

- ① 나머지 연산
- ② 최대공약수
- ③ 소수 (prime Number)

수학

최백준 choi@startlink.io

저장소
#2

$$\frac{1,000,000,000}{9}$$

03

나눗셈 나머지를

출력하기

100,000

나머지 연산

C++, Java

int

long long

(long)

$$2^{31} - 1$$

$$2^{63} - 1$$

$$10^{18}$$

$$10^{50}$$

나머지 연산

Modular Arithmetic

$$\frac{(A+B) \% C}{\times} = \frac{(A \% C + B \% C) \% C}{\times}$$

3

- 컴퓨터의 정수는 저장할 수 있는 범위가 저장되어 있기 때문에, 답을 M으로 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.

- $(A-B) \bmod M = ((A \bmod M) - (B \bmod M) + M) \bmod M$

$$(6/3) \% 3 = 2 \% 3 = 2$$

$$\frac{(6 \% 3 / 3 \% 3) \% 3}{= (0/0) \% 3}$$

나머지

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

(A)(B)(C)

- 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/b8f89204114247a391f019b1f307cd46>

나머지 연산

Modular Arithmetic

- 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.

경우의 수

Modular Arithmetic

- $$\underline{(6-5)} \% 3 = 1$$

$$\begin{aligned} & \frac{(6 \% 3 - 5 \% 3) \% 3}{=} \\ & \frac{(0 - 2) \% 3}{=} \\ & \boxed{-2 \% 3} \end{aligned}$$

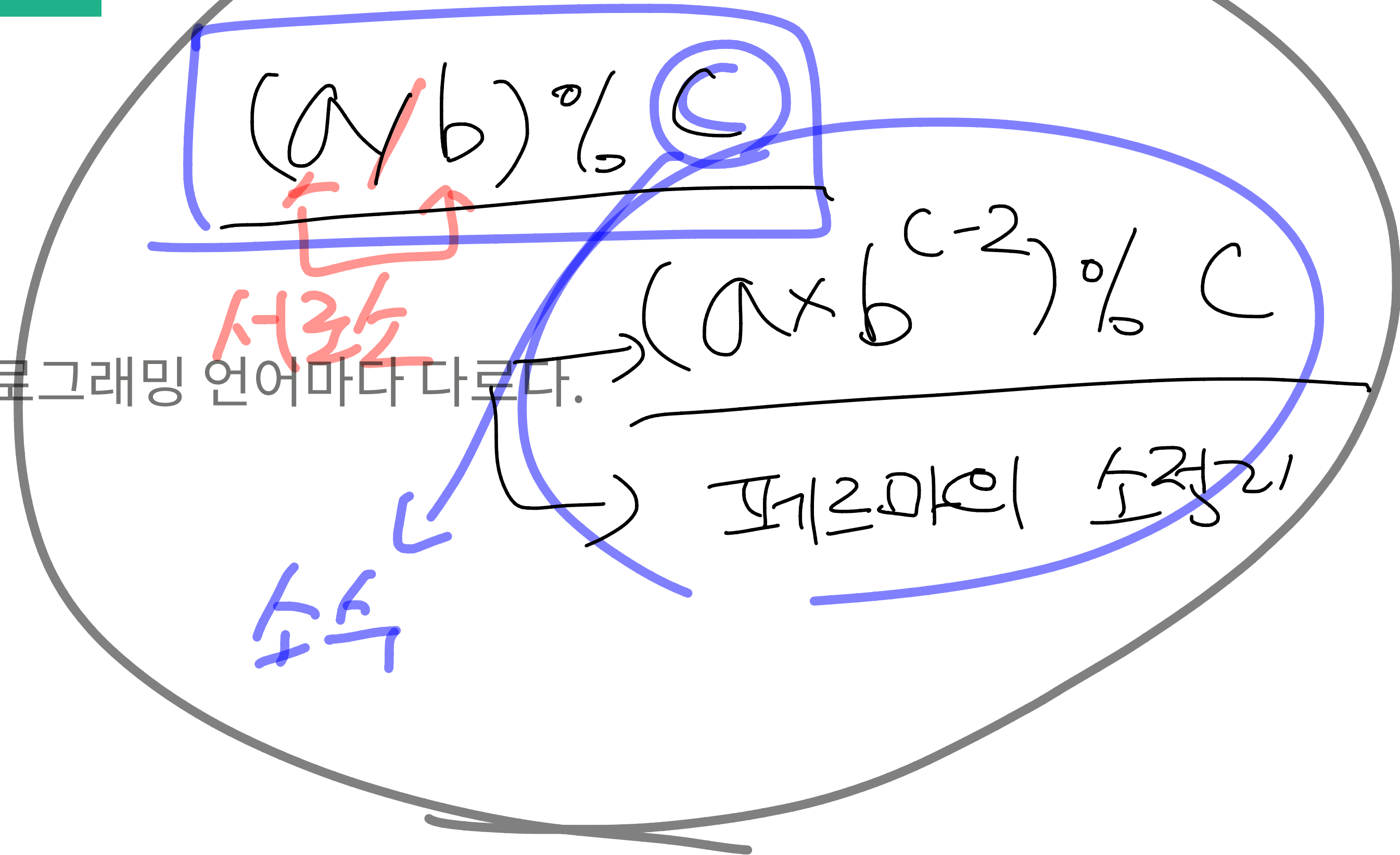
나머지 연산

Modular Arithmetic

~~0/12~~ 벨습의 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

- $-c < (a \% c - b \% c) < 2c$ 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.

$$\underline{(a \% c - b \% c) \% c}$$

$(a \% c - b \% c + c) \% c$

정답: $\left(\frac{4}{2}, 1\right)$

기약분

$$\frac{18}{24} = \frac{3}{4}$$

$$\text{GCD} = 6$$

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.

예를 들어: N을 나눌 수 있는 수

24: 1, 2, 3, 4, 6, 8, 12, 24

18: 1, 2, 3, 6, 9, 18

24와 18의 공통수: 1, 2, 3, 6
최대 " : 6

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

$g < 1$

$O(N)$

최대공약수

Greatest Common Divisor

- 앞 페이지에 있는 방법보다 빠른 방법이 있다.

- 유클리드 호제법(Euclidean algorithm)을 이용하는 방법이다.

- a를 b로 나눈 나머지를 r이라고 했을 때

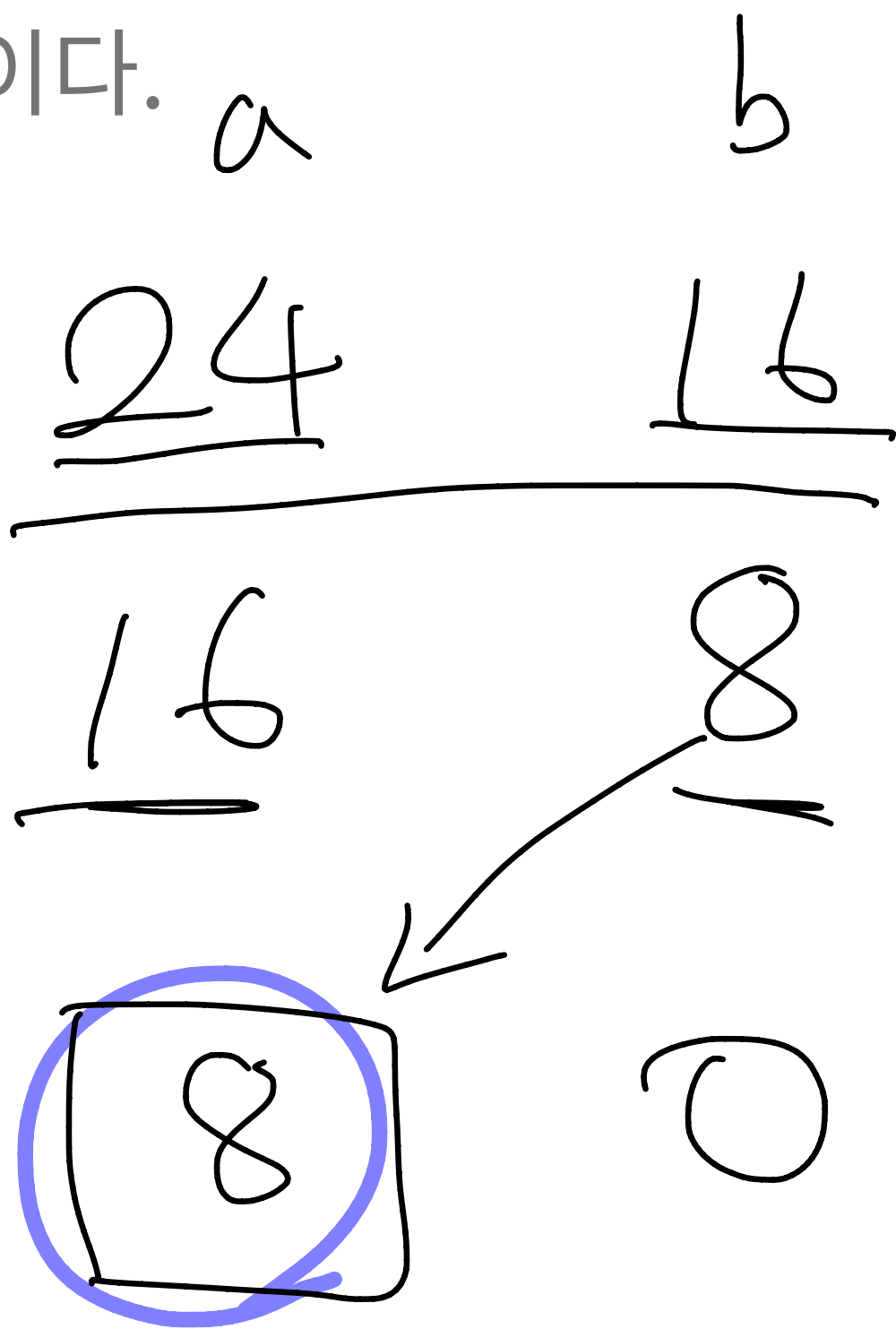
- $GCD(a, b) = GCD(b, r)$ 과 같다

- r이 0이면 그 때 b가 최대 공약수이다.

- $GCD(24, 16) = GCD(16, 8) = GCD(8, 0) = 8$

$GCD(a, b)$

$GCD(b, a \% b)$



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);  
    }  
}
```

$\text{gcd}(\underline{a}, \underline{b})$

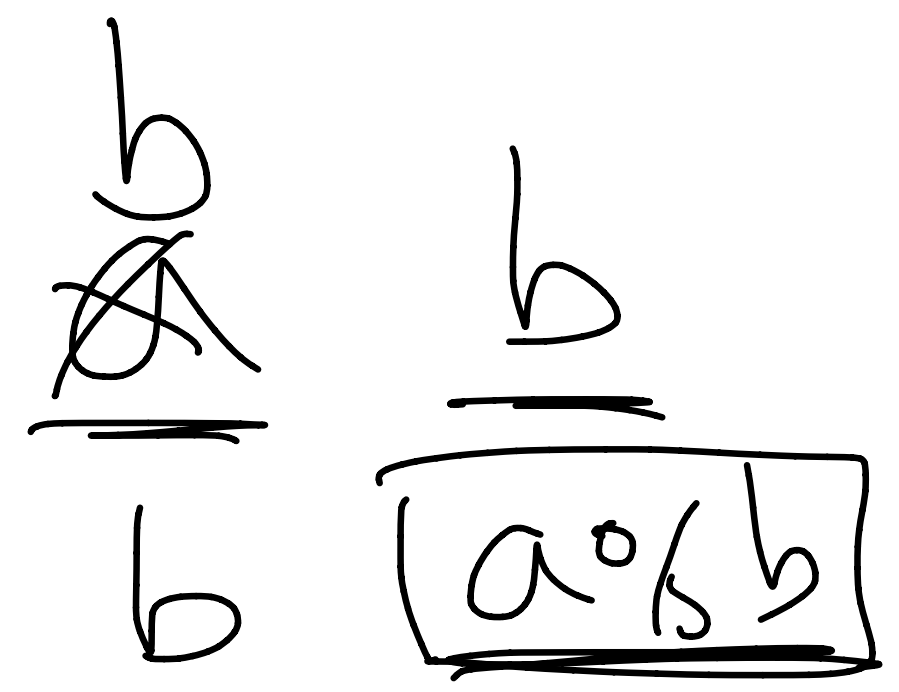
$O(\lg N)$

최대공약수

Greatest Common Divisor

- 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

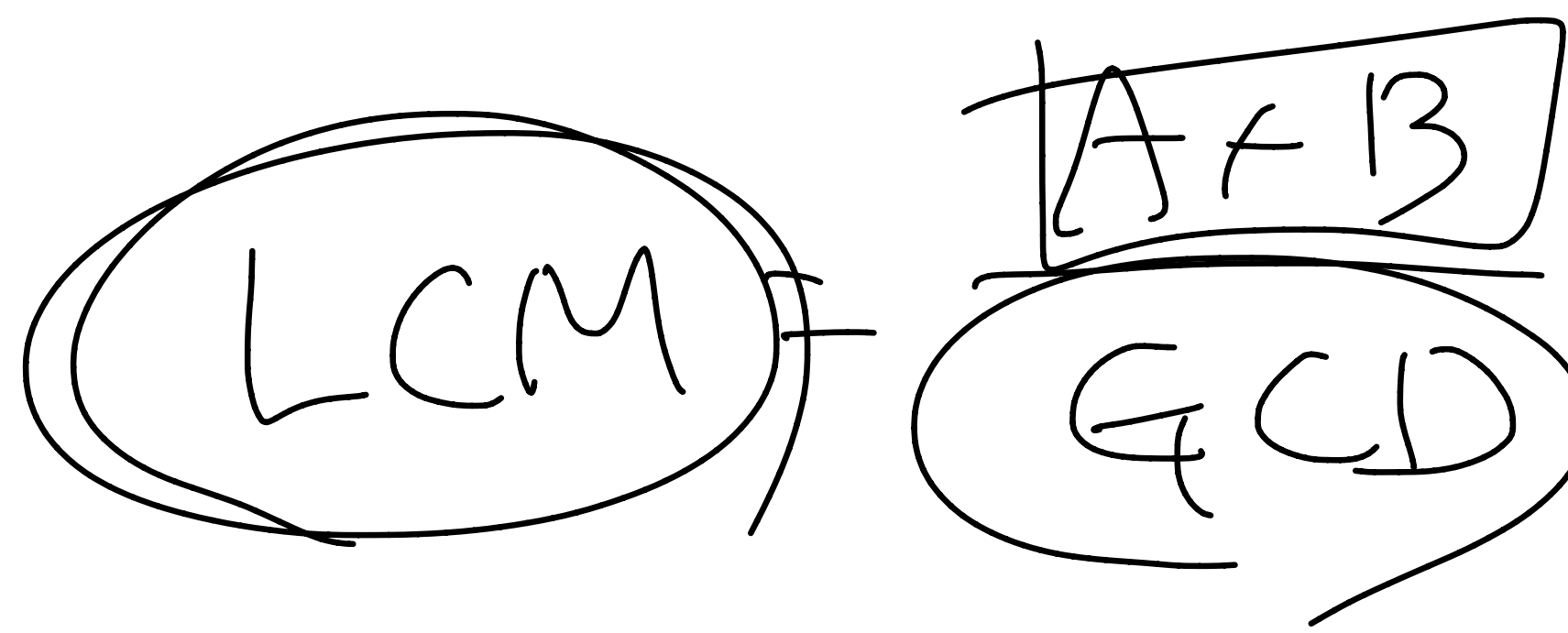
Greatest Common Divisor

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/17ad5944d8f84795915687f8a9d56e69>

최소공배수

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

- 두 수의 최소공배수를 구하는 문제

최소공배수

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

- 소스: <http://codeplus.codes/01fc3f2e0ec648d29e329afdceb31f69>

GCD 합

<https://www.acmicpc.net/problem/9613>

- 수 n 개가 주어졌을 때, 가능한 모든 쌍의 GCD의 합을 구하는 문제

GCD 합

<https://www.acmicpc.net/problem/9613>

- 소스: <http://codeplus.codes/7d9d7d27253f4ce396a487e6fd1f4092>

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

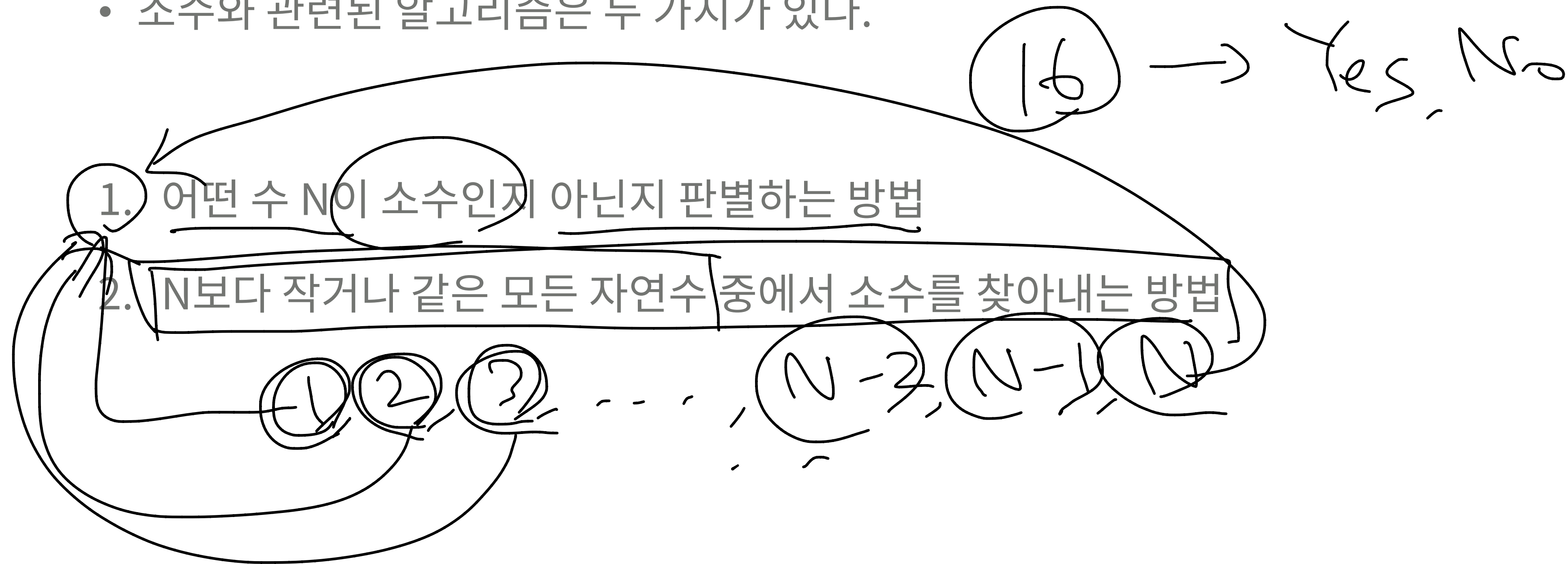
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97

소수

Prime Number

26

- 소수와 관련된 알고리즘은 두 가지가 있다.



소수

Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N 이 소수가 되려면 $(2 \text{보다 크거나 같고 } N-1 \text{보다 작거나 같은 자연수로 나누어 떨어지면 안된다.})$
- 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

```
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;  
}
```

$O(N)$

소수

Prime Number

$N = a \times b$

$a > 1$

29

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N 이 소수가 되려면, 2 보다 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 2 이기 때문에, b 는 $N/2$ 를 넘지 않는다.

소수

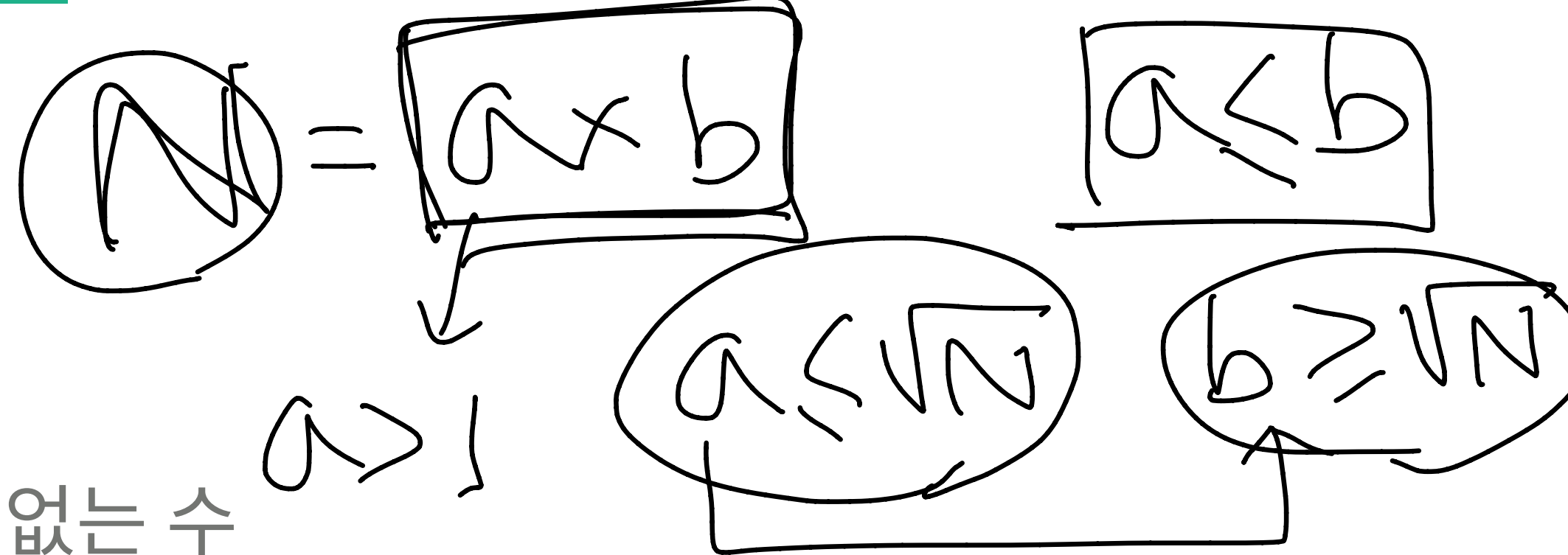
Prime Number

```
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

$$\sqrt{24} = 4.89898 \dots$$



Prime Number

- 소수: 약수가 1과 자기 자신 밖에 없는 수
- N이 소수가 되려면, 2보다 크거나 같고, 루트N 보다 작거나 같은 자연수로 나누어 떨어지면 안된다.
- 이유: N이 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 a와 b의 차이가 가장 작은 경우는 루트 N이다.
- 따라서, 루트 N까지만 검사를 해보면 된다.

$$24 = 1, 2, 3, 4, 6, 8, 12, 24$$

$$\begin{array}{lcl}
 a < \sqrt{N} & & a > \sqrt{N} \\
 b < \sqrt{N} & & b > \sqrt{N} \\
 \hline
 \frac{a \times b}{N} < \frac{\sqrt{N} \times \sqrt{N}}{N} & & \\
 \hline
 N < N & & N > N
 \end{array}$$

소수

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;  
}
```

$O(\sqrt{N})$

2

$$2 \leq i \leq \sqrt{N}$$

$$\begin{array}{r} i \leq \sqrt{N} \\ \hline i^2 \leq N \end{array}$$

소수

Prime Number

33

$$N = 10000$$

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

$$O(N) = 10000$$
$$O(\sqrt{N}) = \cancel{316} \quad 10000$$

$$10000$$

소수 찾기

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

- 입력으로 주어지는 N개의 ~~정수~~ 수 중에서 소수가 몇 개 인지 구하는 문제

소수 찾기

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

35



- 소스: <http://codeplus.codes/a08819ac592241ba9aa97ece2951e8d6>

$$\begin{aligned} \text{어떤 } \leq N \text{ 이하} &\Rightarrow O(\sqrt{N}) \\ 1 \sim N \text{ 이하} &\Rightarrow O(N\sqrt{N}) \end{aligned}$$

소수

Prime Number

- 어떤 수 N 이 소수인지 아닌지 알아내는데 걸리는 시간 복잡도는 $O(\sqrt{N})$ 이었다.

- $N = \text{백만인 경우: } \sqrt{N} = 1,000$

- $N = \text{1억인 경우: } \sqrt{N} = 10,000$

\lg 2 19. —

- 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.

2×2
 2×3
 2×4
...

	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

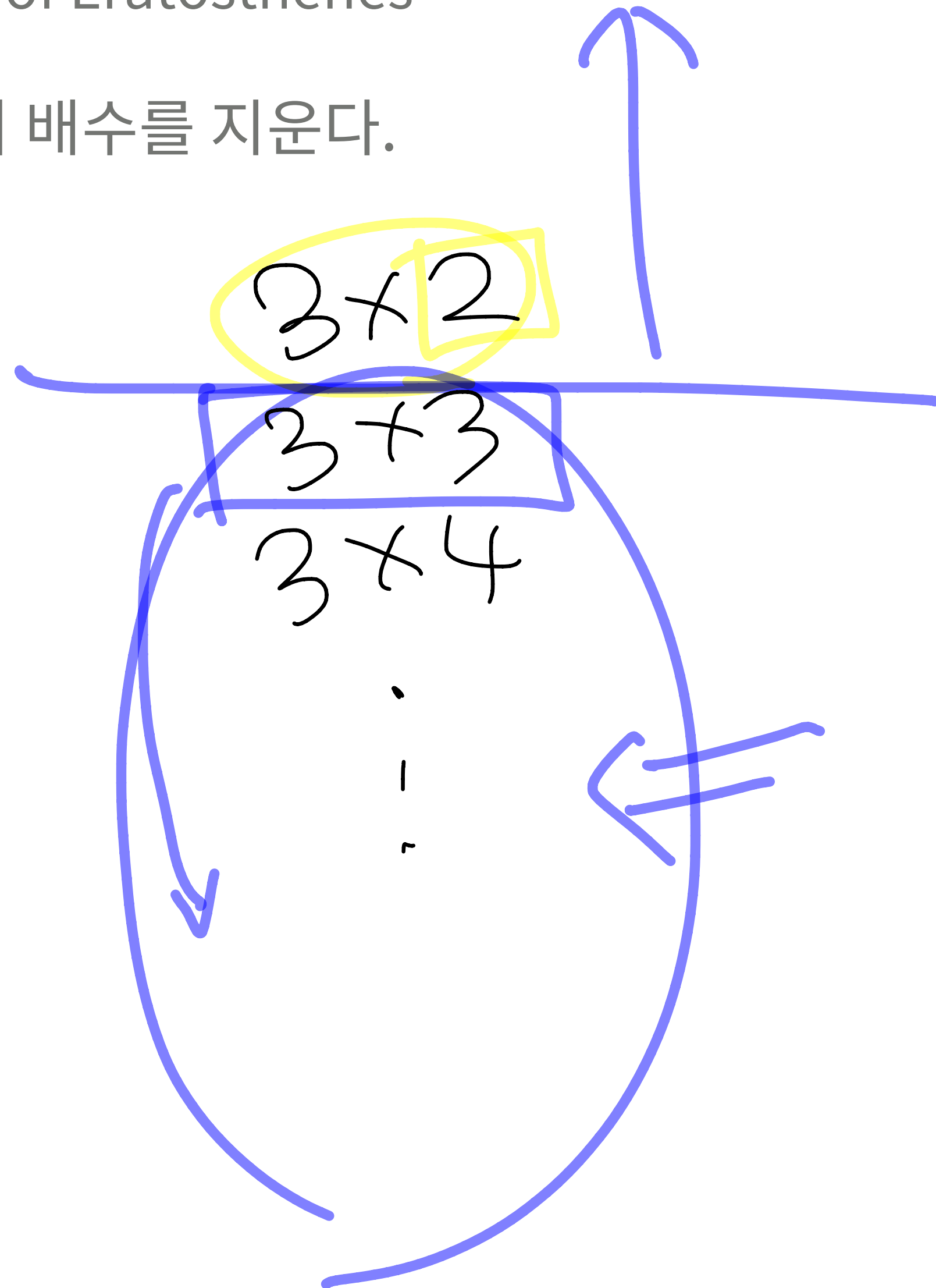
- 지워지지 않은 수 중에서 가장 작은 수는 2이다.
- 2는 소수이고 2의 배수를 모두 지운다.

	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

43

Sieve of Eratosthenes

- 5의 배수를 지운다.

5×2
 5×3
 5×4

 5×5
 5×6
 5×7

확실

	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

47

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

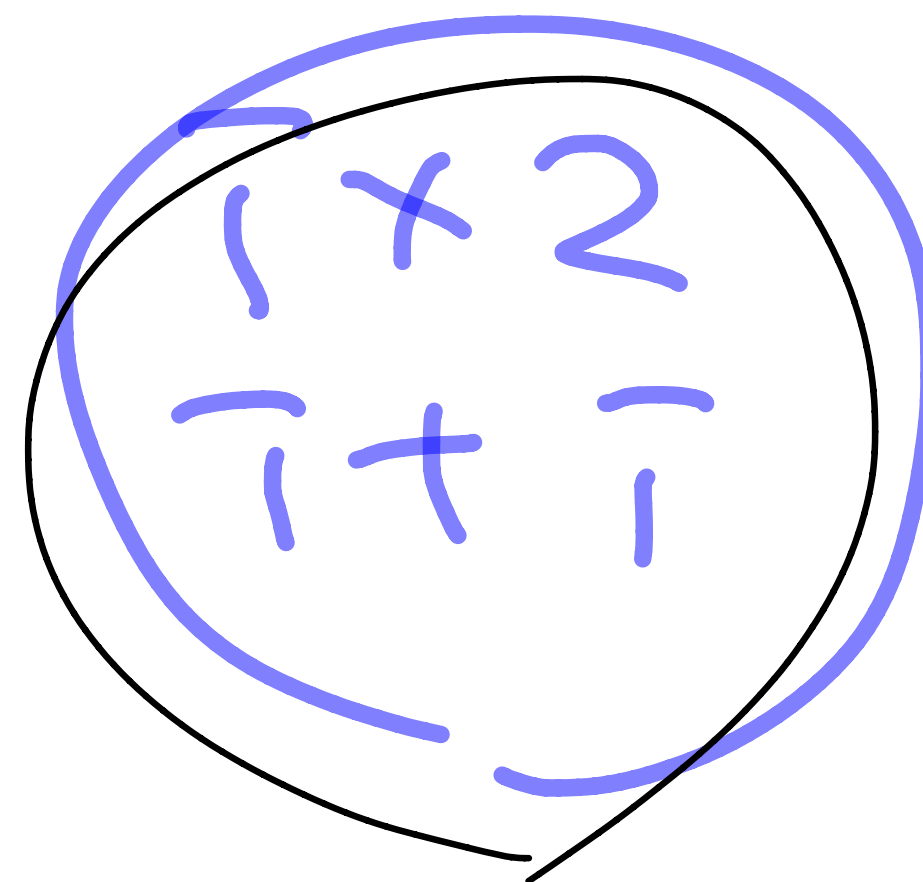
```
int prime[100]; // 소수 저장
int pn=0; // 소수의 개수
bool check[101]; // 지워졌으면 true
int n = 100; // 100까지 소수
for (int i=2; i<=n; i++) {
    if (check[i] == false) {
        prime[pn++] = i;
        for (int j = i*i; j<=n; j+=i) {
            check[j] = true;
        }
    }
}
```

$n = 1, \infty, \infty$

지워짐: true

지워지지 않음: false

$$\sqrt{x} \leq \sqrt{n}$$



에라토스테네스의 체

Sieve of Eratosthenes

$O(N \lg \lg N)$ ~~N^2~~

49

- 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 = 백만인 경우 $i*i$ 는 범위를 넘어가기 때문

Prime Harmonic Series

$$\frac{1}{2} + \frac{1}{3} + \dots \quad \lg n$$

소수 구하기

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

- M이상 N이하 소수를 모두 출력하는 문제

소수 구하기

51

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

- 소스: <http://codeplus.codes/b309225e989842b8b851df2010d57411>

골드바흐의 추측

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

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

53

- 백만 이하의 짝수에 대해서 골드바흐의 추측을 검증하는 문제

Primes ← 짝수

↓ ↓ ↓ ↓

○ ○ ○ ○ ○

P

$$p + q = N$$

$$q = N - p$$

$$N = 24$$

~~$$2 + 2 = 24$$~~

N=11
22

$$5 + 19 = 24$$

$$3 + 21 = 24$$

골드바흐의 추측

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

- 소스: <http://codeplus.codes/6a20a53695a44bffa9ebd66f7b92ac7c9>

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.

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.