Lecture 13: Extended GCD Algorithm

GCD & XGCD

・日・ ・ヨ・・

ъ

- The objective of this lecture is to study the GCD and the Extended GCD algorithms
- Furthremore, we shall find another technique to find the multiplicative inverse of X ∈ Z^{*}_p in the group (Z^{*}_p, ×)

- 4 目 ト - 4 目 ト - 4

Recursive GCD

- Given integers A and B, our objective is to find the GCD of A and B
- We shall rely on the following identity to calculate the GCD efficiently

gcd(A, B) = gcd(B, R),

where R is the remainder of the division of A by B (previous lectures we already saw an efficient algorithm for division).

- Why is this algorithm efficient? Because, if B ≤ A, then the number of bits needed to represent (B, R) is (at least) one less than the number of bits needed to represent (A, B)
- What is the base case of this algorithm? If B = 0, then we know that A = gcd(A, B)
- Let us write the code for this recursive algorithm

GCD(*A*, *B*):

- If B == 0 : return A
- else : return GCD(B, A%B)

• We shall now unroll this recursion to make the code more efficient

```
GCD(A, B):
• While B! = 0:
• R = A\%B
• A = B
• B = R
• return A
```

イロン イロン イヨン イヨン

3

• The extended GCD of (A, B) returns three integers (G, α, β) such that

$$G = \operatorname{gcd}(A, B)$$
 and $G = \alpha \cdot A + \beta \cdot B$.

Note that we can use the extended GCD algorithm to invert X ∈ Z^{*}_p, where p is a prime. Observe that (G, α, β) = XGCD(X, p) shall satisfy the following constraints

$$G = 1$$
 and $G = \alpha \cdot X + \beta \cdot p$.

Taking mod p on both side of the equality, we get that α mod p is the multiplicative inverse of X in the group $(\mathbb{Z}_{p}^{*}, \times)$

• Let us use the template of the recursive GCD algorithm to implement the recursive extended GCD algorithm.

Recursive Extended GCD

- Again, we shall use B = 0 as the base case. In this case we have G = gcd(A, B) = A, and we can express G = 1 · A + 0 · B. Therefore, the base case should return (G, α, β) = (A, 1, 0)
- Now, let us consider the recursive step. Suppose from the recursive call XGCD(B, R) returns (G, α', β'). Now, we need to find what should XGCD(A, B) return. Observe that recursively we have the guarantee that G = α' ⋅ R + β' ⋅ R. Note that R = A γ ⋅ B. Substituting this expression of R, we get

$$G = \alpha' \cdot B + \beta' \cdot (A - \gamma \cdot B) = \beta' \cdot A + (\alpha' - \gamma \beta') \cdot B.$$

Therefore, we can set $\alpha = \beta'$ and $\beta = \alpha' - \gamma \beta'$. So, XGCD(*a*, *b*) should return (*G*, β' , $\alpha' - \gamma \beta'$).

(日本) (日本) (日本)

Here we write down the code.
XGCD(A, B):
If B == 0 : return(A, 1, 0)
Else :

R = A%B
(G, α', β') = XGCD(B, R)
γ = (A - R)/B
return (G, β', α' - γ · β')

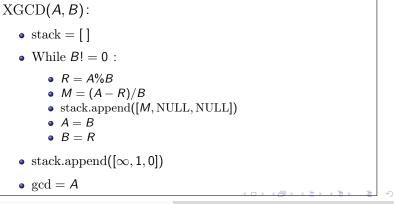
GCD & XGCD

イロン イロン イヨン イヨン

3

Unrolled Extended GCD

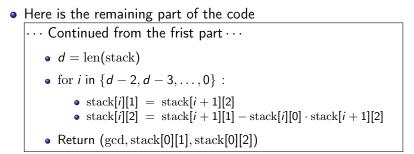
- We shall implement the program stack ourselves
- Let us do this in two steps. First, we shall write the code that implements the recursive calls made by the GCD calculations. In the second part, we shall use the information on the return path up.
- The first part of the code proceeds as follows



 At this point, let us pause and understand how our data structure looks like. Suppose we choose the notation that (m_i, α_i, β_i) are the values of (m, α, β) in the *i*-th depth recursion.

<i>i</i> = 0	<i>i</i> = 1	<i>i</i> = 2	•••	i = d - 2	i = d - 1
<i>m</i> ₁	<i>m</i> ₂	<i>m</i> 3	• • •	m_{d-1}	$m_d = \infty$
NULL	NULL	NULL	• • •	NULL	1
NULL	NULL	NULL		NULL	0

Now, we run an iterator $i \in \{d - 2, d - 3, ..., 0\}$ and update the entries α_i and β_i .



< ロ > < 同 > < 回 > < 回 > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ >

Multiplicative Inverse in \mathbb{Z}_p^*

- Let $X \in \mathbb{Z}_p^*$, where p is a prime
- Therefore, we have gcd(X, p) = 1
- By the extended GCD algorithm, we can find integers α and β such that $1 = \alpha \cdot X + \beta \cdot p$
- Now, we take mod p on both sides of the equality to obtain

$$1 = (\alpha \mod p) \cdot x + 0 \mod p.$$

- That is, we have (α mod p) as the multiplicative inverse of X in the group (Z^{*}_p, ×)
- This computation can be performed by taking mod p in the stack[i][1] and stack[i][2] evaluations in the extended GCD algorithm

イロト イポト イヨト イヨト 二日