

#3. 정수론 (2)

나정휘

<https://justicehui.github.io/>

목차

확장 유클리드 알고리즘

페르마 소정리

중국인의 나머지 정리

월슨 정리

오일러 피 함수

이항 계수

확장 유클리드 알고리즘

확장 유클리드 알고리즘

선형 디오판토스 방정식

- $ax + by = c$ 의 정수해를 구하는 방정식
- $\gcd(a, b) \mid c$ 일 때만 정수해 존재
 - 베주 항등식
- $ax + by = \gcd(a, b)$ 를 해결할 수 있으면 전체 문제를 해결할 수 있음
 - 양변에 $c/\gcd(a, b)$ 를 곱하면 됨

확장 유클리드 알고리즘

확장 유클리드 알고리즘

- $ax + by = \gcd(a, b)$ 의 정수해 (x, y) 를 구하는 알고리즘
- $a = 0$ 이면 $by = b, (x, y) = (0, 1)$
- $b = 0$ 이면 $ax = a, (x, y) = (1, 0)$
- 유클리드 호제법의 종료 조건과 동일
- $bx' + ry' = \gcd(b, r)$ 의 답을 이용해
- $ax + by = \gcd(a, b)$ 의 답을 구할 수 있다면
- 유클리드 호제법과 동일한 방법으로 해결할 수 있음

확장 유클리드 알고리즘

확장 유클리드 알고리즘

- $ax + by = \gcd(a, b) = bx' + ry'$
- $= bx' + (a - qb)y'$
- $= bx' + ay' - qby'$
- $= ay' + b(x' - qy')$
- $\therefore (x, y) = (y', x' - qy') = (y', x' - \left\lfloor \frac{a}{b} \right\rfloor y')$
- 선형 디오판토스 방정식의 특수해를 찾음

```
// return [g,x,y] s.t. ax+by = gcd(a,b) = g
tuple<ll,ll,ll> ext_gcd(ll a, ll b){
    if(b == 0) return {a, 1, 0};
    auto [g,x,y] = ext_gcd(b, a % b);
    return {g, y, x - a/b * y};
}
```

확장 유클리드 알고리즘

선형 디오판토스 방정식

- $ax + by = c$ 를 풀어보자
 - $d = \gcd(a, b) \nmid c$ 이면 해가 존재하지 않음
- 특수해
 - $ax' + by' = d$ 의 특수해 (x', y') 를 구한 다음 (확장 유클리드 알고리즘)
 - $a\left(\frac{c}{d}x'\right) + b\left(\frac{c}{d}y'\right) = c$ 를 이용해
 - $ax + by = c$ 의 특수해 $(x_0, y_0) = (\frac{c}{d}x', \frac{c}{d}y')$ 를 구할 수 있음

확장 유클리드 알고리즘

선형 디오판토스 방정식

- 일반해

- $ax_0 + by_0 = c$ 를 만족하는 특수해 (x_0, y_0) 을 하나 알고 있을 때
- $ax + by = c$ 를 만족하는 모든 정수해 (x, y) 를 구해야 함
- $ax + by = ax_0 + by_0 = c$
- $a(x - x_0) = b(y_0 - y)$
- y 가 존재 $\Leftrightarrow b \mid a(x - x_0)$
- y 가 존재 $\Leftrightarrow b/\gcd(a, b) \mid x - x_0$
 - $a \mid bc \Leftrightarrow b/\gcd(a, b) \mid c$
- $x \equiv x_0 \left(\text{mod} \frac{b}{\gcd(a, b)} \right)$
- $(x, y) = (x_0 + \frac{b}{d}t, y_0 - \frac{a}{d}t)$

확장 유클리드 알고리즘

BOJ 14565 역원(Inverse) 구하기

- n, a 주어진다면 Z_n 에서 a 의 덧셈 역원과 곱셈 역원을 구하는 문제
- 덧셈 역원
 - $a + x = 0 \pmod n$ 을 만족하는 x
 - $x = n - a$
- 곱셈 역원
 - $ax = 1 \pmod n$ 을 만족하는 x
 - $ax + ny = 1$
 - 선형 디오판토스 방정식
 - a, n 이 서로소일 때만 존재
 - 주로 $a^{-1} \pmod n$ 이라고 표기함

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

ll mod(ll a, ll b){ return (a % b >= 0) ? a : a + b; }
tuple<ll, ll, ll> ext_gcd(ll a, ll b){
    if(b == 0) return {a, 1, 0};
    auto [g, x, y] = ext_gcd(b, a % b);
    return {g, y, x - a/b * y};
}

ll inv(ll a, ll n){
    auto [g, x, y] = ext_gcd(a, n);
    if(a == 0 || g != 1) return -1;
    return mod(x, n);
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    ll n, a; cin >> n >> a;
    cout << mod(-a, n) << " " << inv(a, n) << "\n";
}
```

페르마 소정리

페르마 소정리

페르마 소정리

- p 가 소수이고 a, p 가 서로소일 때 $a^{p-1} \equiv 1 \pmod{p}$
- 증명
 - $\{a, 2a, 3a, \dots, (p-1)a\}$ 를 p 로 나눈 나머지의 집합은 $\{1, 2, 3, \dots, p-1\}$
 - $a \times 2a \times 3a \times \dots \times (p-1)a \equiv 1 \times 2 \times 3 \times \dots \times (p-1) \pmod{p}$
 - $(p-1)! a^{p-1} \equiv (p-1)! \pmod{p}$
 - p 는 소수이므로 p 와 $(p-1)!$ 은 서로소
 - 양변을 $(p-1)!$ 로 나누면
 - $a^{p-1} \equiv 1 \pmod{p}$

페르마 소정리

페르마 소정리

- $a^{p-1} \equiv 1 \pmod{p}$
- $a \times a^{p-2} \equiv 1 \pmod{p}$
- p 가 소수이고 a, p 가 서로소이면 a^{p-2} 는 a 의 곱셈 역원
- 확장 유클리드 알고리즘을 모르는 경우에도 곱셈 역원을 쉽고 빠르게 구할 수 있음

페르마 소정리

BOJ 20412 추첨상 사수 대작전! (Hard)

- M, S, X, Y 가 주어짐. M 은 소수
- $X \equiv aS + c \pmod{M}, Y \equiv aX + c \pmod{M}$ 을 만족하는 a, c 를 찾는 문제
- 두 식을 연립하면 $X - Y \equiv a(S - X) \pmod{M}$
- $a \equiv (X - Y) \times (S - X)^{-1} \pmod{M}$
 - M 이 소수이므로 페르마 소정리를 이용해 a 를 구할 수 있음
- $c \equiv X - aS \pmod{M}$

중국인의 나머지 정리

중국인의 나머지 정리

중국인의 나머지 정리

- $\{a_1, a_2, \dots, a_n\}, \{m_1, m_2, \dots, m_n\}$ 이 주어지면 $a \equiv a_i \pmod{m_i}$ 를 만족하는 a 를 구하는 방법
 - $a \equiv a_1 \pmod{m_1}, a \equiv a_2 \pmod{m_2}$ 를 만족하는 a 를 구할 수 있으면
 - 그 방법을 $n - 1$ 번 적용해서 전체 문제를 해결할 수 있음
 - $n = 2$ 인 문제만 생각하자.
- $a \equiv a_1 \pmod{m_1} \Leftrightarrow a = a_1 + m_1x$ 를 만족하는 정수 x 존재
- $a \equiv a_2 \pmod{m_2} \Leftrightarrow a = a_2 - m_2y$ 를 만족하는 정수 y 존재
- 연립하면 $m_1x + m_2y = a_2 - a_1$
 - $\gcd(m_1, m_2) \nmid a_2 - a_1$ 이면 해 없음, 해 존재 $\Leftrightarrow a_1 \equiv a_2 \pmod{g}$
 - 확장 유클리드로 x 구한 다음, $a = a_1 + m_1x$ 계산하면 됨
 - a 는 0 이상 $\text{lcm}(m_1, m_2)$ 미만에서 유일함
 - 증명은 스스로 해보자.

중국인의 나머지 정리



```
pair<ll,ll> crt(ll a1, ll m1, ll a2, ll m2){
    ll g = gcd(m1, m2), m = m1 / g * m2;
    if((a2 - a1) % g) return {-1, -1};
    ll mul = mod((a2-a1)/g, m2);
    ll x = mod(get<1>(ext_gcd(m1, m2)), m2) * mul % m2;
    return { (a1 + x * m1) % m, m };
}

pair<ll,ll> crt(const vector<ll> &a, const vector<ll> &m){
    ll ra = a[0], rm = m[0];
    for(int i=1; i<m.size(); i++){
        auto [aa,mm] = crt(ra, rm, a[i], m[i]);
        if(mm == -1) return {-1, -1}; else tie(ra,rm) = tie(aa,mm);
    }
    return {ra, rm};
}
```

월슨 정리

월슨 정리

월슨 정리

- p 가 소수이면 $(p-1)! \equiv -1 \pmod{p}$
- 증명
 - $p = 2$ 이면 $1! \equiv 1 \equiv -1 \pmod{2}$ 이므로 참
 - 이제 $p > 2$ 인 경우, 즉 p 가 홀수 소수인 경우만 보면 된다.
 - 집합 $S = \{1, 2, \dots, p-1\}$ 와 S 의 한 원소 $r_i \in S$ 를 생각해 보자.
 - $\gcd(r_i, p) = 1$ 이므로 $r_i x \equiv 1 \pmod{p}$ 인 $x \in S$ 가 존재한다.
 - x 를 $x = pq + r$ ($0 < r < p$)로 표현할 수 있으므로 $x \equiv r_j \pmod{p}$ 를 만족하는 $r_j \in S$ 가 존재한다.
 - 다시 말해, 모든 $r_i \in S$ 에 대해 $r_i r_j \equiv 1 \pmod{p}$ 인 $r_j \in S$ 가 존재한다.
 - 이때, $(r_i)^2 \equiv 1 \pmod{p}$ 인 r_i 는 $1, p-1$ 두 개 뿐이다.
 - p 는 홀수이므로 $\{2, 3, 4, \dots, p-2\}$ 의 원소의 개수인 $p-3$ 은 짝수이다.
 - $p-3$ 개의 원소 사이에 완전 매칭이 존재하므로, $2 \times 3 \times \dots \times (p-2) \equiv 1^{\frac{p-3}{2}} \equiv 1 \pmod{p}$
 - 따라서 $(p-1)! \equiv 1 \times 1^{(p-3)/2} \times (p-1) \equiv p-1 \equiv -1 \pmod{p}$

오일러 피 함수

오일러 피 함수

오일러 피 함수

- 양의 정수 n 에 대해, $\phi(n) = 1 \leq s \leq n$ 인 양의 정수 s 중 $\gcd(n, s) = 1$ 을 만족하는 s 의 개수
- 몇 가지 간단한 예시
 - $\phi(1) = 1$
 - p 가 소수면 $\phi(p) = p - 1$
 - p 가 소수, r 이 양의 정수면 $\phi(p^r) = p^r - \frac{p^r}{p} = p^{r-1}(p - 1) = p^r(1 - 1/p)$
- Theorem. 두 양의 정수 α, β 가 서로소라면 $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$
 - $A = \{x \mid 1 \leq x \leq \alpha, \gcd(x, \alpha) = 1\}$
 - $B = \{x \mid 1 \leq x \leq \beta, \gcd(x, \beta) = 1\}$
 - $C = \{x \mid 1 \leq x \leq \alpha\beta, \gcd(x, \alpha\beta) = 1\}$
 - $R = A \times B = \{(a, b) \mid a \in A, b \in B\}$
 - C 와 R 사이에 일대일 대응(bijection) $f: C \rightarrow R$ 이 존재함을 증명하면 됨

오일러 피 함수

오일러 피 함수

- 양의 정수 n 에 대해, $\phi(n) = 1 \leq s \leq n$ 인 양의 정수 s 중 $\gcd(n, s) = 1$ 을 만족하는 s 의 개수
- Theorem. 두 양의 정수 α, β 가 서로소라면 $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$
 - $C = \{x \mid 1 \leq x \leq \alpha\beta, \gcd(x, \alpha\beta) = 1\}$
 - $R = A \times B = \{(a, b) \mid a \in A, b \in B\}$
 - $f(x) = (x \bmod \alpha, x \bmod \beta)$ 가 일대일 대응 함수임을 증명하자.
 - 즉, 모든 $(a, b) \in R$ 에 대해, $x \equiv a \pmod{\alpha}, x \equiv b \pmod{\beta}$ 인 $x \in C$ 가 유일하게 존재함을 증명한다.
 - (존재성)
 - 중국인의 나머지 정리에 의해 $x \equiv a \pmod{\alpha}, x \equiv b \pmod{\beta}$ 인 정수 x 는 $0 \leq x < \alpha\beta$ 범위 내에 존재함
 - $a, b \neq 0$ 이므로 $1 \leq x < \alpha\beta$
 - $\gcd(x, \alpha) = \gcd(x, \beta) = 1$ 이므로 $\gcd(x, \alpha\beta) = 1$
 - 따라서 $x \in C$ 가 항상 존재함

오일러 피 함수

오일러 피 함수

- 양의 정수 n 에 대해, $\phi(n) = 1 \leq s \leq n$ 인 양의 정수 s 중 $\gcd(n, s) = 1$ 을 만족하는 s 의 개수
- Theorem. 두 양의 정수 α, β 가 서로소라면 $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$
 - $C = \{x \mid 1 \leq x \leq \alpha\beta, \gcd(x, \alpha\beta) = 1\}$
 - $R = A \times B = \{(a, b) \mid a \in A, b \in B\}$
 - $f(x) = (x \bmod \alpha, x \bmod \beta)$ 가 일대일 대응 함수임을 증명하자.
 - 즉, 모든 $(a, b) \in R$ 에 대해, $x \equiv a \pmod{\alpha}, x \equiv b \pmod{\beta}$ 인 $x \in C$ 가 유일하게 존재함을 증명한다.
 - (유일성)
 - 두 정수 $x, y \in C$ 에 대해 $f(x) = f(y)$ 라고 하자.
 - $f(x) = f(y)$ 이면 $\alpha \mid (x - y), \beta \mid (x - y)$ 이 성립하고, $\gcd(\alpha, \beta) = 1$ 이므로 $\alpha\beta \mid (x - y)$ 또한 성립한다.
 - $1 \leq x, y \leq \alpha\beta$ 이므로 x 와 y 는 같을 수밖에 없다.
 - 따라서 $(a, b) \in R$ 에 대해 $f(x) = (a, b)$ 인 x 가 존재한다면, x 는 집합 C 에서 유일하다.

오일러 피 함수

오일러 피 함수

- 양의 정수 n 에 대해, $\phi(n) = 1 \leq s \leq n$ 인 양의 정수 s 중 $\gcd(n, s) = 1$ 을 만족하는 s 의 개수
- 오일러 피 함수의 성질
 - $\phi(1) = 1$
 - p 가 소수, r 이 양의 정수면 $\phi(p^r) = p^r - \frac{p^r}{p} = p^{r-1}(p - 1) = p^r(1 - 1/p)$
 - 두 양의 정수 α, β 가 서로소라면 $\phi(\alpha\beta) = \phi(\alpha)\phi(\beta)$
- 오일러 피 함수의 계산
 - $n > 1$ 의 소인수 분해 $n = \prod p_i^{e_i}$ 를 생각하자.
 - $p_i^{e_i}$ 들끼리는 서로소이므로 $\phi(n) = \phi(p_1^{e_1})\phi(p_2^{e_2}) \cdots \phi(p_k^{e_k}) = \prod \phi(p_i^{e_i})$ 이다.
 - $\phi(n) = \prod p_i^{e_i} \left(1 - \frac{1}{p_i}\right) = n \prod \frac{p_i - 1}{p_i}$

오일러 피 함수

오일러 정리

- 양의 정수 a, b 가 서로소이면 $a^{\phi(b)} \equiv 1 \pmod{b}$
 - b 가 소수면 페르마의 소정리와 같음
- 증명
 - $b = 1$ 이면 모든 정수는 $\text{mod } b$ 에서 $1 \equiv 0$ 과 합동이므로 참. $b > 1$ 인 상황만 고려하자.
 - b 로 나눈 나머지가 서로 다른 원소들의 집합 $\{1, 2, 3, \dots, b\}$ 를 생각해 보자.
 - $\gcd(b, s) = 1$ 인 $\phi(b)$ 개의 원소를 차례대로 $r_1, r_2, \dots, r_t$ 라고 하자. (단, $t = \phi(b)$)
 - Claim. $S_1 = \{ar_1, ar_2, \dots, ar_t\}$ 의 원소를 b 로 나눈 원소들의 집합은 $S_2 = \{r_1, r_2, \dots, r_t\}$ 와 같다.
 - $ar_i \equiv ar_j \pmod{b}$ 이면 $r_i \equiv r_j \pmod{b}$ 임을 증명하면 되고, $\gcd(a, b) = 1$ 이므로 양변을 a 로 나누면 쉽게 얻을 수 있다.
 - 따라서 S_1 의 모든 원소를 곱한 값과 S_2 의 모든 원소를 곱한 값은 $\text{mod } b$ 에서 합동이다.
 - $ar_1 \times ar_2 \times \dots \times ar_t \equiv a^t r_1 r_2 \dots r_t \equiv r_1 r_2 \dots r_t \pmod{b}$
 - $\gcd(r_i, b) = 1$ 이므로 양변을 $r_1 r_2 \dots r_t$ 로 나누면 $a^t \equiv a^{\phi(b)} \equiv 1 \pmod{b}$

오일러 피 함수

$a^{b!} \bmod C$

- $1 \leq a, b \leq 10^6$
- $C = 10^9+7$ (소수)
- $\gcd(a, C) = 1$ 이므로 $a^{\phi(C)} \equiv 1 \pmod{C}$
- 따라서 $a^{b!} \equiv a^{b! \bmod \phi(C)} \pmod{C}$
- 오일러 정리를 이용하면 지수의 크기를 줄일 수 있다는 점을 알면 좋다!

이항 계수

이항 계수 (1)

이항 계수

- $nCr = \frac{n!}{r!(n-r)!}$ 를 어떤 수로 나눈 나머지를 빠르게 계산해 보자
 - 소수 p 로 나눈 나머지
 - 전처리 $O(n + \log p)$, 쿼리 $O(1)$
 - 전처리 $O(p + \log p)$, 쿼리 $O(\log n)$
 - 소수의 거듭제곱 p^e 으로 나눈 나머지
 - 전처리 $O(p^e)$, 쿼리 $O(\log n + \log p)$
 - 합성수 $m = p_1^{e_1} p_2^{e_2} p_3^{e_3} \dots p_k^{e_k}$ 로 나눈 나머지
 - 전처리 $O(\sum p_i^{e_i})$, 쿼리 $O(\log n + \log m)$

이항 계수 (1)

$nCr \bmod p - O(n + \log p + q)$

- $\text{fac}[n] = n! \bmod p$
 - $O(n)$ 에 전처리 가능
- $\text{inv}[n] = (n!)^{-1} \bmod p$
 - 매번 곱셈 역원을 구하면 $O(n \log p)$
 - $\frac{1}{n!} = \frac{1}{(n+1)!} \times (n+1)$ 임을 이용하면 $O(n + \log p)$ 에 전처리 가능
- 전처리
 - `fac[0] = 1; for(int i=1; i<=sz; i++) fac[i] = fac[i-1] * i % p;`
 - `inv[sz] = pow_mod(fac[sz], p-2, p);`
 - `for(int i=sz; i>=0; i--) inv[i] = inv[i+1] * (i+1) % p;`
- $\text{fac}[n] * \text{inv}[r] \% p * \text{inv}[n-r] \% p$ 를 이용해 $O(1)$ 에 쿼리 가능
- $n < p$ 일 때만 사용 가능

이항 계수 (2)

$$nCr \bmod p - O(p + \log p + q \log n)$$

- p 가 작고 n 이 클 때 사용
- n, r 을 p 진법으로 전개하자.
 - $n = n_k p^k + n_{k-1} p^{k-1} + \dots + n_1 p^1 + n_0$
 - $r = r_k p^k + r_{k-1} p^{k-1} + \dots + r_1 p^1 + r_0$
- Lucas's Theorem
 - $nCr \equiv \prod n_i Cr_i \pmod{p}$
 - 증명은 어려우니까 생략
- $n_i < p$ 이므로 앞에서 사용한 풀이를 적용할 수 있음
 - p 까지만 전처리해도 되므로 $O(p + \log p)$ 에 전처리 가능
- 앞에서 사용한 풀이를 $\log_p n$ 번 적용해야 하므로 $O(\log n)$ 에 쿼리 가능

이항 계수 (2)

```
struct Lucas{ // init :  $O(P + \log P)$ , query :  $O(\log P)$ 
    const size_t P;
    vector<ll> fac, inv;
    ll Pow(ll a, ll b){
        ll ret = 1;
        for(; b; b>>=1, a=a*a%P) if(b&1) ret=ret*a%P;
        return ret;
    }
    Lucas(size_t P) : P(P), fac(P), inv(P) {
        fac[0] = 1;
        for(int i=1; i<P; i++) fac[i] = fac[i-1] * i % P;
        inv[P-1] = Pow(fac[P-1], P-2);
        for(int i=P-2; ~i; i--) inv[i] = inv[i+1] * (i+1) % P;
    }
    ll small(ll n, ll r) const {
        if(n < r) return 0;
        return fac[n] * inv[r] % P * inv[n-r] % P;
    }
    ll calc(ll n, ll r) const {
        if(n < r || n < 0 || r < 0) return 0;
        if(!n || !r || n == r) return 1;
        return small(n%P, r%P) * calc(n/P, r/P) % P;
    }
};
```

이항 계수 (3)

$$nCr \bmod p^e - O(p^e + q(\log n + \log p))$$

- p 가 곱해진 횟수도 중요함
 - $n!$ 은 p^e 의 배수지만 nCr 은 p^e 의 배수가 아닐 수 있음
- $n!$ 에 p 가 몇 번 곱해졌는지 구해야 함
- 구체적으로, $n! = p^t m$ 일 때 $(t, m \bmod p^e)$ 를 구해야 함
 - 이걸 구해 놓으면 nCr 은 $O(\log p^e)$ 에 구할 수 있음
 - 곱셈 역원을 구해야 하기 때문

이항 계수 (3)

$$nCr \bmod p^e - O(p^e + q(\log n + \log p))$$

- $n! = p^t m$ 으로 표현할 때, $(t, m \bmod p^e)$ 를 구하는 방법을 알아보자. (단, m 은 p 와 서로소)
 - $n = pq + r = p^e q' + r'$
 - $[1, n]$ 에서 p 의 배수가 아닌 수
 - $f(i) := [1, i]$ 에서 p 의 배수가 아닌 수를 곱한 값을 p^e 로 나눈 나머지
 - $[1, p^e - 1]$ 에 있는 수를 곱하면 $f(p^e - 1)$
 - $[p^e + 1, 2p^e - 1], \dots, [(q' - 1)p^e + 1, q'p^e - 1]$ 에 있는 수를 각각 곱한 것도 $f(p^e - 1)$
 - $[q'p^e + 1, q'p^e + r']$ 에 있는 수를 곱하면 $f(r')$
 - 그러므로 모두 곱하면 $f(p^e - 1)^{q'} f(r') \pmod{p^e}$
 - $[1, n]$ 에서 p 의 배수인 수
 - $p, 2p, 3p, \dots, qp$
 - 모두 곱하면 $p^q q!$ 이고, $q!$ 에 대한 문제로 바뀜
 - $q \leq n/p$ 이므로 $\log_p n$ 단계만 거치면 됨

이항 계수 (3)

$$nCr \bmod p^e = O(p^e + q(\log n + \log p))$$

- $n! = p^t m$ 으로 표현할 때, $(t, m \bmod p^e)$ 를 구하는 방법을 알아보자. (단, m 은 p 와 서로소)
 - $n = pq + r = p^e q' + r'$
 - $[1, n]$ 에서 p 의 배수가 아닌 수 : $f(p^e - 1)^{q'} f(r') \pmod{p^e}$
 - $[1, n]$ 에서 p 의 배수인 수 : $p^q q!$
- $q! = p^{t'} m'$ 에서 $(t', m' \bmod p^e)$ 를 알고 있을 때 $(t, m \bmod p^e)$ 를 구하는 방법
 - $t = q + t'$
 - $m \equiv m' f(p^e - 1)^{q'} f(r') \pmod{p^e}$
 - $f(p^e - 1) \equiv \pm 1 \pmod{p^e}$ 라서 $q' = \left\lfloor \frac{n}{p^e} \right\rfloor$ 의 홀짝만 중요함
 - 증명 생략
 - $f(0), f(1), \dots, f(p^e - 1)$ 을 전처리하면 $O(1)$ 에 계산 가능
- 전처리 $O(p^e)$, 쿼리 $O(\log n)$

이항 계수 (3)

$$nCr \bmod p^e - O(p^e + q(\log n + \log p))$$

- $n! = p^{t_1}m_1, r! = p^{t_2}m_2, (n-r)! = p^{t_3}m_3$ 을 모두 계산했다고 하자.
- $nCr = p^{t_1-t_2-t_3} \times m_1 \times m_2^{-1} \times m_3^{-1} \pmod{p^e}$
- 전처리 $O(p^e)$, 쿼리 $O(\log n + \log p)$
 - $f(0), f(1), \dots, f(p^e - 1)$ 계산 : $O(p^e)$
 - $n! = p^t m$ 계산 : $O(\log n)$
 - 곱셈 역원 계산 : $O(\log p)$

이항 계수 (3)

```
template<ll p, ll e> struct CombinationPrimePower {
    vector<ll> f; ll pe;
    CombinationPrimePower(){
        pe = 1; for(int i=0; i<e; i++) pe *= p;
        f.resize(pe); f[0] = 1;
        for(int i=1; i<pe; i++) f[i] = f[i-1] * (i % p != 0 ? i : 1) % pe;
    }
    pair<ll, ll> factorial(ll n){
        if(n < p) return {0, f[n]};
        ll q = n / p, qp = n / pe, rp = n % pe;
        auto [tp, mp] = factorial(q);
        ll t = q + tp, m = mp * Pow(f[pe-1], qp%2, pe) % pe * f[rp] % pe;
        return {t, m};
    }
    ll calc(ll n, ll r){
        if(n < 0 || r < 0 || n < r) return 0;
        auto [t1, m1] = factorial(n);
        auto [t2, m2] = factorial(r);
        auto [t3, m3] = factorial(n-r);
        ll t = t1 - t2 - t3, m = m1 * inv(m2, pe) % pe * inv(m3, pe) % pe;
        if(t >= e) return 0;
        else return m * Pow(p, t, pe) % pe;
    }
};
```

이항 계수 (4)

$$nCr \bmod m$$

- $m = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ 를 소인수분해하고
- $nCr \bmod p^e$ 를 구해서 CRT로 합치면 됨
- 끝!

이항 계수 (4)

BOJ 14854 이항 계수 6

- $nCr \bmod 142857$ 을 구하는 문제
- $142857 = 3^3 * 11 * 13 * 37$

```
int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int T; cin >> T;
    CombinationPrimePower<3,3> a;
    CombinationPrimePower<11,1> b;
    CombinationPrimePower<13,1> c;
    CombinationPrimePower<37,1> d;
    while(T--){
        int n, r; cin >> n >> r;
        cout << crt(
            {a.calc(n, r), b.calc(n, r), c.calc(n, r), d.calc(n, r)},
            {27, 11, 13, 37}
        ).first << "\n";
    }
}
```