

수학 1

최백준 choi@startlink.io

나머지 연산

나머지 연산

Modular Arithmetic

- 컴퓨터의 정수는 저장할 수 있는 범위가 저장되어 있기 때문에, 답을 M 으로 나눈 나머지를 출력하라는 문제가 등장한다.
- $(A+B) \bmod M = ((A \bmod M) + (B \bmod M)) \bmod M$
- $(A \times B) \bmod M = ((A \bmod M) \times (B \bmod M)) \bmod M$
- 나누기의 경우에는 성립하지 않는다. (Modular Inverse를 구해야 함)
- 뺄셈의 경우에는 먼저 mod 연산을 한 결과가 음수가 나올 수 있기 때문에 다음과 같이 해야 한다.
- $(A-B) \bmod M = ((A \bmod M) - (B \bmod M) + M) \bmod M$

나머지

<https://www.acmicpc.net/problem/10430>

- 첫째 줄에 $(A+B)\%C$
- 둘째 줄에 $(A\%C + B\%C)\%C$
- 셋째 줄에 $(A \times B)\%C$
- 넷째 줄에 $(A\%C \times B\%C)\%C$
- 를 출력하는 문제

나머지

<https://www.acmicpc.net/problem/10430>

- $(A+B)\%C$ 와 $(A\%C + B\%C)\%C$ 는 같고
- $(A \times B)\%C$ 와 $(A\%C \times B\%C)\%C$ 는 같다.
- 첫째 줄과 둘째 줄, 셋째 줄과 넷째 줄의 결과가 같다.

나머지

<https://www.acmicpc.net/problem/10430>

- 소스: <http://codeplus.codes/27e6b96d80b64ec9b4e71b91793eafbd>

나머지 연산

Modular Arithmetic

7

- 문제에서 "정답을 ~~~로 나눈 나머지를 출력하라" 라는 말이 있는 이유는 정답이 int나 long long과 같은 자료형의 범위를 넘어가기 때문이다.
- 앞에서 본 것처럼 매번 나누면 된다.

나머지 연산

Modular Arithmetic

- $(6 - 5) \% 3 = 1 \% 3 = 1$ 이다.
- $(6\%3 - 5\%3) \% 3 = (0 - 2) \% 3 = -2 \% 3 = ?$

나머지 연산

Modular Arithmetic

- 음수의 경우 결과의 부호가 프로그래밍 언어마다 다르다.
- $(6\%3 - 5\%3) \% 3$
- C11, C++14: -2
- Java: -2
- Python3: 1
- 참고: https://en.wikipedia.org/wiki/Modulo_operation

나머지 연산

Modular Arithmetic

- $0 \leq a \% c < c$
- $0 \leq b \% c < c$
- 이기 때문에
- $(a \% c - b \% c)$ 의 결과는
- $-c < (a \% c - b \% c) < c$ 를 만족한다.
- 따라서, $(a \% c - b \% c + c)$ 는 0보다 큰 값을 갖기 때문에, 이 상태에서 다시 c 로 나눠주면 원하는 결과를 얻을 수 있다.

최대공약수

최대공약수

Greatest Common Divisor

- 최대공약수는 줄여서 GCD라고 쓴다.
- 두 수 A와 B의 최대공약수 G는 A와 B의 공통된 약수 중에서 가장 큰 정수이다.
- 최대공약수를 구하는 가장 쉬운 방법은 2부터 $\min(A, B)$ 까지 모든 정수로 나누어 보는 방법
- 최대공약수가 1인 두 수를 서로소(Coprime)라고 한다.

```
int g = 1;
for (int i=2; i<=min(a,b); i++) {
    if (a % i == 0 && b % i == 0) {
        g = i;
    }
}
```

최대공약수

Greatest Common Divisor

- 앞 페이지에 있는 방법보다 빠른 방법이 있다.
- 유클리드 호제법(Euclidean algorithm)을 이용하는 방법이다.
- a 를 b 로 나눈 나머지를 r 이라고 했을 때
- $\text{GCD}(a, b) = \text{GCD}(b, r)$ 과 같다
- r 이 0이면 그 때 b 가 최대 공약수이다.
- $\text{GCD}(24, 16) = \text{GCD}(16, 8) = \text{GCD}(8, 0) = 8$

최대공약수

Greatest Common Divisor

- 재귀함수를 사용해서 구현한 유클리드 호제법

```
int gcd(int a, int b) {  
    if (b == 0) {  
        return a;  
    } else {  
        return gcd(b, a%b);  
    }  
}
```

최대공약수

Greatest Common Divisor

15

- 재귀함수를 사용하지 않고 구현한 유클리드 호제법

```
int gcd(int a, int b) {  
    while (b != 0) {  
        int r = a%b;  
        a = b;  
        b = r;  
    }  
    return a;  
}
```

최대공약수

Greatest Common Divisor

16

- 세 수의 최대공약수는 다음과 같이 구할 수 있다.
- $\text{GCD}(a, b, c) = \text{GCD}(\text{GCD}(a, b), c)$
- 네 수, N개의 숫자도 위와 같은 식으로 계속해서 구할 수 있다.

최소공배수

Least Common Multiple

- 최소공배수는 줄여서 LCM이라고 한다.
- 두 수의 최소공배수는 두 수의 공통된 배수 중에서 가장 작은 정수
- 최소공배수는 GCD를 응용해서 구할 수 있다.
- 두 수 a, b 의 최대공약수를 g 라고 했을 때
- 최소공배수 $l = g * (a/g) * (b/g)$ 이다.

최대공약수와 최소공배수

<https://www.acmicpc.net/problem/2609>

- 두 수의 최대공약수와 최소공배수를 구하는 문제

최대공약수와 최소공배수

<https://www.acmicpc.net/problem/2609>

- 소스: <http://codeplus.codes/41fd6b8af1e3447ebd0096a18dd1f7af>

최소공배수

<https://www.acmicpc.net/problem/1934>

- 두 수의 최소공배수를 구하는 문제

최소공배수

<https://www.acmicpc.net/problem/1934>

- 소스: <http://codeplus.codes/5c176e0d1fa34269ba9d3922aec8774b>

소수

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N 이 소수가 되려면, 2보다 크거나 같고, $N-1$ 보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 1부터 100까지 소수
- 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

소수

Prime Number

- 소수와 관련된 알고리즘은 두 가지가 있다.
1. 어떤 수 N 이 소수인지 아닌지 판별하는 방법
 2. N 보다 작거나 같은 모든 자연수 중에서 소수를 찾아내는 방법

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N 이 소수가 되려면, 2보다 크거나 같고, $N-1$ 보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 1부터 100까지 소수
- 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97



Prime Number

```
bool prime(int n) {  
    if (n < 2) {  
        return false;  
    }  
    for (int i=2; i<=n-1; i++) {  
        if (n % i == 0) {  
            return false;  
        }  
    }  
    return true;  
}
```

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N 이 소수가 되려면, 2보다 크거나 같고, $N/2$ 보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 이유: N 의 약수 중에서 가장 큰 것은 $N/2$ 보다 작거나 같기 때문
- $N = a \times b$ 로 나타낼 수 있는데, a 가 작을수록 b 는 크다.
- 가능한 a 중에서 가장 작은 값은 2이기 때문에, b 는 $N/2$ 를 넘지 않는다.



Prime Number

```
bool prime(int n) {  
    if (n < 2) {  
        return false;  
    }  
    for (int i=2; i<=n/2; i++) {  
        if (n % i == 0) {  
            return false;  
        }  
    }  
    return true;  
}
```

소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N 이 소수가 되려면, 2보다 크거나 같고, 루트 N 보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 이유: N 이 소수가 아니라면, $N = a \times b$ 로 나타낼 수 있다. ($a \leq b$)
- $a > b$ 라면 두 수를 바꿔서 항상 $a \leq b$ 로 만들 수 있다.
- 두 수 a 와 b 의 차이가 가장 작은 경우는 루트 N 이다.
- 따라서, 루트 N 까지만 검사를 해보면 된다.



Prime Number

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

소수

Prime Number

- 컴퓨터에서 실수는 근사값을 나타내기 때문에, 루트 N과 같은 경우는 앞 페이지처럼 나타내는 것이 좋다.
- 루트 $i \leq N$ 은
- $i \leq N*N$ 과 같다.
- 어떤 수 N이 소수인지 아닌지 판별하는데 걸리는 시간 복잡도: $O(\sqrt{N})$

소수 찾기

<https://www.acmicpc.net/problem/1978>

- 입력으로 주어지는 N개의 소수 중에서 소수가 몇 개 인지 구하는 문제

소수 찾기

<https://www.acmicpc.net/problem/1978>

- 소스: <http://codeplus.codes/dd79a3ffcf2d4b90b5d392e7195724b9>

소수

Prime Number

- 어떤 수 N 이 소수인지 아닌지 알아내는데 걸리는 시간 복잡도는 $O(\sqrt{N})$ 이었다.
- $N = \text{백만인 경우: } \sqrt{N} = 1,000$
- $N = \text{1억인 경우: } \sqrt{N} = 10,000$
- 그럼, 1부터 1,000,000까지 모든 소수를 구하는데 걸리는 시간 복잡도는 몇일까?
- 각각의 수에 대해서 소수인지 아닌지 검사해야 한다.
- 각각의 수에 대해서 $O(\sqrt{N})$ 의 시간이 걸린다.
- 수는 총 N 개이기 때문에, $O(N\sqrt{N})$ 이 걸린다.
- $1,000,000 * 1,000 = 1,000,000,000 = 10\text{억} = 10\text{초}$
- 너무 긴 시간이 필요하다.

에라토스테네스의 체

Sieve of Eratosthenes

- 1부터 N까지 범위 안에 들어가는 모든 소수를 구하려면 에라토스테네스의 체를 사용한다.
 1. 2부터 N까지 모든 수를 써놓는다.
 2. 아직 지워지지 않은 수 중에서 가장 작은 수를 찾는다.
 3. 그 수는 소수이다.
 4. 이제 그 수의 배수를 모두 지운다.

에라토스테네스의 체

Sieve of Eratosthenes

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	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
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

에라토스테네스의 체

Sieve of Eratosthenes

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	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
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

에라토스테네스의 체

Sieve of Eratosthenes

- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	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	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

에라토스테네스의 체

Sieve of Eratosthenes

- 3의 배수를 지운다.

	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	
51		53		55		57		59	
61		63		65		67		69	
71		73		75		77		79	
81		83		85		87		89	
91		93		95		97		99	

에라토스테네스의 체

Sieve of Eratosthenes

- 3의 배수를 지운다.

	2	3		5		7			
11		13				17		19	
		23		25				29	
31				35		37			
41		43				47		49	
		53		55				59	
61				65		67			
71		73				77		79	
		83		85				89	
91				95		97			

에라토스테네스의 체

Sieve of Eratosthenes

- 5의 배수를 지운다.

	2	3		5		7			
11		13				17		19	
		23		25				29	
31				35		37			
41		43				47		49	
		53		55				59	
61				65		67			
71		73				77		79	
		83		85				89	
91				95		97			

에라토스테네스의 체

Sieve of Eratosthenes

- 5의 배수를 지운다.

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47		49	
		53						59	
61						67			
71		73				77		79	
		83						89	
91						97			

에라토스테네스의 체

Sieve of Eratosthenes

- 7의 배수를 지운다.

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47		49	
		53						59	
61						67			
71		73				77		79	
		83						89	
91						97			

에라토스테네스의 체

Sieve of Eratosthenes

- 7의 배수를 지운다.

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

에라토스테네스의 체

Sieve of Eratosthenes

- 11의 배수는 이미 지워져 있다.
- 2, 3, 5, 7로 인해서
- 11×11 은 121로 100을 넘기 때문에
- 더 이상 수행할 필요가 없다.
- 남아있는 모든 수가 소수이다.

	2	3		5		7			
11		13				17		19	
		23						29	
31						37			
41		43				47			
		53						59	
61						67			
71		73						79	
		83						89	
						97			

에라토스테네스의 체

Sieve of Eratosthenes

```
int prime[100]; // 소수 저장
int pn=0; // 소수의 개수
bool check[101]; // 지워졌으면 true
int n = 100; // 100까지 소수
for (int i=2; i<=n; i++) {
    if (check[i] == false) {
        prime[pn++] = i;
        for (int j = i*i; j<=n; j+=i) {
            check[j] = true;
        }
    }
}
```

에라토스테네스의 체

Sieve of Eratosthenes

- 1부터 N까지 모든 소수를 구하는 것이 목표이기 때문에, 구현할 때는 바깥 for문 (i)를 N까지 돌린다.
- 안쪽 for문 (j)는 N의 크기에 따라서, $i*i$ 또는 $i*2$ 로 바꾸는 것이 좋다.
- $i = \text{백만인 경우}$ $i*i$ 는 범위를 넘어가기 때문

소수 구하기

<https://www.acmicpc.net/problem/1929>

- M이상 N이하 소수를 모두 출력하는 문제

소수 구하기

<https://www.acmicpc.net/problem/1929>

- 소스: <http://codeplus.codes/b2bdde2729f94a0a8bed73b159cead48>

골드바흐의 추측

50

Goldbach's conjecture

- 2보다 큰 모든 짝수는 두 소수의 합으로 표현 가능하다.
- 위의 문장에 3을 더하면
- 5보다 큰 모든 홀수는 세 소수의 합으로 표현 가능하다.
- 로 바뀐다.
- 아직 증명되지 않은 문제
- 10^{18} 이하에서는 참인 것이 증명되어 있다.

골드바흐의 추측

51

<https://www.acmicpc.net/problem/6588>

- 백만 이하의 짝수에 대해서 골드 바흐의 추측을 검증하는 문제

골드바흐의 추측

52

<https://www.acmicpc.net/problem/6588>

- 소스: <http://codeplus.codes/b48af5f23dd44012a019f69a11fb6723>

에라토스테네스의 체

Sieve of Eratosthenes

- 에라토스테네스의 체를 사용한 경우
- 어떤 수 N 이 소수인지 아닌지 판별하기 위해 루트 N 방법을 사용할 필요가 없다.
- 에라토스테네스의 결과에서 지워지지 않았으면 소수, 아니면 소수가 아니기 때문이다.

팩토리얼

팩토리얼

Factorial

55

- $N! = 1 \times 2 \times \cdots \times N$
- 팩토리얼은 매우 큰 값
- $6! = 720$
- $8! = 40320$
- $10! = 3628800$

팩토리얼

<https://www.acmicpc.net/problem/10872>

- $N!$ 을 출력하는 문제

팩토리얼

<https://www.acmicpc.net/problem/10872>

- 소스: <http://codeplus.codes/975e6b0eb6c849d5b339121d946da34b>

팩토리얼 0의 개수

<https://www.acmicpc.net/problem/1676>

- $N! = 1 \times 2 \times \cdots \times N$
- 의 0이 몇 개 인지 알아내는 문제
- $10! = 3628800$
- $10!$ 이 0이 2개인 이유는 $10!$ 을 소인수분해 해보면 알 수 있다.
- $10! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 \times 8 \times 9 \times 10$
- $10! = 1 \times 2 \times 3 \times 2^2 \times 5 \times 2 \times 3 \times 7 \times 2^3 \times 3^2 \times 2 \times 5$
- $10! = 2^8 \times 3^4 \times 5^2 \times 7$
- $10! = 2^6 \times 3^4 \times 7 \times (2^2 \times 5^2) = 2^6 \times 3^4 \times 7 \times 10^2$

팩토리얼 0의 개수

<https://www.acmicpc.net/problem/1676>

- $N! = 1 \times 2 \times \cdots \times N$
- 의 0이 몇 개 인지 알아내려면 $N!$ 을 소인수분해 했을 때, 2와 5가 몇 개 나오는지 알아야 한다.
- 5의 개수가 항상 2의 개수 보다 적기 때문에, 5의 개수만 세어주면 된다.
- $N! 0\text{의 개수} = \lfloor N/5 \rfloor + \lfloor N/5^2 \rfloor + \lfloor N/5^3 \rfloor + \cdots$

팩토리얼 0의 개수

60

<https://www.acmicpc.net/problem/1676>

- 100!의 경우
- 인수로 5가 들어가는 것을 찾아보자.

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
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

팩토리얼 0의 개수

<https://www.acmicpc.net/problem/1676>

- 100!의 경우
- 인수로 5가 들어가는 것을 찾아보자.
- 여기서 25, 50, 75, 100은
- $25 \times 1, 25 \times 2, 25 \times 3, 25 \times 4 =$
- $5 \times 5 \times 1, 5 \times 5 \times 2, 5 \times 5 \times 3, 5 \times 5 \times 4$ 로
- 5가 두 개씩 들어간다.

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
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

팩토리얼 0의 개수

<https://www.acmicpc.net/problem/1676>

- $100/5$ 를 했을 때 세는 5의 개수
- = 20개
- 25, 50, 75, 100도 5의 개수를 1개로 센다
- 따라서 $100/25$ 를 한 번 더 해서
- 5의 개수를 한 번 더 세어줘야 한다.

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
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

팩토리얼 0의 개수

<https://www.acmicpc.net/problem/1676>

- $100/25 = 4$
- 25, 50, 75, 100
- 따라서, 100!의 0의 개수는 $20+4 = 24$ 개이다.

933262154439441526816992388562667
004907159682643816214685929638952
175999932299156089414639761565182
862536979208272237582511852109168
640000000000000000000000000000

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
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

팩토리얼 0의 개수

64

<https://www.acmicpc.net/problem/1676>

- 소스: <http://codeplus.codes/a8b5f043201e4a07bb62fc86f4760240>

조합 0의 개수

65

<https://www.acmicpc.net/problem/2004>

- nCm 의 0의 개수를 구하는 문제

조합 0의 개수

<https://www.acmicpc.net/problem/2004>

- 팩토리얼은 2의 개수가 5의 개수 보다 항상 많기 때문에, 5의 개수만 세어줬는데
- 조합은 어떻게 될 지 모르기 때문에, 2의 개수와 5의 개수를 동시에 세어줘야 한다.

조합 0의 개수

<https://www.acmicpc.net/problem/2004>

- 소스: <http://codeplus.codes/e3f9b8bcc33e43269e0690d237deae11>

끝

코드 플러스

<https://code.plus>

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.