2025 겨울방학 알고리즘 스터디

기초 정수론

컴퓨터 공학과 20230546 서보경

목차

1. 소수를 어떻게 빠르게 구할까?
2. 에라토스테네스의 체
3. 역원이 뭘까?
4. 페르마의 소정리를 이용한 역원 구하기
5. 모듈러 역원과 분할정복을 이용한 이항계수

소수를 구하기 위해선?

지금까지 소수를 구하기 위해서는 어떤 방법을 사용했을까?

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

int main(void)
{
    cin.tie(0)->sync_with_stdio(0);
    int n = 0;
    cin >> n;
    for (int j = 0; j < n; ++j)
    {
        int k = 0;
        cin >> k;
        for (int i = 2; i <= k; ++i)
        {
            if (k % i == 0)
            {
                cout << format("{} is not prime\n", k);
                return 0;
            }
        }
        cout << format("{} is prime!\n", k);
    }
    return 0;
}
```

브루트 포스 : O(N * K)

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

int main(void)
{
    cin.tie(0)->sync_with_stdio(0);
    int n = 0;
    cin >> n;
    for (int j = 0; j < n; ++j)
    {
        int k = 0;
        cin >> k;
        for (int i = 2; i <= sqrt(k); ++i)
        {
            if (k % i == 0)
            {
                cout << format("{} is not prime\n", k);
                return 0;
            }
        }
        cout << format("{} is prime!\n", k);
    }
    return 0;
}
```

최적화 브루트 포스 : $O(N * \sqrt{K})$

너무 느리다!!

입력

자연수 K가 주어진다.(1 ≤ K ≤ 500,000)

출력

K번째 소수를 출력하자.

서브태스크 1 (5점)

- K ≤ 5000

서브태스크 2 (20점)

- K ≤ 40000

서브태스크 3 (75점)

문제에서 주어진 제약 조건 외 조건 없음

예제 입력 1 복사

```
1
```

예제 출력 1 복사

```
2
```

예제 입력 2 복사

```
3
```

예제 출력 2 복사

```
5
```

브루트 포싱을 이용해서 검사를 한다고 치면…

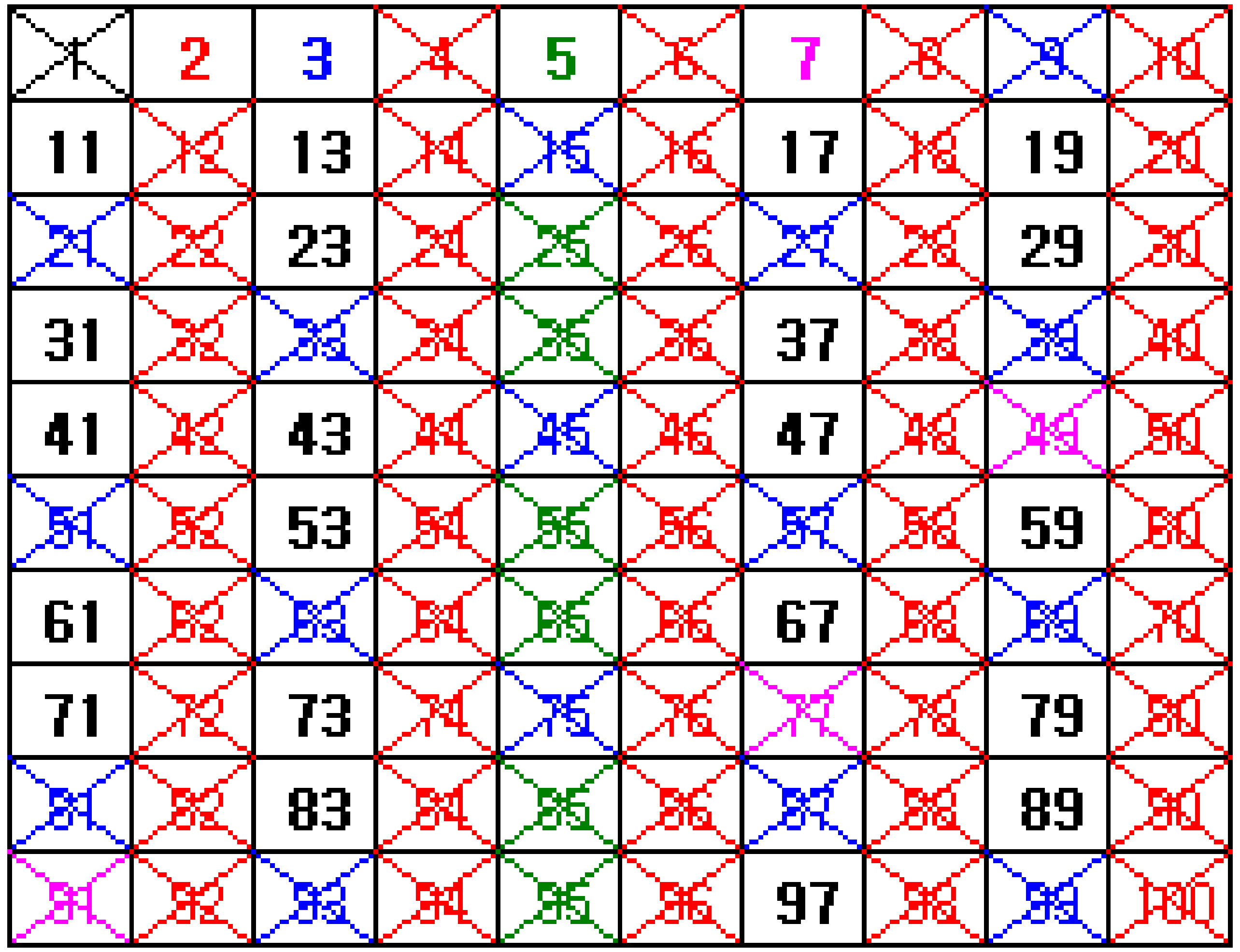
$$\sum_{i=1}^{n} isprime(i)$$

isprime() 함수가 대충 root(i)만큼 걸린다고 하자.
또한 50만번째 소수는 7500000 이하의 범위에 존재한다.

브루트 포스 연산량은 무려 13689316869개.

-> 시간내에 통과 하기 어렵다 (약 136초).

에라토스테네스의 체



고대 그리스의 수학자 에라토스테네스가 만들어 낸
특정 범위의 모든 소수를 찾는 방법이다.
체로 치듯이 수를 걸어낸다고 하여 그 이름이 명명 되었다.

시간 복잡도 : O(n $loglogn$)

tmi
에라토스테네스의 체는 특정 i ~ j 구간에 있는 모든 소수를
구하는 알고리즘 중에 제일 빠르다.

50까지의 소수를 구해보자!

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

STEP 1

먼저 n개 만큼의 배열(리스트)를 선언한다.
그 수가 소수인지 아닌지 체크하기 위함이다.

50까지의 소수를 구해보자!

~~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

STEP 2

먼저, 1은 소수가 아님에 자명하니 1을 지운 후 다음수로 넘어간다.
(배열의 경우 false 혹은 0 할당)

50까지의 소수를 구해보자!

	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

STEP 3

현재 선택 된 수는 2이다. 2는 소수이다.

50까지의 소수를 구해보자!

	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~~

STEP 4

본인을 제외한 n이하의 2의 배수를 싹 지운다.
삭제 작업을 완료 한 후, 다음 소수로 넘어간다.

50까지의 소수를 구해보자!

	2	3		5
	7		9	
11		13		15
	17		19	
21		23		25
	27		29	
31		33		35
	37		39	
41		43		45
	47		49	

STEP 2 + 3

현재 선택한 소수가 $\sqrt{50}$ 이하일때까지 삭제 작업을 반복한다.
(3의 배수 삭제, 5의 배수 삭제……)

50까지의 소수를 구해보자!

	2	3		5
	7			
11		13		
	17		19	
		23		
			29	
31				
	37			
41		43		
	47		49	

결과

소수
[2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23, 29 , 31, 37 , 41, 43, 47, 49]

왜 $\sqrt{n}$ 까지만 봐도 충분할까?

이유

어떤 숫자 n이 소수가 아니면, 즉 합성수라면 n은 두개의 약수 a와 b의 곱으로 나타낼 수 있다.

- 만약 a가 $\sqrt{n}$보다 크다면, b는 반드시 $\sqrt{n}$보다 작다.
- 그래서 $\sqrt{n}$까지만 확인해도 n이 소수인지 알 수 있다.

예를 들어, 36의 약수 쌍은 다음과 같다.
(1, 36), (2,18), (3,12), (4,9), (6,6), 여기서 $\sqrt{36}$ = 6이다.

6보다 큰 약수들은 모두 6보다 작은 약수와 짝을 이룬다. ex) 9 x 4 / 12 x 3 등등

따라서 $\sqrt{36}$ 인 6까지만 검사하면 충분하다.

만약 $\sqrt{n}$ 이상의 숫자 p가 남아 있다면

- p x p > n 이므로, 그 배수들은 이미 n 범위를 벗어난다.
- p 이전의 작은 소수들이 n 이하에서 모든 배수를 이미 지워버렸다.

$\sqrt{n}$까지만 확인하면 그 이후가 이미 다른 작은 소수들로 처리된 상태이기 때문에 $\sqrt{n}$ 범위로도 충분하다.

역원이 뭘까?

항등원

어떤 숫자와 계산해도 그 숫자가 변하지 않게 해주는 특별한 숫자.
덧셈에서는 0 / 곱셈에서는 1이다.

역원

어떤 숫자와 계산 했을 때 항등원이 되는 숫자.
덧셈에서는 –a(즉, 그 숫자의 부호를 반대로 바꾼 것)
곱셈에서는 $\frac{1}{a}$ (즉, 그 숫자의 나누기 형태)

예시

5의 덧셈 역원은 -5 -> 5 + (-5) = 0
5의 곱셈 역원은 $\frac{1}{5}$ -> 5 * ($\frac{1}{5}$) = 1

모듈러 연산과 모듈러 역원

모듈러 연산

모듈러 연산은 나머지 연산이라고도 불리며, 어떤 숫자를 다른 숫자로 나눈 뒤 나머지를 구하는 연산이다.
a mod n 라고 표현한다.

모듈러 연산은 다음과 같은 성질을 가진다.
(a + b) mod n = [(a mod n) + (b mod n)] mod n
(a x b) mod n = [(a mod n) x (b mod n)] mod n

모듈러 역원

모듈러 연산에서도 곱셈의 역원이 있다.
a x b ≡ 1 (mod n)를 만족하는 b를 a의 모듈러 역원이라고 한다. 즉 a 와 b를 곱했을 때 나머지가 항등원 1이 된다.
(a와 n이 서로소(gcd(a,n) = 1) 일 때만 역원이 존재.

예시

3 × b ≡ 1 (mod7) | 3 x 1 = 3 -> mod7 : 3 | 3 x 2 = 6 -> mod7 : 6 | 3 x 5 = 15 -> mod7 : 1
고로 위 식의 모듈러 역원은 5

페르마의 소정리를 이용한 역원 구하기

페르마의 소정리

소수 p 와 p와 서로소인 a가 있다고 하자. 그러면 $a^{p-1} \equiv 1 \pmod p$ 가 성립한다.
이 성질은 p 가 소수일 때 항상 성립한다.

페르마의 소정리를 이용한 역원 계산

a의 모듈러 역원은 $b \equiv a^{-1} (mod\ p)$로 표현되며, 페르마의 소정리에 의해서

$$a^{p-1} \equiv 1 (mod\ p) \rightarrow a^{p-2} \equiv a^{-1} (mod\ p)$$

즉, $a^{-1} \equiv a^{p-2}\ (mod\ p)$로 계산 할 수 있다!

예시

$8^{-1}\ mod\ 17 = 8^{17-2}\ mod\ 17 = 8^{15}\ mod\ 17 = 15$

$22^{-1}\ mod\ 211 = 22^{211-2}\ mod\ 211 = 22^{209}\ mod\ 211 = 48$

그래서 이걸 어따 쓰는데??

분할 정복을 이용한 거듭제곱 법

```cpp
#include <bits/stdc++.h>
#define MOD 1000000007
using namespace std;
typedef long long ll;

int main(void)
{
    cin.tie(0)->sync_with_stdio(0);
    ll a = 0, b = 0, ans = 1;
    for (ll i = 1; i <= b; ++i)
    {
        ans = (ans * a) % MOD;
    }
    return 0;
}
```

b가 작다면 충분히 빠르게 계산 할 수 있지만,
b가 10억, 아니 1000억 이라면 너무 느리다!

지수를 절반으로 나누어 계산량을 줄일 수 없을까?
-> 분할 정복을 이용한 거듭 제곱 : O($logb$)

정직하게 구한 a^b : O(b)

분할 정복을 이용한 거듭제곱 법

예시 - 2^{40} 구하기 (초기값 : a = 2, b = 40, result = 1)

단계	a	b	result
1단계	2 * 2 = 4	40 / 2 = 20	1
2단계	4 * 4 = 16	20 / 2 = 10	1
3단계	16 * 16 = 256	10 / 2 = 5	1
4단계	256 * 256 = 65536	5 / 2 = 2	1 * 256 = 256
5단계	65536 * 65536 = 4294967296	2 / 2 = 1	256
6단계	의미 없음	1 / 2 = 0	256 * 4294967296 = 1099511627776

b가 홀수 인 경우

result = result * a

홀수인 경우를 포함하여 일반적인 경우

a = a * a
b = b / 2

모듈러 역원과 분할 정복을 이용한 이항 계수

$$\binom{n}{k} = nCk = \frac{n!}{(n-k)!k!} (\text{단}, 0 \le k \le n)$$

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. ($1 \le N \le 4{,}000{,}000$, $0 \le K \le N$)

출력

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

푸는 방법

1 – 먼저, 1부터 n까지의 팩토리얼을 모두 구한다.

2 – 모듈러 역원은 단순한 모듈러 연산으로 구할 수 없다.
그러기 때문에 아까 배운 페르마의 소정리를 이용하여
역원을 구한다.

3 – 잘 조합 해주면 답이 나온다!!

주의 사항

$\prod(n-k)^{1000000005}$ 등을 구할 때도 당연히 오버 플로우의 위험성이 있기 때문에 항상 a를 제곱하거나, result에 a를 곱해줄 때도 나머지($mod\ 1000000007$) 연산을 활용하여야 한다.

또한, 모듈러 p 자체가 소수가 아니라면 페르마의 소정리를 쓸 수 없다. (확장 유클리드 호제법 사용해야함)