정수론
중급 3주차

서강대학교 전해성(seastar105)

정수론은 왜 공부해야 하는가?

대회에 자주 나옵니다. 그래서 준비해야 합니다.

그럼 정수론 수업 들어야 해요?

기본 도구만 공부해두고 필요할 때마다 보충합시다.

목차

소수의 판별과 소인수분해

주어진 수 N이 소수인지 판단하는 문제

가장 간단하지만 어려운 문제

쉬운 풀이만 머리에 넣어둡시다.

소수의 판별과 소인수분해

Brute-Force

$[2, N]$까지의 수로 전부 N을 나눠본다. -> $O(N)$

Faster!

$[2, \sqrt{N}]$까지의 수로 전부 N을 나눠본다. -> $O(\sqrt{N})$

소수의 판별과 소인수분해

왜 $\sqrt{N}$까지만 나눠봐도 충분할까?

N이 i로 나누어 떨어진다는 뜻은 $\frac{N}{i}$으로도 나누어 떨어진다는 뜻입니다.

$i \leq \sqrt{N}$으로 N이 나누어 떨어진다면 $\frac{N}{i} \geq \sqrt{N}$으로도 나누어 떨어져야 합니다.

이는 역도 성립해야 합니다.

따라서 $\sqrt{N}$까지의 수로 나누어 떨어지는 수가 없었다면 이후로도 없습니다.

```
bool primality_test(int x) {
    if(x == 1) return false;
    for(int i = 2; i*i <= x; ++i) {
        if (x % i == 0) return false;
    }
    return true;
}
```

소수의 판별과 소인수분해

주어진 수 N의 소인수 분해를 찾는 문제

Brute-Force

$[2, N]$까지의 소수로 전부 N을 나눠본다.

Faster!

$[2, \sqrt{N}]$까지의 소수로 전부 N을 나눠본다.

소수의 판별과 소인수분해

$[2, \sqrt{N}]$의 수들로 나눠 볼 때 나누는 수가 소수인지 확인할 필요가 없다.

만약에 N이 2로 나누어 떨어진다면 더 이상 나누어 떨어지지 않을 때까지 2로 나눠주면 됩니다.
이 과정을 반복하면 소수가 아닌 인수는 자연스레 없어집니다.
따라서 $O(\sqrt{N})$

```
vector<int> factorize(int x) {
    vector<int> ret;
    for(int i=2;i*i<=x;++i) {
        while(x%i == 0) {
            ret.push_back(i);
            x /= i;
        }
    }
    if(x > 1) ret.push_back(x);
    return ret;
}
```

소수의 판별과 소인수분해

소수 판별도 소인수 분해도 $O(\sqrt{N})$보다 빠르게 하는 방법이 존재합니다.

궁금하시면 밀러-라빈 소수 판정과, 폴라드 로 소인수분해를 공부해주세요.

이런 거까지 쓰이는 일은 잘 없습니다.

에라토스테네스의 체

지금까진 주어진 수 하나에 대해서 소수 판정과 소인수 분해를 했습니다.

만약 이런 게 쿼리로 들어온다면 매번 $O(\sqrt{N})$만큼 수행해야 할까요?

만약 주어지는 수들의 범위가 적당히 작은 범위로 고정되어 있다면 전처리를 통해서 빠르게 수행할 수 있습니다.

에라토스테네스의 체

[1, N] 범위 내에서 소수를 전부 구하고 싶습니다. $N \leq 10^7$

이를 위한 효율적인 방법은 에라토스테네스의 체입니다.

$O(N)$만큼의 메모리를 사용해서 $O(NloglogN)$만에 [1, N] 범위 내의 소수를 전부 구할 수 있습니다.

에라토스테네스의 체

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

에라토스테네스의 체

2보다 큰 2의 배수 지우기

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

에라토스테네스의 체

3보다 큰 3의 배수 지우기

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

에라토스테네스의 체

5보다 큰 5의 배수 지우기

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

에라토스테네스의 체

1과 50까지의 소수는 비어있다

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

에라토스테네스의 체 - 시간복잡도

조금 비효율적인 에라토스테네스의 체를 살펴봅시다.

[1, N] 사이의 모든 i에 대해서 i보다 큰 i의 배수를 지웁니다.

이 방식의 시간은 아래처럼 걸립니다.

$$\frac{N}{1}+\frac{N}{2}+\frac{N}{3}+\cdots+\frac{N}{N}=N(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N})$$

조화급수

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N}$$

위 식은 조화급수에 N을 넣은 것으로 아래와 같은 upper bound를 가집니다.

$$\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{N}\leq\int_{1}^{N}\frac{1}{x}dx=\ln N$$

따라서 비효율적인 방식의 시간복잡도는 $O(N\log N)$이 됩니다.

에라토스테네스의 체 - 시간복잡도

에라토스테네스의 체는 소수에 대해서만 진행하기 때문에 $O(NlogN)$보다 더 빠른 $O(NloglogN)$의 시간복잡도를 가집니다.

에라토스테네스의 체도 조화급수 형식의 방식도 굉장히 많이 쓰입니다.

```
int MAX = 1000000;
bool is_prime[1000005];

void sieve() {
    for(int i=2;i<=MAX;++i) is_prime[i] = true;
    for(int i=2;i<=MAX;++i) {
        if(is_prime[i]) {
            for(int j=i+i;j<=MAX;j+=i) is_prime[j] = false;
        }
    }
}
```

백준 16563번 어려운 소인수분해

어려운 소인수분해 성공 출처

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	512 MB	2144	592	354	25.035%

문제

지원이는 대회에 출제할 문제에 대해서 고민하다가 소인수분해 문제를 출제해야겠다고 마음을 먹었다. 그러나 그 이야기를 들은 동생의 반응은 지원이의 기분을 상하게 했다.

> "소인수분해? 그거 너무 쉬운 거 아니야?"

지원이는 소인수분해의 어려움을 알려주고자 엄청난 자신감을 가진 동생에게 2와 500만 사이의 자연수 N개를 주고 소인수분해를 시켰다. 그러자 지원이의 동생은 기겁하며 쓰러졌다. 힘들어하는 지원이의 동생을 대신해서 여러분이 이것도 쉽다는 것을 보여주자!

백준 16563번 어려운 소인수분해

주어지는 수가 최대 500만이고 100만개가 주어집니다.

$O(\sqrt{N})$이 걸리는 소인수분해를 100만번 하면 시간초과를 받습니다.

백준 16563번 어려운 소인수분해

에라토스테네스의 체를 수행할 때 소수가 아닌 i에 대해서 is_prime[i]가 처음으로 false가 되는 때를 생각합시다.

그 때의 iteration을 돌고 있는 소수가 i의 가장 작은 소인수입니다.

이를 이용합시다.

백준 16563번 어려운 소인수분해

i의 가장 작은 소인수를 p[i]라고 합시다. (p[1] = 1)

p[i] = i인 i는 1과 소수 외엔 없습니다.

그리고 이 p[i]는 에라토스테네스의 체를 수행하면서 구할 수 있습니다.

```
int MAX = 5000000;
int p[MAX+1];

void sieve() {
    for(int i=1;i<=MAX;++i) p[i] = i;
    for(int i=2;i<=MAX;++i) {
        if(p[i] == i) {
            for(int j=i+i;j<=MAX;j+=i) p[j] = i;
        }
    }
}
```

백준 16563번 어려운 소인수분해

p[i]를 전부 구했다면 소인수분해 과정은 쉬워집니다.

x를 소인수분해 한다고 치면 p[x] != x 일동안 x를 p[x]로 나눠주면 됩니다.

x를 2 이상인 수로 나누기 때문에 소인수 분해의 시간복잡도는 $O(\log x)$입니다.

백준 1222번 홍준 프로그래밍 대회

홍준 프로그래밍 대회 성공 혼처 다국어

★ 한국어

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	1814	408	275	21.089%

문제

홍준이는 프로그래밍 대회를 개최했다. 이 대회는 사람들이 팀을 이루어서 참가해야 하며, 팀원의 수는 홍준이가 정해준다. 팀원이 홍준이가 정한 값보다 부족하다면, 그 팀은 대회에 참여할 수 없다. 모든 팀은 같은 수의 팀원으로 이루어져 있다.

대회에 참여 의사를 밝힌 학교는 총 N개이다. 각 학교는 모든 학생이 참여할 수 있는 경우에만 대회에 참가한다. 즉, 남는 사람 없이 모든 학생이 팀에 들어갈 수 있어야 한다.

대회는 예선과 본선으로 구성되어 있다. 모든 팀은 같은 학교 소속으로 이루어져 있어야 한다. 예선에서 각 학교 1등팀만 본선에 진출한다.

홍준이의 대회는 올해가 첫 해이기 때문에, 많은 관심이 필요하다. 따라서, 본선에 참가하는 사람의 수를 최대가 되도록 팀원의 수를 정하려고 한다. 또, 본선이 지루해지는 것을 막기 위해 적어도 두 팀이 본선에 참가할 수 있어야 한다.

홍준이가 팀원을 몇 명으로 정해야 본선에 참가하는 사람의 수가 최대가 되는지 구하는 프로그램을 작성하시오.

백준 1222번 홍준 프로그래밍 대회

홍준이가 정한 팀원의 수를 x라고 합시다.

그러면 학생 수가 x로 나누어 떨어지는 학교만 본선에 참가가 가능합니다.

여기서 주어지는 학생 수는 최대 200만입니다.

백준 1222번 홍준 프로그래밍 대회

최대 200만의 수를 20만번 소인수분해 하면 시간초과를 받습니다.

문제를 푸는 데 중요한 관찰은 시도해볼 수 있는 모든 x가 200만개라는 점입니다.

백준 1222번 홍준 프로그래밍 대회

x를 고정하면 학생 수가 kx 꼴인 학교는 본선에 진출이 가능합니다.

학생 수가 i인 학교가 몇 개 있는지 확인 하는 것은 O(1)에 가능합니다.

그러면 모든 x를 확인하는 데는 시간이 얼마나 걸릴까요?

백준 1222번 홍준 프로그래밍 대회

[1, 2,000,000]까지의 모든 x에 대해서 x의 배수는 각각 2,000,000 / x 개 있습니다.

따라서 모든 x를 확인하는 것은 아래와 같은 시간이 걸립니다.

$$\frac{MAX}{1}+\frac{MAX}{2}+\frac{MAX}{3}+\cdots+\frac{MAX}{MAX}\quad(MAX=2,000,000)$$

위는 조화급수 꼴의 형태로 시간복잡도는 $O(MAX\log MAX)$로 시간 내에 해결이 가능합니다.

Modular Arithmetic (모듈러 연산)

문제를 풀다 보면 답을 1,000,000,007로 나눈 나머지를 출력하라는 말이 자주 보입니다.

정수끼리의 사칙연산을 어떤 주어진 수로 나눈 나머지들로 수행하는 것을 modular arithmetic이라고 부릅니다.

정수 a를 n으로 나눈 나머지를 $a \pmod{n}$으로 씁니다. 코드에선 a%n 이죠.

그리고 두 수 a, b가 n으로 나눈 나머지가 같을 때 아래와 같이 표현합니다.

$$a \equiv b \pmod{n}$$

Modular Arithmetic (모듈러 연산)

다음과 같은 사실들이 성립합니다.

$$a \equiv b \pmod{n} \text{이고 k가 정수일 때,}$$
$$a + k \equiv b + k \pmod{n}$$
$$ka \equiv kb \pmod{n}$$

이를 바꿔 말하면 모듈러 연산을 취하면서도 정수끼리의 덧셈과 곱셈, 뺄셈은 가능하다는 뜻입니다.

그러면 나눗셈은 어떻게 할까요?

Modular Arithmetic (모듈러 연산)

정수끼리의 연산이며 결과는 정수인데 올바른 나눗셈을 생각하기는 어렵습니다.

대신에 역원이라는 개념을 갖고 와서 나눗셈을 생각해봅시다.

곱셈의 항등원은 1입니다.

그러면 어떤 수 a의 곱셈에 대한 역원은 a에다가 어떤 수 x를 곱했을 때 결과가 1이 나오는 x입니다.

$$ax \equiv 1 \pmod{n}$$

Modular Arithmetic (모듈러 연산)

b를 a로 나눈 결과가 k라고 합시다.

그러면 k에다가 a를 곱한 것이 b여야 합니다.

$$b \div a \equiv k \pmod{n}$$

$$b \equiv ak \pmod{n}$$

여기서 a의 곱셈에 대한 역원 x를 양변에 곱해주면 나눗셈의 결과인 k만 남습니다.

$$bx \equiv k \pmod{n}$$

확장 유클리드 알고리즘

어떤 수 a의 n에 대한 모듈러 곱셈의 역원을 구하는 방법을 알아봅시다.

$$ax \equiv 1 \pmod{n}$$
$$ax + ny = 1$$

x를 구하기 위해서는 위와 같은 선형 디오판토스 방정식에서 (x,y)가 정수인 해를 찾아야 합니다.

확장 유클리드 알고리즘

$$ax + by = c$$

위와 같은 방정식에서 x, y가 둘 다 정수인 해가 존재하기 위해서는 c가 a, b의 최대공약수 gcd(a, b)의 정수배여야 함이 알려져 있습니다.

c를 a, b의 최대공약수 g로 놓았을 때 정수해 (x,y)를 구하는 알고리즘이 바로 확장 유클리드 알고리즘입니다.

확장 유클리드 알고리즘

$$ax + by = g$$

a = bq+r이라고 쓸 수 있습니다. q는 a를 b로 나눈 몫이고 r은 그 나머지입니다.

a에 이를 대입하면 식을 아래와 같이 변형할 수 있습니다.

$$\begin{aligned}(bq + r)x + by &= g \\ b(qx + y) + rx &= g \\ bx' + ry' &= g\end{aligned}$$

$$gcd(a, b) = gcd(b, r)$$

bx′ + ry′ = g의 정수해 (x′,y′)를 구하면 (x,y)도 구할 수 있게 됩니다.

확장 유클리드 알고리즘

ax+by=g의 문제를 bx'+ry'=g의 문제로 바꿀 수 있었고 이 과정은 유클리드 알고리즘과 유사합니다.

이제 base case만 정의하면 문제를 해결할 수 있습니다.

(a,b) -> (b,r)의 과정을 반복하다보면 r이 0이 되는 순간이 옵니다.

이 때의 a가 최대공약수 g가 되고 정수해는 (1,0)입니다.

$$a \cdot x + 0 \cdot y = g$$

확장 유클리드 알고리즘

(x′, y′)를 구했으면 이를 토대로 (x, y)를 구합니다.

$$x = y'$$
$$y = x' - qy'$$
$$q = a/b$$

```
pair<ll,pair<ll,ll>> xGCD(ll a, ll b) {    // it returns {g, {x,y}}, ax+by=g
    if(b == 0) return {a,{1,0}};
    pair<ll,pair<ll,ll>> ret = xGCD(b, a%b);
    ll g, x, y;
    g = ret.first;
    tie(x,y) = ret.second;
    return {g,{y,x-(a/b)*y}};
}
```

모듈러 역원

$$ax \equiv 1 \pmod{n}$$
$$ax + ny = 1$$

다시 역원을 구하는 문제로 돌아옵시다.

위 식에서 정수해 (x, y)가 해를 가질 조건은 우변이 a, n의 최대공약수의 정수배일 것입니다.

그러면 a, n이 서로소가 아니라면 역원은 없습니다.

모듈러 역원

$$ax \equiv 1 \pmod{n}$$
$$ax + ny = 1$$

이제 a, n이 서로소라고 가정합시다.

확장 유클리드 알고리즘으로 정수해 (x, y)를 구할 수 있고 그렇게 구한 x가 우리가 원하는 역원이 됩니다.

다만, 구현 시에는 x의 범위를 조절해줄 필요가 있습니다.

백준 14565번 역원 구하기

역원(Inverse) 구하기 성공 출처

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	759	467	360	63.380%

문제

집합 Zn을 0부터 n-1까지의 정수 집합이라고 하자. Zn ∋ a, b, c 일 때, (a+b) mod n = 0이면 b는 a의 덧셈역이라고 하고 (a*c) mod n = 1이면 c는 a의 곱셈역이라고 한다.

정수 N, A가 주어졌을 때 Zn에서의 A의 덧셈역과 곱셈역을 구하시오.

단, 곱셈역을 구할 수 없으면 -1을 출력한다.

백준 14565번 역원 구하기

A, N이 주어지고 A의 N에 대한 모듈러 곱셈 역원과 덧셈 역원을 구해야 합니다.

덧셈 역원은 N-A로 자명합니다.

곱셈역이 존재하는지 확인하는 것은 A와 N이 서로소인지 확인하면 됩니다.

이제 곱셈 역원을 구합시다.

백준 14565번 역원 구하기

A, N으로 Ax+Ny = 1의 식을 놓고 확장 유클리드를 돌리면 끝입니다.

다만 역원의 범위가 0부터 N사이가 되도록 조절해줄 필요가 있습니다.

```
pair<ll,pair<ll,ll>> xGCD(ll a, ll b) {    // it returns {g, {x,y}}, ax+by=g
    if(b == 0) return {a,{1,0}};
    pair<ll,pair<ll,ll>> ret = xGCD(b, a%b);
    ll g, x, y;
    g = ret.first;
    tie(x,y) = ret.second;
    return {g,{y,x-(a/b)*y}};
}

// return -1 if there's no mod inverse
ll mod_inverse(ll a, ll mod) {
    auto res = xGCD(a,mod);
    if(res.first > 1) return -1;
    return (res.second.first + mod) % mod;
}
```

N이 소수일 때

모듈러를 하는 수 N이 소수일 때 역원을 구하는 쉬운 방법이 있습니다.

페르마의 소정리라고 아래와 같은 성질이 성립합니다.

$$a^{N-1} \equiv 1 \pmod{N}$$

이 때, 지수를 분리하면 $aa^{N-2} \equiv 1 \pmod{N}$로 a^{N-2}이 a의 N에 대한 역원입니다.

지수승은 지난주에 배운 분할정복을 이용한 거듭제곱을 쓰면 됩니다.

백준 20412번 추첨상 사수 대작전! (Hard)

문제

입력 제한 외 난이도에 따른 문제의 차이는 없다.

APC는 매년 교내 참가자들에게 추첨상을 지급하고 있다. 올해 추첨상은 공정한 추첨을 위해 준표가 직접 작성한 난수생성기를 통해 추첨을 하고자 한다. 난수생성기란, 이론적으로 예측을 더 할 수 없도록 일련의 숫자나 심볼을 생성하는 장치이다.

> 주헌 : 형이 짠 난수생성기가 공정하다는 걸 어떻게 알아 ?
>
> 준표 : 걱정 마! c언어에서 ANSI 표준으로 사용하는 '선형합동법(Linear Congruential)' 을 구현할 거니까 ~
>
> 주헌 : 선형합동법이 뭔데 ?
>
> 준표 : 그게 뭐냐면 ..

준표의 설명을 간단히 정리해보면,

$X_1 = (a \times Seed + c) \% m$

$X_2 = (a \times X_1 + c) \% m$

...

$X_{n+1} = (a \times X_n + c) \% m$

이런 식으로 준표가 몰래 정하는 a, c, m 와 참가자들이 정하는 $Seed$ 값을 바탕으로 위 공식에 따라 난수를 생성한다는 것이었다.

> 주헌 : 음... a, c, m을 아무렇게나 잡으면 안 되지 않을까 ?
>
> 준표 : 응. Hull-Dobell 정리에 따르면 그게 맞아. 그런데 귀찮아서 그냥 m을 대충 내가 좋아하는 소수로 하려구.
>
> 주헌 : (형이 좋아하는 소수..? 씨익..)

사실 주헌이는 올해에는 추첨상을 반드시 받아내겠다는 야망이 있었다! 위 대화는 그를 위한 초석이었던 것이다! 주헌이는 준표를 너무 잘 알기 때문에 준표가 좋아하는 소수를 이미 알고 있었고, 준표가 자신이 직접 작성한 난수생성기에 문제가 없음을 참가자들에게 알려주기 위해 실제 추첨 전 난수생성기가 잘 작동한다는 것을 모두의 앞에서 시연하기로 되어있었다.

주헌이는 계략을 짰다. 주헌이는 시연 중 참가자들이 정한 $Seed$와 이를 이용해 만들어진 X_1, X_2 를 이용해 준표가 몰래 정한 a, c를 찾아낼 것이다. 만약 주헌이가 추첨상을 받지 못한다면, 찾아낸 a, c를 폭로해 모든 것이 조작되었다고 주장하며 추첨 자체를 무효로 만들 계략이다! 주헌이는 a, c를 자동으로 찾아주는 프로그램을 만들고자 한다.

입력

한 줄에 걸쳐 준표가 좋아하는 소수 m, 참가자들이 정한 Seed, 시연으로 공개된 X_1, X_2 이 주어진다. 항상 가능한 상황만 입력으로 주어진다.

백준 20412번 추첨상 사수 대작전! (Hard)

a, c를 구해야 합니다. a부터 먼저 차근차근 구합시다.

주어진 식을 가지고 놀다보면 아래와 같이 만들 수 있습니다.

$$X_1 \equiv a \times seed + c \pmod{m}$$
$$X_2 \equiv a \times X_1 + c \pmod{m}$$
$$X_2 - X_1 \equiv a(X_1 - seed) \pmod{m}$$

a 빼고는 전부 알고 있는 값입니다. m이 항상 소수라고 했기 때문에 역원은 항상 존재합니다.

따라서 $X_1 - seed$의 역원을 구해서 곱하면 a를 구할 수 있습니다.

백준 20412번 추첨상 사수 대작전! (Hard)

a를 구했다면 주어진 식에 값을 대입해서 c도 구할 수 있습니다.

m이 소수기 때문에 $X_1 - seed$의 역원은 $(X_1 - seed)^{m-2}$로 구할 수 있습니다.

시간복잡도는 $O(\log m)$이 됩니다.

백준 11401번 이항계수 3

이항 계수 3 성공

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	11775	4549	3262	43.586%

문제

자연수 N과 정수 K가 주어졌을 때 이항 계수 $\binom{N}{K}$를 1,000,000,007로 나눈 나머지를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N과 K가 주어진다. ($1 \leq N \leq 4,000,000$, $0 \leq K \leq N$)

출력

$\binom{N}{K}$를 1,000,000,007로 나눈 나머지를 출력한다.

백준 11401번 이항계수 3

경우의 수를 세는 문제에서 이항계수는 자주 쓰입니다.

N이 적당히 크고 이항계수를 소수로 나눈 값을 구하는 방법을 알아봅시다.

백준 11401번 이항계수 3

이항계수는 다음과 같은 식으로 나타낼 수 있습니다.

$$\binom{N}{K} = \frac{N!}{(N-K)!K!}$$

이 값을 그대로 구하는 것은 숫자가 매우 커져서 힘들지만 적당히 큰 소수 P로 나눈 나머지를 구하는 것은 가능합니다.

백준 11401번 이항계수 3

P는 소수이기 때문에 역원이 항상 존재합니다.

그렇다면 1부터 N까지의 팩토리얼에 대한 값과 각각의 역원도 구할 수 있다는 소리입니다.

$$\binom{N}{K} = \frac{N!}{(N-K)!K!}$$

따라서 이 식을 사용하면 이항계수를 구할 수 있습니다.

백준 11401번 이항계수 3

아래는 해당 문제를 푸는 코드입니다.

```cpp
#include<bits/stdc++.h>
using namespace std;
using ll = long long;
const ll mod = 1e9+7;

ll ipow(ll a, ll b) {
    ll ret = 1;
    while(b) {
        if(b&1) ret = ret * a % mod;
        b >>= 1; a = a * a % mod;
    }
    return ret;
}

int main() {
    int N, K; cin >> N >> K;
    ll nom = 1;
    for(ll i=2;i<=N;++i) nom = nom * i % mod;
    ll denom = 1;
    for(ll i=2;i<=K;++i) denom = denom * i % mod;
    for(ll i=2;i<=N-K;++i) denom = denom * i % mod;
    cout << nom * ipow(denom, mod-2) % mod << '\n';
    return 0;
}
```

Problem Set

- 필수 문제
  - 16563 어려운 소인수분해
  - 1222 홍준 프로그래밍 대회
  - 20412 추첨상 사수 대작전! (Hard)
  - 13977 이항계수와 쿼리

Problem Set

- 연습 문제
  - 4948 베르트랑 공준
  - 6064 카잉 달력
  - 3474 교수가 된 현우
  - 10166 관중석
  - 2725 보이는 점의 개수
  - 20946 합성인수분해
  - 17425 약수의 합
  - 1747 소수 & 팰린드롬
  - 13206 Professor KCM
  - 15897 잘못 구현한 에라토스테네스의 체
  - 11414 LCM
  - 9359 서로소
  - 15907 Acka의 리듬세상
  - 1644 소수의 연속합
  - 1630 오민식
  - 2824 최대공약수
  - 3955 캔디 분배

“Any Question?”