2. Mathematic
#1

2017010698
수학과 오서영

Permutation (순열)

: n가지 종류에서 r개를 순서[위치, 자리]를 고려해서 뽑는 경우

$$ {}_nP_r = n(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!} = (n)_r $$

Combination(조합)

: n가지 종류에서 r개를 순서를 고려하지 않고 뽑는 경우

$$ {}_nP_r = {}_nC_r \cdot r! \quad \Rightarrow \quad {}_nC_r = \binom{n}{r} = \frac{n!}{(n-r)!r!} = \frac{(n)_r}{r!} $$

Prime Number (소수)

: 약수가 1 과 자기 자신밖에 없는 1보다 큰 양수

```
def is_prime(num):
    if num <= 1:
        return False
    for i in (2, num):
        if num % i == 0:
                return False
    return True
```

Sieve of Eratosthenes(에라토스테네스의 체)

: 소수를 찾는 방법

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

1. 모든 수를 소수라고 가정 (1은 제외)

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

2. 2 자신을 제외한 모든 2의 배수를 소수 목록에서 제거

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

3. 제거되지 않은 수 중 가장 작은 값은 3
3 자신을 제외한 3 의 배수를 모두 제거한다.

Result

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

위와 같은 과정을 나열된 수의 가장 큰 값인 50 까지 반복하여 최종적으로 남아있는 수가 소수가 된다.

GCD(Greatest Common Divisor) : 최대 공약수
LCM(Least Common Multiple) : 최소 공배수

Euclidean Algorithm (유클리드 호제법)

: 2개의 자연수 또는 정식의 최대공약수를 구하는 알고리즘

2개의 자연수 a, b에 대해서 a를 b로 나눈 나머지를 r (단, a>b)
a와 b의 최대공약수는 b와 r의 최대공약수와 같다.
이 성질에 따라, b를 r로 나눈 나머지 r′를 구하고,
다시 r을 r′로 나눈 나머지를 구하는 과정을 반복하여
나머지가 0이 되었을 때 나누는 수가 a와 b의 최대공약수

1071과 1029의 최대공약수?

1. 1071은 1029로 나누어떨어지지 않기 때문에,
1071을 1029로 나눈 나머지를 구한다. => 42
2. 1029는 42로 나누어떨어지지 않기 때문에,
1029를 42로 나눈 나머지를 구한다. => 21
42는 21로 나누어떨어진다.
따라서, 최대공약수는 21이다.

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

Matrix(행렬)

행렬 : 직사각형 형태로 수가 배열된 것

행 : 행렬의 가로줄
열 : 행렬의 세로줄

m X n 행렬 : m개의 행과 n개의 열로 이루어진 행렬

$$A = \overset{\text{열}}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}}\ \text{행} \Rightarrow \begin{array}{c} 2 \times 3 \\ \text{행렬} \end{array}$$

Matrix Multiplication (행렬곱)

$$\begin{bmatrix} \mathrm{a} & b \\ \mathrm{c} & d \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ax + by + cz \\ dx + ey + fz \\ gx + hy + iz \end{bmatrix}$$

```
for i in range(len(X)):
    for j in range(len(Y[0])):
        for k in range(len(Y)):
                result[i][j] += X[i][k] * Y[k][j]
```

Element-Wise Multiplication ?

Reference

[1] 구종만,『 프로그래밍 대회에서 배우는 알고리즘 문제 해결 전략 』,
인사이트(2012)