Lecture 13: Extended GCD Algorithm

Objective

- 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 $x \in \mathbb{Z}_p^*$ in the group $(\mathbb{Z}_p^*, \times)$

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.
- Why is this algorithm efficient? Because, if $b \leqslant a$, 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, $a\%b$)

Unrolled GCD

- 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

- The extended GCD of (a, b) returns three integers (g, α, β) such that

$$g = \gcd(a, b) \text{ and } g = \alpha \cdot a + \beta \cdot b.$$

- Note that we can use the extended GCD algorithm to invert $x \in \mathbb{Z}_p^*$, where p is a prime. Observe that $(g, \alpha, \beta) = \mathrm{XGCD}(x, p)$ shall satisfy the following constraints

$$g = 1 \text{ and } g = \alpha \cdot x + \beta \cdot p.$$

  Taking $\mod p$ on both side of the equality, we get that $\alpha \mod p$ 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.

- Again, we shall use $b = 0$ as the base case. In this case we have $g = \gcd(a, b) = a$, and we can express $g = 1 \cdot a + 0 \cdot b$. Therefore, the base case should return $(g, \alpha, \beta) = (a, 1, 0)$
- Now, let us consider the recursive step. Suppose from the recursive call $\mathrm{XGCD}(b, r)$ returns (g, α', β'). Now, we need to find what should $(XGCD(a, b)$ return.
  Observe that recursively we have the guarantee that $g = \alpha' \cdot b + \beta' \cdot r$. Note that $r = a - \gamma \cdot b$. 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, $\mathrm{XGCD}(a, b)$ should return $(g, \beta', \alpha' - \gamma\beta')$.

- Here we write down the code.

XGCD(a, b):

- If $b == 0$: return$(a, 1, 0)$
- Else :
  - $r = a\%b$
  - $(g, \alpha', \beta') = \text{XGCD}(b, r)$
  - $\gamma = (a - r)/b$
  - return $(g, \beta', \alpha' - \gamma \cdot \beta')$

Unrolled Extended GCD

- We shall implement the program stack outselves
- 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

XGCD(a, b):

- stack = [ ]
- While $b! = 0$:
  - $r = a\%b$
  - $m = (a - r)/b$
  - stack.append($[m, \text{NULL}, \text{NULL}]$)
  - $a = b$
  - $b = r$
- stack.append($[\infty, 1, 0]$)
- gcd = a

- At this point, let us pause and understand how our datastructure 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.

$i = 0$	$i = 1$	$i = 2$	$\cdots$	$i = d-2$	$i = d-1$
m_1	m_2	m_3	$\cdots$	m_{d-1}	$m_d = \infty$
NULL	NULL	NULL	$\cdots$	NULL	1
NULL	NULL	NULL	$\cdots$	NULL	0

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

- Here is the remaining part of the code

$\cdots$ Continued from the frist part $\cdots$

- $d = \mathrm{len}(\mathrm{stack})$
- for i in $\{d-2, d-3, \ldots, 0\}$:
    - $\mathrm{stack}[i][1] \;=\; \mathrm{stack}[i+1][2]$
    - $\mathrm{stack}[i][2] \;=\; \mathrm{stack}[i+1][1] - \mathrm{stack}[i][0] \cdot \mathrm{stack}[i+1][2]$
- Return $(\mathrm{gcd}, \mathrm{stack}[0][1], \mathrm{stack}[0][2])$

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 $(\alpha \mod p)$ as the multiplicative inverse of x in the group $(\mathbb{Z}_p^*, \times)$
- This computation can be performed by taking $\mod p$ in the $\mathrm{stack}[i][1]$ and $\mathrm{stack}[i][2]$ evaluations in the extended GCD algorithm