Number Theory

나정휘

https://justiceHui.github.io/

목차

- 용어 정의
- 유클리드 호제법
- 확장 유클리드 알고리즘
- 페르마 소정리
- 중국인의 나머지 정리
- 이항 계수

용어 정의

용어 정의

- 약수, 배수, 최대공약수, 최소공배수, 서로소
  - 정수 a, b에 대해 $b = an$을 만족하는 정수 n이 존재하면 a는 b의 약수, b는 a의 배수
    - $a|b$: a가 b를 나눈다, b는 a로 나누어진다.
  - $a = b = 0$이 아닌 정수 a, b에 대해, $g|a, g|b$를 만족하는 가장 큰 자연수 g : 최대공약수
  - $a|l, b|l$를 만족하는 가장 작은 자연수 l: 최소공배수
  - a와 b의 최대공약수가 1이면 a와 b는 서로소
  - $ab = gl$
    - $a = ga', b = gb'$라고 하면 a'과 b'은 서로소
    - $l = ga'b'$이므로 $ab = g^2a'b' = gl$임
    - $l = ab/g$

용어 정의

- 몫, 나머지, 합동
  - 정수 a와 0이 아닌 정수 b가 있을 때, $a = bq + r, 0 \leq r < |b|$를 만족하는 정수 q, r은 유일함
    - q는 몫, r은 나머지
    - 주의: C/C++에서 음수 나눗셈은 나머지가 음수가 나올 수 있음
      - 5 % 3 = 2, -5 % -3 = -3
      - -5 % 3 = -3, 5 % -3 = 2
      - 결과가 음수인 경우 b를 더하면 됨
  - 정수 a, b와 0이 아닌 정수 n이 있을 때, $n|(a - b)$이면 a와 b가 $(mod\ n)$에서 합동
    - $a \equiv b\ (mod\ n)$
    - a와 b를 n으로 나눈 나머지가 동일

용어 정의

- 합동
  - 반사성, 대칭성, 추이성
    - $a \equiv a(mod\ n)$
    - $a \equiv b(mod\ n)$이면 $b \equiv a(mod\ n)$
    - $a \equiv b(mod\ n)$이고 $b \equiv c(mod\ n)$이면 $a \equiv c(mod\ n)$
  - 사칙연산
    - $a \equiv b(mod\ n)$이면
    - 정수 c에 대해, $a \pm c \equiv b \pm c(mod\ n)$
    - 0이 아닌 정수 c에 대해, $ac \equiv bc(mod\ n)$
    - 나눗셈은 성립 안 함

용어 정의

- 소수
  - 약수가 1과 자기 자신밖에 없는 2 이상의 자연수
  - 소수는 무한히 많음
    - 소수가 유한하다고 가정하고 귀류법 사용하면 증명 가능
  - 임의의 자연수 n에 대해, 소수가 등장하지 않는 길이 n인 구간 존재
    - $2 \le k \le n+1$일 때 $(n+1)!+k$는 k의 배수이므로 소수가 아님
  - 임의의 자연수 n에 대해, $n < p \le 2n$인 소수 p가 존재
    - 베르트랑 공준
  - $\pi(x)$를 x이하 소수의 개수라고 하면, $\lim_{n\to\infty} \frac{\pi(x)\log x}{x} = 1$이 성립함
    - x이하 소수의 개수는 $O(\frac{x}{\log x})$개
    - 소수 정리

용어 정의

- 2 이상의 자연수 n을 소수들의 곱으로 표현하는 것: 소인수분해
  - $n = {p_1}^{e_1}{p_2}^{e_2}\cdots{p_k}^{e_k}$
  - 소수들의 순서만 다른 경우를 같은 표현으로 보면, 소인수분해의 결과는 유일하게 존재
    - 2개 이상이라고 가정하고 귀류법 사용하면 증명 가능
  - n의 약수는 ${p_1}^{f_1}{p_2}^{f_2}\cdots{p_k}^{f_k}$꼴 (단, $0 \le f_i \le e_i$)

질문?

유클리드 호제법

유클리드 호제법

- 최대공약수의 성질
  - $\gcd(a, b) = \gcd(|a|, |b|)$
  - $\gcd(a, 0) = |a|$
  - $\gcd(a, b) = \gcd(b, a)$
  - $\gcd(a, b) = \gcd(a \pm b, b)$
    - $d|a \ and \ d|b \Leftrightarrow d|(a+b) \ and \ d|b$
    - 이므로 (a, b)의 공약수 집합과 (a+b, b)의 공약수 집합 동일함
    - a-b도 동일하게 증명 가능
  - $\gcd(a, b) = \gcd(a + nb, b)$
    - 위의 결과에서 수학적 귀납법 적용
  - $\gcd(a, b) = \gcd(a \ mod \ b, b)$
    - $a \ mod \ b = r$ 이라고 하면 $a = nb + r$ 인 정수 n 존재
    - a에서 b를 여러 번 뺀다고 생각해도 됨

유클리드 호제법

- 유클리드 호제법
  - 두 정수 a, b의 최대공약수를 구하는 과정
    - 음수인 경우 절댓값을 취하면 되므로 $a, b \geq 0$ 인 경우만 생각
    - $\gcd(a, b) = \gcd(b, a)$ 이므로 $a \geq b$ 인 경우만 생각
    - $\gcd(a, 0) = a$ 이므로 $a \geq b \geq 1$ 인 경우만 생각, $b = 0$ 이면 알고리즘 종료
  - $\gcd(a, b) = \gcd(a\ mod\ b, b)$
    - $a = bq + r$ 이라고 하면 $\gcd(a, b) = \gcd(b, r)$, $a \geq b > r$
    - (a, b)를 (b, r)로 축소
    - 이대로 b를 0까지 끌고 내려가면 됨
      - $r \leq a/2$ 이므로 $br \leq ab/2$
        - $q \geq 1$이므로 $2r \leq (q + 1)r = qr + r \leq qb + r = a$
      - $O(\log ab)$번의 축소를 거치면 $b = 0$

유클리드 호제법

- BOJ 2609 최대공약수와 최소공배수
  - 두 자연수의 최대공약수와 최소공배수를 출력하는 문제
  - lcm(a, b) = a * b / gcd(a, b)
    - 계산 과정에서 나올 수 있는 최댓값은 ab
  - lcm(a, b) = a / gcd(a, b) * b
    - 계산 과정에서 나올 수 있는 최댓값은 lcm ≤ ab

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

질문?

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

  - 선형 디오판토스 방정식의 특수해를 찾음

```cpp
// 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};
}
```

질문?

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

    - $ax + by = ax_0 + by_0 = c$
    - $a(x - x_0) = b(y_0 - y)$
    - y가 존재 $\Leftrightarrow b \mid a(x - x_0)$
    - y가 존재 $\Leftrightarrow b/\gcd(a, b) \mid x - x_0$
      - $a \mid bc \Leftrightarrow b/\gcd(a, b) \mid c$
    - $x \equiv x_0 \left(mod\,\frac{b}{\gcd(a,b)}\right)$
    - $(x, y) = (x_0 + \frac{b}{d}t, y_0 - \frac{a}{d}t)$

확장 유클리드 알고리즘

- BOJ 14565 역원(Inverse) 구하기
  - n, a 주어지면 Zn에서 a의 덧셈 역원과 곱셈 역원을 구하는 문제

  - 덧셈 역원
    - a + x = 0 (mod n) 을 만족하는 x
    - x = n – a
  - 곱셈 역원
    - ax = 1 (mod n) 을 만족하는 x
    - ax + ny = 1
      - 선형 디오판토스 방정식
      - a, n이 서로소일 때만 존재
    - 주로 a^{-1} (mod n) 이라고 표기함

```cpp
#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";
}
```

질문?

페르마 소정리

페르마 소정리

- 페르마 소정리
  - p가 소수이고 a, p가 서로소일 때 $a^{p-1} \equiv 1\ (mod\ p)$

  - 증명
    - $\{a, 2a, 3a, \cdots, (p-1)a\}$를 p로 나눈 나머지의 집합은 $\{1,2,3, \cdots, p-1\}$
    - $a \times 2a \times 3a \times \cdots \times (p-1)a \equiv 1 \times 2 \times 3 \times \cdots \times (p-1)\ (mod\ p)$
    - $(p-1)!\, a^{p-1} = (p-1)!\ (mod\ p)$
    - p는 소수이므로 p와 $(p-1)!$ 은 서로소
    - 양변을 $(p-1)!$ 로 나누면
    - $a^{p-1} \equiv 1\ (mod\ p)$

페르마 소정리

- 페르마 소정리
  - $a^{p-1} \equiv 1 \ (mod\ p)$
  - $a \times a^{p-2} \equiv 1 \ (mod\ p)$
  - p가 소수이고 a, p가 서로소이면 a^{p-2}는 a의 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, Y가 주어짐. M은 소수
  - $X \equiv aS + c \ (mod\ M), Y \equiv aX + c \ (mod\ M)$을 만족하는 a, c를 찾는 문제

  - 두 식을 연립하면 $X - Y \equiv a(S - X) \ (mod\ M)$
  - $a \equiv (X - Y) \times (S - X)^{-1} \ (mod\ M)$
    - M이 소수이므로 페르마 소정리를 이용해 a를 구할 수 있음
  - $c \equiv X - aS \ (mod\ M)$

질문?

중국인의 나머지 정리

중국인의 나머지 정리

- 중국인의 나머지 정리
  - $\{a_1, a_2, \cdots, a_n\}, \{m_1, m_2, \cdots, m_n\}$이 주어지면 $a \equiv a_i (mod\ m_i)$를 만족하는 a를 구하는 방법
    - $a \equiv a_1 (mod\ m_1), a \equiv a_2 (mod\ m_2)$를 만족하는 a를 구할 수 있으면
    - 그 방법을 $n - 1$번 적용해서 전체 문제를 해결할 수 있음
    - $n = 2$인 문제만 생각하자.

  - $a \equiv a_1 (mod\ m_1) \iff a = a_1 + m_1 x$를 만족하는 정수 x 존재
  - $a \equiv a_2 (mod\ m_2) \iff a = a_2 - m_2 y$를 만족하는 정수 y 존재
  - 연립하면 $m_1 x + m_2 y = a_2 - a_1$
    - $\gcd(m1, m2) \nmid a_2 - a_1$이면 해 없음, 해 존재 $\iff a_1 \equiv a_2\ (mod\ g)$
    - 확장 유클리드로 x 구한 다음, $a = a_1 + m_1 x$ 계산하면 됨
    - a는 0 이상 $lcm(m_1, m_2)$ 미만에서 유일함
      - 증명은 스스로 해보자.

중국인의 나머지 정리

```cpp
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};
}
```

이항 계수

이항 계수

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

    - p^e부터는 심화 문제로 드릴 테니 걱정하지 마세요.

이항 계수

- $nCr\ mod\ p$ - $O(n + \log p + q)$
  - fac[n] = $n!\, mod\ p$
    - $O(n)$에 전처리 가능
  - inv[n] = $(n!)^{-1}\ mod\ p$
    - 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;`
  - `fac[n] * inv[r] % p * inv[n-r] % p` 를 이용해 $O(1)$에 쿼리 가능
  - $n < p$ 일 때만 사용 가능

질문?

이항 계수

- $nCr\ mod\ p$ - $O(p + \log p + q \log n)$
  - p가 작고 n이 클 때 사용
  - n, r을 p진법으로 전개하자.
    - $n = n_k p^k + n_{k-1} p^{k-1} + \cdots + n_1 p^1 + n_0$
    - $r = r_k p^k + r_{k-1} p^{k-1} + \cdots + r_1 p^1 + r_0$
  - Lucas's Theorem
    - $nCr \equiv \prod n_i C r_i\ (mod\ p)$
    - 증명은 어려우니까 생략

  - $n_i < p$ 이므로 앞에서 사용한 풀이를 적용할 수 있음
    - p까지만 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

```
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;
    }
};
```

질문?

이항 계수

- $nCr\ mod\ p^e$ - $O(p^e + q(\log n + \log p))$
  - p가 곱해진 횟수도 중요함
    - $n!$은 p^e의 배수지만 nCr은 p^e의 배수가 아닐 수 있음

  - $n!$에 p가 몇 번 곱해졌는지 구해야 함
  - 구체적으로, $n! = p^t m$일 때 $(t, m\ mod\ p^e)$를 구해야 함
    - 이걸 구해 놓으면 nCr은 $O(\log p^e)$에 구할 수 있음
    - 곱셈 역원을 구해야 하기 때문

이항 계수

- $nCr\ mod\ p^e$ - $O(p^e + q(\log n + \log p))$
  - $n! = p^t m$ 으로 표현할 때, $(t, m\ mod\ p^e)$를 구하는 방법을 알아보자. (단, m은 p와 서로소)
    - $n = pq + r = p^e q' + r'$
    - $[1, n]$에서 p의 배수가 아닌 수
      - $f(i) \coloneqq [1, i]$에서 p의 배수가 아닌 수를 곱한 값을 p^e로 나눈 나머지
      - $[1, p^e - 1]$에 있는 수를 곱하면 $f(p^e - 1)$
      - $[p^e + 1, 2p^e - 1], \cdots, [(q' - 1)p^e + 1, q'p^e - 1]$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 $f(p^e - 1)^{q'} f(r')\ (mod\ p^e)$
    - $[1, n]$에서 p의 배수인 수
      - $p, 2p, 3p, \cdots, qp$
      - 모두 곱하면 $p^q q!$ 이고, $q!$에 대한 문제로 바뀜
      - $q\ \le n/p$이므로 $\log_p n$단계만 거치면 됨

이항 계수

- $nCr\ mod\ p^e$ - $O(p^e + q(\log n + \log p))$
  - $n! = p^t m$ 으로 표현할 때, $(t, m\ mod\ p^e)$를 구하는 방법을 알아보자. (단, m은 p와 서로소)
    - $n = pq + r = p^e q' + r'$
    - $[1, n]$에서 p의 배수가 아닌 수 : $f(p^e - 1)^{q'} f(r')\ (mod\ p^e)$
    - $[1, n]$에서 p의 배수인 수 : $p^q q!$

    - $q! = p^{t'} m'$ 에서 $(t', m' mod\ p^e)$를 알고 있을 때 $(t, m\ mod\ p^e)$를 구하는 방법
      - $t = q + t'$
      - $m \equiv m' f(p^e - 1)^{q'} f(r')\ (mod\ p^e)$
      - $f(p^e - 1) \equiv \pm 1\ (mod\ p^e)$ 라서 $q' = \left\lfloor \frac{n}{p^e} \right\rfloor$의 홀짝만 중요함
        - 증명 생략
      - $f(0), f(1), \cdots, f(p^e - 1)$을 전처리하면 $O(1)$에 계산 가능
    - 전처리 $O(p^e)$, 쿼리 $O(\log n)$

이항 계수

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

이항 계수

```cpp
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;
    }
};
```

질문?

이항 계수

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

이항 계수

- BOJ 14854 이항 계수 6
  - nCr mod 142857을 구하는 문제
  - 142857 = $3^3 * 11 * 13 * 37$

```cpp
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";
    }
}
```

질문?

과제

- 필수
  - 2609 최대공약수와 최소공배수
  - 14565 역원 구하기
  - 20412 추첨상 사수 대작전! (Hard)
  - 23062 백남이의 여행의 준비의 준비
  - 13977 이항 계수와 쿼리
  - 11402 이항 계수 4
- 심화
  - 11661 해의 개수
  - 15718 돌아온 떡파이어
  - 14884* ACG
  - 14854 이항 계수 6