离散对数

定义

前置知识：阶与原根。

离散对数的定义方式和对数类似。取有原根的正整数模数 m，设其一个原根为 g. 对满足 $(a,m)=1$ 的整数 a，我们知道必存在唯一的整数 $0\leq k<\varphi(m)$ 使得

$$g^k\equiv a\pmod m$$

我们称这个 k 为以 g 为底，模 m 的离散对数，记作 $k=\operatorname{ind}_g a$，在不引起混淆的情况下可记作 $\operatorname{ind} a$.

显然 $\operatorname{ind}_g 1=0$，$\operatorname{ind}_g g=1$.

性质

离散对数的性质也和对数有诸多类似之处。

性质

设 g 是模 m 的原根，$(a,m)=(b,m)=1$，则：

1. $\operatorname{ind}_g(ab)\equiv\operatorname{ind}_g a+\operatorname{ind}_g b\pmod{\varphi(m)}$

   进而 $(\forall n\in\mathbf{N})$，$\operatorname{ind}_g a^n\equiv n\operatorname{ind}_g a\pmod{\varphi(m)}$
2. 若 g_1 也是模 m 的原根，则 $\operatorname{ind}_g a\equiv\operatorname{ind}_{g_1} a\cdot\operatorname{ind}_g g_1\pmod{\varphi(m)}$
3. $a\equiv b\pmod m\iff\operatorname{ind}_g a=\operatorname{ind}_g b$

证明

1. $g^{\operatorname{ind}_g(ab)}\equiv ab\equiv g^{\operatorname{ind}_g a}g^{\operatorname{ind}_g b}\equiv g^{\operatorname{ind}_g a+\operatorname{ind}_g b}\pmod m$
2. 令 $x=\operatorname{ind}_{g_1} a$，则 $a\equiv g_1^x\pmod m$. 又令 $y=\operatorname{ind}_g g_1$，则 $g_1\equiv g^y\pmod m$.

   故 $a\equiv g^{xy}\pmod m$，即 $\operatorname{ind}_g a\equiv xy\equiv\operatorname{ind}_{g_1} a\cdot\operatorname{ind}_g g_1\pmod{\varphi(m)}$
3. 注意到

$$\begin{aligned}\operatorname{ind}_g a=\operatorname{ind}_g b&\iff\operatorname{ind}_g a\equiv\operatorname{ind}_g b\pmod{\varphi(m)}\\&\iff g^{\operatorname{ind}_g a}\equiv g^{\operatorname{ind}_g b}\pmod m\\&\iff a\equiv b\pmod m\end{aligned}$$

大步小步算法

目前离散对数问题仍不存在多项式时间经典算法（离散对数问题的输入规模是输入数据的位数）。在密码学中，基于这一点人们设计了许多非对称加密算法，如 Ed25519。

在算法竞赛中，BSGS（baby-step giant-step，大步小步算法）常用于求解离散对数问题。形式化地说，对 $a,b,m \in \mathbf{Z}^+$，该算法可以在 $O(\sqrt{m})$ 的时间内求解

$$a^x \equiv b \pmod m$$

其中 $a \perp m$。方程的解 x 满足 $0 \le x < m$.（注意 m 不一定是素数）

算法描述

令 $x = A\left\lceil \sqrt{m} \right\rceil - B$，其中 $0 \le A, B \le \left\lceil \sqrt{m} \right\rceil$，则有 $a^{A\left\lceil \sqrt{m} \right\rceil - B} \equiv b \pmod m$，稍加变换，则有 $a^{A\left\lceil \sqrt{m} \right\rceil} \equiv ba^B \pmod m$.

我们已知的是 a,b，所以我们可以先算出等式右边的 ba^B 的所有取值，枚举 B，用 `hash` / `map` 存下来，然后逐一计算 $a^{A\left\lceil \sqrt{m} \right\rceil}$，枚举 A，寻找是否有与之相等的 ba^B，从而我们可以得到所有的 x，$x = A\left\lceil \sqrt{m} \right\rceil - B$.

注意到 A, B 均小于 $\left\lceil \sqrt{m} \right\rceil$，所以时间复杂度为 $\Theta\left(\sqrt{m}\right)$，用 `map` 则多一个 $\log$.

为什么要求 a 与 m 互质

注意到我们求出的是 A, B，我们需要保证从 $a^{A\left\lceil \sqrt{m} \right\rceil} \equiv ba^B \pmod m$ 可以推回 $a^{A\left\lceil \sqrt{m} \right\rceil - B} \equiv b \pmod m$，后式是前式左右两边除以 a^B 得到，所以必须有 $a^B \perp m$ 即 $a \perp m$.

进阶篇

对 $a, b \in \mathbf{Z}^+$，$p \in \mathbf{P}$，求解

$$x^a \equiv b \pmod p$$

该问题可以转化为 BSGS 求解的问题。

由于式子中的模数 p 是一个质数，那么 p 一定存在一个原根 g. 因此对于模 p 意义下的任意的数 x $(1 \le x < p)$ 有且仅有一个数 i $(0 \le i < p-1)$ 满足 $x = g^i$.

方法一

我们令 $x = g^c$，g 是 p 的原根（我们一定可以找到这个 g 和 c），问题转化为求解 $(g^c)^a \equiv b \pmod p$. 稍加变换，得到

$$(g^a)^c \equiv b \pmod p$$

于是就转换成了 BSGS 的基本模型了，可以在 $O(\sqrt{p})$ 解出 c，这样可以得到原方程的一个特解 $x_0 \equiv g^c \pmod p$.

方法二

我们仍令 $x = g^c$，并且设 $b = g^t$，于是我们得到

$$g^{ac} \equiv g^t \pmod p$$

方程两边同时取离散对数得到

$$ac \equiv t \pmod{\varphi(p)}$$

我们可以通过 BSGS 求解 $g^t \equiv b \pmod p$ 得到 t，于是这就转化成了一个线性同余方程的问题。这样也可以解出 c，求出 x 的一个特解 $x_0 \equiv g^c \pmod p$.

找到所有解

在知道 $x_0 \equiv g^c \pmod p$ 的情况下，我们想得到原问题的所有解。首先我们知道 $g^{\varphi(p)} \equiv 1 \pmod p$，于是可以得到

$$\forall\, t \in \mathbf{Z},\ x^a \equiv g^{c \cdot a + t \cdot \varphi(p)} \equiv b \pmod p$$

于是得到所有解为

$$\forall\, t \in \mathbf{Z}, a \mid t \cdot \varphi(p),\ x \equiv g^{c + \frac{t \cdot \varphi(p)}{a}} \pmod p$$

对于上面这个式子，显然有 $\frac{a}{(a, \varphi(p))} \mid t$. 因此我们设 $t = \frac{a}{(a, \varphi(p))} \cdot i$，得到

$$\forall\, i \in \mathbf{Z}, x \equiv g^{c + \frac{\varphi(p)}{(a, \varphi(p))} \cdot i} \pmod p$$

这就是原问题的所有解。

实现

下面的代码实现的找原根、离散对数解和原问题所有解的过程。

参考代码

```
int gcd(int a, int b) { return a ? gcd(b % a, a) : b; }

int powmod(int a, int b, int p) {
  int res = 1;
  while (b > 0) {
    if (b & 1) res = res * a % p;
    a = a * a % p, b >>= 1;
  }
  return res;
}

// Finds the primitive root modulo p
int generator(int p) {
  vector<int> fact;
  int phi = p - 1, n = phi;
  for (int i = 2; i * i <= n; ++i) {
    if (n % i == 0) {
      fact.push_back(i);
      while (n % i == 0) n /= i;
    }
  }
  if (n > 1) fact.push_back(n);
  for (int res = 2; res <= p; ++res) {
    bool ok = true;
    for (int factor : fact) {
```

```
      if (powmod(res, phi / factor, p) == 1) {
        ok = false;
        break;
      }
    }
    if (ok) return res;
  }
  return -1;
}

// This program finds all numbers x such that x^k=a (mod n)
int main() {
  int n, k, a;
  scanf("%d %d %d", &n, &k, &a);
  if (a == 0) return puts("1\n0"), 0;
  int g = generator(n);
  // Baby-step giant-step discrete logarithm algorithm
  int sq = (int)sqrt(n + .0) + 1;
  vector<pair<int, int>> dec(sq);
  for (int i = 1; i <= sq; ++i)
    dec[i - 1] = {powmod(g, i * sq * k % (n - 1), n), i};
  sort(dec.begin(), dec.end());
  int any_ans = -1;
  for (int i = 0; i < sq; ++i) {
    int my = powmod(g, i * k % (n - 1), n) * a % n;
    auto it = lower_bound(dec.begin(), dec.end(), make_pair(my, 0));
    if (it != dec.end() && it->first == my) {
      any_ans = it->second * sq - i;
      break;
    }
  }
  if (any_ans == -1) return puts("0"), 0;
  // Print all possible answers
  int delta = (n - 1) / gcd(k, n - 1);
  vector<int> ans;
  for (int cur = any_ans % delta; cur < n - 1; cur += delta)
    ans.push_back(powmod(g, cur, n));
  sort(ans.begin(), ans.end());
  printf("%d\n", ans.size());
  for (int answer : ans) printf("%d ", answer);
}
```

扩展篇（扩展 BSGS）

对 $a,b,m\in\mathbf{Z}^+$，求解

$$a^x \equiv b \pmod m$$

其中 a,m 不一定互质。

当 $(a,m)=1$ 时，在模 m 意义下 a 存在逆元，因此可以使用 BSGS 算法求解。于是我们想办法让他们变得互质。

具体地，设 $d_1=(a,m)$. 如果 $d_1\nmid b$，则原方程无解。否则我们把方程同时除以 d_1，得到

$$\frac{a}{d_1}\cdot a^{x-1}\equiv\frac{b}{d_1}\pmod{\frac{m}{d_1}}$$

如果 a 和 $\frac{m}{d_1}$ 仍不互质就再除，设 $d_2=\left(a,\frac{m}{d_1}\right)$. 如果 $d_2\nmid\frac{b}{d_1}$，则方程无解；否则同时除以 d_2 得到

$$\frac{a^2}{d_1d_2}\cdot a^{x-2}\equiv\frac{b}{d_1d_2}\pmod{\frac{m}{d_1d_2}}$$

同理，这样不停的判断下去，直到 $a\perp\dfrac{m}{d_1d_2\cdots d_k}$.

记 $D=\prod_{i=1}^{k}d_i$，于是方程就变成了这样：

$$\frac{a^k}{D}\cdot a^{x-k}\equiv\frac{b}{D}\pmod{\frac{m}{D}}$$

由于 $a\perp\dfrac{m}{D}$，于是推出 $\dfrac{a^k}{D}\perp\dfrac{m}{D}$. 这样 $\dfrac{a^k}{D}$ 就有逆元了，于是把它丢到方程右边，这就是一个普通的 BSGS 问题了，于是求解 $x-k$ 后再加上 k 就是原方程的解啦。

注意，不排除解小于等于 k 的情况，所以在消因子之前做一下 $\Theta(k)$ 枚举，直接验证 $a^i\equiv b\pmod m$，这样就能避免这种情况。

习题

- SPOJ MOD 模板
- SDOI2013 随机数生成器
- SGU261 Discrete Roots 模板
- SDOI2011 计算器 模板
- Luogu4195【模板】exBSGS/Spoj3105 Mod 模板
- Codeforces - Lunar New Year and a Recursive Sequence
- LOJ6542 离散对数 index calculus 方法，非模板

参考资料

1. Discrete logarithm - Wikipedia
2. 潘承洞，潘承彪。初等数论。
3. 冯克勤。初等数论及其应用。

本页面最近更新：2023/11/5 12:51:17，更新历史
发现错误？想一起完善？ 在 GitHub 上编辑此页！
本页面贡献者：H-J-Granger, StudyingFather, Alpha1022, ChungZH, countercurrent-time, dkz051, Entertainer, FFjet, Great-designer, Henry-ZHR, hly1204, hsfzLZH1, iamtwz, Ir1d, isdanni, Kelatte, ksyx, Lampese,