

# 수학 2

최백준 [choi@startlink.io](mailto:choi@startlink.io)

---

$a^b$

---

$a^b$

# $a^b$

$a^b$

- $a$ 의  $b$ 제곱을 빠르게 구해야 한다.

```
int ans = 1;
for (int i=1; i<=b; i++) {
    ans = ans * a;
}
```

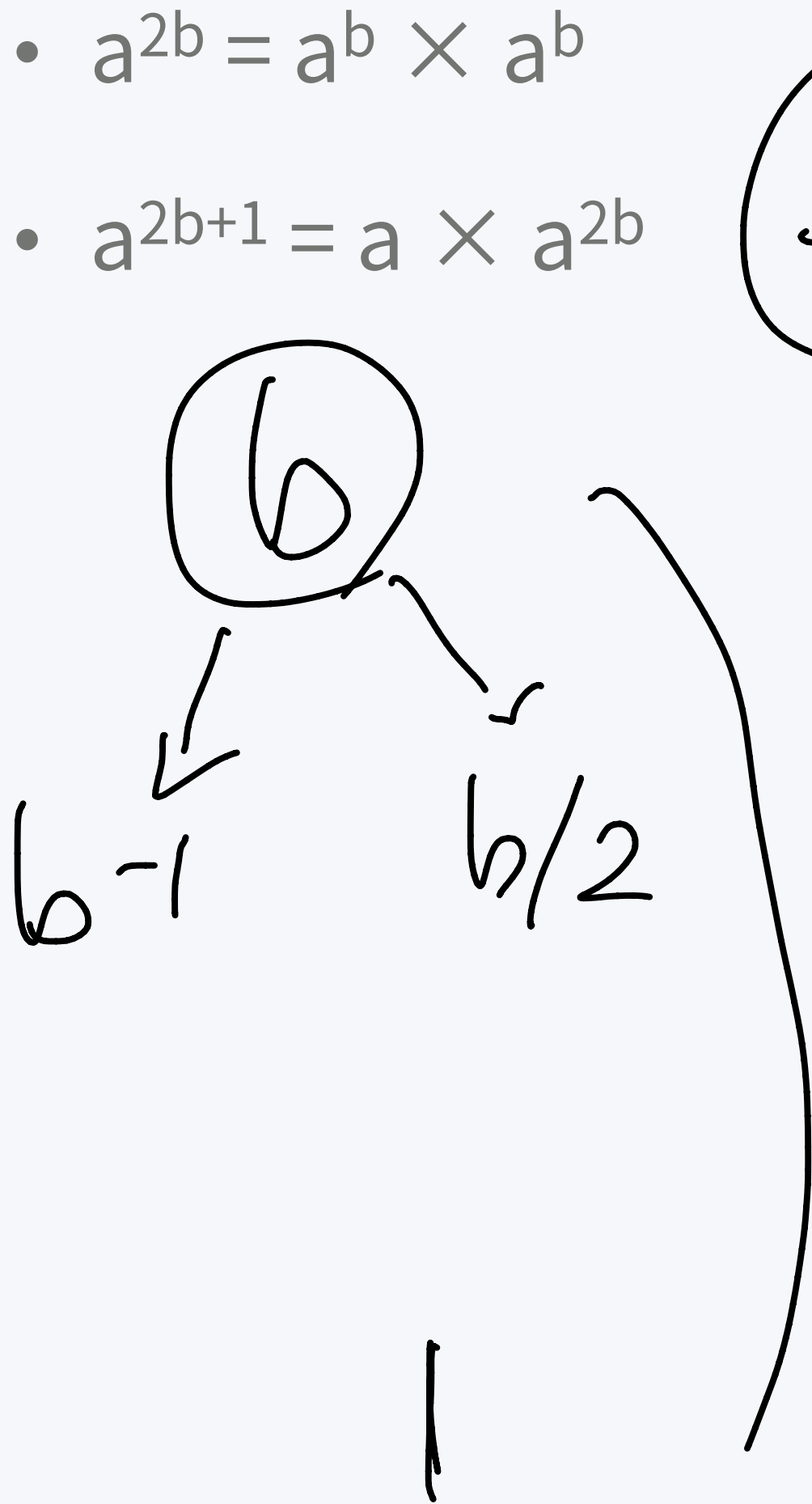
- 직관적인 방법이지만  $O(b)$ 라는 시간이 걸리게 된다.
- 따라서, 조금 더 빠른 방법이 필요하다.

$$a^b = \underbrace{a \times a \times \dots \times a}_{b\text{번}}$$

# $a^b$

분할정복 이용하기

- 분할정복을 이용해서 구할 수 있다.
- $a^{2b} = a^b \times a^b$
- $a^{2b+1} = a \times a^{2b}$



$b$

①  $\frac{b}{2}$   $\frac{b}{2}$  정복

재귀 :  $a^{2b} = a^b \cdot a^b$

$\frac{b-1}{2}$   $\frac{b+1}{2}$   $a^{2b+1} = a \cdot a^{2b}$

- ①  $a^{27} = a \cdot a^{26}$
- ②  $a^{26} = a^{13} \cdot a^{13}$
- ③  $a^{13} = a \cdot a^{12}$
- ④  $a^{12} = a^6 \cdot a^6$
- ⑤  $a^6 = a^3 \cdot a^3$
- ⑥  $a^3 = a \cdot a^2$
- ⑦  $a^2 = a \cdot a$

# $a^b$

분할정복 이용하기

$a^b$  계산하는 방법

```
int calc(int a, int b) {
  if (b == 0) {
    return 1;
```

$a^0 = 1$

```
} else if (b == 1) {
  return a;
```

$a^1 = a$

```
} else if (b % 2 == 0) {
  int temp = calc(a, b/2);
  return temp * temp;
```

$a^b = a^{b/2} \cdot a^{b/2}$  (재귀)

```
} else { // b % 2 == 1
  return a * calc(a, b-1);
}
```

$a^b = a \cdot a^{b-1}$  (재귀)

```
}
```

# $a^b$

분할정복 이용하기

- 이 부분을

```
} else if (b % 2 == 0) {  
    int temp = calc(a, b/2);  
    return temp * temp;  
}
```

- 아래와 같이 구현 하면  $O(N)$  이다. (호출이 2배가 되어버린다)

```
} else if (b % 2 == 0) {  
    return calc(a, b/2) * calc(a, b/2);  
}
```

# $a^b$

분할정복 이용하기

$(\lg b)$

$$3^{27} = 3^{1+2+8+16}$$

7

- 이진수의 원리를 이용해서도 구할 수 있다.

```
int calc(int a, int b) {  
    int ans = 1;  
    while (b > 0) {  
        if (b % 2 == 1) {  
            ans *= a;  
        }  
        a = a * a;  
        b /= 2;  
    }  
    return ans;  
}
```

$$27 = 11011_2 = 2^0 + 2^1 + 2^3 + 2^4 = 1 + 2 + 8 + 16$$

$$> 3^1 \times 3^2 \times 3^8 \times 3^{16}$$

$$\begin{aligned} a = 27 (11011) &\rightarrow 13 (1101) \rightarrow 6 (110) \rightarrow 3 (11) \\ b = 3 &\rightarrow 3^2 \rightarrow 3^4 \rightarrow 3^8 \end{aligned}$$

# $a^b$

## 분할정복 이용하기

- 예를 들어, 3의 27 제곱인 경우를 생각해보자.
- 27은 이진수로 11011 이다.
- $27 = 2^0 + 2^1 + 2^3 + 2^4$
- $27 = 1 + 2 + 8 + 16$
- $3^{27} = 3^{1+2+8+16}$
- $3^{27} = 3^1 \times 3^2 \times 3^8 \times 3^{16}$
- 을 이용해서 a를 계속해서  $a \times a$ 로 곱해가면서 제곱을 구하게 된다.



# 곱셈

<https://www.acmicpc.net/problem/1629>

- 자연수 A를 B번 곱한 수를 C로 나눈 나머지를 구하는 문제

$$(A^B) \% C$$

$$\left[ \frac{2/5}{+2} \right]$$
$$112121$$

$$(A \times B) \% C$$

$$= (A \% C \times B \% C) \% C$$

$$2(4)48364)$$

$$\left( \frac{2/5}{+2} \right)$$

$$(\lg(3))$$

# 곰셈

<https://www.acmicpc.net/problem/1629>

- C++ (분할 정복): <https://gist.github.com/Baekjoon/d0012f7c7b47cd5ad166>
- C++ (이진수 응용): <https://gist.github.com/Baekjoon/8b832ab8508fab1e1a42>

# 에라토스테네스의 체

---

# 제곱 LL 수

<https://www.acmicpc.net/problem/1016>

( 소수: 2 ~ N-1로 나누어 떨어지지 않는 수  
 제곱 LL 수:  $2^2, 3^2, \dots$  제곱수로 나누어 떨어지지 않는 수 ) 비슷.

- 소수를 구하는 방법인 에라토스 테네스의 체를 응용해서 문제를 풀 수 있다.
- max-min의 차이가 1,000,000이기 때문에, 배열을 이용할 수 있다.

# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- 1부터 100까지 제곱 L L 수를 구해보자

$2^2$  배수 = 4의 배수

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

# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- 2의 제곱의 배수를 모두 지운다.

$3^2$  배수 = 9 배수

1	2	3		5	6	7		9	10
11		13	14	15		17	18	19	
21	22	23		25	26	27		29	30
31		33	34	35		37	38	39	
41	42	43		45	46	47		49	50
51		53	54	55		57	58	59	
61	62	63		65	66	67		69	70
71		73	74	75		77	78	79	
81	82	83		85	86	87		89	90
91		93	94	95		97	98	99	

# 제곱 L L 수

15

<https://www.acmicpc.net/problem/1016>

- 3의 제곱의 배수를 모두 지운다.

1	2	3		5	6	7			10
11		13	14	15		17		19	
21	22	23		25	26			29	30
31		33	34	35		37	38	39	
41	42	43			46	47		49	50
51		53		55		57	58	59	
61	62			65	66	67		69	70
71		73	74	75		77	78	79	
	82	83		85	86	87		89	
91		93	94	95		97	98		

# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- 4의 제곱의 배수를 모두 지운다.



1	2	3		5	6	7			10
11		13	14	15		17		19	
21	22	23		25	26			29	30
31		33	34	35		37	38	39	
41	42	43			46	47		49	50
51		53		55		57	58	59	
61	62			65	66	67		69	70
71		73	74	75		77	78	79	
	82	83		85	86	87		89	
91		93	94	95		97	98		



# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- 5의 제곱의 배수를 모두 지운다.

1	2	3		5	6	7			10
11		13	14	15		17		19	
21	22	23			26			29	30
31		33	34	35		37	38	39	
41	42	43			46	47		49	
51		53		55		57	58	59	
61	62			65	66	67		69	70
71		73	74			77	78	79	
	82	83		85	86	87		89	
91		93	94	95		97	98		

# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- 6의 제곱의 배수를 모두 지운다.

1	2	3		5	6	7			10
11		13	14	15		17		19	
21	22	23			26			29	30
31		33	34	35		37	38	39	
41	42	43			46	47		49	
51		53		55		57	58	59	
61	62			65	66	67		69	70
71		73	74			77	78	79	
	82	83		85	86	87		89	
91		93	94	95		97	98		

# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- 7의 제곱의 배수를 모두 지운다.

정답이 버스  
응답

1	2	3		5	6	7			10
11		13	14	15		17		19	
21	22	23			26			29	30
31		33	34	35		37	38	39	
41	42	43			46	47			
51		53		55		57	58	59	
61	62			65	66	67		69	70
71		73	74			77	78	79	
	82	83		85	86	87		89	
91		93	94	95		97			

# 제곱 L L 수

<https://www.acmicpc.net/problem/1016>

- C/C++: <https://gist.github.com/Baekjoon/390c28220b7ed484f7ea11817b2b9be2>
- Java: <https://gist.github.com/Baekjoon/b3e73785626d7522c4c7cfcff213a67d>

# 행렬

---

# 행렬 덧셈

<https://www.acmicpc.net/problem/2738>

- 두 행렬을 입력받고 덧셈을 수행하는 문제

```
for (int i=0; i<n; i++) {  
    for (int j=0; j<m; j++) {  
        c[i][j] = a[i][j] + b[i][j];  
    }  
}
```

# 행렬 곱셈

<https://www.acmicpc.net/problem/2740>

- 두 행렬을 입력받고 곱셈을 수행하는 문제

```
for (int i=0; i<n; i++) {  
    for (int j=0; j<r; j++) {  
        c[i][j] = 0;  
        for (int k=0; k<m; k++) {  
            c[i][j] += a[i][k]*b[k][j];  
        }  
    }  
}
```

$$\frac{n \times r, \quad r \times m}{n \times r \times m}$$

$$N^3$$

# 행렬 제곱

<https://www.acmicpc.net/problem/10830>

- 행렬 A의 B제곱을 구하는 문제

크기  $N \times N$

공:  $N^3 \times B$

$((A \times A) \times A) \times A$

---

분할 정복, 이진 탐색 등등  $O(N^3 \lg B)$



# 행렬 제공

<https://www.acmicpc.net/problem/10830>

- C++: <https://gist.github.com/Baekjoon/53da6550ac5ca0c7608d>
- C++ (연산자 오버로딩): <https://gist.github.com/Baekjoon/a5d855c9f20ad45ef6c9>

# 피보나치 수

---

# 피보나치 수

Fibonacci Number

- $F_0 = 0$
- $F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$
- 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

$$F_n = F_{n-1} + F_{n-2}$$

재귀 (naive)  $O(2^N)$   
DP (memo)  $O(N)$

# 피보나치 수

<https://www.acmicpc.net/problem/2747>

- N번째 피보나치 수를 구하는 문제 ( $N \leq 45$ )

int

- <https://gist.github.com/Baekjoon/7e3535257c280c231f57>

# 피보나치 수 2

29

<https://www.acmicpc.net/problem/2748>

- N번째 피보나치 수를 구하는 문제 ( $N \leq 90$ )

- 90번째 피보나치 수는 ~~int~~ 범위를 넘어간다.

- <https://gist.github.com/Baekjoon/f179bfa10c7d10ac5c3d>

long long

# 피사노 주기

Pisano Period

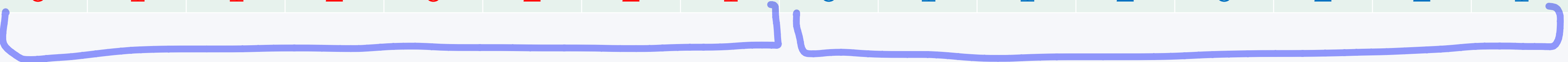
- 피보나치 수를  $K$ 로 나눈 나머지는 주기를 갖는다.
- 이것을 피사노 주기라고 한다.
- 3으로 나누었을 때의 주기는 8이다.

$F_n \% K$

$F_n \% 3$

$O(\text{주기의 길이})$   
8

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$F_n$	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610
$F_n \% 3$	0	1	1	2	0	2	2	1	0	1	1	2	0	2	2	1



반복

# 피보나치 수 3

<https://www.acmicpc.net/problem/2749>

31

$O(N)$  X

- N번째 피보나치 수를  $M = 1,000,000$ 으로 나눈 나머지를 구하는 문제
- $N \leq 1,000,000,000,000,000,000$
- 피사노 주기를 이용해서 주기를 찾고 문제를 풀 수 있다.
- 주기의 길이가 K이면
- N번째 피보나치 수를 M으로 나눈 나머지는  $N \% K$  번째 피보나치 수와 같다.
- $M = 10^k$  일 때,  $k > 2$  라면, 주기는 항상  $15 \times 10^{k-1}$  이다.
- 이 사실을 모른다고 해도, 주기를 구하는 코드를 이용해서 정답을 구할 수 있다.

$10^{18}$

1,500,000

$N \% K$

$$M = 10^k$$

$$\text{주기} : 15 \times 10^{k-1}$$

# 피보나치 수 3

32

<https://www.acmicpc.net/problem/2749>

- C++: <https://gist.github.com/Baekjoon/967bc3b5f8f7638db71c>



# 피보나치 수

Fibonacci Number

$$\begin{pmatrix} F_{n+2} = F_{n+1} + F_n \\ F_{n+1} = F_n + 0 \times F_{n-1} \end{pmatrix}$$

•  $\begin{pmatrix} F_{n+2} \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix}$

행렬의 제곱  $N^3 \lg B$   
상승

•  $\begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$

N번재  $(\lg N)$

•  $\sum_{i=1}^n F_i = F_{n+2} - 1$  피보나치 수의 합

•  $\sum_{i=1}^n F_{2i} = F_{2n+1} - 1$  짝수 번째 피보나치 수의 합

•  $\sum_{i=0}^n F_{2i+1} = F_{2n+2}$  홀수 번째

•  $\sum_{i=1}^n F_i^2 = F_n F_{n+1}$  피보나치 수 제곱

•  $\gcd(F_n, F_m) = F_{\gcd(n,m)}$  최대공약수

# 피보나치 수

Fibonacci Number

- $F_{2n-1} = \underbrace{F_n^2 + F_{n-1}^2}_{\text{정수}}$
- $\underline{F_{2n}} = \underbrace{(F_{n-1} + F_{n+1})F_n}_{\text{정수}} = \underbrace{(2F_{n-1} + F_n)F_n}_{\text{정수}}$

등식의 작은 용어)

$O(N^2)$

# 피보나치 수 6

<https://www.acmicpc.net/problem/11444>

• N번째 피보나치 수를  $M = 1,000,000,007$ 으로 나눈 나머지를 구하는 문제

•  $N \leq 1,000,000,000,000,000,000$

• 주기가 어떻게 될 지 알 수 없기 때문에,

• 20 페이지의 분할 정복 방법이나

• 19 페이지의 행렬 곱셈을 이용해서 풀어야 한다.

$\log t$

$(3) \Rightarrow 8$

# 피보나치 수 6

<https://www.acmicpc.net/problem/11444>

- C++ (행렬 제공): <https://gist.github.com/Baekjoon/eef05420ac8ac0491fc2>
- C++ (분할 정복): <https://gist.github.com/Baekjoon/b0828d05c8321f4c2e0e4a1c40fc60f8>

# 그 외의 피보나치 수 문제

## Fibonacci Number

- 피보나치 수 4: <https://www.acmicpc.net/problem/10826>
- 피보나치 수 5: <https://www.acmicpc.net/problem/10870>
- 피보나치 수의 확장: <https://www.acmicpc.net/problem/1788>
- 피사노 주기: <https://www.acmicpc.net/problem/9471>
- 피보나치 수의 합: <https://www.acmicpc.net/problem/2086>
- 피보나치 수의 제곱의 합: <https://www.acmicpc.net/problem/11440>
- 홀수번째 피보나치 수의 합: <https://www.acmicpc.net/problem/11442>
- 짝수번째 피보나치 수의 합: <https://www.acmicpc.net/problem/11443>
- 피보나치 수와 최대공약수: <https://www.acmicpc.net/problem/11778>

# 이항 계수

---

# 이항 계수

Binomial Coefficient

- n개중에 k개를 순서 없이 고르는 방법  $nCr$
- $\binom{n}{k}$ 로 쓴다.
- $\frac{n!}{k!(n-k)!}$  이다.
- $\frac{n \times (n-1) \times \dots (n-k+1)}{k!}$  와 같다.
- 구해보자!

n개 중에서 k개를  
순서 없이 고르는 방법

$\binom{n}{k}$      $nCk$

$$\begin{aligned} &+ O(n) \\ &+ O(k) \\ &+ O(n-k) \end{aligned} \quad \frac{n!}{k!(n-k)!} = O(n)$$

$4C_2 = 6$   
 $\frac{2 \cancel{4} \cdot 3 \cdot \cancel{2} \cdot 1}{\cancel{2} \cdot \cancel{1} \cdot \cancel{2} \cdot \cancel{1}}$

# 이항 계수 1

<https://www.acmicpc.net/problem/11050>

- $\binom{n}{k}$  를 구하는 문제
- $1 \leq n \leq 10, 0 \leq k \leq n$  이기 때문에
- $\frac{n!}{k!(n-k)!}$  값을 그냥 구하면 된다.

$10! = 3628800$

3백만

$11! = 332$

$20!$

(int)

$21^3$

$20!$

$\binom{n}{k} \leftarrow \text{long long}$

$20C5 = \frac{20 \times 19 \times 18 \times 17 \times 16}{5 \times 4 \times 3 \times 2 \times 1}$

$20^3 \times 3 = 8000 \times 3 = 24000$   
 $19 \times 17 \times 16 \times 3$



# 이항 계수 1

<https://www.acmicpc.net/problem/11050>

- C++: <https://gist.github.com/Baekjoon/28960ad31bc2c134a6e4>

# 파스칼의 삼각형

Pascal's Triangle

42

- 이항 계수를 삼각형 모양으로 배열
- $n$ 번 줄에는 수를  $n$ 개만 쓴다.
- 각 줄의 첫 번째와 마지막 수는 1이다.
- 나머지 수는 위 줄의 왼쪽 수와 오른쪽 수를 더해서 만든다.

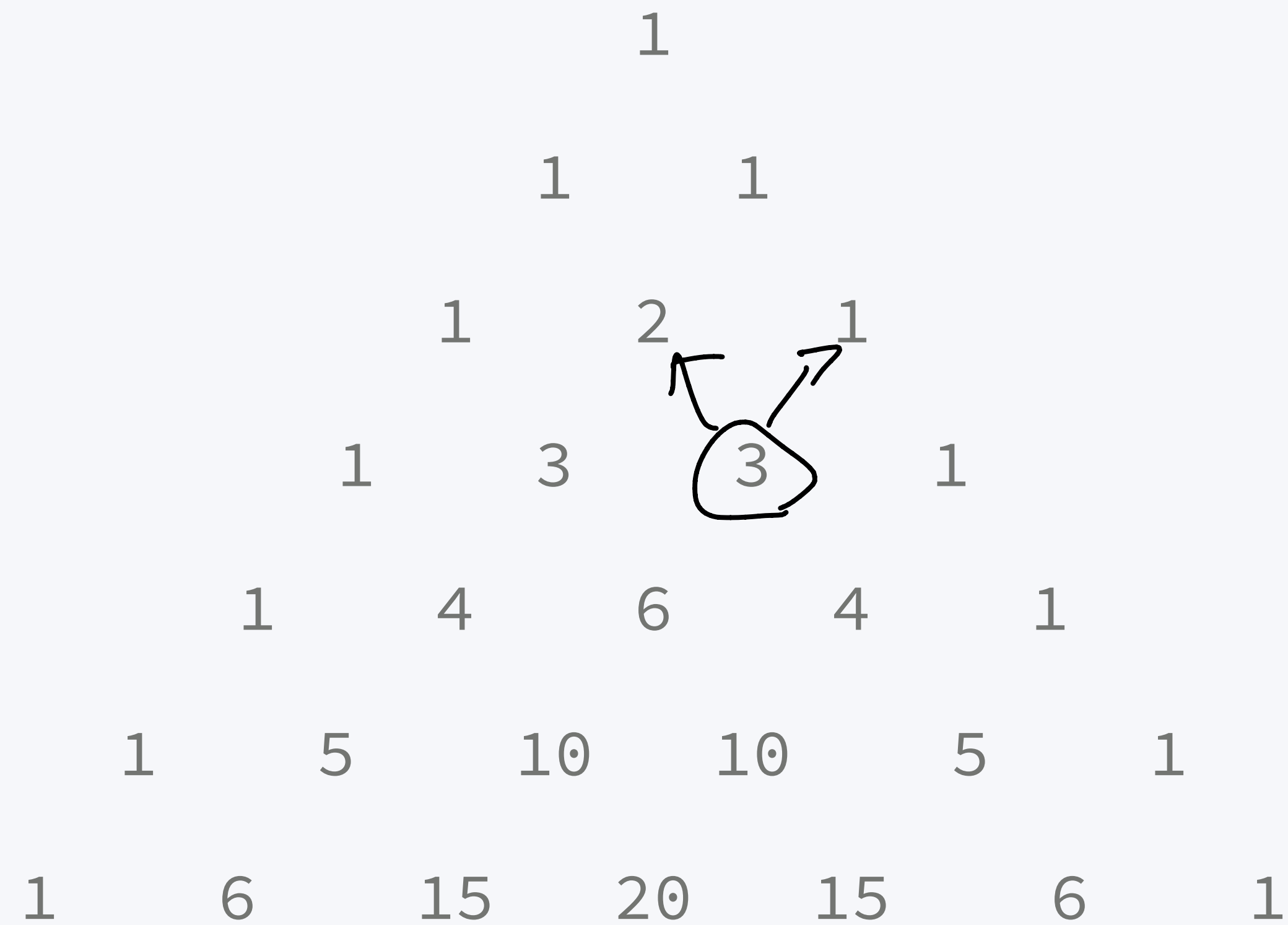
# 파스칼의 삼각형

43

Pascal's Triangle

- 5까지 파스칼의 삼각형

$$n\text{행 } k\text{열} = nC_k = \binom{n}{k}$$



# 파스칼의 삼각형

Pascal's Triangle

- $C[n][k]$  = n번줄의 k번째 수라고 했을 때
- $C[n][1] = 1, C[n][n] = 1$
- $C[n][k] = C[n-1][k-1] + C[n-1][k]$
- 로 정의할 수 있음
- $C[n][k]$ 는  $\binom{n}{k}$  이다.

$$C[n][k] = C[n-1][k-1] + C[n-1][k]$$

# 파스칼의 삼각형

Pascal's Triangle

- $C[n][k] = C[n-1][k-1] + C[n-1][k]$

- $C[n][k]$  는  $\binom{n}{k}$  를 나타내기 때문에,  $n$ 개 중에  $k$ 개를 순서 없이 고르는 방법이다.  $n-1$ 개

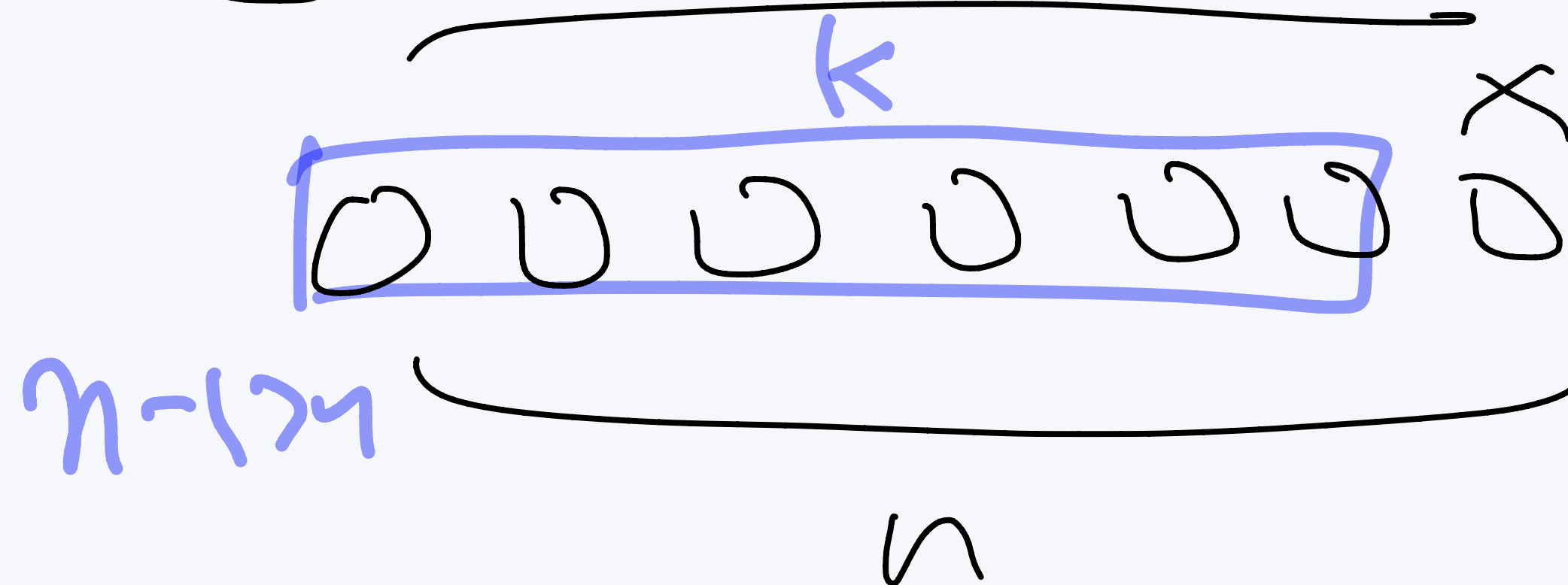
- $n$ 개 중에  $k$ 개를 순서 없이 고른다면 다음과 같은 두 가지 경우가 가능하다

1.  $n$ 번째를 고른 경우

①  $n$ 번째를 고른 경우,  $C[n-1][k-1]$

2.  $n$ 번째를 고르지 않은 경우

②  $n$ 번째를 고르지 않은 경우,  $C[n-1][k]$

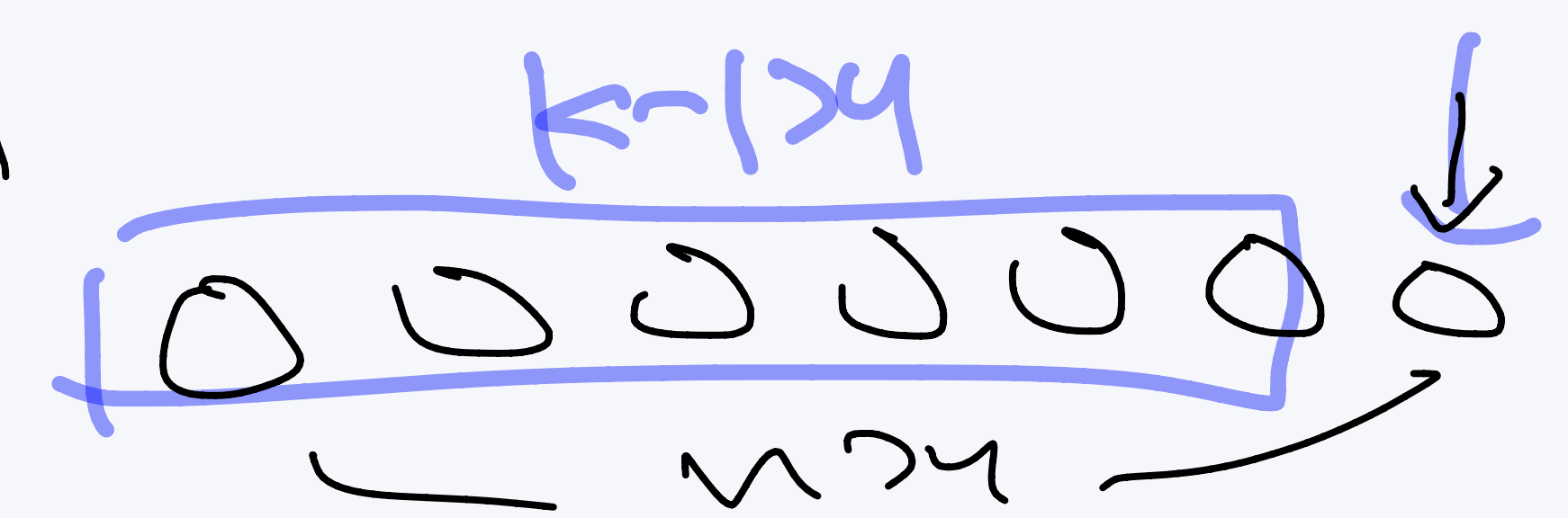


$C[n][k]$

$n$ 개  $k$ 개

고른:  $k$ 개

$k-1$ 개



# 파스칼의 삼각형

Pascal's Triangle

- $C[n][k] = C[n-1][k-1] + C[n-1][k]$
- $C[n][k]$  는  $\binom{n}{k}$  를 나타내기 때문에,  $n$ 개 중에  $k$ 개를 순서 없이 고르는 방법이다.
- $n$ 개 중에  $k$ 개를 순서 없이 고른다면 다음과 같은 두 가지 경우가 가능하다
  1.  $n$ 번째를 고른 경우
    - $n$ 번째를 골랐기 때문에,  $n-1$ 개 중에  $k-1$ 개를 골랐어야 한다.  $C[n-1][k-1]$
  2.  $n$ 번째를 고르지 않은 경우
    - $n$ 번째를 고르지 않았기 때문에,  $n-1$ 개 중에  $k$ 개를 골랐어야 한다.  $C[n-1][k]$

# 이항 계수 2

<https://www.acmicpc.net/problem/11051>

- $\binom{n}{k}$  을 10,007로 나눈 나머지를 구하는 문제
- $1 \leq n \leq 1,000, 0 \leq k \leq n$  이기 때문에
- 파스칼의 삼각형을 이용해서 구하면 된다.

$$n^2$$

# 이항 계수 2

48

<https://www.acmicpc.net/problem/11051>

- C++: <https://gist.github.com/Baekjoon/318692441d185275e5a3>



# 이항 계수

Binomial Coefficient

- $\underline{(x + y)^n} = \sum_{k=0}^n \binom{n}{k} x^{n-k} y^k$

- $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad (1 \leq k \leq n-1)$

파스칼의 삼각형

- $\underline{\binom{n}{k} = \binom{n}{n-k}}$

$$nC_k = nC_{n-k}$$

# 이항 계수

Binomial Coefficient

50

- $n$ 개 중에  $k$ 개를 중복 없이 뽑는 방법의 수  $\binom{n}{k}$
- $n$ 개 중에  $k$ 개를 중복을 허용하면서 뽑는 방법의 수  $\binom{n+k-1}{k}$
- 0과 1로만 이루어진 문자열의 개수  $\binom{n+k}{k}$
- 0과 1로만 이루어진 문자열의 개수 (1은 연속하지 않음)  $\binom{n+1}{k}$
- 카탈란 수  $\frac{1}{n+1} \binom{2n}{n}$

# 뤼카의 정리

Lucas' Theorem

- 음이 아닌 정수  $n, m$ 과 소수  $p$ 에 대해서 다음이 성립한다.
- $\binom{n}{m} = \prod_{i=0}^k \binom{n_i}{m_i} \pmod{p}$
- 여기서  $n_i$ 와  $m_i$ 는  $n$ 과  $m$ 을  $p$ 진법으로 나타낸 것이다. 즉, 다음과 같다.
- $n = n_k p^k + n_{k-1} p^{k-1} + \dots + n_1 p + n_0$
- $m = m_k p^k + m_{k-1} p^{k-1} + \dots + m_1 p + m_0$

# 이항 계수 4

<https://www.acmicpc.net/problem/11402>

- $\binom{n}{k} \bmod M$ 을 구하는 문제 과거
- $1 \leq n \leq 10^{18}$ ,  $0 \leq k \leq n$ ,  $2 \leq m \leq 2000$ ,  $m$ 은 소수
- $m$ 이 2000보다 작은 소수이기 때문에, 파스칼의 삼각형을 만들고
- 뢰카의 정리를 이용하면 된다.  $n \geq k$   $m$ 전방

# 이항 계수 4

<https://www.acmicpc.net/problem/11402>

- $\binom{n}{k} \bmod M$ 을 구하는 문제
- $1 \leq n \leq 10^{18}, 0 \leq k \leq n, 2 \leq m \leq 2000, m$ 은 소수
- $m$ 이 2000보다 작은 소수이기 때문에, 파스칼의 삼각형을 만들고
- 뫼비우스의 정리를 이용하면 된다.

# 이항 계수 4


<https://www.acmicpc.net/problem/11402>

- C++: <https://gist.github.com/Baekjoon/69fb9c3f786e0e1855da>

# 이항 계수 5

55

<https://www.acmicpc.net/problem/11439>

$N!$  

- $\binom{n}{k} \bmod M$ 을 구하는 문제
- $1 \leq n \leq 4 \times 10^6, 0 \leq k \leq n, 2 \leq m \leq 4 \times 10^6, m$ 은 소수
- $\binom{n}{k}$ 를 소인수 분해 하면서 풀어야 한다.
- 팩토리얼 0의 개수 문제를 풀 때,  $N!$ 를 소인수 분해를 하면  $5^k$ 의  $k$ 가 몇 개 인지 구하는 방법을 배웠다.
- 이 방법을 응용해서 푼다.

$O(m)$

# 이항 계수 5

<https://www.acmicpc.net/problem/11439>

- C++: <https://gist.github.com/Baekjoon/237269bdf7fa2ea98520>

$$\binom{20}{5} = \frac{20!}{5! \cdot 15!} = \frac{19 \times 17 \times 16 \times 3}{2^4 \cdot 3^1}$$

$$20! \quad 2 \times 10! : 20/2 + 20/4 + 20/8 + 20/16 = 10 + 5 + 2 + 1 = 18$$

$$5! \quad 2 \times 10! : 5/2 + 5/4 = 2 + 1 = 3$$

$$15! \quad 2 \times 10! : 15/2 + 15/4 + 15/8 = 7 + 3 + 1 = 11$$

$$\frac{2^{18}}{2^3 \cdot 2^1} = 2^4$$

$$20! \quad 3 \times 10! : 20/3 + 20/9 = 6 + 2 = 8$$

$$5! \quad 3 \times 10! : 5/3 = 1 = 1$$

$$15! \quad 3 \times 10! : 15/3 + 15/9 = 5 + 1 = 6$$

$$8 - 1 - 6 = 1 \quad 3^1$$



# 카탈란 수

---

# 카탈란 수

Catalan Number

이항 계수 ~

•  $C_n = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{(n+1)!n!} = \prod_{k=2}^n \frac{n+k}{k} \quad (n \geq 0)$

•  $C_n = \binom{2n}{n} - \binom{2n}{n+1}$  ← Pascal

•  $C_0 = 1$

•  $C_{n+1} = \sum_{i=0}^n C_i C_{n-i}$

• 1, 1, 2, 5, 14, 42, 132, ...

$O(n)$

$O(n^2)$

DP

# 카탈란 수

## Catalan Number

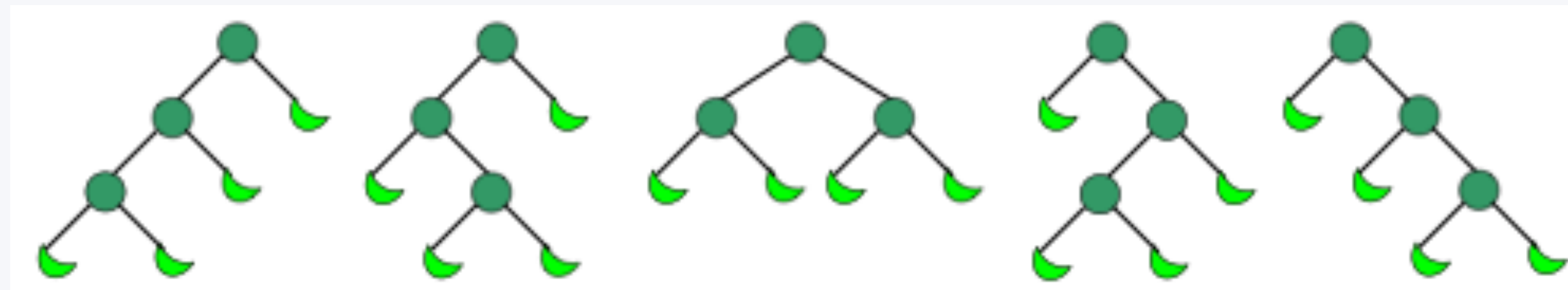
- 여는 괄호  $n$ 개와 닫는 괄호  $n$ 개로 이루어진 올바른 괄호 문자열의 개수  $C_n$

((())) ()(()) ()()() ((())()) ((())())

- 항  $n+1$ 개를 곱하는 순서의 수 괄호  $n$

((ab)c)d (a(bc))d (ab)(cd) a((bc)d) a(b(cd))

- 단말 정점이  $n+1$ 개인 풀 바이너리 트리의 개수 (풀 바이너리 트리: children이 2개 or 0개)



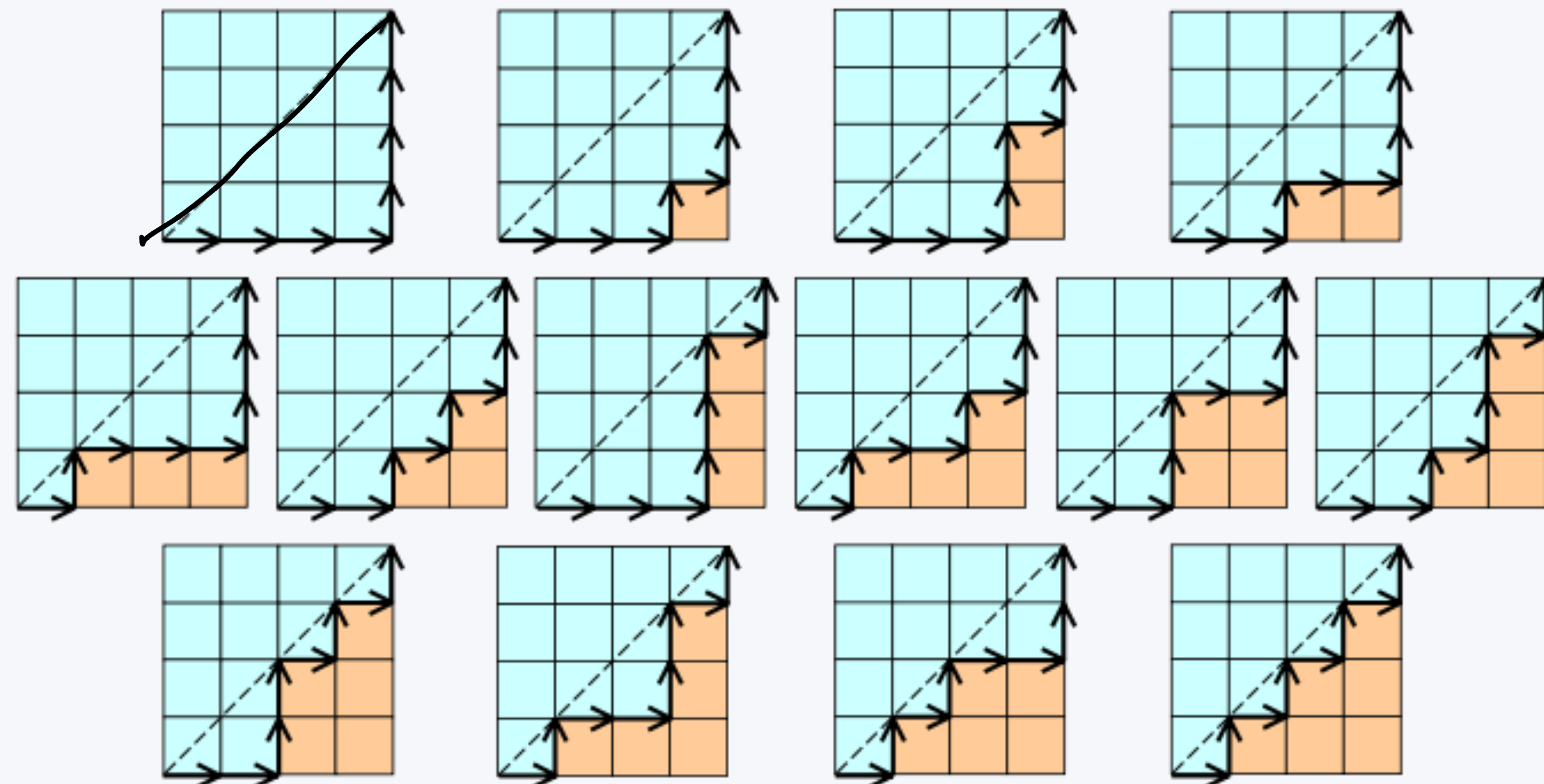
# 카탈란 수

Catalan Number

- $n \times n$  개의 정사각형으로 이루어져 있는 격자에서 대각선을 지나가지 않고 가장 왼쪽 아래에서 오른쪽 위로 지나가는 방법의 수

$$C_n$$

- 오른쪽으로 가는 것을 (로, 위로 가는 것을 )로 생각하면 올바른 괄호 문자열의 수와 같다



# 괄호

<https://www.acmicpc.net/problem/10422>

- 올바른 괄호 문자열의 개수를 구하는 문제

# 괄호

<https://www.acmicpc.net/problem/10422>

- C++: <https://gist.github.com/Baekjoon/58302ab9793b7e861fa8>

# 오일러 피 함수

---

# 오일러 피 함수

Euler's phi Function

- $\phi(n)$ 로 나타낸다.
- $\phi(n) = \gcd(n, k) = 1$  인  $1 \leq k \leq n$ 의 개수
- $\phi(9) = 6$
- $k = 1, 2, 4, 5, 7, 8$

n보다

작은 수 가능

자신과  
공배수

n

$1 \leq k \leq n$

n과  $\phi(n)$



# 오일러 피 함수

Euler's phi Function

- $\phi(p) = p-1$  ( $p$ 가 소수인 경우)
- $\phi(nm) = \phi(n) \times \phi(m)$  ( $n, m$ 이 서로소인 경우)
- $\phi(n) = n \prod_{p|n} (1 - \frac{1}{p})$
- $p$ 는  $n$ 의 소인수
- $\phi(9) = 9 * (1 - 1/3) = 9 * 2/3 = 6$

# 오일러 피 함수

Euler's phi Function

```
long long phi(long long n) {  
    long long ans = n;  
    for (long long i=2; i*i<=n; i++) {  
        if (n % i == 0) {  
            while (n % i == 0)  
                n /= i;  
            ans -= ans / i;  
        }  
    }  
    if (n > 1)  
        ans -= ans / n;  
    return ans;  
}
```

소인수 분해 응용

# $\text{GCD}(n, k) = 1$

<https://www.acmicpc.net/problem/11689>

- 오일러 피 함수를 구현해보는 문제

# $\text{GCD}(n, k) = 1$

<https://www.acmicpc.net/problem/11689>

- C++: <https://gist.github.com/Baekjoon/47e8d9b2a2e60253e22f>

나-2학기

# 나머지 연산 2

# 확장 유클리드 알고리즘

70

## Extended Euclidean Algorithm

- $ax + by = \gcd(a, b)$  의 해를 구할 수 있는 알고리즘
- 유클리드 알고리즘과 다르게 4개의 변수를 이용해서 사용한다.
- $\gcd(a, b)$ 를 조금 어렵게 써보면 다음과 같다.
- 몫:  $q_0, \dots, q_k$ , 나머지:  $r_0 \dots, r_k$
- $r_0 = a$
- $r_1 = b$
- ...
- $r_{i+1} = r_{i-1} - q_i r_i$  ( $0 \leq r_{i+1} < |r_i|$ , 이 부등식으로  $q_i$ 를 결정할 수 있다)

# 확장 유클리드 알고리즘

## Extended Euclidean Algorithm

- 유클리드 알고리즘에 두 변수  $s$ 와  $t$ 를 추가해야 한다.
- $r_0 = a, r_1 = b$
- $s_0 = 1, s_1 = 0$
- $t_0 = 0, t_1 = 1$
- ...
- $r_{i+1} = r_{i-1} - q_i r_i$  ( $0 \leq r_{i+1} < |r_i|$ )
- $s_{i+1} = s_{i-1} - q_i s_i$
- $t_{i+1} = t_{i-1} - q_i t_i$

# 확장 유클리드 알고리즘

72

## Extended Euclidean Algorithm

- $a = 240, b = 46$ 인 경우를 풀어보자.
- $240x + 46y = \gcd(240, 46) = 2$

$i$	$q_{i-1}$	$r_i$	$s_i$	$t_i$
0		240	1	0
1		46	0	1
2	$240/46 = 5$	$240 - 5 \times 46 = 10$	$1 - 5 \times 0 = 1$	$0 - 5 \times 1 = -5$
3	$46/10 = 4$	$46 - 4 \times 10 = 6$	$0 - 4 \times 1 = -4$	$1 - 4 \times -5 = 21$
4	$10/6 = 1$	$10 - 1 \times 6 = 4$	$1 - 1 \times -4 = 5$	$-5 - 1 \times 21 = -26$
5	$6/4 = 1$	$6 - 1 \times 4 = 2$	$-4 - -1 \times 5 = -9$	$21 - 1 \times -26 = 47$
6	$4 \times 2 = 2$	$4 - 2 \times 2 = 0$	$5 - 2 \times -9 = 23$	$-26 - 2 \times 47 = -120$



# 확장 유클리드 알고리즘

Extended Euclidean Algorithm

- $ax + by = \gcd(a, b)$  의 해를 쉽게 구할 수 있는 알고리즘
- 대부분의 경우에  $a$ 와  $b$ 중 하나는 음수가 나온다.
- $240x + 46y = \gcd(240, 46) = 2$
- 확장 유클리드 알고리즘의 마지막 이전  $s$ 와  $t$ 값이  $x$ 와  $y$ 값이 된다.
- $240 \times -9 + 46 \times 47 = 2$

# 해의 개수

<https://www.acmicpc.net/problem/11661>

- $Ax + By + C = 0$ 의 해의 개수를 구하는 문제

# 확장 유클리드 알고리즘

75

Extended Euclidean Algorithm

- $ax + by = \gcd(a,b)$  의 해를 쉽게 구할 수 있는 알고리즘
- 이 알고리즘은  $\gcd(a,b)$ 가 1인 경우에 유용하게 사용할 수 있다.
- $ax + by = 1$  일 때,  $x$ 는  $a$ 의 나머지 연산의 곱셈 역원이 되기 때문

# 나머지 연산의 곱셈 역원

## Modular Multiply Inverse

- 정수  $a$ 을  $m$ 으로 나눈 나머지의 곱셈 역원은  $a \times a^{-1} \equiv 1 \pmod{m}$  을 만족하는  $a^{-1}$  을 말한다.
- 즉,  $a^{-1} \equiv x \pmod{m}$  을 만족하는  $x$ 를 말한다.
- 역원은  $a$ 와  $m$ 이 서로소인 경우에만 존재한다.

```
for (int i=1; i<m; i++) {  
    if ((a*i) % m == 1) {  
        x = i;  
    }  
}
```

- 위 소스의 시간 복잡도는  $O(m)$  이다.

# 나머지 연산의 곱셈 역원

Modular Multiplicative Inverse

- 확장 유클리드 알고리즘을 이용해서 구할 수도 있다.
- $ax = 1 \pmod{m}$  을 구해야 하기 때문에
- $ax = 1 + my$ 로 바꿔서 쓸 수 있다.
- $ax - my = 1$  로 다시 쓸 수 있고,  $x$ 와  $y$ 는 음수가 되어도 상관 없기 때문에
- $ax + my = 1$  로 다시 바꿔 쓸 수 있다.
- 이제 확장 유클리드 알고리즘을 이용해서  $x$ 의 값을 구할 수 있다.

# 나머지 연산의 곱셈 역원

## Modular Multiplicate Inverse

- $m$ 이 소수인 경우에는 페르마의 소정리를 이용해서 구할 수도 있다.
- $m$ 이 소수이고,  $a$ 가  $m$ 과 서로소라면,  $a^{m-1}$ 은  $m$ 으로 나눈 나머지는 1이다.
- 즉
- $a^{m-1} \equiv 1 \pmod{m}$  이라는 의미이다.
- 따라서,  $a \times a^{m-2} \equiv 1 \pmod{m}$  이고,
- $a^{m-2}$  가  $a \times x \equiv 1 \pmod{m}$  을 만족하는  $x$ 가 되기 때문에
- 역원은  $a^{m-2}$ 가 된다.

$$O(\log C)$$

$$(A/B)^{C-2} \% C$$

|| 소수

$$(A \cdot B^{C-2}) \% C$$

$a^b$

# 이항 계수 3

<https://www.acmicpc.net/problem/11401>

79

- $\binom{n}{k} \bmod M$ 을 구하는 문제

- $1 \leq n \leq 4,000,000, 0 \leq k \leq n, M = 1,000,000,007$

- 나머지 연산의 곱셈 역원을 이용해서 풀 수 있다.

$$\left( \underbrace{N!}_A / \underbrace{k! / (N-k)!}_B \right) \% M$$

$$(A/B) \% M$$

$$(A/B) \% M$$

$$(A \cdot B^{M-2}) \% M$$

# 이항 계수 3

80

<https://www.acmicpc.net/problem/11401>

- C++: <https://gist.github.com/Baekjoon/24a439e38aceb9946b2f>