大步小步算法

Baby-step Giant-step

BSGS

解决**离散对数**问题：已知 $a \perp m$，求解：

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

其中，$0 \leqslant x < m$.

注意：只需要 $a \perp m$，不需要 m 是质数。

Idea：假设解 $x = np - q$，其中 n 是一个特定的数（事实上取 $n = \lceil\sqrt{m}\rceil$），这里，$p \in \left[1, \left\lceil\frac{m}{n}\right\rceil\right]$，$q \in [0, n]$。那么：$a^{np-q} \equiv b \pmod m$. 根据互质的条件，得到：

$$a^{np} \equiv ba^q \pmod m$$

所以我们可以枚举 q，把 ba^q 的值存在哈希表中（`unordered_map`），然后枚举 p，查找哈希表中是否有该值，如果有，那么 $np - q$ 就是一个解（从小到大枚举 p，那找到的第一个解就是**最小非负整数解**，因为 p 是"大步"）。复杂度显然是 $O(n + \frac{m}{n})$，自然取 $n = \lceil\sqrt{m}\rceil$ 时达到最小。

Complexity：$O(\sqrt{p})$

ATT：特殊定义 $0^0 = 1$，如果题目不认可该定义，认为 $a = 0, b = 1$ 时无解，特判即可。

Code：

```
int BSGS(int a, int b, int m){
    // solve a^x = b (mod m)
    unordered_map<int, int> val;
    int sq = sqrt(m) + 1;
    LL an = 1;
    for(int i = 1; i <= sq; i++)     an = an * a % m;
    for(LL q = 0, cur = b; q <= sq; cur = cur * a % m, q++)
        val[cur] = q;
    for(LL p = 1, cur = an; p <= sq; cur = cur * an % m, p++)
        if(val.count(cur))
            return sq * p - val[cur];
    return -1;
}
```

exBSGS

当 a 和 m 不互质时，求解：

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

Idea：a 和 m 不互质，那我们就把它变互质。设 $d_1 = \gcd(a, m)$，根据贝祖定理，若 $d_1 \nmid b$，则无解；否则，根据同余式的除法法则，有：$\frac{a}{d_1}a^{x-1} \equiv \frac{b}{d_1} \pmod{\frac{m}{d_1}}$；如果 a 和 $\frac{m}{d_1}$ 仍然不互质，就继续设 $d_2 = \gcd\left(a, \frac{m}{d_1}\right)$，得到：$\frac{a^2}{d_1d_2}a^{x-2} \equiv \frac{b}{d_1d_2} \pmod{\frac{m}{d_1d_2}}$；一直这么做下去直到互质为止，得到方程：

$$\frac{a^k}{d_1d_2\cdots d_k}a^{x-k} \equiv \frac{b}{d_1d_2\cdots d_k} \pmod{\frac{m}{d_1d_2\cdots d_k}}$$

这就变成了一个普通的 BSGS，求解后加上 k 即是原解。

如果解出来 $x<k$ 怎么办？这意味着执行过程中出现了 $\frac{a^{k'}}{d_1\cdots d_{k'}}\equiv\frac{b}{d_1\cdots d_{k'}}$ 的情况，判断一下即可。

Complexity：$O(\sqrt{m}+\lg^2 m)$

Code：

```cpp
int exBSGS(int a, int b, int m){
    // solve a^x = b (mod m)
    int A = 1, k = 0, d;
    while((d = gcd(a, m)) > 1){
        if(b == A)  return k;
        if(b % d)   return -1;
        A = 1ll * A * (a / d) % m;
        b /= d, m /= d, k++;
    }

    unordered_map<int, int> val;
    int sq = sqrt(m) + 1;
    LL an = 1;
    for(int i = 1; i <= sq; i++)    an = an * a % m;
    for(LL q = 0, cur = b; q <= sq; cur = cur * a % m, q++)
        val[cur] = q;
    for(LL p = 1, cur = an * A % m; p <= sq; cur = cur * an % m, p++)
        if(val.count(cur))
            return sq * p - val[cur] + k;
    return -1;
}
```