

Extended GCD

Lecture 13: Euclid's algorithm and modular inverses

Lecture overview

Euclid's algorithm finds a greatest common divisor. Keeping track of the same divisions also gives the coefficients needed to compute a modular inverse.

Lecture slides and worked examples.

Primary source: Lecture 13 [Maj26].

Abstract and lecture roadmap

We want to compute $X^{-1} \bmod p$. The useful starting point is an ordinary integer identity:

$$1 = \alpha X + \beta p.$$

Reducing it modulo p leaves $(\alpha \bmod p)X \equiv 1 \pmod{p}$.

The plan

1. Find the gcd using division with remainder.
2. Recover coefficients by following the recursion back up.
3. Implement that return path with an explicit stack.
4. Reduce the coefficients modulo p to find an inverse.

[HPS14, §1.2 and Remark 1.15]; [BS23, §A.3, items 2–3].

Main results

Euclid and extended Euclid

For $A, B \geq 0$, not both zero, the algorithms compute

$$G = \gcd(A, B), \quad G = \alpha A + \beta B$$

with $\alpha, \beta \in \mathbb{Z}$. For inputs of at most n bits, they use $O(n)$ divisions and $O(n^3)$ bit operations with an elementary analysis.

Modular inversion

For $M \geq 2$, the residue X has an inverse modulo M exactly when $\gcd(X, M) = 1$. If $\text{XGCD}(X, M) = (1, \alpha, \beta)$, then

$$X^{-1} \equiv \alpha \pmod{M}.$$

[HPS14, Thms. 1.7, 1.11; Prop. 1.13(b)]; [BS23, §A.3].

An inverse hidden in an integer identity

Suppose we want to invert 17 modulo 43.

$$1 = -5 \cdot 17 + 2 \cdot 43.$$

Taking this identity modulo 43 gives

$$(-5) \cdot 17 \equiv 1 \pmod{43}.$$

The inverse

$$17^{-1} \equiv -5 \equiv 38 \pmod{43}.$$

We can check the answer immediately: $17 \cdot 38 = 646 = 15 \cdot 43 + 1$.

The question is how to find the identity systematically.

The gcd problem

Let $A, B \geq 0$, not both zero. We use $\gcd(A, B)$ for their largest positive common divisor.

For $B > 0$, division with remainder gives

$$A = \gamma B + R, \quad \gamma = \lfloor A/B \rfloor, \quad 0 \leq R < B.$$

One smaller instance

$$\gcd(A, B) = \gcd(B, R).$$

The base case is $\gcd(A, 0) = A$ for $A > 0$.

If $A < B$, the first quotient is 0. The same algorithm still works.

Lecture 13, p. 3 [Maj26]; [HPS14, Thm. 1.7].

Why the gcd stays unchanged

Proof roadmap

Show that (A, B) and (B, R) have exactly the same common divisors.

Proof. Since $A = \gamma B + R$, for every positive integer d :

- If $d \mid A$ and $d \mid B$, then $d \mid A - \gamma B = R$.
- If $d \mid B$ and $d \mid R$, then $d \mid \gamma B + R = A$.

Thus the two pairs have the same common divisors. Taking the largest gives

$$\gcd(A, B) = \gcd(B, R). \quad \square$$

[HPS14, proof of Thm. 1.7, Eq. (1.2), p. 14].

Recursive GCD

```
GCD( $A, B$ )  
  if  $B = 0$ : return  $A$   
   $R \leftarrow A \bmod B$   
  return GCD( $B, R$ )
```

Proof of correctness. The recursive call has the same gcd. Each positive second argument becomes a smaller nonnegative integer, so eventually it becomes 0.

At that point, the answer is $\text{gcd}(A, 0) = A$. $\square$

Correctness comes from the gcd identity. Termination comes from $0 \leq R < B$.

Lecture 13, p. 3 [Maj26]; [HPS14, Thm. 1.7].

Unrolled GCD

```
GCD( $A, B$ )  
  while  $B \neq 0$ :  
     $R \leftarrow A \bmod B$   
     $(A, B) \leftarrow (B, R)$     (simultaneously)  
  return  $A$ 
```

Loop invariant

The gcd of the current pair equals the gcd of the original inputs.

Proof. Each update preserves the gcd. When $B = 0$, the invariant says that the current A is the desired answer. $\square$

Lecture 13, p. 4 [Maj26].

A complete Euclidean computation

For $(A, B) = (240, 46)$:

A	B	γ	R	Division identity
240	46	5	10	$240 = 5 \cdot 46 + 10$
46	10	4	6	$46 = 4 \cdot 10 + 6$
10	6	1	4	$10 = 1 \cdot 6 + 4$
6	4	1	2	$6 = 1 \cdot 4 + 2$
4	2	2	0	$4 = 2 \cdot 2 + 0$

The last nonzero remainder

$$\gcd(240, 46) = \gcd(46, 10) = \cdots = \gcd(2, 0) = 2.$$

Why there are only $O(n)$ divisions

Suppose $A \geq B > 0$, with $A < 2^n$.

Proof. There are two cases.

- If $B \leq A/2$, the first coordinate is at most $A/2$ after one step.
- If $B > A/2$, then $\gamma = 1$ and $R = A - B < A/2$. After two steps, the first coordinate is R , unless the algorithm has already stopped.

Halving within two steps

Every two divisions at least halve the first coordinate. A positive integer starting below 2^n permits only $O(n)$ such divisions. $\square$

An initial $A < B$ adds just one step with quotient 0.

[HPS14, proof of Thm. 1.7, p. 14].

The bit cost of GCD

Assume the two inputs have at most n bits.

Number of divisions	$O(n)$
Largest operand length	n bits
Schoolbook cost per division	$O(n^2)$
Total bit cost	$O(n^3)$

What the bound tells us

Euclid's algorithm runs in polynomial time in the binary input length.

This elementary bound suffices here. A more careful analysis gives a better bound.

Arithmetic costs: [Sho08, Thm. 3.3, p. 62]; Euclid: [HPS14, Thm. 1.7].

What extended GCD returns

The specification

$$\text{XGCD}(A, B) = (G, \alpha, \beta), \quad G = \gcd(A, B) = \alpha A + \beta B.$$

The coefficients α, β are ordinary integers. They may be negative.

For example,

$$\text{XGCD}(240, 46) = (2, -9, 47), \quad 2 = -9 \cdot 240 + 47 \cdot 46.$$

The expression $\alpha A + \beta B$ is a **Bézout identity** for the gcd.

We will compute it using exactly the same divisions as before.

Lecture 13, p. 5 [Maj26]; [HPS14, Thm. 1.11, pp. 16–17].

The coefficient update

Proof roadmap

Trust the recursive answer for (B, R) . Substitute $R = A - \gamma B$ and collect the coefficients of A and B .

Proof. Suppose the recursive call gives

$$G = \alpha' B + \beta' R.$$

Then

$$\begin{aligned} G &= \alpha' B + \beta' (A - \gamma B) \\ &= \beta' A + (\alpha' - \gamma \beta') B. \end{aligned}$$

The return rule

$$(\alpha, \beta) = (\beta', \alpha' - \gamma \beta'). \quad \square$$

Recursive extended GCD

```
XGCD( $A, B$ )  
  if  $B = 0$ : return ( $A, 1, 0$ )  
   $\gamma \leftarrow \lfloor A/B \rfloor$ ,  $R \leftarrow A \bmod B$   
  ( $G, \alpha', \beta'$ )  $\leftarrow$  XGCD( $B, R$ )  
  return ( $G, \beta', \alpha' - \gamma\beta'$ )
```

Proof of correctness. At the base case,

$$A = 1 \cdot A + 0 \cdot 0.$$

At each return, the coefficient update preserves the Bézout identity, and the gcd identity preserves G . Induction up the recursion proves the specification. $\square$

Lecture 13, pp. 6–7 [Maj26]; [HPS14, Thm. 1.11].

Back substitution by hand

Return to the divisions for 240 and 46. Working backward:

$$\begin{aligned}2 &= 6 - 4 \\&= 6 - (10 - 6) = 2 \cdot 6 - 10 \\&= 2(46 - 4 \cdot 10) - 10 = 2 \cdot 46 - 9 \cdot 10 \\&= 2 \cdot 46 - 9(240 - 5 \cdot 46) \\&= -9 \cdot 240 + 47 \cdot 46.\end{aligned}$$

The same update at every line

Replace a remainder by its division identity, then collect coefficients.

The final check is $-2160 + 2162 = 2$.

How large can the coefficients become?

For positive A, B , write $G = \gcd(A, B)$.

A useful bound for the returned coefficients

$$|\alpha| \leq B/G, \quad |\beta| \leq A/G.$$

Proof. If $R = 0$, the return is $(G, 0, 1)$, so the bounds hold. Otherwise the child call on (B, R) gives, inductively,

$$|\alpha'| \leq R/G, \quad |\beta'| \leq B/G.$$

The update therefore gives

$$|\alpha| = |\beta'| \leq B/G, \quad |\beta| \leq (R + \gamma B)/G = A/G. \quad \square$$

Extended GCD is also efficient

There are $O(n)$ calls, and each return performs

$$(\alpha, \beta) = (\beta', \alpha' - \gamma\beta').$$

The coefficient bound shows that the returned integers have $O(n)$ bits, including their signs. The zero-input cases return immediately or after one call.

An elementary running-time bound

Each call uses an $O(n^2)$ division. Each return uses an $O(n^2)$ multiplication and a subtraction. Thus

$$\text{Total bit cost } O(n^3).$$

The product $\gamma\beta'$ has at most $O(n)$ bits as well, since both factors do.

For the extended Euclidean algorithm, see [HPS14, Thm. 1.11].

Unrolling the extended algorithm

Ordinary GCD only needs the current pair. Extended GCD must also recover the coefficients on the return path.

Pass 1: go down

Run Euclid's algorithm. Save the quotient γ_i at each depth i .

Pass 2: come back up

At the base case, initialize $(\alpha, \beta) = (1, 0)$. Read the saved quotients in reverse order and apply the coefficient update.

We will first store the coefficients in the stack, as in the lecture. Then we will see that storing only the quotients is enough.

Lecture 13, pp. 8–10 [Maj26]; compare [HPS14, Ex. 1.12].

The downward pass

```
StackXGCD( $A, B$ )  
   $S \leftarrow []$   
  while  $B \neq 0$ :  
     $\gamma \leftarrow \lfloor A/B \rfloor$ ,  $R \leftarrow A \bmod B$   
    append  $[\gamma, \perp, \perp]$  to  $S$   
     $(A, B) \leftarrow (B, R)$   
   $G \leftarrow A$   
  append  $[\perp, 1, 0]$  to  $S$ 
```

A row stores $[\gamma_i, \alpha_i, \beta_i]$. The symbol $\perp$ means “not filled in.” The base row needs no quotient.

The stack at the base case

For $(A, B) = (240, 46)$, the downward pass produces:

Depth i	A_i	B_i	γ_i	α_i	β_i
0	240	46	5	$\perp$	$\perp$
1	46	10	4	$\perp$	$\perp$
2	10	6	1	$\perp$	$\perp$
3	6	4	1	$\perp$	$\perp$
4	4	2	2	$\perp$	$\perp$
5	2	0	$\perp$	1	0

The pair (A_i, B_i) describes the recursive call at depth i . We show it for explanation; the implementation need not store it.

The base row records $2 = 1 \cdot 2 + 0 \cdot 0$.

The upward pass

Continue the algorithm. Let d be the number of rows in S .

```
for  $i = d - 2, d - 3, \dots, 0$ :  
     $S[i].\alpha \leftarrow S[i + 1].\beta$   
     $S[i].\beta \leftarrow S[i + 1].\alpha - S[i].\gamma \cdot S[i + 1].\beta$   
return  $(G, S[0].\alpha, S[0].\beta)$ 
```

The invariant on the way up

After filling row i ,

$$G = \alpha_i A_i + \beta_i B_i.$$

Proof. The base row satisfies the identity. The same substitution used for recursive XGCD proves it for each preceding row. $\square$

Lecture 13, pp. 9–10 [Maj26].

The completed stack

Read this table from the bottom row upward.

i	A_i	B_i	γ_i	α_i	β_i	Identity
0	240	46	5	-9	47	$2 = -9 \cdot 240 + 47 \cdot 46$
1	46	10	4	2	-9	$2 = 2 \cdot 46 - 9 \cdot 10$
2	10	6	1	-1	2	$2 = -10 + 2 \cdot 6$
3	6	4	1	1	-1	$2 = 6 - 4$
4	4	2	2	0	1	$2 = 0 \cdot 4 + 1 \cdot 2$
5	2	0	$\perp$	1	0	$2 = 1 \cdot 2 + 0 \cdot 0$

For example, row 1 uses row 2 and $\gamma_1 = 4$:

$$(\alpha_1, \beta_1) = (2, -1 - 4 \cdot 2) = (2, -9).$$

A smaller stack implementation

Only the next coefficient pair is needed during the upward pass.

```
StackXGCD( $A, B$ )  
   $Q \leftarrow []$   
  while  $B \neq 0$ :  
     $\gamma \leftarrow \lfloor A/B \rfloor$ ,  $R \leftarrow A \bmod B$   
    append  $\gamma$  to  $Q$ ;   $(A, B) \leftarrow (B, R)$   
   $G \leftarrow A$ ;   $(\alpha, \beta) \leftarrow (1, 0)$   
  while  $Q$  is nonempty:  
     $\gamma \leftarrow \text{pop}(Q)$   
     $(\alpha, \beta) \leftarrow (\beta, \alpha - \gamma\beta)$   (simultaneously)  
  return  $(G, \alpha, \beta)$ 
```

Both entries on the right use the old values of α, β .

When does an inverse exist?

Let $M \geq 2$ and $0 \leq X < M$.

The criterion

$$X \text{ is invertible modulo } M \iff \gcd(X, M) = 1.$$

Proof. If $G = 1$, extended GCD gives

$$1 = \alpha X + \beta M,$$

so $\alpha X \equiv 1 \pmod{M}$.

Conversely, if $XY \equiv 1 \pmod{M}$, then $XY - kM = 1$ for some integer k . Every common divisor of X, M divides 1, so their gcd is 1. $\square$

[HPS14, Prop. 1.13(b), p. 20; Remark 1.15, p. 21]; [BS23, §A.3, items 2–3].

The inversion algorithm

```
Inverse( $X, M$ )  ( $M \geq 2$ )  
   $X \leftarrow X \bmod M$   
   $(G, \alpha, \beta) \leftarrow \text{XGCD}(X, M)$   
  if  $G \neq 1$ : return "no inverse"  
  return  $\alpha \bmod M$ 
```

For a prime modulus p

Every $X \in \mathbb{Z}_p^* = \{1, \dots, p-1\}$ has gcd 1 with p . Thus the algorithm always returns its inverse.

For a composite modulus, the same algorithm works whenever the gcd is 1. For instance, $5^{-1} \equiv 5 \pmod{12}$.

Lecture 13, p. 11 [Maj26]; [HPS14, Remark 1.22].

Computing $17^{-1} \bmod 43$

The divisions

$$17 = 0 \cdot 43 + 17,$$

$$43 = 2 \cdot 17 + 9,$$

$$17 = 1 \cdot 9 + 8,$$

$$9 = 1 \cdot 8 + 1,$$

$$8 = 8 \cdot 1 + 0.$$

Back substitution

$$1 = 9 - 8$$

$$= 9 - (17 - 9)$$

$$= 2 \cdot 9 - 17$$

$$= 2(43 - 2 \cdot 17) - 17$$

$$= -5 \cdot 17 + 2 \cdot 43.$$

The call is $\text{XGCD}(17, 43) = (1, -5, 2)$. The coefficient of 17 is -5 , so the returned inverse is 38.

Reducing the coefficients modulo p

For inversion, we only need the coefficient residues.

The modular update on the return path

$$(\alpha, \beta) \leftarrow (\beta \bmod p, (\alpha - \gamma\beta) \bmod p).$$

Keep the Euclidean divisions and their quotients as exact integer computations.

Proof. At depth i , preserve the congruence

$$G \equiv \alpha_i A_i + \beta_i B_i \pmod{p}.$$

Substituting $R_i = A_i - \gamma_i B_i$ proves the same update modulo p . At the top, $G = 1$ and $(A_0, B_0) = (X, p)$, so $\alpha_0 X \equiv 1 \pmod{p}$. $\square$

The reduced coefficients certify a congruence; they need not give an exact integer Bézout identity.

A modular stack computation

For $\text{XGCD}(17, 43)$, the quotients are $[0, 2, 1, 1, 8]$. Start at $(\alpha, \beta) = (1, 0)$ and pop from the right.

Popped quotient	$\alpha \bmod 43$	$\beta \bmod 43$
8	0	1
1	1	42
1	42	2
2	2	38
0	38	2

The final congruence is $38 \cdot 17 + 2 \cdot 43 \equiv 1 \pmod{43}$.

Its left side is 732, not 1. Reduction preserved exactly the information needed for the inverse.

A quick checkpoint

1. Run Euclid on $(99, 78)$. Find G, α, β with $G = \alpha \cdot 99 + \beta \cdot 78$.
2. Compute $7^{-1} \bmod 26$ using extended GCD.
3. Explain why 6 has no inverse modulo 15.
4. In the coefficient update, what goes wrong if we overwrite α before computing the new β ?

A useful habit

Check the final integer identity, or multiply the proposed inverse by the original residue.

Checkpoint answers

1. $\gcd(99, 78) = 3$, and

$$3 = -11 \cdot 99 + 14 \cdot 78.$$

2. $1 = -11 \cdot 7 + 3 \cdot 26$, so

$$7^{-1} \equiv -11 \equiv 15 \pmod{26}.$$

3. $\gcd(6, 15) = 3$. Any integer combination of 6 and 15 is divisible by 3, so none equals 1.
4. The expression $\alpha - \gamma\beta$ needs the old α . Use simultaneous assignment or save the old values.

Technical inputs used in the proofs

Division with remainder

For $A \geq 0$ and $B > 0$, there are unique integers γ, R with

$$A = \gamma B + R, \quad 0 \leq R < B.$$

Hoffstein–Pipher–Silverman, 2nd ed., §1.2, p. 12, and Ex. 1.14, p. 51 [HPS14].

Binary arithmetic cost model

For operands of at most n bits, schoolbook addition and subtraction cost $O(n)$; multiplication and division cost $O(n^2)$.

Shoup, *A Computational Introduction to Number Theory and Algebra*, 2nd ed., §3.3.5, Thm. 3.3(i)–(iii), p. 62 [Sho08].

The earlier lectures supply these arithmetic facts. All other algorithmic claims were proved here.

References

- [BS23] Dan Boneh and Victor Shoup.
A graduate course in applied cryptography, 2023.
Version 0.6, January 2023.
URL: <https://toc.cryptobook.us/book.pdf>.
- [HPS14] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman.
An Introduction to Mathematical Cryptography.
Undergraduate Texts in Mathematics. Springer, New York, second edition, 2014.
doi:10.1007/978-1-4939-1711-2.
- [Maj26] Hemanta K. Maji.
CS355: Lecture 13: Extended GCD algorithm, 2026.
Instructor-supplied lecture slides, 11 slides.
- [Sho08] Victor Shoup.
A Computational Introduction to Number Theory and Algebra.
Cambridge University Press, second edition, 2008.
URL: <https://shoup.net/ntb/ntb-v2.pdf>.