



- 將變數 p 設為變數 $n1$ 內容的絕對值，
將變數 q 設為變數 $n2$ 內容的絕對值
- 將 p 除以 q 的餘數設為 r
- 當 r 大於 0 時
 - 3.1 拷貝 q 的數值至 p
 - 3.2 拷貝 r 的數值至 q
 - 3.3 將 p 除以 q 的餘數設為 r
- q 即為 $n1$ 和 $n2$ 的最大公因數

```
int find_gcd(int n1, int n2) {  
    int p, q, r;  
    p = abs(n1);  
    q = abs(n2);  
    r = p % q;  
    while (r != 0) {  
        p = q;  
        q = r;  
        r = p % q;  
    }  
    return q;  
}
```


Recursive Solution



```
01 int gcd(int p, int q) // assume p >= q
02 {
03     int ans;
04
05     if (p % q == 0)
06         ans = q;
07     else
08         ans = gcd(q, p % q);
09
10     return ans;
11 }
```

Recursive Solution



```
01 int gcd(int p, int q) // assume p >= q
02 {
03     int ans;
04
05     if (p % q == 0)
06         ans = q;
07     else
08         ans = gcd(q, p % q);
09
10     return ans;
11 }
```

```
01 int gcd(int p, int q)
02 {
03     int r = p%q;
04     if (r == 0)
05         return q;
06     return gcd(q, r);
07 }
```



Integer Multiplicative Group Z_p^*

- A **group** G is a finite or infinite set of elements and a binary operation \times which together satisfy

1. Closure: $\forall a, b \in G \quad a \times b = c \in G$ 封閉性
2. Associativity: $\forall a, b, c \in G \quad (a \times b) \times c = a \times (b \times c)$ 結合性
3. Identity: $\forall a \in G \quad 1 \times a = a \times 1 = a$ 單位元素
4. Inverse: $\forall a \in G \quad a \times a^{-1} = 1 = a^{-1} \times a$ 反元素

- e.g. $p=11, Z_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

binary operation: $\times \pmod{p}$, identity: 1, inverse: x^{-1} can be found using the “extended Euclidean Algorithm”

$1 * 1 \equiv 1 \pmod{11}$	$6 * 2 = 12 \equiv 1 \pmod{11}$
$2 * 6 = 12 \equiv 1 \pmod{11}$	$7 * 8 = 56 \equiv 1 \pmod{11}$
$3 * 4 = 12 \equiv 1 \pmod{11}$	$8 * 7 = 56 \equiv 1 \pmod{11}$
$4 * 3 = 12 \equiv 1 \pmod{11}$	$9 * 5 = 45 \equiv 1 \pmod{11}$
$5 * 9 = 45 \equiv 1 \pmod{11}$	$10 * 10 = 100 \equiv 1 \pmod{11}$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$d = 2 = 50 - 3 \cdot 16$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$d = 2 = 50 - 3 \cdot 16$$

$$50 = 482 - 2 \cdot 216$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$d = 2 = 50 - 3 \cdot 16$$

$$50 = 482 - 2 \cdot 216$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$d = 2 = 50 - 3 \cdot 16$$

$$50 = 482 - 2 \cdot 216$$

$$16 = 216 - 4 \cdot 50$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$d = 2 = 50 - 3 \cdot 16$$
$$50 = 482 - 2 \cdot 216$$
$$16 = 216 - 4 \cdot 50$$

Diagram illustrating the Extended Euclidean Algorithm. The equation $d = 2 = 50 - 3 \cdot 16$ is shown on the left. Two dashed orange arrows point from the right-hand side of this equation to the right-hand sides of the two equations below it: $50 = 482 - 2 \cdot 216$ and $16 = 216 - 4 \cdot 50$. The numbers 50 and 16 in the first equation are colored blue and green respectively. In the second equation, 50 is blue and 216 is red. In the third equation, 216 is red and 50 is blue.

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) \\
 50 &= 482 - 2 \cdot 216 \\
 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 & 216 &= 1180 - 2 \cdot 482 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) & 50 &= 482 - 2 \cdot 216 \\
 & & 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 & 216 &= 1180 - 2 \cdot 482 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) & 50 &= 482 - 2 \cdot 216 \\
 & & 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 & 216 &= 1180 - 2 \cdot 482 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) & 50 &= 482 - 2 \cdot 216 \\
 & & 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 & 216 &= 1180 - 2 \cdot 482 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) & 50 &= 482 - 2 \cdot 216 \\
 & & 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 & 216 &= 1180 - 2 \cdot 482 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) & 50 &= 482 - 2 \cdot 216 \\
 &= \dots = 1180 \cdot (-29) + 482 \cdot 71 & 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

Extended Euclidean algorithm for finding the inverse in \mathbb{Z}_p^*



- Def: a and b are **relatively prime**: $\gcd(a, b) = 1$
- **Theorem**: Let a and b be two nonzero integers, and let $d = \gcd(a, b)$. Then there exist integers x, y such that $a \cdot x + b \cdot y = d$
 - Constructive proof: Use the following **Extended Euclidean Algorithm** to find x and y

$$\begin{aligned}
 d = 2 &= 50 - 3 \cdot 16 & 216 &= 1180 - 2 \cdot 482 \\
 &= (482 - 2 \cdot 216) - 3 \cdot (216 - 4 \cdot 50) & 50 &= 482 - 2 \cdot 216 \\
 &= \dots = 1180 \cdot (-29) + 482 \cdot 71 & 16 &= 216 - 4 \cdot 50
 \end{aligned}$$

$\begin{matrix} \uparrow & \nearrow & \uparrow & \nearrow \\ a & x & b & y \end{matrix}$

Extended Euclidean Algorithm



Let $\gcd(a, b) = d$

- Looking for s and t , $\gcd(s, t) = 1$ s.t. $a \cdot s + b \cdot t = d$
- When $d = 1$, $t \equiv b^{-1} \pmod{a}$

$$\begin{aligned}
 a &= q_1 \cdot b + r_1 & \textcircled{1} \\
 b &= q_2 \cdot r_1 + r_2 & \textcircled{2} \\
 r_1 &= q_3 \cdot r_2 + r_3 & \textcircled{3} \\
 r_2 &= q_4 \cdot r_3 + d & \textcircled{4} \\
 r_3 &= q_5 \cdot d + 0 & \textcircled{5}
 \end{aligned}$$

write r_1, r_2, \dots, d as
linear combination
of a and b

Ex. $1180 = 2 \cdot 482 + 216$

$$\begin{aligned}
 1180 - 2 \cdot 482 &= 216 \\
 482 &= 2 \cdot 216 + 50 \\
 482 - 2 \cdot (1180 - 2 \cdot 482) &= 50 \\
 -2 \cdot 1180 + 5 \cdot 482 &= 50 \\
 216 &= 4 \cdot 50 + 16 \\
 (1180 - 2 \cdot 482) - 4 \cdot (-2 \cdot 1180 + 5 \cdot 482) &= 16 \\
 9 \cdot 1180 - 22 \cdot 482 &= 16 \\
 50 &= 3 \cdot 16 + 2 \\
 (-2 \cdot 1180 + 5 \cdot 482) - 3 \cdot (9 \cdot 1180 - 22 \cdot 482) &= 2 \\
 -29 \cdot 1180 + 71 \cdot 482 &= 2
 \end{aligned}$$

Derive the Iteration Formula



- in the forward direction, let $r_0 = a$, $r_1 = b$, $\gcd(a,b) = 1$

$$r_0 = a_0 r_0 + b_0 r_1$$

$$r_1 = a_1 r_0 + b_1 r_1$$

$$r_0 = q_0 r_1 + r_2$$

$$r_0 - q_0 r_1 = r_2 = a_2 r_0 + b_2 r_1$$

$$r_1 = q_1 r_2 + r_3$$

$$r_1 - q_1 r_2 = r_3 = a_3 r_0 + b_3 r_1$$

$$r_2 = q_2 r_3 + r_4$$

$$r_2 - q_2 r_3 = r_4 = a_4 r_0 + b_4 r_1$$

...

...

$$r_{i-2} - q_{i-2} r_{i-1} = r_i = a_i r_0 + b_i r_1$$

$$\begin{cases} a_i = a_{i-2} - q_{i-2} a_{i-1} \\ b_i = b_{i-2} - q_{i-2} b_{i-1} \end{cases}$$

if $r_i == 0$ then $r_{i-1} = 1$
 $a_{i-1} r_0 + b_{i-1} r_1 = r_{i-1} = 1$
 i.e. $a_{i-1} r_0 \equiv 1 \pmod{r_1}$

where $a_0 = 1$, $b_0 = 0$, $a_1 = 0$, $b_1 = 1$

Implementation (w/o using array)



```
int x, p, r, a, am1=0, am2=1;
int b, bm1=1, bm2=0, q, qm1, qm2;

printf("Please input (x, p), where "
      "gcd(x,p)=1 > "); /* assume p >=x */
scanf("%d %d", &x, &p);
printf("inverse(%d) (mod %d) = ", x, p);

if (x == 1) printf("%d\n", x), return 0;

qm2 = p / x; r = p % x; p = x; x = r;
qm1 = p / x; r = p % x; p = x; x = r;

a = am2 - qm2 * am1;
b = bm2 - qm2 * bm1;
```

```
while (r != 0)
{
    q = p / x;
    r = p % x;

    p = x;
    x = r;
    am2 = am1; am1 = a;
    bm2 = bm1; bm1 = b;
    qm2 = qm1; qm1 = q;
    a = am2 - qm2 * am1;
    b = bm2 - qm2 * bm1;
}
printf ("%d\n", b);
```

Another Implementation



```
01 int main(void) {
02     int x, y, a, b;
03     int quotient[50], nIter, gcd;

04     printf("Please input two integers > ");
05     scanf("%d%d", &x, &y);

06     gcd = euclidean(&x, &y, &nIter, quotient);
07     printf("gcd(%d, %d) = %d\n", x, y, gcd);

08     extendedEuclidean(quotient, nIter, &a, &b);
09     if (x < 0) a = -a;
10     if (y < 0) b = -b;
11     prettyPrint(gcd, a, x, b, y);

12     system("pause");
13     return 0;
14 }
```

using Array

Euclidean Algorithm



```
01 int euclidean(int *x, int *y, int *nIter, int quotient[ ]) {
02     int divisor, remainder, dividant, iter=0;
03     if (*x < *y) swap(x, y);
04     dividant = *x < 0 ? -(*x) : *x;
05     divisor = *y < 0 ? -(*y) : *y;
06     do
07     {
08         quotient[iter++] = dividant / divisor;
09         remainder = dividant % divisor;
10         dividant = divisor;
11         divisor = remainder;
12     } while (remainder != 0);
13     *nIter = iter;
14     return dividant;
15 }
```

Extended Euclidean Algorithm



```
01 void extendedEuclidean(int q[], int nIter, int *a, int *b) {
02     int a1[3] = {1, 0, -1}, b1[3] = {0, 1, -1}, i;
03     for (i=0; i<nIter-1; i++) {
04         a1[2] = a1[0] - q[i] * a1[1];
05         b1[2] = b1[0] - q[i] * b1[1];
06         a1[0] = a1[1];
07         a1[1] = a1[2];
08         b1[0] = b1[1];
09         b1[1] = b1[2];
10     }
11     *a = a1[1];
12     *b = b1[1];
13 }
```

Recursive Solution

- Given $a > b \geq 0$, find $g = \text{GCD}(a, b)$ and x, y s.t. $a x + b y = g$ where $|x| \leq b+1$ and $|y| \leq a+1$

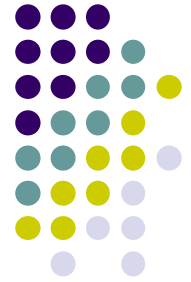


Recursive Solution



- Given $a > b \geq 0$, find $g = \text{GCD}(a, b)$ and x, y s.t. $ax + by = g$ where $|x| \leq b+1$ and $|y| \leq a+1$
- Let $a = qb + r, b > r \geq 0 \Rightarrow (qb + r)x + by = g$
 $\Rightarrow b(qx + y) + rx = g$
 $\Rightarrow by' + rx = g$, where $y' = qx + y$

Recursive Solution



- Given $a > b \geq 0$, find $g = \text{GCD}(a, b)$ and x, y s.t. $a x + b y = g$ where $|x| \leq b + 1$ and $|y| \leq a + 1$
- Let $a = q b + r, b > r \geq 0 \Rightarrow (q b + r) x + b y = g$
 $\Rightarrow b (q x + y) + r x = g$
 $\Rightarrow b y' + r x = g, \text{ where } y' = q x + y$
- This means that if we can find y' and x satisfying $b y' + (a \% b) x = g$ then x and $y = y' - q x = y' - (a/b) x$ satisfies $a x + b y = g$
Note that in this way r will eventually be 0

Recursive Solution



- Given $a > b \geq 0$, find $g = \text{GCD}(a, b)$ and x, y s.t. $a x + b y = g$ where $|x| \leq b + 1$ and $|y| \leq a + 1$
- Let $a = q b + r, b > r \geq 0 \Rightarrow (q b + r) x + b y = g$
 $\Rightarrow b (q x + y) + r x = g$
 $\Rightarrow b y' + r x = g$, where $y' = q x + y$
- This means that if we can find y' and x satisfying $b y' + (a \% b) x = g$ then x and $y = y' - q x = y' - (a/b) x$ satisfies $a x + b y = g$
Note that in this way r will eventually be 0

```
01 void extgcd(int a, int b, int *g, int *x, int *y) { // a > b >= 0
02     if (b == 0)
03         *g = a, *x = 1, *y = 0;
04     else {
05         extgcd(b, a % b, g, y, x);
06         *y = *y - (a / b) * (*x);
07     }
08 }
```