알고리즘 스터디 17회

간단한 정수론과 실수

CONTENTS

소수 판정

완전 탐색

시간 복잡도 : O(N)

```
static boolean solution(int N) {
    for (int i = 2; i < N; i++) {
        if (N % i == 0) return false;
    }
    return true;
}
```

소수 판정

완전 탐색

시간 복잡도 : O(N)

```
static boolean solution(int N) {
    for (int i = 2; i < N; i++) {
        if (N % i == 0) return false;
    }
    return true;
}
```

범위 줄이기

시간 복잡도 : O(logN)

```
static boolean solution(int N) {
    for (int i = 2; i <= Math.sqrt(N); i++) {
        if (N % i == 0) return false;
    }
    return true;
}
```

약수 구하기

완전 탐색

시간 복잡도 : O(N)

```java
static List<Integer> solution(int N) {
    List<Integer> result = new ArrayList<>();

    for (int i = 1; i <= N; i++) {
        if (N % i == 0) result.add(i);
    }
    return result;
}
```

범위 줄이기

시간 복잡도 : O(logN)

```java
static List<Integer> solution(int N) {
    List<Integer> result = new ArrayList<>();

    for (int i = 1; i <= Math.sqrt(N); i++) {
        if (N % i == 0) {
            if (i * i != N) result.add(N / i);
            result.add(i);
        }
    }
    return result;
}
```

소인수 분해

완전 탐색

시간 복잡도 : O(N)

```java
static List<Integer> solution(int N) {
    List<Integer> result = new ArrayList<>();

    for (int i = 2; i <= N; i++) {
        while (N % i == 0) {
            result.add(i);
            N /= i;
        }
    }
    return result;
}
```

범위 줄이기

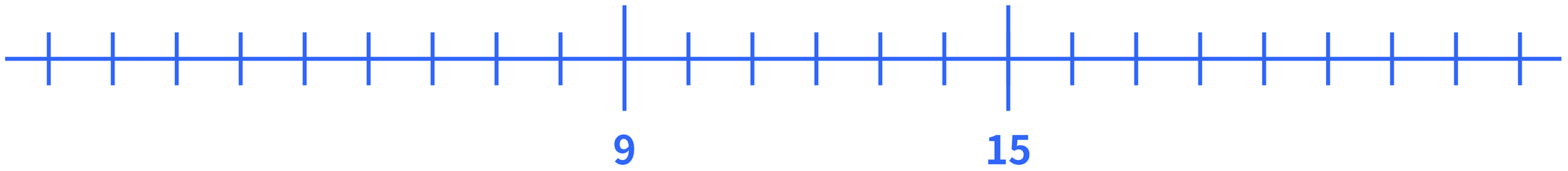
시간 복잡도 : O(logN)

```java
static List<Integer> solution(int N) {
    List<Integer> result = new ArrayList<>();

    for (int i = 2; i <= Math.sqrt(N); i++) {
        while (N % i == 0) {
            result.add(i);
            N /= i;
        }
    }
    if (N > 1) result.add(N);
    return result;
}
```

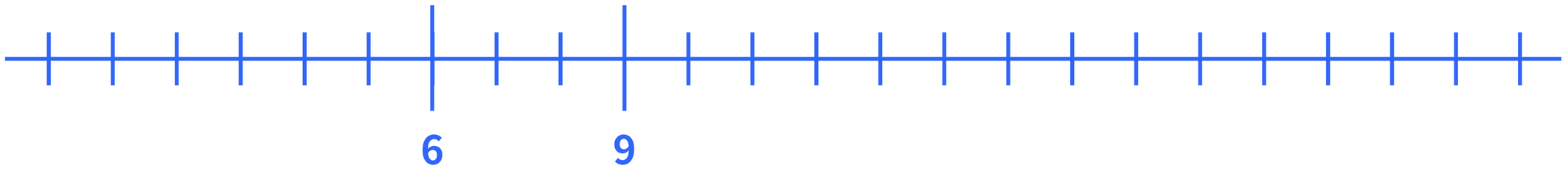
최대 공약수, 최대 공배수

유클리드 호제법(GCD)

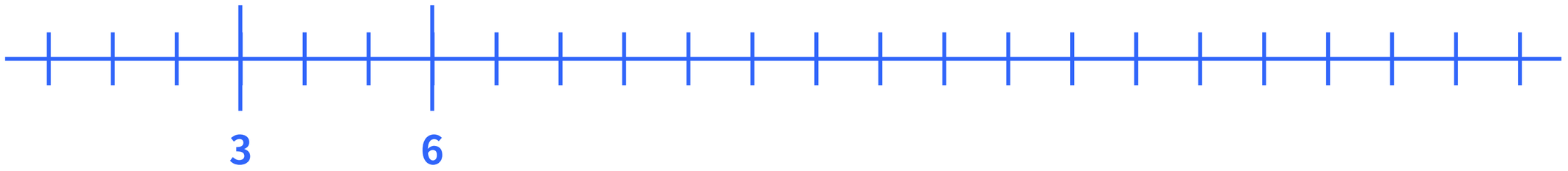
유클리드 호제법(GCD)



유클리드 호제법(GCD)



유클리드 호제법(GCD)



유클리드 호제법(GCD)

두 수의 최대 공약수는 두 수의 차도 같은 약수를 가진다

유클리드 호제법(GCD)

완전 탐색

```
static int gcd(int a, int b) {
    if (a == b) return a;
    int n = a - b;
    if (n > b) return gcd(n, b);
    return gcd(b, n);
}
```

유클리드 호제법(GCD)

하지만 3억과 3이라면?
3억이 3이 될때까지 계속 뺄 것

유클리드 호제법(GCD)

완전 탐색

```
static int gcd(int a, int b) {
    if (a == b) return a;
    int n = a - b;
    if (n > b) return gcd(n, b);
    return gcd(b, n);
}
```

불필요한 반복 줄이기

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

유클리드 호제법(GCD)

완전 탐색

```
static int gcd(int a, int b) {
    if (a == b) return a;
    int n = a - b;
    if (n > b) return gcd(n, b);
    return gcd(b, n);
}
```

불필요한 반복 줄이기 (ver 메모리 아끼기)

```
static int gcd(int a, int b) {
    while (b > 0) {
        a %= b;
        a = a ^ b;
        b = a ^ b;
        a = a ^ b;
    }
    return a;
}
```

최대 공약수, 최대 공배수

최대 공약수

```
static int solution(int a, int b) {
    return gcd(Math.max(a, b), Math.min(a, b));
}
```

최소 공배수

```
static int solution(int a, int b) {
    return a * b / gcd(Math.max(a, b), Math.min(a, b));
}
```

05

에라토스테네스의 채

1	2	3	4	5	6
7	8	9	10	11	12

05

에라토스테네스의 채

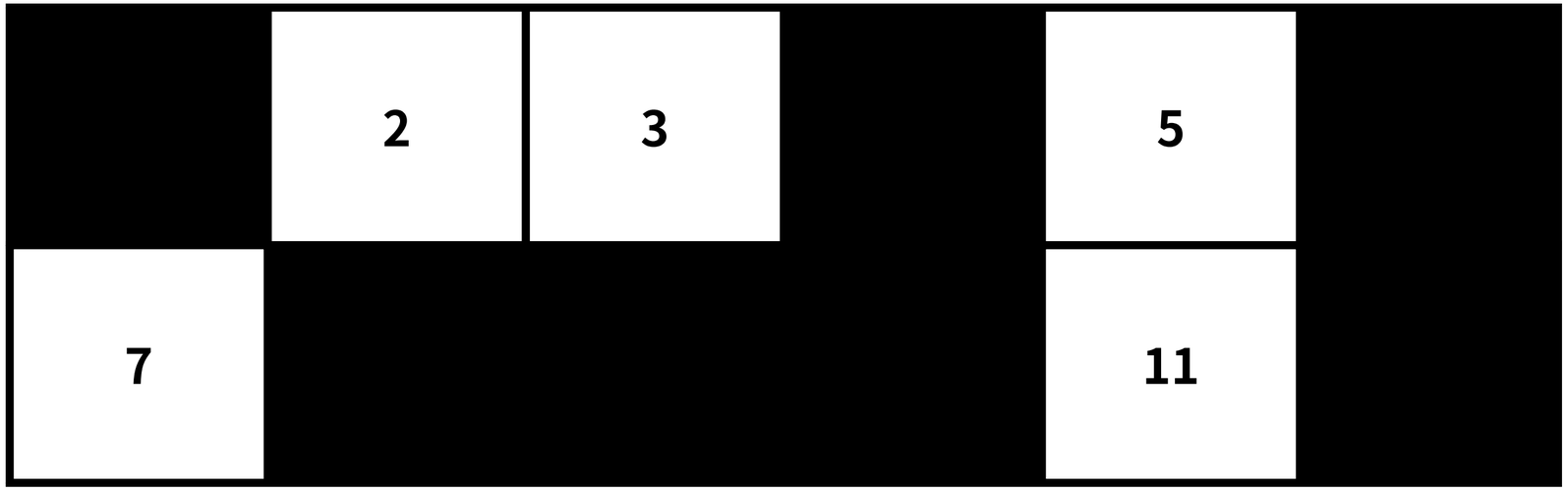
	2	3	4	5	6
7	8	9	10	11	12

05

에라토스테네스의 채

05

에라토스테네스의 채



05

에라토스테네스의 채 (완탐 : O(N^2))

```java
static List<Integer> solution(int N) {
    boolean[] isS = new boolean[N + 1];

    isS[0] = isS[1] = true;

    List<Integer> result = new ArrayList<>();

    for (int i = 2; i <= N; i++) {
        if (isS[i]) continue;

        result.add(i);

        int now = i << 1;

        while (now <= N) {
            isS[now] = true;
            now += i;
        }
    }

    return result;
}
```

05

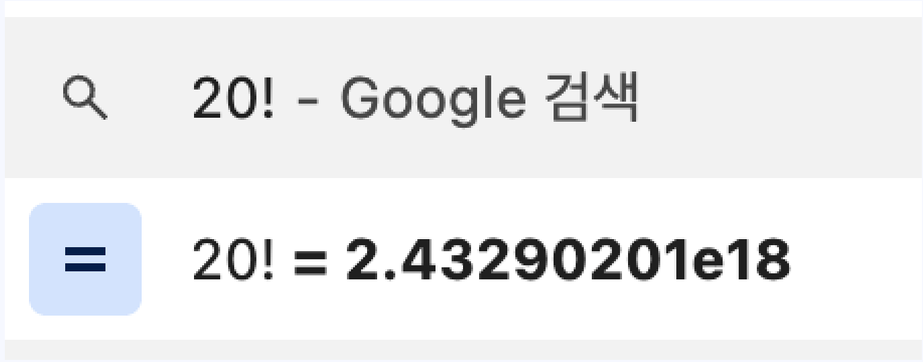
에라토스테네스의 채
(덜 탐색하기 : O(NlogN))

```java
static List<Integer> solution(int N) {
    boolean[] isS = new boolean[N + 1];

    isS[0] = isS[1] = true;

    List<Integer> result = new ArrayList<>();

    for (int i = 2; i <= Math.sqrt(N); i++) {
        if (isS[i]) continue;

        int now = i << 1;

        while (now <= N) {
            isS[now] = true;
            now += i;
        }
    }

    for (int i = 2; i <= N; i++) {
        if (isS[i]) continue;
        result.add(i);
    }

    return result;
}
```

nCr 오차없이 구하기

문제

nCr은 일단 팩토리얼 계산이라 몹시 큰 수



해결 방법

N개의 연속된 수안에는 N의 배수가 반드시 존재한다.

```
1    2    3    4    5
120  121  122  123  124
```

nCr 오차없이 구하기

```java
static long solution(int n, int r) {
    long result = 1;

    for (int i = 0; i < r; i++) {
        result *= n - i;
        result /= i + 1;
    }

    return result;
}
```

07

실수 오차 줄이기

가능하면 정수에서 끝내기

정수에서 계산하면 실수 연산을 통해
나오는 실수를 예방할 수 있다.

07

실수 오차 줄이기

선그리기 성공 다국어

골드 II

☆ 한국어

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	1605	148	98	9.674%

문제

정문이는 2차원 평면에 N개의 선을 그리려 한다. 그런데 선이 너무 많아 하나하나 다 그리려면 매우 힘들다는 것을 깨달았다. 그래서 서로 이어진 선 같은 경우에는 하나의 선으로 생각하여서 그리려 한다.

예를 들어 (1,1) 에서 (3,3)로 가는 선이 있고 (2,2)에서 (4,4)으로 가는 선이 있다면 이는 이어서 그릴 수 있으므로 하나의 선으로 생각할 수 있을 것이다. N개의 선의 정보가 주어져 있을 때, 몇 개의 선으로 그릴 수 있는지 판단하는 프로그램을 작성하시오.

입력

첫째 줄에 선분의 개수 N(1<=N<=10000) 이 주어지고 그리고 둘째 줄부터 N+1번째 줄까지 네 개의 소수가(x1, y1, x2, y2) 주어진다(최대 소숫점 둘째 자리까지). 이 말은 (x1,y1) 에서 (x2,y2)까지 선이 이어진다는 것이다. (좌표는 0 이상 1000 이하이고 선의 길이는 0보다 크다.)

출력

선의 개수를 출력하시오.

예제 입력 1 복사

```
3
1.0 10.0 3.0 14.0
0.0 0.0 20.0 20.0
10.0 28.0 2.0 12.0
```

예제 출력 1 복사

```
2
```

07

실수 오차 줄이기

```
    static int convert(String s) {
        String[] ss = s.split("\\.");
        if (ss[0].isEmpty()) {
            return Integer.parseInt(ss[1]);
        }
        if (ss.length == 1) return Integer.parseInt(ss[0]) * 100;
        return Integer.parseInt(ss[0]) * 100 + (ss[1].length() == 1 ? Integer.parseInt(ss[1]) * 10 :
Integer.parseInt(ss[1]));
    }
```

07

실수 오차 줄이기

가능하면 정수에서 끝내기

정수에서 계산하면 실수 연산을 통해
나오는 실수를 예방할 수 있다.

함수 활용

자바의 Math의 pow와 같은
함수를 활용

08

실수 이분 탐색

출력

In first and only line you should write y coordinate where you will put a lamp, real decimal number rounded to 2 decimal places.

Your output value must be within 0.01 absolute or relative error of the correct value.

예제 입력 1 복사

```
6
0 0
10 0
11 1
15 1
16 0
25 0
```

예제 출력 1 복사

```
3.00
```

소수점 2번째 자리에서 반올림

08

실수 이분 탐색

```
while (s + 0.001 <= e) {
    double m = (s + e) / 2;

    if (isP(m, lines)) {
        e = m;
    } else {
        s = m;
    }
}
```

그럼 사용되는 부분은 3번째 자리까지니까
0.001 차이나면 끝