整数論テクニック集 DRAFT

夕叢霧香 (Kirika Yuumura, @kirika_comp)

2018 年 2 月 22 日

謝辞

全ての助けの中で、dr.ken (@drken1215) によるものが一番大きいです。彼の協力により、問題がたくさん集まりました。これを読んでくださる読者の方にも感謝の念を禁じえません。

1　はじめに

これは、競技プログラミングで使える整数論のテクニックをまとめた文章です。AtCoder の青から赤下位程度をターゲットにしています。

整数論の問題は得てして「地頭ゲー」などと呼ばれやすいですが、実はそうではなく、知識を持っていることで解ける問題が多いです。しかし、その知識については、日本の競技プログラマーたちには、あまり知られていないように見えます。そのため、この文章を書くことにしました。

基本的に、実際に出題されている問題で利用されるテクニックを、実際の問題と共に紹介するという方式で書いていきます。そのため、ネタバレは非常に多いです。

各セクションにはレベルを振ってあります。筆者が感じたおよその難易度を表しています。

コメント等は Twitter (@kirika_comp) までお願いします。

2　mod p 上の計算

2.1　基本: 整数の加減乗除 (Lv. 1)

計算結果が大きすぎるため、 mod (10^9+7) で出力せよ、という問題が結構あります。このような問題の場合は、最終結果を mod (10^9+7) するのではなく、途中結果も mod (10^9+7) することで、途中結果が大きくなりすぎるのを防ぐことができます。

2.2　基本: 分数の加減乗除 (Lv. 2)

たまに、分数についての言及があることがあります。大抵以下のような形をしています。

答えは分数 A/B になる。このとき、B の mod (10^9+7) での逆元を B^{-1} として、$A \times B^{-1}$ mod (10^9+7) を出力せよ。

これも、特別な配慮などはせずに、途中結果を $\bmod (10^9+7)$ で保持しておくだけで、計算が正しく行えます。

例題

- Codeforces Round #465 (Div. 2) D. Fafa and Ancient Alphabet

COLUMN

専門用語を使うと、これは環準同型 $\mathbb{Z} \to \mathbb{Z}/(10^9+7)\mathbb{Z}$ が、体の準同型 $\mathbb{Q} \to \mathbb{Z}/(10^9+7)\mathbb{Z}$ に拡張できる、ということができます (ただし、細かい点を除く)。興味のある人は、調べてみてください。

3 二分累乗法 (Lv. 1)

二分累乗法は基本 TODO

3.1 群的構造を見つけて二分累乗する (Lv. 2)

例題

$\cos\theta = \frac{d}{l}$ であるような θ に対して、$\cos t\theta$ は有理数であることが証明できる。$l\cos t\theta = \frac{p}{q}$ であるとき、$pq^{-1} \bmod (10^9+7)$ を求めよ。

- 部分点 (15/100 点): t は 2 冪である。つまり、ある 0 以上の整数 p について $t = 2^p$。
- 満点 (100 点): $1 \le t \le 10^{18}$, t は整数。

(出典: CodeChef February Challenge 2018 (FEB18) » Broken Clock (BROCLK))

部分点解法は、2 倍角の公式 $\cos 2\theta = 2\cos^2\theta - 1$ を利用して、p 回の計算を行うことでできます。

問題は満点解法の方で、愚直にやると t 倍角の公式が必要になってきて、不可能とは言わないまでも面倒です。そこで、ド・モアブルの公式 (de Moivre's formula)

$$\cos t\theta + i\sin t\theta = (\cos\theta + i\sin\theta)^t$$

を利用して、強引に冪乗公式に持っていくことを考えましょう。式の形から、$\cos\theta + i\sin\theta$ なる数の計算、およびそれの累乗の計算ができれば良いことになります。ここで、以下のようにペアを用いて数を表現することにします:

$$\langle a, b\rangle \mapsto a + ib\sin\theta$$

これにより掛け算、それゆえ冪乗が、整数ペアの上の演算として実装できます。どのようにするかみていきましょう。

まず、$\cos\theta + i\sin\theta$ はもちろん、$\langle\cos\theta, 1\rangle = \langle d/l, 1\rangle$ として表現されます。掛け算についてですが、

$$(a,b)\times(c,d) = (a+ib\sin\theta)(c+id\sin\theta) = (ac - bd\sin^2\theta) + i(ad+bc)\sin\theta = \langle ac + bd(\cos^2\theta - 1), ad+bc\rangle$$

より、問題なく実装することができます。これによりべき乗も問題なく実装でき、問題が解けます。

ソースコード 1　BROCLK.cpp

```
1 #include<iostream>
```

```cpp
using namespace std;
typedef long long lint;
typedef pair<lint,lint>pll;
const lint mod=1e9+7;

lint powmod(lint x,lint e){
  lint c=1;
  for(int i=63;i>=0;--i){
    c=c*c%mod;
    if(e&1LL<<i)c=c*x%mod;
  }
  return c;
}

pll mul_pll(pll a,pll b,lint c){
  lint x=a.first*b.first%mod;
  lint nx=a.second*b.second%mod;
  x=(x+nx*c)%mod;
  lint y=(a.first*b.second+a.second*b.first)%mod;
  return pll(x,y);
}

pll pow_pll(pll a,lint e,lint c){
  pll p(1,0);
  for(int i=63;i>=0;--i){
    p=mul_pll(p,p,c);
    if(e&1LL<<i)p=mul_pll(p,a,c);
  }
  return p;
}

int main(){
  int tt;
  cin>>tt;
  while(tt--){
    lint l,d,t;
    cin>>l>>d>>t;
    lint cos=d*powmod(l,mod-2)%mod;
    lint s2=(cos*cos%mod)+mod-1;
    s2%=mod;
    lint ans=pow_pll(pll(cos,1),t,s2).first;
    cout<<ans*l%mod<<endl;
  }
}
```

4 mod p のアルゴリズム

4.1 基礎知識

4.1.1 フェルマーの小定理 (Lv. 1)

p が素数で $a \not\equiv 0 \pmod{p}$ のとき、 $a^{p-1} \equiv 1 \pmod{p}$ が成立します。

4.1.2 平方剰余 (Lv. 3)

$a \equiv x^2 \pmod{p}$ となる x が存在する場合、a を $\bmod p$ における **平方剰余** (*quadratic residue*)、そうでない場合 a を**平方非剰余** (*quadratic non-residue*) と呼びます。p が奇素数の時、0 を除くと平方剰余と平方非剰余の割合は 1:1 です。また、$a \not\equiv 0 \pmod{p}$ のとき $a^{(p-1)/2}$ は $\bmod p$ で 1 か -1 かのどちらかですが、a が平方剰余のとき 1、平方非剰余のとき -1 です。

例 **4.1.** $p = 13$ の場合を考えます。このとき、平方剰余は $0 = 0^2, 1 = 1^2, 3 \equiv 4^2, 4 = 2^2, 9 = 3^2, 10 \equiv 6^2, 12 \equiv 5^2 \pmod{13}$ の 7 個です。0 を除外すると 1,3,4,9,10,12 の 6 個で、mod 13 の 0 以外の同値類 12 個のうち、ちょうど半分が平方剰余、もう半分が平方非剰余です。なお、適当な平方非剰余 z をとると、0 以外の平方剰余に z を掛けたものはすべて平方非剰余です。例えば、$z = 2$ とすると、1,3,4,9,10,12 に 2 を掛けて mod 13 したもの (2,6,8,5,7,11) はすべて平方非剰余です。

4.1.3 加法群 (Lv. 4)

$\mathbb{Z}/p\mathbb{Z}$ の話 TODO

4.1.4 乗法群 (Lv. 4)

$(\mathbb{Z}/p\mathbb{Z})^*$ の話 TODO

平方剰余全体は $(\mathbb{Z}/p\mathbb{Z})^*$ の部分群 TODO

4.2 mod_sqrt, Tonelli-Shanks のアルゴリズム (Lv. 3)

4.2.1 問題

ある x について $a \equiv x^2 \pmod{p}$ が成り立つ a が与えられる。この時、x を求めよ。

4.2.2 解法

まず、簡単のため $p = 2$ の場合を除外します (このときは $a^2 \equiv a \pmod{2}$ なので簡単)。また $a \equiv 0 \pmod{p}$ の場合も除外します ($0^2 = 0$ なので簡単)。p が $\bmod 4$ で 3 の時は簡単です。$x \equiv a^{(p+1)/4}$ とすると、$x^2 \equiv a^{(p+1)/2}$ です。ここで、$a \equiv y^2$ となる y が存在するので、$a^{(p-1)/2} \equiv y^{p-1} \equiv 1 \pmod{p}$ です。だから、$x^2 \equiv a$ が成り立ちます。

p が $\bmod 4$ で 1 の時は結構複雑なことをします。ここでは Tonelli-Shanks の方法と呼ばれるアルゴリズムを説明します。

4.2.3 Tonelli-Shanks (トネリ-シャンクス) のアルゴリズム

Reference: `https://en.wikipedia.org/wiki/Tonelli%E2%80%93Shanks_algorithm`

注意: 以下の疑似コードでは代入は全部同時に行います。特に 5. で、t に代入する値は前の c によって決まります。

注意 2: 本来の Tonelli-Shanks とは違いますが、不変量を考えることで筆者が復元できたのが以下のアルゴリズムなので、こちらの方が理解しやすいと思います。(効率は悪い)

終了時には $m = 1$ なので、$t \equiv 1 \pmod{p}$ になっているはずで、そのときの r が求める値です。(不変量

Algorithm 1 単純化された Tonelli-Shanks のアルゴリズム

入力: $p(\geq 3)$: 奇素数, $a(\not\equiv 0 \pmod p)$: 平方剰余

出力: $r^2 \equiv a \pmod p$ を満たす r

$p = q \times 2^s + 1$ とします。 ($s \geq 1, q$ は奇数)

1. $z^{(p-1)/2} \equiv -1 \pmod p$ となるような z を選ぶ。このような z は確率 1/2 でヒットするため、何個か試せば必ず見つかる。

2. $m := s, c :\equiv z^q, t :\equiv a^q, r :\equiv a^{(q+1)/2}$ とする。以降不変量 $\boldsymbol{r^2 \equiv at \pmod p, t^{2^{m-1}} \equiv 1 \pmod p, c^{2^{m-1}} \equiv -1 \pmod p}$ を崩さないように注意して操作する。

3. 以降 m を減らしていく。m が 1 なら終了。そうでなければ、$t^{2^{m-2}} \equiv 1 \pmod p$ なら 4. へ、そうでなければ 5. へ行く。

4. $c :\equiv c^2 \pmod p, m := m - 1$, 6. へ行く。

5. $c :\equiv c^2 \pmod p, t :\equiv c^2 t \pmod p, r :\equiv cr \pmod p, m := m - 1$ を全て同時に代入する, 6. へ行く。

6. 3. へ行く。

$r^2 \equiv at \pmod p$ に注意。)

C++ での実装はソースコード 2 のようになります。

ソースコード 2　tonelli-shanks.cpp

```cpp
#include<random>
using namespace std;
typedef long long lint;

lint powmod(lint a,lint e,lint p){
  lint r=1;
  for(int i=63;i>=0;--i){
    r=r*r%p;
    if(e&1LL<<i)r=r*a%p;
  }
  return r;
}

// p:素数, a は 0 でなく、平方剰余
lint simplified_tonelli_shanks(lint p,lint a){
  mt19937 mt;
  if(powmod(a,(p-1)/2,p)!=1)return -1;
  lint q=p-1;
  lint m=0;
  while(q%2==0)q/=2,m++;
  lint z;
  do{
    z=mt()%p;
  }while(powmod(z,(p-1)/2,p)!=p-1);
  lint c=powmod(z,q,p);
  lint t=powmod(a,q,p);
  lint r=powmod(a,(q+1)/2,p);
  for(;m>1;--m){
    lint tmp=powmod(t,1<<(m-2),p);
    if(tmp!=1)
      r=r*c%p,t=t*(c*c%p)%p;
    c=c*c%p;
```

```
33    }
34    return r;
35  }
```

例を挙げて見ていきましょう。$p = 41, a = 8$ とします。

$p = 5 * 2^3 + 1$ なので、$q = 5, s = 3$ です。z として、ここでは 7 をとります。

$m := 3, c :\equiv 7^5 = 16807 \equiv 38, t :\equiv 8^5 = 32768 \equiv 9, r \equiv 8^3 = 512 \equiv 20$ となります。 (mod 41 は適宜省略)

m	c	t	r
3	38	9	20
2	9	40	22
1	40	1	34

よって、$x \equiv \pm 34 (= \mp 7)$ が答えになります。

以上のアルゴリズムで、4. のパートに無駄があります。4. では c と m しか変更していないので、$t^{2^i} \not\equiv 1 \pmod{p}$ となる最大の i が見つけられれば、4. の操作をまとめることができます。このアイデアを使うのが、本来の Tonelli-Shanks のアルゴリズムです。

(TODO Wikipedia の Tonelli-Shanks の説明)

COLUMN

群論的なアプローチをすると、もっと綺麗な見方が得られます。$(\mathbb{Z}/p\mathbb{Z})^* \cong \mathbb{Z}/(p-1)\mathbb{Z} \cong (\mathbb{Z}/2^s\mathbb{Z}) \times (\mathbb{Z}/q\mathbb{Z})$ の 2 冪成分 (2-Sylow 部分群) $H = \mathbb{Z}/2^s\mathbb{Z}$ を考えます。このとき、z と t は H の元であることがわかります。t が 1 になるように、うまく H^2 の元で調整しているわけです。

5 平方剰余の相互法則 (Lv. 4)

5.1 ルジャンドル記号

TODO ルジャンドル記号

5.2 平方剰余の相互法則

以下の定理が成り立つことが知られています。

定理 **5.1** (平方剰余の相互法則)**.** $p, q \geq 3$ を奇素数とする。このとき、以下が成立する。

$$\left(\frac{p}{q}\right)\left(\frac{q}{p}\right) = (-1)^{\frac{p-1}{2} \times \frac{q-1}{2}}$$

定理 **5.2** (補充法則)**.** $p \geq 3$ を奇素数とする。このとき、以下が成立する。

$$\left(\frac{-1}{p}\right) = (-1)^{\frac{p-1}{2}}, \left(\frac{2}{p}\right) = (-1)^{\frac{p^2-1}{8}}$$

これを利用することで、ルジャンドル記号を計算できます。

例 **5.3.** $p \neq 2, 3$ を、2,3 以外の素数とします。このとき、$\left(\frac{3}{p}\right)$ は、p を 12 で割った余りで完全に決まります。

$$\left(\frac{3}{p}\right) = (-1)^{\frac{p-1}{2}} \left(\frac{p}{3}\right)$$

ここで、

$$\left(\frac{p}{3}\right) = \begin{cases} 1 & \text{if } p \equiv 1 \pmod 3 \\ -1 & \text{if } p \equiv 2 \pmod 3 \end{cases}$$

なので、

$$(-1)^{\frac{p-1}{2}} = \begin{cases} 1 & \text{if } p \equiv 1 \pmod 4 \\ -1 & \text{if } p \equiv 3 \pmod 4 \end{cases}$$

と合わせ、

$$\left(\frac{3}{p}\right) = \begin{cases} 1 & \text{if } p \equiv 1, 11 \pmod{12} \\ -1 & \text{if } p \equiv 5, 7 \pmod{12} \end{cases}$$

が得られます。

5.3 2 次体

5.4 有限体

任意の素数 p と正の整数 e に対して、p^e 要素の有限体が存在します。逆に、有限体の要素数は、必ず p^e の形で表せます。このような有限体は、一意に存在します。これを $GF(p^e)$ と表記することにします。

5.5 フロベニウス写像

$\mathrm{Frob}\colon GF(p^e) \to GF(p^e), \mathrm{Frob}(x) := x^p$ を**フロベニウス写像**と呼びます。 TODO

命題 **5.4.** Frob は e 回適用すると元に戻る。つまり、$\mathrm{Frob}^e(x) = x$。

命題 **5.5.** $x, \mathrm{Frob}(x), \mathrm{Frob}^2(x), \ldots, \mathrm{Frob}^{e-1}(x)$ は全て共役 (TODO definition)。つまり、TODO。

5.6 応用例

例題

数列 $a_0 = 2, a_{n+1} = a_n(a_n + 4)$ がある。このとき、素数 M に対して、$a_N \bmod M$ を求めよ。
(出典: yukicoder No.613 Solitude by the window)

この問題は、一般項を求めるところが一番難しく、一般項を求めた後は数論的な考察を進めるだけで解けます。ここでは、$a_n = (2+\sqrt{3})^{2^n} + (2-\sqrt{3})^{2^n} - 2$ であることがわかっているとして、この状態から問題を解いてみましょう。$a_n \bmod M$ が計算できれば良いです。

$(2+\sqrt{3})^{2^n} \bmod M$ が計算できれば万事解決です。簡単のため、M が 2 でも 3 でもないとしましょう。先にも述べた (TODO) 通り、3 が $\bmod M$ で平方剰余なら (つまり $\left(\frac{3}{M}\right) = 1$ なら)、議論は $GF(M)$ の中で完結

できます。3 が mod M で平方非剰余 (つまり $\left(\frac{3}{M}\right) = -1$) の場合を考えます。このとき、$GF(M)$ に $\sqrt{3}$ を添加して拡大したものは、$GF(M^2)$ と同型になります。

$$GF(M)(\sqrt{3}) \cong GF(M^2)$$

ここで、$\mathrm{Frob}(2+\sqrt{3}) = (2+\sqrt{3})^M \in GF(M^2)$ がどのような元になるかを考えてみましょう。$2+\sqrt{3}$ の共役は自分自身と $2-\sqrt{3}$ のみなので、$\mathrm{Frob}(2+\sqrt{3}) = 2-\sqrt{3}$ でなければなりません。これから、$(2+\sqrt{3})^{M+1} = (2-\sqrt{3})(2+\sqrt{3}) = 1$ であることが分かります。

6 mod_sqrt その 2(Lv. 4)

http://pekempey.hatenablog.com/entry/2017/02/03/220150
Cipolla のアルゴリズム

7 ペル方程式 (Lv. 4)

例題

一辺 a メートルの正方形がある。この正方形から、一辺 b メートルの正方形を n 個切り出すことを考える。ただし、切り出されなかった部分の面積は、元の正方形の面積の 50% 以上でなければならない。つまり、$a^2 - nb^2 \geq a^2/2$ が必要である。
ここで切り出されなかった部分のうち、a^2 平方メートルの 50% を越える部分の面積 (つまり $(a^2/2 - nb^2)$ 平方メートル) を最小化したい。n が与えられるので、最小値を与える正の整数 a, b を与えよ。
制約: $1 \leq n \leq 10000$
(出典: Aizu Online Judge 2116: Subdividing a Land (ACM-ICPC Japan Alumni Group Practice Contest, for World Finals, Tokyo, Japan, 2008-02-23), `http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=2116`)

まず、 $2n$ が平方数の場合、$a = \sqrt{2n}, b = 1$ が最小値 0 を与えることが自明です。そうでない場合を考えます。n が十分に大きい場合には、(気の遠くなるような計算 TODO) $a^2 - 2nb^2 \geq 0$ ならば n 個詰めることが確実にできることが保証されるので、結局 $a^2 - 2nb^2 \geq 0$ の条件つきで、$a^2 - 2nb^2$ の最小値を与える a, b を計算すれば良いことが分かります。

ここで、以下の事実が知られています。

定理 **7.1.** n が平方数でない正の整数のとき、方程式 $x^2 - ny^2 = 1$ は、正の整数解 (x, y) を必ず持つ。

このような方程式を**ペル方程式** (*Pell's equation*) と呼び、このような (x, y) のうち、x が最小のものを**基本解** (*fundamental solution*) と呼びます。この問題では、基本解が計算できれば良いでしょう。

ペル方程式の基本解を計算するアルゴリズムを説明します。連分数を使う、以下のアルゴリズムが広く知られています。TODO

索引