BABY STEP GIANT STEP

TEJAS KHAIRNAR

★ Discrete Log (x): ★ Naive : O(p)

★ Block size : b

$\rightarrow a^x \equiv n \pmod p \longrightarrow x \in [0, p-1] \Rightarrow 1 \le i \le \left\lceil \frac{p-1}{b} \right\rceil$

★ $x = ib - j$

$\rightarrow i = \left\lceil \frac{x}{b} \right\rceil \rightarrow j = \left\lceil \frac{x}{b} \right\rceil b - x = (b - (x \% b)) \% b \Rightarrow 0 \le j \le b-1$

★ Giant Step $\longrightarrow i$

★ Baby Step $\longrightarrow j$

$\rightarrow a^{ib} \equiv na^j \pmod p$

$\rightarrow p/b$ values b

- Sort these values
- Iterate & do Bin Search

★ T.C. $\longrightarrow O\left(\frac{p}{b}\right) + O(b) + O\left(\frac{p}{b}\log\frac{p}{b} + b\log\frac{p}{b}\right)$

$\longrightarrow O\left(\frac{p}{b}\log\frac{p}{b} + b\log\frac{p}{b}\right)$

$\rightarrow$ Use Hash-Map $\rightarrow O\left(\frac{p}{b} + b\right)$

$\Rightarrow b = \sqrt{p} \Rightarrow$ T.C. $\rightarrow O(\sqrt{p})$

★ b : block size
★ n : argument
★ a : base
★ p : prime
★ j : Base power
★ x : Exponent

★ CODE SNIPPET

```
int discrete_log(int arg, int base, int p)
{
        int block = sqrt(p - 1);
        vector<pii> arg_base_power(block);  ---> (na^j, j)

        int base_power = 1;  ---> a^j
        for (int j = 0; j < block; j++)
        {
                arg_base_power[j] = mp((1ll * arg * base_power) % p, j);
                base_power = (1ll * base * base_power) % p;
        }

        sort(all(arg_base_power));
        int base_power_block = base_power;  ---> a^b

        int exponent = -1;
        for (int i = 1; i <= (((p - 1) + block - 1) / block); i++)   --> ceil((p-1)/b)
        {
                auto temp = lb(all(arg_base_power), mp(base_power_block, 0));
                if (temp != arg_base_power.end()
                     &&temp->first == base_power_block)
                {
                        exponent = (i * block - temp->second) % (p - 1);
                        break;
                }
                base_power_block = (1ll * base_power * base_power_block) % p;  ---> a^ib
        }

        return exponent;
}
```