Mathematics Basic - 1

수학 기초 1

競技プログラミングの鉄則

KPSC Algorithm Study 25/1/2 Thu.

by Haru_101

Mathematics

- 이번 시간부터는 수학과 관련된 개념들을 이용하는 기본 문제들에 대해 살펴보겠습니다.

Mathematics

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_z)

問題文

以下の Q 個の質問に答えるプログラムを作成してください。

- 質問 1 ：整数 X_1 は素数ですか？
- 質問 2 ：整数 X_2 は素数ですか？
- 質問 3 ：整数 X_3 は素数ですか？
- . . .
- 質問 Q ：整数 X_Q は素数ですか？

制約

- 入力は全て整数
- $1 \leq Q \leq 10000$
- $2 \leq X_i \leq 300000$

- 아래 쿼리 Q개에 대해 답하는 프로그램을 작성하세요.
  - 쿼리 1 : 정수 X_1은 소수인가?
  - 쿼리 2 : 정수 X_2는 소수인가?
  - ...
  - 쿼리 Q : 정수 X_3은 소수인가?

Mathematics

- 소수란, 1과 자기 자신(X)을 제외한 나머지 $2..X-1$인 수들로 나눴을 때 나눠 떨어지지 않는 수를 말합니다.
- 즉, $X \ mod \ k \neq 0, k = 2, 3, \cdots, X-1$입니다.
- 단순하게 이를 2부터 $X-1$까지 모든 수에 대해서 $X \ mod \ k$가 0인지 검증하고 소수인지 판정하면 될 듯 합니다.
- 하지만, X가 최대 30만이고, Q가 1만개니 QX가 대충 30억이 됩니다. 따라서 시간 초과가 날 수 밖에 없습니다.

Mathematics

- 여기에서 시간복잡도를 줄이는 방법을 몇 가지 소개드리겠습니다.
- X가 소수든 소수가 아니든 $X = x_1 \times x_2$ 가 성립하는 임의의 정수 x_1, x_2를 생각해봅시다.
- X를 알고, x_1을 정한다면 x_2는 $X = x_1 \times x_2$의 식에 의해 저절로 정해지게 됩니다.
- 그렇다면 x_1을 어떤 범위의 수로 정해야 할까요?

Mathematics

- 일단, $X = 15$라고 하고, x_1, x_2에 대한 표를 만들어보겠습니다.

- 표의 값이 0인 경우 그 수로 나누어 떨어짐을 의미합니다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	-														-
x_2	-														-

Mathematics

- 먼저, $x_1 = 2$일때 $x_2 = 7.5$입니다.

- $X\ mod\ x_1 \neq 0$이므로, $x_1 = 2$일때는 나누어 떨어지지 않는다(X)라고 표시합시다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	-	X													-
x_2	-														-

Mathematics

- 그 다음 $x_1 = 3$일때 $x_2 = 5$이므로, 일단 두 수는 정수입니다.
- $X\ mod\ x_1 = 0$이고, $X\ mod\ x_2 = 0$이므로 $x_1 = 3, x_2 = 5$일때 O를 표시합시다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	-	X	O												-
x_2	-				O										-

Mathematics

- 그 다음 $x_1 = 4$일때 $x_2 = \frac{15}{4}$입니다.

- $X\ mod\ x_1 \neq 0$이므로, $x_1 = 4$일때는 나누어 떨어지지 않는다(X)라고 표시합시다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	-	X	O	X											-
x_2	-				O										-

Mathematics

- 그 다음 $x_1 = 5$일때 $x_2 = 3$입니다.
- 근데, 이 두 수의 순서를 바꾸면, 아까 $x_1 = 3$일때 했던 것을 또 하는 것과 마찬가지가 됩니다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	-	X	O	X	?										-
x_2	-		?		O										-

Mathematics

- 즉, 이 수부터는 더 이상 계산을 하지 않아도 소수를 판정할 수 있음을 의미합니다.

- 좀 더 명확하게는 x_1과 x_2의 대소관계가 바뀌는 시점인 $\sqrt{X}$까지만 계산을 해도 이 수가 소수인지 판정할 수 있게 됩니다. 즉, $X = \sqrt{X} \times \sqrt{X}$이기 때문에 $x_1 = 2, \cdots, \sqrt{X}$까지만 나누어 떨어지는지 계산하면 됩니다.

- 이때 x_1, x_2는 정수이므로, 정확히는 $\lfloor\sqrt{X}\rfloor$입니다. ($\lfloor\sqrt{X}\rfloor$는 $\sqrt{X}$보다 작거나 같은 정수를 의미)

$\lfloor\sqrt{X}\rfloor$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_1	-	X	O	X	?										-
x_2	-		?		O										-

Mathematics

- 이때 시간복잡도는 $O(Q\sqrt{X})$가 되므로, 시간 내에 해결할 수 있습니다.

Mathematics

- 이를 코드로 구현하면 다음과 같습니다.

```cpp
int main() {
    fastio();
    int T;
    cin >> T;
    while(T--) {
        int X;
        cin >> X;
        bool flag = true;
        for(int i=2; i<=sqrt(X); i++) {
            if(X%i==0) flag = false;
        }
        cout << (flag ? "Yes" : "No") << '\n';
    }
}
```

Mathematics

- 이 문제는 아니지만, $Q = 1{,}000{,}000$이고 $X = 1{,}000{,}000$일때 어떻게 풀어야할까요?
- Q가 매우 커서 미리 소수를 다 구해놔야 하는 경우는 조금 힘들어보입니다.
- 이때는 '에라토스테네스의 체'라는 알고리즘이 유용합니다.

Mathematics

- 먼저 아래 표는 특정 수가 소수인지 저장하는 표입니다.

- 일단 1은 소수가 아니므로 X 표시합니다.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X																							

Mathematics

- 그 다음 2부터 쭉 반복문을 돌면서, X가 아닌 칸의 경우는 그 수가 소수라고 판정합니다. (O 표시)

- 소수라고 판정한 경우에는 그 수의 배수들을 모두 X표시합니다.
- (p가 소수면 pq는 소수가 아니다라는 논리)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X	O		X		X		X		X		X		X		X		X		X		X		X

Mathematics

- 그 다음 2부터 쭉 반복문을 돌면서, X가 아닌 칸의 경우는 그 수가 소수라고 판정합니다. (O 표시)

- 소수라고 판정한 경우에는 그 수의 배수들을 모두 X표시합니다.
- (p가 소수면 pq는 소수가 아니다라는 논리)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X	O	O	X		X		X	X	X		X		X	X	X		X		X	X	X		X

Mathematics

- 그 다음 2부터 쭉 반복문을 돌면서, X가 아닌 칸의 경우는 그 수가 소수라고 판정합니다. (O 표시)

- 소수라고 판정한 경우에는 그 수의 배수들을 모두 X표시합니다.
- (p가 소수면 pq는 소수가 아니다라는 논리)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X	O	O	X	O	X		X	X	X		X		X	X	X		X		X	X	X		X

Mathematics

- 그 다음 2부터 쭉 반복문을 돌면서, X가 아닌 칸의 경우는 그 수가 소수라고 판정합니다. (O 표시)

- 소수라고 판정한 경우에는 그 수의 배수들을 모두 X표시합니다.
- (p가 소수면 pq는 소수가 아니다라는 논리)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X	O	O	X	O	X	O	X	X	X		X		X	X	X		X		X	X	X		X

Mathematics

- 그 다음 2부터 쭉 반복문을 돌면서, X가 아닌 칸의 경우는 그 수가 소수라고 판정합니다. (O 표시)

- 소수라고 판정한 경우에는 그 수의 배수들을 모두 X표시합니다.
- (p가 소수면 pq는 소수가 아니다라는 논리)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X	O	O	X	O	X	O	X	X	X	O	X		X	X	X		X		X	X	X		X

Mathematics

- 그 다음 2부터 쭉 반복문을 돌면서, X가 아닌 칸의 경우는 그 수가 소수라고 판정합니다. (O 표시)

- 소수라고 판정한 경우에는 그 수의 배수들을 모두 X표시합니다.
- (p가 소수면 pq는 소수가 아니다라는 논리)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
x	X	O	O	X	O	X	O	X	X	X	O	X	O	X	X	X	O	X	O	X	X	X	O	X

Mathematics

- 이를 코드로 구현하면 다음과 같습니다. (is_prime은 기본적으로 모두 true로 초기화해야 합니다.)

```
bool is_prime[300005];
void eratos() {
    for(int i=1; i<=300000; i++) {
        is_prime[i] = true;
    }
    is_prime[1] = false;
    for(int i=2; i<=300000; i++) {
        if(is_prime[i] == true) {
            for(int j=2*i; j<=300000; j+=i) {
                is_prime[j] = false;
            }
        }
    }
}

int main() {
    fastio();
    int T;
    cin >> T;
    eratos();
    while(T--) {
        int X;
        cin >> X;
        cout << (is_prime[X] ? "Yes" : "No") << '\n';
    }
}
```

Mathematics

- j의 초기값을 다르게 하는 방법도 있습니다. (이 방법이 더 빠릅니다. (상수차이))
- 이때는 j가 오버플로우가 날 수 있으므로 i, j 모두 long long int를 사용해야합니다.

```
bool is_prime[300005];
void eratos() {
    for(int i=1; i<=300000; i++) {
        is_prime[i] = true;
    }
    is_prime[1] = false;
    for(ll i=2; i<=300000; i++) {
        if(is_prime[i] == true) {
            for(ll j=i*i; j<=300000; j+=i) {
                is_prime[j] = false;
            }
        }
    }
}

int main() {
    fastio();
    int T;
    cin >> T;
    eratos();
    while(T--) {
        int X;
        cin >> X;
        cout << (is_prime[X] ? "Yes" : "No") << '\n';
    }
}
```

Mathematics

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/math_and_algorithm_o)

問題文

A と B の最大公約数を求めてください。

制約

- $1 \leq A, B \leq 10^9$
- A, B は整数

- 정수 A, B의 최대공약수를 구하는 프로그램을 작성하세요.

Mathematics

- 단순히 1부터 $\min(A, B)$까지로 두 수를 나눴을때 나누어 떨어지면 그 수들 중 최대가 최대공약수가 됩니다.
- 하지만 $A, B \le 10^9$인 상황에서 $O(\min(A, B))$는 어려워 보입니다.
- 이를 빠르게 구하는 방법인 유클리드 호제법을 소개하겠습니다.

Mathematics

- 유클리드 호제법은 다음과 같이 동작합니다.

- A, B중에 큰 값을 작은값으로 나눈 나머지로 교체
- 이걸 계속 반복하다가 둘 중 하나라도 0이 되면 종료하고 0이 아닌 값이 최소공배수가 됨

- $A = 117, B = 432$인 경우를 예로 들어봅시다.
- $\min(A, B) = 117$이므로 117번의 연산이 필요하지만, 유클리드 호제법을 이용하면 더 빠르게 풀 수 있습니다.

Mathematics

- 먼저 $A = 117, B = 432$이므로 B를 $B \ mod \ A$로 교체합니다.
  - $B \ mod \ A = 432 \ mod \ 117 = 81$
- 그 다음 $A' = 117, B' = 81$이므로 A'를 $A' \ mod \ B'$로 교체합니다.
  - $A' mod \ B' = 117 \ mod \ 81 = 36$
- 그 다음 $A'' = 36, B'' = 81$이므로 B''를 $B'' mod \ A''$로 교체합니다.
  - $B'' mod \ A'' = 81 \ mod \ 36 = 9$
- 그 다음 $A''' = 36, B''' = 9$이므로 A'''를 $A''' mod \ B'''$로 교체합니다.
  - $A''' mod \ B''' = 36 \ mod \ 9 = 0$
- 그 다음 $A'''' = 0, B'''' = 9$이므로 $B'''' = 9$가 최소공배수가 됩니다.

Mathematics

- 유클리드 호제법의 성질은 $A' + B'$의 값이 이전 $A + B$값의 $\frac{2}{3}$이하가 된다는 성질을 띄고 있어서 간단하게만 말하면, 시간복잡도는 $O(\log_{1.5}(A + B)) = O(\log(A + B))$ 가 됩니다.

- $(A + B) \geq \frac{3}{2}(A' + B')$

Mathematics

- 코드로 구현하면 다음과 같습니다.

```
ll gcd(ll a, ll b) {
    while(a > 0 && b > 0) {
        if(a > b) {
            a = a%b;
        } else {
            b = b%a;
        }
    }
    if(a==0) return b;
    else return a;
}

int main() {
    fastio();
    ll A, B;
    cin >> A >> B;
    cout << gcd(A, B);
}
```

Mathematics

- 번외로 최소공배수를 구하는법도 있습니다.

- 최소공배수는 다음 성질을 갖고 있습니다.
- $lcm(A, B) \times \gcd(A, B) = A \times B$

- 따라서 식을 정리하면 $lcm(A, B) = \frac{A \times B}{\gcd(A,B)}$가 됩니다.

- 이를 이용해서 최소공배수를 구하면 됩니다.

Mathematics

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/math_and_algorithm_aq)

問題文

a^b を $1000000007\ (= 10^9 + 7)$ で割った余りを計算してください。

制約

- $1 \le a \le 100$
- $1 \le b \le 10^9$
- 入力はすべて整数

- 정수 a, b에 대해 a^b를 1,000,000,007로 나눈 나머지를 구하는 프로그램을 작성하세요.

Mathematics

- 먼저, 이 문제를 풀기 위한 나머지에 대한 성질을 몇 가지 소개하겠습니다.

- $(A+B)\ mod\ C = (A\ mod\ C) + (B\ mod\ C)$
- $AB\ mod\ C = (A\ mod\ C) \times (B\ mod\ C)$

- 또한 중학교때 배웠던 거듭제곱의 성질에 대해 상기해보면…

- $a^x \times a^y = a^{x+y}$가 성립하므로, $x = y$라고 하면, $a^x \times a^x = a^{2x}$가 됩니다.
- $2x = t$라고 하면, $a^t = a^{\frac{t}{2}} \times a^{\frac{t}{2}}$가 됩니다. 이때 $a^{\frac{t}{2}}$가 2번 있으므로 $a^{\frac{t}{2}}$를 1번만 구하고 나서 2번 곱하면 됩니다.

- 만약 $2x+1 = t$인 경우 (t가 홀수) $a^t = a^{\frac{t}{2}} \times a^{\frac{t}{2}} \times a^1$이 되므로, $2x = t$인 경우에서 a를 곱해주면 됩니다.

Mathematics

- 하지만 $a^{\frac{t}{2}}$값이 매우 클 수 있으므로 이 값을 $10^9 + 7$로 나눈 나머지를 취해야 합니다.
- 따라서 $a^t = \left(a^{\frac{t}{2}}\ mod\ 10^9 + 7\right) \times \left(a^{\frac{t}{2}}\ mod\ 10^9 + 7\right)$를 이용해서 구하면 됩니다.

Mathematics

- 코드로 구현하면 다음과 같습니다. ($m=10^9+7$)

```
ll pow(ll a, ll b, ll m) {
    if(b==1) return a%m;
    else {
        ll tmp = pow(a, b/2, m);
        ll ret = ((tmp%m) * (tmp%m))%m;
        if(b%2==1) ret = (ret * (a%m))%m;
        return ret%m;
    }
}
int main() {
    fastio();
    ll A, B;
    cin >> A >> B;
    cout << pow(A, B, mod);
}
```

Mathematics

- 다음 문제를 풀어봅시다. (https://atcoder.jp/contests/tessoku-book/tasks/tessoku_book_ae)

問題文

1 以上 N 以下の整数のうち、$3, 5$ のいずれかで割り切れるものは何個ありますか。

制約

- N は 1 以上 10^{12} 以下の整数

- 1 이상 N이하의 정수 중에서 3 혹은 5로 나누어 떨어지는 수는 몇 개인지 구하는 프로그램을 작성하세요.
- N은 1 이상, 10^{12}이하의 정수

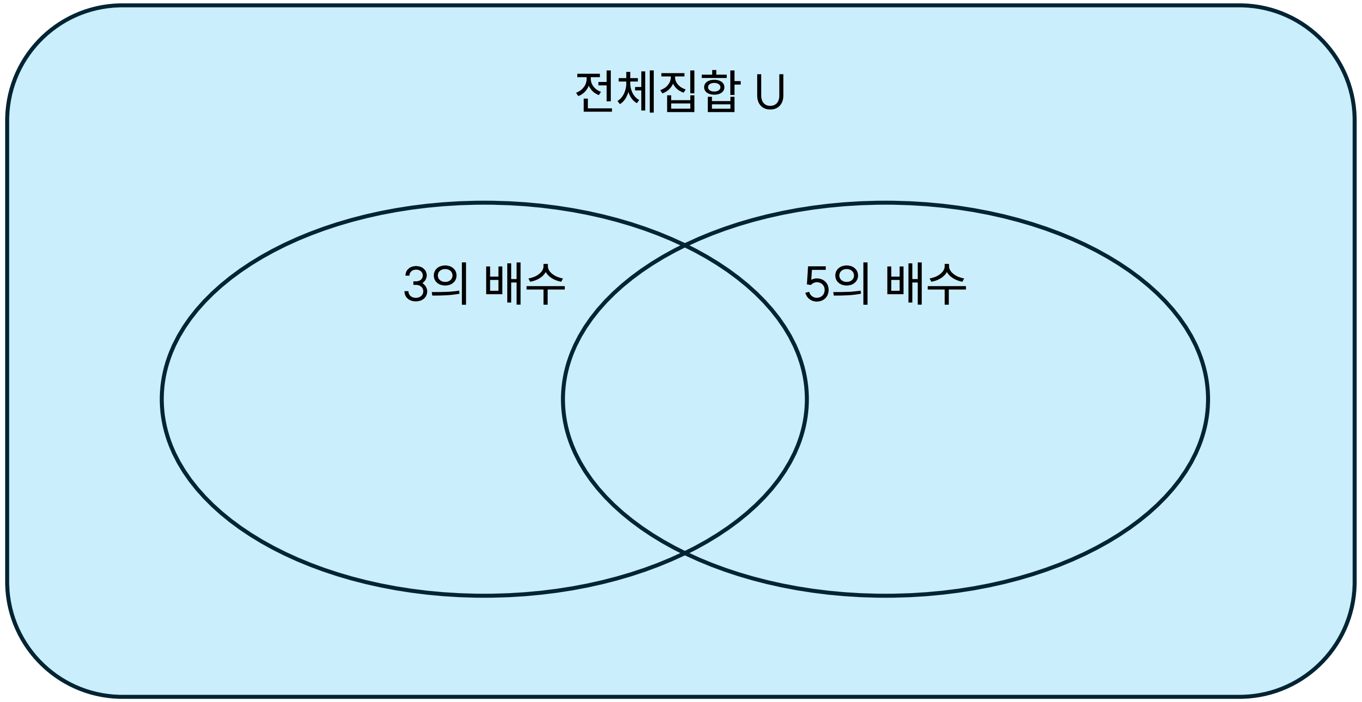
Mathematics

- 1부터 N까지 모든 정수에 대해 3의 배수인지, 5의 배수인지 검사하는 것은 $O(N)$이기 때문에 이 문제에선 시간초과가 납니다.

- 따라서 이를 효율적으로 계산하는 방법이 필요합니다.

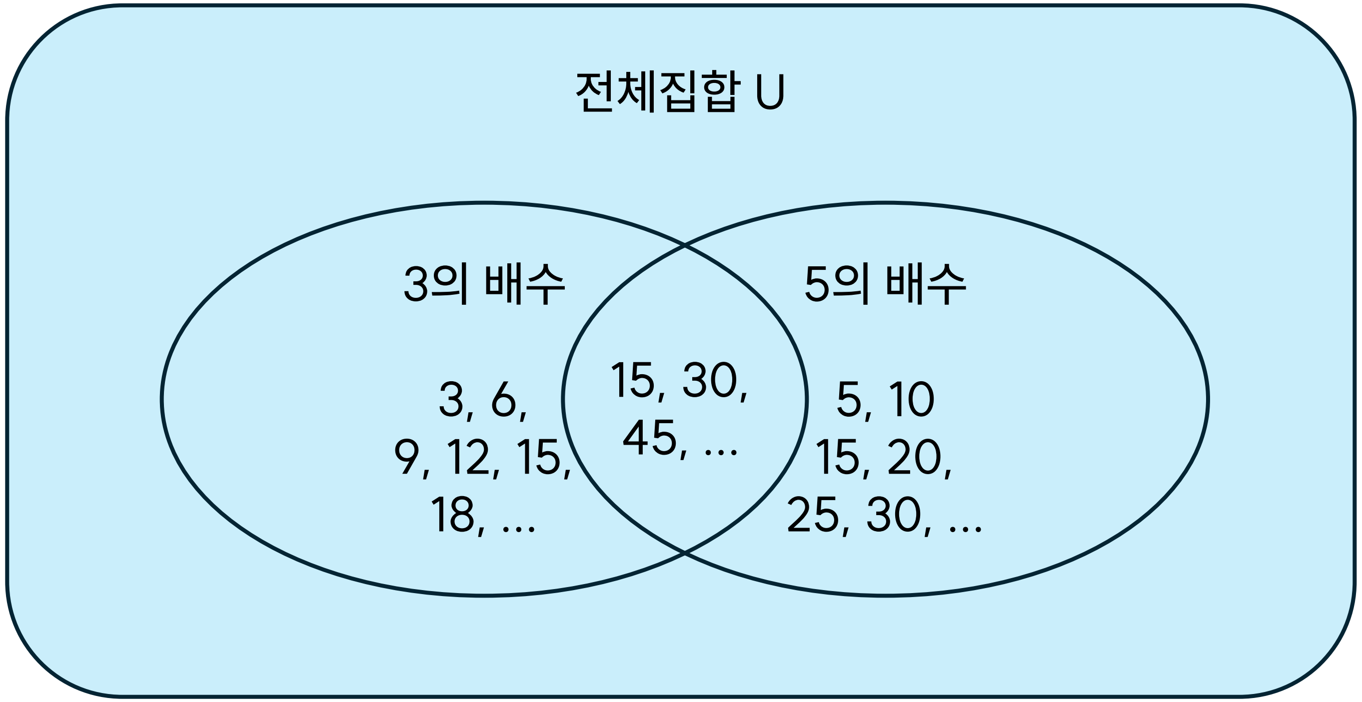
Mathematics

- 이 문제를 집합으로 생각해봅시다.



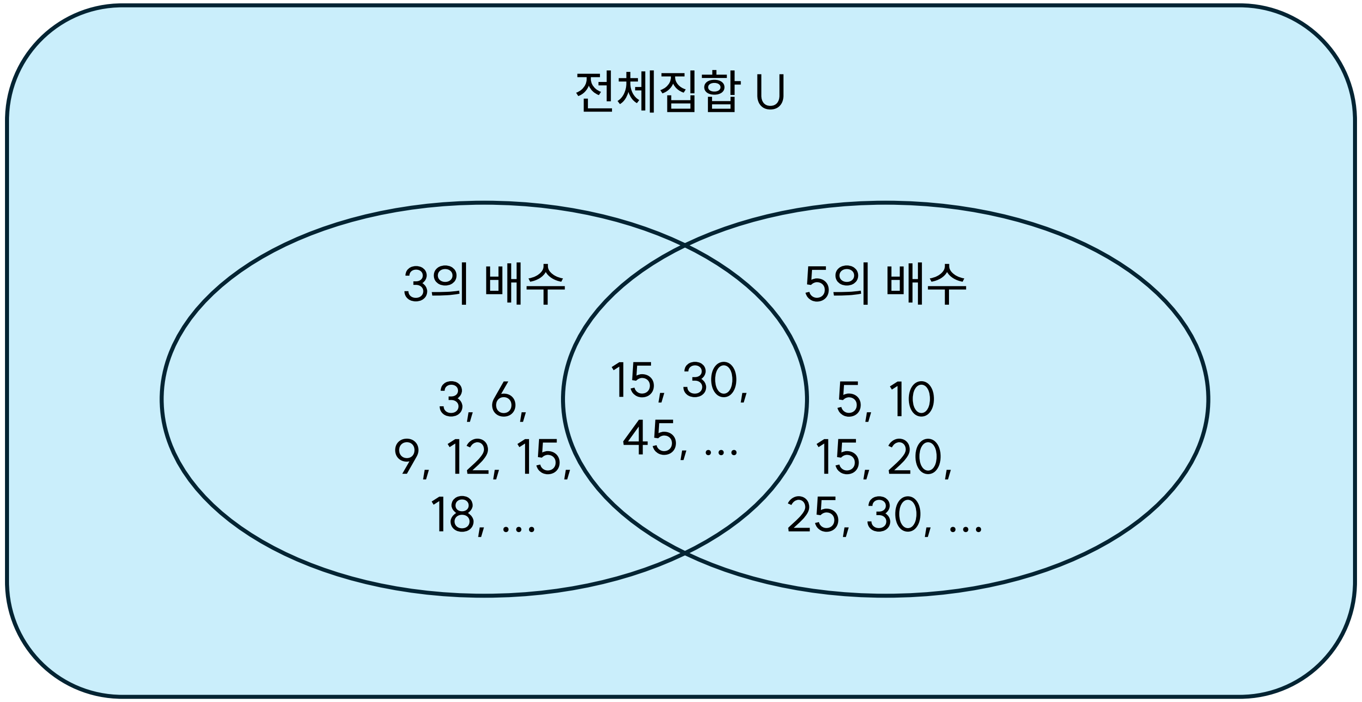
Mathematics

- 먼저 3의 배수 집합에는 3, 6, 9, 12, 15, 18, ...과 같은 수가, 5의 배수 집합에는 5, 10, 15, ...과 같은 수가 있습니다.
- 이 집합에서 겹치는 수들은 15, 30, 45, 60, ...과 같이 15의 배수가 겹치게 됩니다. (즉 카운팅이 2번됨)



Mathematics

- 즉, 3의 배수 집합과 5의 배수 집합의 합집합에 있는 요소의 개수를 구해야 합니다.
- 집합의 성질에 따라 $|A \cup B| = |A| + |B| - |A \cap B|$가 성립하므로,
  (3 혹은 5의 배수의 개수) = 3의 배수의 개수 + 5의 배수의 개수 – 15의 배수의 개수가 됩니다.
- 이런식으로 여러 개의 합집합의 크기를 구하는 원리를 포함배제의 원리라고 합니다.



Mathematics

- 즉, $\left\lfloor\frac{N}{3}\right\rfloor + \left\lfloor\frac{N}{5}\right\rfloor - \left\lfloor\frac{N}{15}\right\rfloor$가 정답이 됩니다. (코드는 간단해서 첨부하진 않겠습니다.)

Mathematics Basic - 1

끝.

다음주에는 Mathematics Basic - 2로 찾아뵙겠습니다.