整数論テクニック集 DRAFT

夕叢霧香 (Kirika Yuumura, @kirika_comp)

2018 年 2 月 26 日

謝辞

全ての助けの中で、dr.ken (@drken1215) によるものが最大です。彼の協力により、問題がたくさん集まりました。その他、以下の方々にも助けをいただきました:

- minus3theta (@minus3theta)
- noicwt (@noicwt)
- p 進大好き bot (@non_archimedean)
- かならい (@sugarknri)

感謝します。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。ご注意ください。

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　基本: アルゴリズムについて

2.1　アルゴリズムにおける不変量

アルゴリズムを考える時は、ループや再帰呼び出しのある地点で、どのような条件が常に成り立っているか、どのような値が保たれているかを考察することが、アルゴリズムの理解を助けることがあります。このような条件や値のことを、**不変量** (*invariant*) と呼びます。

例として、拡張ユークリッドの互除法の実装を与えます。ここで、各再帰呼び出しの `return` 直前に

$ax + by = \gcd(a, b)$ が成り立っていることに注意してください。

ソースコード 1　extgcd.cpp

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

lint ext_gcd(lint a,lint b,lint&x,lint&y){
  if(b==0){
    x=1;y=0;return a;
  }
  lint q=a/b;
  lint g=ext_gcd(b,a-q*b,x,y);
  lint z=x-q*y;
  x=y;y=z;
  return g;
}


int main(){
  lint a,b;
  cin>>a>>b;
  lint x,y;
  lint g=ext_gcd(a,b,x,y);
  cout<<a<<" "
      <<b<<" "<<x<<" "<<y
      <<endl;
  cout<<g<<endl;

}
```

3　mod p 上の計算

3.1　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。

なお除算については、次のサブセクションで説明する逆元を掛けることで実装することができます。

3.2　基本: 逆元 (Lv. 1)

p を素数とします。$1 \leq r < p$ の時、$rs \equiv 1 \pmod{p}, 1 \leq s < p$ を満たす s がただ一つ存在します。これを r の mod p における**逆元** (*inverse element*) と呼びます。逆元の計算には、以下の 2 種類の方法があります:

1. $r^{p-1} \equiv 1 \pmod{p}$ (フェルマーの小定理、定理 5.1) を利用して、$r^{p-2} \bmod p$ を計算する。
2. 拡張ユークリッドの互除法を使う。

実装が単純なのは 1. ですが、多くの場合で 2. の方が高速に動作します。ライブラリを作る場合は、2. の方で作ると良いでしょう。

以下に両方の実装を与えます。 実装 TODO

3.3 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 $\bmod (10^9+7)$ での逆元を B^{-1} として、$A \times B^{-1} \bmod (10^9+7)$ 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。

問題

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

COLUMN

専門用語を使うと、これは素数 p について自然に定まる環準同型 $\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ が、環準同型 $\mathbb{Z}_{(p)} \to \mathbb{Z}/p\mathbb{Z}$ に拡張できる、ということができます (ただし、$\mathbb{Z}_{(p)}$ は、$\mathbb{Z}$ のイデアル (p) による**局所化**と呼ばれるもので、分子が任意、分母が p の倍数でないような分数全体からなる環)。興味のある人は、「局所化」や「附値環」などの単語で調べてみてください。

4 二分累乗法 (Lv. 1)

二分累乗法とよばれる基本テクニックがあります。

ある種の「掛け算」$x \cdot y$ が定義されているような代数的構造 (**モノイド**と呼びます) の上で、x の e 乗 $x^e = x \cdots\cdots x$ は $2\log_2 e$ 回以内の掛け算で計算できます。この方法を**二分累乗法** (*exponentiation by squaring*) と呼びます。

二分累乗法には 2 種類の方法があります。一つ目は、e の 2 進表記の小さい桁の方から計算をする方法で、もう一つは e の 2 進表記の大きい桁の方から計算をする方法です。整数の mod つき掛け算を例にして、両方のアルゴリズムを紹介します。

まず一つ目から紹介します。ソースコード 2 に実装を載せました。ループの各イテレーション終了時に $prod * cur^e = (answer) = x^e$ が成立していることに注意しましょう。最終的に $e = 0$ となって $prod$ に答えが入ります。

ソースコード 2　powmod-1.cpp

```
1  typedef long long lint;
2
3  lint powmod(lint x,lint e,lint mod){
4    lint prod=1;
5    lint cur=x;
6    while(e>0){
7      if(e%2==1)prod=prod*cur%mod;
8      cur=cur*cur%mod;
9      e/=2;
10   }
11   return prod;
12 }
```

次に二つ目を紹介します。e の上の桁から見ていくアルゴリズムです。ループの各イテレーション終了時に $prod^{2^i} * x^{e\%2^i} = (answer) = x^e$ が成り立つことに注意しましょう。

ソースコード 3　powmod-2.cpp

```
1  typedef long long lint;
2
3  lint powmod(lint x,lint e,lint mod){
4    lint prod=1;
5    for(int i=63;i>=0;--i){
6      prod=prod*prod%mod;
7      if(e&1LL<<i)prod=prod*x%mod;
8    }
9    return prod;
10 }
```

4.1　群的構造を見つけて二分累乗する (Lv. 2)

問題

$\cos\theta = \frac{d}{l}$ 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、

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

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 4　BROCLK.cpp

```
1  #include<iostream>
2  using namespace std;
```

```cpp
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;
  }
}
```

4.2 うまい変形で除算を回避する (Lv. 2)

問題

整数 A が、次のような 10 進表記で与えられる。

$$(A)_{10} = a_1^{L_1} \cdot a_2^{L_2} \cdot \cdots \cdot a_N^{L_N}$$

ここで、a_i は 10 進表記で与えられた整数、L_i は整数である。また $a_i^{L_i}$ は、a_i を文字列としてみなして、L_i 個連結したものを表し、$s \cdot t$ は文字列 s, t の連結を表す。このとき、A を B で割った余りを求めよ。
制約: $1 \leq N \leq 10000, 1 \leq a_i \leq 10^9, 1 \leq L_i \leq 10^9$

- 部分点 (99/100 点): $B = 10^9 + 7$。
- 満点 (100 点): $1 \leq B \leq 10^9 + 7$。B は素数とは限らない。

(出典: ARC020 C - A mod B Problem)

B が素数の場合、A は等比数列の総和の公式を使って、以下のような閉じた式の形で書けるので、計算することは簡単です。ここで、b_i は a_i の桁数です。

$$A = a_N f(10^{b_N}, l_N) + 10^{b_N \times l_N}(a_{N-1} f(10^{b_{N-1}}, l_{N-1}) + 10^{b_{N-1} \times l_{N-1}}(\ldots)), f(y,z) = 1 + y + \cdots + y^{z-1} = \frac{y^z - 1}{y - 1}$$

問題は B が素数でない場合です。$f(y,z)$ の計算の中で、$y-1$ による除算を行っていますが、$y-1$ と B が互いに素でない場合に、これは失敗します。これを回避するために、y^z を二分累乗法で計算するのを諦め、$f(y,z)$ を直接二分累乗法に似た方法で計算することを考えましょう。以下の等式が成立します:

$$f(y, 2z) = 1 + y + \cdots + y^{2z-1} = (1 + y + \cdots + y^{z-1})(1 + y^z) = (1 + y^z) f(y,z)$$

$$f(y, z+1) = 1 + y + \cdots + y^z = 1 + y(1 + y + \cdots + y^{z-1}) = 1 + y f(y,z)$$

これによって、z の偶奇に応じて場合分けしながら再帰を行うことで、$f(y,z)$ の値を除算なしで計算することができます。ソースコード TODO

5 mod p のアルゴリズム

5.1 基礎知識

5.1.1 フェルマーの小定理 (Lv. 1)

定理 **5.1.** p が素数で $a \not\equiv 0 \pmod{p}$ のとき、$a^{p-1} \equiv 1 \pmod{p}$ が成立する。

5.1.2 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 mod 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 mod 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 **5.2.** $p = 13$ 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。

5.1.3 加法群 (Lv. 4)

$\mathbb{Z}/p\mathbb{Z}$ の話 TODO 素数 p に対して、p で割った余り全体の集合を $\mathbb{Z}/p\mathbb{Z}$ と書きます。$\mathbb{Z}/p\mathbb{Z}$ は、加法を演算として、群の構造を持ちます。

5.1.4 乗法群 (Lv. 4)

$\mathbb{Z}/p\mathbb{Z}$ のうち、0 以外の元には乗法の逆元が存在することは、3.2 で確かめました。これらの元からなる集合を $(\mathbb{Z}/p\mathbb{Z})^*$ と表記し、$\mathbb{Z}/p\mathbb{Z}$ の**乗法群** (*multiplicative group*) と呼びます。

平方剰余全体は $(\mathbb{Z}/p\mathbb{Z})^*$ の部分群 TODO 平方剰余全体の集合をここでは H と書くことにします。H は $(\mathbb{Z}/p\mathbb{Z})^*$ の部分群です。5.1.2 でみたように、平方非剰余 z を適当にとると、zH と H は共通部分を持たず、また $H \cup zH = (\mathbb{Z}/p\mathbb{Z})^*$ です。

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

5.2.1 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。

5.2.2 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 mod 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 mod 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。

5.2.3 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 $r^2 \equiv at \pmod{p}$ に注意。)

C++ での実装はソースコード 5 のようになります。

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。

ソースコード 5　tonelli-shanks.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;
  }
  return r;
}
```

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。

6 平方剰余の相互法則 (Lv. 4)

6.1 ルジャンドル記号

ルジャンドル記号とは、以下で定義されるものです。

$$\left(\frac{a}{p}\right) := \begin{cases} 1 & a \text{ が平方剰余のとき} \\ -1 & a \text{ が平方非剰余のとき} \\ 0 & a \equiv 0 \pmod{p} \text{ のとき} \end{cases}$$

命題 **6.1.**

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

命題 **6.2.** ルジャンドル記号は乗法的である。つまり、

$$\left(\frac{a}{p}\right)\left(\frac{b}{p}\right) = \left(\frac{ab}{p}\right)$$

系 **6.3.** 対応 $a \mapsto \left(\frac{a}{p}\right)$ は群準同型 $(\mathbb{Z}/p\mathbb{Z})^* \to \{+1, -1\}$ を定める。

6.2 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 **6.4** (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 **6.5** (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 **6.6.** $p \neq 2, 3$ 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。

6.3 2 次体

有理数体 $\mathbb{Q}$ に有理数の平方根 $\sqrt{d}$ を添加した体 $\mathbb{Q}(\sqrt{d})$ を、*2* **次体** (*quadratic field*) と呼びます。

6.4 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 $\mathrm{GF}(p^e)$ と表記することにします。

6.5 フロベニウス写像

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

命題 **6.7.** Frob は e 回適用すると元に戻る。つまり、$\mathrm{Frob}^e(x) = x$。

命題 **6.8.** $x, \mathrm{Frob}(x), \mathrm{Frob}^2(x), \ldots, \mathrm{Frob}^{e-1}(x)$ は全て共役 (TODO definition)。つまり、TODO。

6.6 応用例

問題

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 $\bmod M$ 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。

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

ここで、$\mathrm{Frob}(2+\sqrt{3}) = (2+\sqrt{3})^M \in \mathrm{GF}(M^2)$ 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。

7 mod_sqrt その 2(Lv. 3)

Karp (1991) [1]

8 mod_sqrt その 3(Lv. 4)

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

9 多項式を使うテク (Lv. 4)

9.1 FFT

TODO 説明

9.2 フェルマーの小定理

定理 **9.1.** 素数 p に対して、以下の等式が成り立つ:

$$(x+1) \times (x+2) \times \cdots \times (x+(p-1)) \equiv x^{p-1} - 1 \pmod{p}$$

証明. まず、$(x+k)$ 同士は互いに素です。フェルマーの小定理から、$1 \le k \le p-1$ なる各 k について、$x+k | x^{p-1}-1$ が成り立ちます (整除は $\mathrm{GF}(p)[x]$ の上のもの)。以上より、$(x+1)\times(x+2)\times\cdots\times(x+(p-1)) | x^{p-1}-1$

です。この整除関係の左辺も右辺も $(p-1)$ 次なので、両者は定数倍の違いしかありません。x^{p-1} の係数を比べることで、その定数倍は 1 倍である、つまり両者が等しいことがわかります。 □

注意として、p が素数でない場合にこれを拡張しようとしても、失敗します。例えば $p=8$ の時、$(x+1)(x+3)(x+5)(x+7) \equiv x^4+6x^2+1 \not\equiv x^{\varphi(8)}-1 \pmod 8$ です。これは $\mathbb{Z}/p\mathbb{Z}$ が体にならず、$(\mathbb{Z}/p\mathbb{Z})[x]$ において (0 以外の) 0 次多項式が必ずしも可逆元ではないことに由来します。

この事実を応用して解ける問題を紹介します。以下の問題を考えましょう。

問題

正の整数 n と素数 p が与えられる。$[n] := \{1, \ldots, n\}$ として、整数 k に対して $f(n,k)$ を以下で定める:

$$f(n,k) := \sum_{S \subseteq [n], |S|=k} \prod_{x \in S} x$$

このとき、$p \nmid f(n,k)$ となるような $0 \le k \le n$ の個数を、$\bmod 10^9+7$ で求めよ。(テストケースは T ケース与えられる。)

制約: $1 \le T \le 4, 2 \le p \le 100000$, p は素数

- 部分点 (10/100 点): $n \le 5000$
- 部分点 (50/100 点): $n \le 100000$
- 満点 (100 点): $n < 10^{501}$

(出典: CodeChef February Challenge 2018 (FEB18) » Lucas Theorem (LUCASTH))

簡単な式変形、DP、あるいは実験などで、

$$f(n,k) = ((t+1)\times(2t+1)\times\cdots\times(nt+1) \text{ の } t^k \text{ の係数 })$$

であることがわかります。$P(n) := (t+1)\times(2t+1)\times\cdots\times(nt+1)$ と置きましょう。$P(n)$ の係数を愚直に計算することで、$O(n^2)$ 解法が得られます (10/100 点)。

これを高速化することを考えましょう。多項式の乗算を高速化したいので、FFT を使うことが思いつきます。分割統治で計算すれば、$O(n(\log n)^2)$ 解法が得られます (50/100 点)。

満点解法について考えましょう。先ほど紹介したフェルマーの小定理を少し修正することで、以下が分かります:

$$P(p-1) = (t+1)\times(2t+1)\times\cdots\times((p-1)t+1) \equiv -t^{p-1}+1 \pmod p$$

$(kt+1) \bmod p$ は周期 p なので、$P(pk) \equiv P(p)^k = (-t^{p-1}+1)^k \pmod p$ が分かります。

よって、$n = qp+r$ $(0 \le r < p)$ としたとき、$P(n) \equiv P(qp)P(r) \equiv (-t^{p-1}+1)^q P(r) \pmod p$ が計算できればよいです。

$(-t^{p-1}+1)^q$ について考えましょう。今は p の倍数かどうかにだけ関心があるので、$(t^{p-1}+1)^q$ を考えても同じです。ここで、次のような事実が実験によって分かります:

非負整数 a の p 進表記を $a = (d_{e-1}d_{e-2}\cdots d_1 d_0)_p$ とすると、$(x+1)^a \bmod p$ の 0 でない係数は、ちょうど
$(d_{e-1}+1)\times(d_{e-2}+1)\times\cdots\times(d_0+1)$ 個ある。

これに $x = t^{p-1}$ を代入すると、$q = (d_{e-1}d_{e-2}\cdots d_1 d_0)_p$ として、$(-t^{p-1}+1)^q \bmod p$ の 0 でない係数は $(d_{e-1}+1)\times(d_{e-2}+1)\times\cdots\times(d_0+1)$ 個あって、しかもそれぞれの次数は $p-1$ 次以上離れていることが分か

ります。p 進表記を計算する方法は色々ありますが、n の桁数が小さいので、$O((\log n)^2)$ 時間かけて愚直に多倍長整数演算をすれば良いでしょう。

次に $P(r)$ ですが、$P(r)$ は r 次なので、$r < p-1$ ならば $(-t^{p-1}+1)^q$ と干渉しないことが分かります。よって答えは $(d_{e-1}+1) \times (d_{e-2}+1) \times \cdots \times (d_0+1) \times (P(r) \bmod p$ の 0 でない係数) であることが分かります。$r = p-1$ の時は、$P(r) \equiv P(p) \pmod p$ より、$n = (q+1)p$ の時と同じ答えになることが分かります。

以上から $O(p(\log p)^2 + (\log n)^2)$ 時間の解法が得られました。以下にソースコードを載せます。`https://www.codechef.com/viewsolution/17544576` です。

ソースコード 6　FEB18-LUCASTH.cpp

```
#include<algorithm>
#include<cassert>
#include<iostream>
#include<vector>
using namespace std;
typedef long long lint;
#define rep(i,n)for(int i=0;i<(int)(n);++i)

// http://math314.hateblo.jp/entry/2015/05/07/014908
namespace math314{
  typedef long long ll;
  typedef pair<int, int> Pii;

#define FOR(i,n) for(int i = 0; i < (n); i++)
#define sz(c) ((int)(c).size())

  template<class T> T extgcd(T a, T b, T& x, T& y) { for (T u = y = 1, v = x = 0; a;) { T q = b / a; swap(x -= q * u, u); swap(y -= q * v, v); swap(b -= q * a, a); } return b; }
  template<class T> T mod_inv(T a, T m) { T x, y; extgcd(a, m, x, y); return (m + x % m) % m; }
  ll mod_pow(ll a, ll n, ll mod) { ll ret = 1; ll p = a % mod; while (n) { if (n & 1) ret = ret * p % mod; p = p * p % mod; n >>= 1; } return ret; }

  template<int mod, int primitive_root>
  class NTT {
  public:
    int get_mod() const { return mod; }
    void _ntt(vector<ll>& a, int sign) {
      const int n = sz(a);
      assert((n ^ (n&-n)) == 0); //n = 2^k

      const int g = 3; //g is primitive root of mod
      int h = (int)mod_pow(g, (mod - 1) / n, mod); // h^n = 1
      if (sign == -1) h = (int)mod_inv(h, mod); //h = h^-1 % mod

      //bit reverse
      int i = 0;
      for (int j = 1; j < n - 1; ++j) {
        for (int k = n >> 1; k >(i ^= k); k >>= 1);
        if (j < i) swap(a[i], a[j]);
      }

      for (int m = 1; m < n; m *= 2) {
        const int m2 = 2 * m;
        const ll base = mod_pow(h, n / m2, mod);
        ll w = 1;
        FOR(x, m) {
```

```
45              for (int s = x; s < n; s += m2) {
46                ll u = a[s];
47                ll d = a[s + m] * w % mod;
48                a[s] = u + d;
49                if (a[s] >= mod) a[s] -= mod;
50                a[s + m] = u - d;
51                if (a[s + m] < 0) a[s + m] += mod;
52              }
53              w = w * base % mod;
54            }
55          }
56
57          for (auto& x : a) if (x < 0) x += mod;
58        }
59        void ntt(vector<ll>& input) {
60          _ntt(input, 1);
61        }
62        void intt(vector<ll>& input) {
63          _ntt(input, -1);
64          const int n_inv = mod_inv(sz(input), mod);
65          for (auto& x : input) x = x * n_inv % mod;
66        }
67
68        // 畳み込み演算を行う
69        vector<ll> convolution(const vector<ll>& a, const vector<ll>& b){
70          int ntt_size = 1;
71          while (ntt_size < sz(a) + sz(b)) ntt_size *= 2;
72
73          vector<ll> _a = a, _b = b;
74          _a.resize(ntt_size); _b.resize(ntt_size);
75
76          ntt(_a);
77          ntt(_b);
78
79          FOR(i, ntt_size){
80            (_a[i] *= _b[i]) %= mod;
81          }
82
83          intt(_a);
84          return _a;
85        }
86      };
87
88      typedef NTT<167772161, 3> NTT_1;
89      typedef NTT<469762049, 3> NTT_2;
90      typedef NTT<1224736769, 3> NTT_3;
91
92      // garner のアルゴリズムを直書きした version，速い
93      vector<ll> fast_int32mod_convolution(vector<ll> a, vector<ll> b,int mod){
94        for (auto& x : a) x %= mod;
95        for (auto& x : b) x %= mod;
96
97        NTT_1 ntt1; NTT_2 ntt2; NTT_3 ntt3;
98        assert(ntt1.get_mod() < ntt2.get_mod() && ntt2.get_mod() < ntt3.get_mod());
99        auto x = ntt1.convolution(a, b);
100       auto y = ntt2.convolution(a, b);
101       auto z = ntt3.convolution(a, b);
```

```
102
103     // garner のアルゴリズムを極力高速化した
104     const ll m1 = ntt1.get_mod(), m2 = ntt2.get_mod(), m3 = ntt3.get_mod();
105     const ll m1_inv_m2 = mod_inv<ll>(m1, m2);
106     const ll m12_inv_m3 = mod_inv<ll>(m1 * m2, m3);
107     const ll m12_mod = m1 * m2 % mod;
108     vector<ll> ret(sz(x));
109     FOR(i, sz(x)){
110       ll v1 = (y[i] - x[i]) * m1_inv_m2 % m2;
111       if (v1 < 0) v1 += m2;
112       ll v2 = (z[i] - (x[i] + m1 * v1) % m3) * m12_inv_m3 % m3;
113       if (v2 < 0) v2 += m3;
114       ll constants3 = (x[i] + m1 * v1 + m12_mod * v2) % mod;
115       if (constants3 < 0) constants3 += mod;
116       ret[i] = constants3;
117     }
118
119     return ret;
120   }
121 }
122
123 pair<string,lint> divide(const string &n,lint r){
124   string res;
125   lint rem=0;
126   bool cont_zero=1;
127   rep(i,n.size()){
128     rem=10*rem+(n[i]-'0');
129     lint q=rem/r;
130     rem%=r;
131     if(cont_zero&&q==0)continue;
132     if(q!=0)cont_zero=0;
133     res+='0'+q;
134   }
135   return make_pair(res,rem);
136 }
137
138
139 vector<lint> parse(const string &n,lint r){
140   vector<lint> dig;
141   string cur(n);
142   while(1){
143     pair<string,lint>sub=divide(cur,r);
144     dig.push_back(sub.second);
145     cur=sub.first;
146     if(cur=="")break;
147   }
148   return dig;
149 }
150
151 vector<lint> g(lint p,lint l,lint r){
152   if(l>r){
153     return vector<lint>(1,1);
154   }
155   if(l==r){
156     vector<lint> ret(2);
157     ret[0]=1;
158     ret[1]=l;
```

```
159       return ret;
160     }
161     lint mid=(l+r+1)/2;
162     vector<lint>fst=g(p,l,mid-1);
163     vector<lint>snd=g(p,mid,r);
164     return math314::fast_int32mod_convolution(fst,snd,p);
165   }
166
167
168   lint f(string n,lint p){
169     const lint mod=1e9+7;
170     vector<lint> dig=parse(n,p);
171     if(dig[0]==p-1){
172       //propagate
173       dig[0]=0;
174       int car=1;
175       for(int pos=1;pos<(int)dig.size();++pos){
176         dig[pos]+=car;
177         car=dig[pos]/p;
178         dig[pos]%=p;
179       }
180       if(car>0)dig.push_back(car);
181     }
182     vector<lint> ans=g(p,1,dig[0]);
183     lint ret=0;
184     rep(i,dig[0]+1)
185       if(ans[i]!=0)ret++;
186     rep(i,dig.size()-1)
187       ret=ret*(dig[i+1]+1)%mod;
188     return ret;
189   }
190
191   int main(){
192     int t;
193     cin>>t;
194     while(t--){
195       string n;
196       lint p;
197       cin>>n>>p;
198       cout<<f(n,p)<<endl;
199     }
200   }
```

10　ペル方程式 (Lv. 4)

問題

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 **10.1.** 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

11　単項イデアル整域 (Lv. 5)

定義 **11.1.** 可換環 R が**整域** (*integral domain*) であるとは、$x, y \in R$ が $xy = 0$ を満たすならば、x か y の少なくとも一方は 0 であることである。

定義 **11.2.** 可換環 R の**イデアル** (*ideal*) とは、R の部分集合 I であって、以下の 2 条件を満たすものである:

- $x \in R, y \in R \to x + y \in R$
- $x \in R, y \in I \to xy \in R$

定義 **11.3.** 可換環 R とその要素 $x \in R$ に対して、x **によって生成される単項イデアル** (*a principal ideal generated by x*) とは、$xR = \{xy \mid y \in R\}$ のことである。これを (x) と表記する。

定義 **11.4.** **単項イデアル整域** (*principal ideal domain*, PID) (独: *Hauptidealbereich*, ハウプトイデアールベライヒ) とは、全てのイデアルが単項生成であるような整域である。

例 **11.5.** PID の例として有名なのは、$\mathbb{Z}, \mathbb{Z}[i]$ (i は虚数単位)、 $K[x]$ (係数が K の元であるような多項式全体のなす集合、K は体) などです。逆に PID でない例として有名なものには、$\mathbb{Z}[\sqrt{-5}]$ などがあります。

以下の問題を考えてみましょう。

問題

(P,Q)-サンタがいる。(P,Q)-サンタは最初原点におり、(x,y) からは $(x\pm P, y\pm Q)$ または $(x\pm Q, y\pm P)$ の 8 種類の点に移動できる。N 人の子供の座標 (X_i, Y_i) が与えられるので、(P,Q)-サンタが到達できる子供の数を求めよ。
(出典: yukicoder No.321 (P,Q)-サンタと街の子供たち)

この問題に限り、添字との混同を防ぐため、虚数単位を $\sqrt{-1}$ と表記します。座標 (x,y) に到達可能として、$x+y\sqrt{-1}$ がどのような条件を満たすべきかを考えてみましょう。問題の移動は、$x+y\sqrt{-1}$ から $x+y\sqrt{-1}\pm(P+Q\sqrt{-1}), x+y\sqrt{-1}\pm(P-Q\sqrt{-1}), x+y\sqrt{-1}\pm(Q+P\sqrt{-1}), x+y\sqrt{-1}\pm(Q-P\sqrt{-1})$ の 8 種類の点に移動するものと考えることができます。$Q-P\sqrt{-1} = -\sqrt{-1}(P+Q\sqrt{-1})$, $Q+P\sqrt{-1} = \sqrt{-1}(P-Q\sqrt{-1})$ に注意すると、結局移動できるのは

$$\alpha(P+Q\sqrt{-1}) + \beta(P-Q\sqrt{-1})$$

(ただし、$\alpha, \beta \in \mathbb{Z}[\sqrt{-1}]$) で表される点ということが分かります。集合

$$\{\alpha(P+Q\sqrt{-1}) + \beta(P-Q\sqrt{-1}) \mid \alpha, \beta \in \mathbb{Z}[\sqrt{-1}]\}$$

は、他でもないイデアル $(P+Q\sqrt{-1}, P-Q\sqrt{-1})$ であり、$\mathbb{Z}[\sqrt{-1}]$ が PID であるという性質から、$P+Q\sqrt{-1}$ と $P-Q\sqrt{-1}$ の最大公約数を γ と置くと、このイデアルは γ によって生成される単項イデアル (γ) です。よって、$\gamma | X_i + Y_i\sqrt{-1}$ かどうかの判定を行うことで、この問題が解けました。

また、以下の問題を考えてみましょう。

問題

N 個の非負整数 A_i が黒板に書かれている。以下の操作を何度でも行える:

- 黒板にある数を一つ選び、それを x とする。$2x$ を新しく書き込む。
- 黒板にある数を二つ選び、それらを x, y とする (同じ数でも良い)。x xor y を新しく書き込む。

最終的に書き込める数のうち、X 以下のものは何種類あるか? 998244353 で割った余りを求めよ。
(出典: ARC084 F - XorShift)

まず、演算が 2 倍と xor なので、整数を以下のようにして、GF(2)[x] の元 (つまり、GF(2) 係数の多項式) として表すという発想が自然です:

$$t = b_u 2^u + b_{u-1}2^{u-1} + \cdots + b_1 2 + b_0 \mapsto b_u x^u + b_{u-1}x^{u-1} + \cdots b_1 x + b_0$$

多項式の上の演算として考えると、2 倍する操作は x をかける操作、xor をとる操作は多項式の足し算です。(GF(2) の上の足し算が xor であることに注意してください。) GF(2)[x] は PID なので、この問題は、GF(2)[x] の上の最大公約数が計算できれば解けます。つまり、整数 t が黒板に書き込めることと、以下が同値です:

$$t \in (A_1, \ldots, A_N) = (\gcd(A_1, \ldots, A_N))$$

この場合、最大公約数の計算一回には $O(|A_i|^2)$ 時間かかるので、全体の計算量は $O(N|A_i|^2)$ となり、十分間に合います。ビット並列のテクニックを使うことで、$O(N|A_i|^2/64)$ にしても良いでしょう。

ソースコードは以下のようになります。`https://arc084.contest.atcoder.jp/submissions/2137652` です。

ソースコード 7　ARC084F.cpp

```
#include<algorithm>
#include<bitset>
#include<cassert>
#include<iostream>
#include<vector>
using namespace std;
typedef long long lint;
#define rep(i,n)for(int i=0;i<(int)(n);++i)

const int N=4000;
typedef bitset<N> bs;
const lint MOD=998244353;

bs read(){
  string s;
  cin>>s;
  int l=s.length();
  bs ret;
  rep(i,l)ret[i]=s[l-i-1]=='1';
  return ret;
}

// a%b
bs rem(bs a,bs b){
  int hi=-1;
  rep(i,N)
    if(b[i])hi=i;
  assert(hi>=0);
  for(int i=N-hi-1;i>=0;--i)
    if(a[i+hi])a^=b<<i;
  return a;
}

bs gcd(bs a,bs b){
  while(b.count()!=0){
    a=rem(a,b);
    swap(a,b);
  }
  return a;
}

int main(){
  int n;
  cin>>n;
  bs x=read();
  vector<bs> s(n);
  rep(i,n)s[i]=read();
  bs g=s[n-1];
  rep(i,n-1)g=gcd(s[i],g);
  lint ans=0;
```

```
51    lint cur=1;
52    int hi=-1;
53    rep(i,N)
54      if(g[i])hi=i;
55    rep(i,N-hi){
56      if(x[i+hi])ans=(ans+cur)%MOD;
57      cur=cur*2%MOD;
58    }
59    bs y=x^rem(x,g);
60    bool lt=false; //x<y?
61    for(int i=N-1;i>=0;--i){
62      if(x[i]!=y[i]){
63        lt=x[i]<y[i];
64        break;
65      }
66    }
67    if(!lt)ans=(ans+1)%MOD; // y<=x; counts y
68    cout<<ans<<endl;
69  }
```

索引

参考文献

[1] Richard M. Karp. An introduction to randomized algorithms. *Discrete Applied Mathematics*, Vol. 34, No. 1-3, pp. 165–201, 1991.