

# #03-1. 사칙연산

나정휘

<https://justicehui.github.io/>

# 목차

나눗셈과 합동식

C/C++의 나눗셈 연산

거듭 제곱의 빠른 계산

# 나눗셈과 합동식

## 나눗셈 정리 (Division Algorithm)

- 정수  $a$ 와 0이 아닌 정수  $b$ 가 있을 때,  $a = bq + r, 0 \leq r < |b|$ 를 만족하는 정수  $q, r$ 은 유일함
  - 이때  $q$ 를 몫,  $r$ 을 나머지라고 부름
  - 8을 3으로 나누었을 때의 몫은 2, 나머지는 2 ( $8 = 3 \times 2 + 2$ )
  - 8을 -3으로 나누었을 때의 몫은 -2, 나머지는 2 ( $8 = (-3) \times (-2) + 2$ )
  - -8을 3으로 나누었을 때의 몫은 -3, 나머지는 1 ( $-8 = 3 \times (-3) + 1$ )
  - -8을 -3으로 나누었을 때의 몫은 3, 나머지는 1 ( $-8 = (-3) \times 3 + 1$ )

# 나눗셈과 합동식

## 약수와 배수

- 정수  $a, b$ 에 대해  $b = an$ 을 만족하는 정수  $n$ 이 존재하면  $a$ 는  $b$ 의 약수,  $b$ 는  $a$ 의 배수
  - $a|b$  :  $a$ 가  $b$ 를 나눈다,  $b$ 는  $a$ 로 나누어진다.
  - 정의에 의해 모든 정수는 0을 나눌 수 있음
- $a \neq b \neq 0$ 인 정수  $a, b$ 에 대해,  $g|a, g|b$ 를 만족하는 가장 큰 자연수  $g$ : 최대공약수
- $a|l, b|l$ 를 만족하는 가장 작은 자연수  $l$ : 최소공배수
- $a$ 와  $b$ 의 최대공약수가 1이면  $a$ 와  $b$ 는 서로소
- $ab = gl$ 
  - $a = ga', b = gb'$ 라고 하면  $a'$ 과  $b'$ 은 서로소
  - $l = ga'b'$ 이므로  $ab = g^2a'b' = gl$ 임
  - $l = ab/g$

# 나눗셈과 합동식

## 합동

- 정수  $a, b$ 와 0이 아닌 정수  $n$ 이 있을 때,  $n|(a - b)$ 이면  $a$ 와  $b$ 가  $(\text{mod } n)$ 에서 합동
  - $a \equiv b (\text{mod } n)$
  - $a$ 와  $b$ 를  $n$ 으로 나눈 나머지가 동일하다는 뜻

## 합동의 성질

- 반사성, 대칭성, 추이성
  - $a \equiv a (\text{mod } n)$
  - $a \equiv b (\text{mod } n)$ 이면  $b \equiv a (\text{mod } n)$
  - $a \equiv b (\text{mod } n)$ 이고  $b \equiv c (\text{mod } n)$ 이면  $a \equiv c (\text{mod } n)$
- 사칙연산
  - $a \equiv b (\text{mod } n)$ 이면
  - 두 정수  $c \equiv d (\text{mod } n)$ 에 대해,  $a \pm c \equiv b \pm c (\text{mod } n)$
  - 0이 아닌 두 정수  $c \equiv d (\text{mod } n)$ 에 대해,  $ac \equiv bc (\text{mod } n)$
  - 나눗셈은 성립 안 함
    - $9 \equiv 12 (\text{mod } 3), 9/3 \not\equiv 12/3 (\text{mod } 3)$

# C/C++의 나눗셈 연산

## C/C++의 나눗셈 연산

- a, b가 정수일 때  $(a / b) * b + (a \% b) = a$ 를 만족하는 몫과 나머지를 반환함
- b가 a의 약수가 아니면 몫 0에 가까워지도록 자름 (truncation towards zero)

- $8 / 3 = 2$	$8 \% 3 = 2$	$2 * 3 + 2 = 8$
- $8 / (-3) = -2$	$8 \% (-3) = 2$	$(-2) * (-3) + 2 = 8$
- $(-8) / 3 = -2$	$(-8) \% 3 = -2$	$(-2) * 3 + (-2) = -8$
- $(-8) / (-3) = 2$	$(-8) \% (-3) = -2$	$2 * (-3) + (-2) = -8$

- 나머지를  $0 \leq r < |b|$  범위에 넣고 싶을 때는
  - $r = r + (r \geq 0 ? 0 : |b|)$
  - 또는  $r = (r + |b|) \% |b|$

# C/C++의 나눗셈 연산

## BOJ 17466. N! mod P (1)

- N과 P가 주어지면 N!을 P로 나눈 나머지를 구하는 문제
- $N, P \leq 10^8$  이므로 계산 과정 도중에 최대  $10^{16}$ 까지의 수를 볼 수 있음
- long long을 사용하면서, 곱셈 연산을 할 때마다 나머지를 취하면 됨



```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    ll N, P, R = 1;
    cin >> N >> P;
    for(int i=1; i<=N; i++) R = R * i % P;
    cout << R;
}
```

# C/C++의 나눗셈 연산

## C/C++의 나눗셈 연산

- floor와 ceil 직접 구현하기
  - math.h (또는 cmath)에 있는 floor/ceil은 부동 소수점 자료형을 사용하기 때문에 정확하지 않음
  - 정수 간의 나눗셈의 결과를 버림/올림하는 것은 정수만 사용해서 정확하게 수행할 수 있음
- $p / q$ 에서 항상  $q > 0$ 이라고 가정하자.
  - $q < 0$ 이면  $p$ 와  $q$ 에 각각  $-1$  곱하면 됨
- $p \geq 0$  인 경우
  - 버림:  $p / q$
  - 올림:  $(p + q - 1) / q$
- $p < 0$  인 경우
  - 버림:  $(p - q + 1) / q$
  - 올림:  $p / q$



```
int floor(int p, int q){
    if(q < 0) p = -p, q = -q;
    return p >= 0 ? p / q : (p - q + 1) / q;
}

int ceil(int p, int q){
    if(q < 0) p = -p, q = -q;
    return p >= 0 ? (p + q - 1) / q : p / q;
}
```



질문?

# 거듭제곱

목표: 음이 아닌 정수  $a, b, c$ 에 대해  $a^b \pmod c$ 를 구하는 것

- $b = 0$ 이면  $a^b = 1$
- $b \geq 1$ 이면 두 가지 경우로 나뉘서 생각할 수 있음
  - $2 \mid b$ 이면  $a^b = a^{b/2} \times a^{b/2}$
  - $2 \nmid b$ 이면  $a^b = a \times a^{(b-1)/2} \times a^{(b-1)/2}$
- 곱셈 연산을 할 때마다  $c$ 로 나눈 나머지를 취하면
- $a < c$ 일 때 계산 과정 도중에 나오는 값은  $c^2$ 미만

# 거듭제곱

## BOJ 1629. 곱셈

- $2147483647 (= 2^{31} - 1)$  이하의 자연수  $a, b, c$ 가 주어졌을 때
- $a^b \bmod c$ 를 구하는 문제
- 계산 도중에 int 범위를 넘어갈 수 있으므로 long long 사용
- Pow 함수의 호출 횟수:  $\lfloor \log_2 b \rfloor + 2$ 번

```
● ● ●

#include <bits/stdc++.h>
using namespace std;
using ll = long long;

ll Pow(ll a, ll b, ll c){
    if(b == 0) return 1;
    ll half = Pow(a, b / 2, c);
    if(b % 2 == 0) return half * half % c;
    else return a * half % c * half % c;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    ll a, b, c;
    cin >> a >> b >> c;
    cout << Pow(a % c, b, c);
}
```

# 거듭제곱

## BOJ 1629. 곱셈

- $b = 5$  일 때  $b$ 를 이진법으로 나타내 보면  $101_{(2)}(2^2 + 2^0)$
- $a^5$ 는  $a^4 \times a^1$ 로 나타낼 수 있음
- $a^1, a^2, a^4, a^8, \dots$ 를 알고 있다면 최대  $\log_2 b$ 번의 곱셈으로  $a^b$ 를 구할 수 있음



```
ll Pow(ll a, ll b, ll c){
    ll res = 1;
    while(b > 0){
        if(b % 2 == 1) res = res * a % c;
        b /= 2;
        a = a * a % c;
    }
    return res;
}
```

질문?

# 거듭제곱

## BOJ 11819. The Shortest does not Mean the Simplest

- 똑같은 문제인데  $1 \leq a, b, c \leq 10^{18}$ 
  - 곱셈을 하는 순간 long long 범위를 넘어감
- 거듭제곱과 동일한 방법으로  $a \times b$ 를  $\log_2 b$ 번의 덧셈으로 구하면 됨
  - $a, b < c$ 일 때  $ab < c^2$ 이지만  $a + b < 2c$ 라서 long long 범위를 넘어가지 않음

```
ll Add(ll a, ll b, ll c){ return (a + b) % c; }

ll Mul(ll a, ll b, ll c){
    ll res = 0;
    while(b > 0){
        if(b % 2 == 1) res = Add(res, a, c);
        b /= 2; a = Add(a, a, c);
    }
    return res;
}

ll Pow(ll a, ll b, ll c){
    ll res = 1;
    while(b > 0){
        if(b % 2 == 1) res = Mul(res, a, c);
        b /= 2; a = Mul(a, a, c);
    }
    return res;
}
```

# 거듭제곱

## BOJ 10830. 행렬 제곱

- 행렬의 거듭제곱도 계산할 수 있음
- 행렬 곱셈의 항등원은 단위 행렬

```
vector<vector<int>> Mul(vector<vector<int>> a, vector<vector<int>> b){
    int n = a.size();
    vector<vector<int>> c(n, vector<int>(n));
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            for(int k=0; k<n; k++){
                c[i][j] = (c[i][j] + a[i][k] * b[k][j]) % 1000;
            }
        }
    }
    return c;
}

vector<vector<int>> Pow(vector<vector<int>> a, long long b){
    int n = a.size();
    vector<vector<int>> res(n, vector<int>(n));
    for(int i=0; i<n; i++) res[i][i] = 1;
    while(b > 0){
        if(b % 2 == 1) res = Mul(res, a);
        b /= 2; a = Mul(a, a);
    }
    return res;
}
```

질문?