수학

25.03.31 (월) 오후 5시 ~ 6시

목차

기초 정수론

기초 정수론 | 01 개요

- 스터디 주제들을 학습하기 위한 꼭 알아야 하는 것들을 설명하겠습니다!
    - 소수, 합성수
    - 최대공약수, 최소공배수, 최대공약수와 최소공배수의 관계
    - 모듈러 연산의 활용

기초 정수론 | 02 소수와 합성수

- 소수
  - 1과 자기 자신만을 약수로 가지는 1보다 큰 자연수

  $ex)\ 2, 3, 5, 7, 11, 13, 17, \ldots$

- 합성수
  - 1과 자기 자신 외에도 다른 약수를 가지는 1보다 큰 자연수
  - 두 개 이상의 소수들의 곱으로 표현될 수 있는 수

기초 정수론 | 03 GCD, LCD

- 최대 공약수 (GCD, Greatest Common Divisor)

  두 수 a, b에 대해 최대 공약수는 $GCD(a, b)$라고 표기한다.

- 최소 공배수 (LCD, Least Common Multiple)

  두 수 a, b에 대해 최소 공배수는 $LCD(a, b)$라고 표기한다.

예) a = 12, b = 9에 대한 GCD, LCD 구하기

- $a = 2^2 \times 3^1$
- $b = 2^0 \times 3^2$
- $LCD(a, b) = 2^2 \times 3^2$
- $GCD(a, b) = 2^0 \times 3^1$

기초 정수론 | 03 GCD, LCD

- **LCD와 GCD의 관계**

  GCD(a, b)를 구할 수 있으면 LCD(a, b)를 구할 수 있다는 의미로 사용된다.

  - $LCD(a, b) \times GCD(a, b) = a \times b$

- 증명

  - $a = p_1^{e_1} p_2^{e_2} p_3^{e_3} p_4^{e_4} \dots p_n^{e_n}$
  - $b = p_1^{f_1} p_2^{f_2} p_3^{f_3} p_4^{f_4} \dots p_n^{f_n}$
  - $LCD(a,\ b) = p_1^{\max(e_1, f_1)} p_2^{\max(e_2, f_2)} p_3^{\max(e_3, f_3)} p_4^{\max(e_4, f_4)} \dots p_n^{\max(e_n, f_n)}$
  - $GCD(a,\ b) = p_1^{\min(e_1, f_1)} p_2^{\min(e_2, f_2)} p_3^{\min(e_3, f_3)} p_4^{\min(e_4, f_4)} \dots p_n^{\min(e_n, f_n)}$
  - $LCD(a, b)\ \times\ GCD(a, b) = p_1^{\max(e_1, f_1)+\min(e_1, f_1)} p_2^{\max(e_2, f_2)+\min(e_2, f_2)} \dots \mathrm{p}_\mathrm{n}^{\max(e_n, f_n)+\min(e_n, f_n)} = a\ \times\ b$

기초 정수론 | 04 모듈러 연산

- 모듈러 연산이란, 어떤 한 숫자를 다른 숫자로 나눈 나머지를 구하는 연산이다.
- **모듈러 연산의 특징**
  - $(A + B) \bmod C = (A \bmod C + B \bmod C) \bmod C$
  - $(A - B) \bmod C = (A \bmod C - B \bmod C) \bmod C$
  - $(A \times B) \bmod C = \left((A \bmod C) \times (B \bmod C)\right) \bmod C$
- **표기법**
  - 정수 d가 정수 a를 나누어 떨어지게 한다면 **d | a** 라고 표기한다.

소인수 분해

소인수 분해 | 01 개요

- 소인수 분해는 어떤 자연수를 소수들의 곱으로 나타내는 방법이다.
- $O(N)$ 소인수 분해
  - 2부터 N까지 하나씩 나누면서 나누어 떨어지는 경우, 나누어 떨어질 때까지 나누기
- $O(\sqrt{N})$소인수 분해
  - N까지 검사할 필요 없이 $\sqrt{N}$까지만 검사해도 됨.
  - 모든 약수는 그 짝이 존재하며, 그 하나는 항상 $\sqrt{N}$ 이하이기 때문임.

소인수 분해 | 02 예제 : 백준 11653번 소인수분해

- **문제 요약**
  - 정수 N이 주어졌을 때, 소인수 분해하는 프로그램을 작성하시오.
- **입력**
  - 첫째 줄에 정수 N (1 ≤ N ≤ 10,000,000)이 주어진다.
- **출력**
  - N의 소인수분해 결과를 한 줄에 하나씩 오름차순으로 출력한다. N이 1인 경우 아무것도 출력하지 않는다.

예제 입력 1 복사

```
72
```

예제 출력 1 복사

```
2
2
2
3
3
```

소인수 분해 | 02 예제 : 백준 11653번 소인수분해

- **풀이**

Line 9 ~ 14

소인수 분해를 진행한다.

N의 제곱근까지만 살피면 된다.

Line 15 ~ 17

소인수 분해가 완전히 끝나지 않는 경우를 처리하기 위해

위 코드가 반드시 필요하다.

ex) 29를 생각해보면 됨.

```cpp
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      ios::sync_with_stdio(false);
6
7      int N;
8      cin >> N;
9      for(int i = 2; i * i <= N; i++){
10         while(N % i == 0){
11             cout << i << "\n";
12             N /= i;
13         }
14     }
15     if(N > 1){
16         cout << N << "\n";
17     }
18
19     return 0;
20 }
```

에라토스테네스의 체

에라토스테네스의 체 | 01 개요

- 에라토스테네스의 체는 여러 개의 자연수에서 소수를 빠르게 찾는 알고리즘이다.
- 배수를 제거하는 방식을 이용하여 소수만을 남긴다.
- **알고리즘 동작 방식**
  1. 2부터 시작하여 2의 배수(4, 6, 8, ...)를 제거
     - I. 3의 배수(6, 9, 12, ...)를 제거
     - II. 4의 배수는 이미 제거되었으므로 PASS
     - III. 5의 배수...
  2. 남아있는 수는 모두 소수!

- **최적화**
  - **어떤 수 p에 대해 p^2부터 시작해서 배수를 제거하면 된다.**
  - p보다 작은 배수는 이미 이전 단계에서 제거되었기 때문이다. 15를 생각해보면 3 x 5이므로 3의 배수를 지울 때 이미 지워졌다.

에라토스테네스의 체 | 02 예제 : 백준 1929번 소수 구하기

- **문제 요약**
  - M이상 N이하의 소수를 모두 출력하는 프로그램을 작성하시오.
- **입력**
  - 첫째 줄에 자연수 M과 N이 빈 칸을 사이에 두고 주어진다. (1 ≤ M ≤ N ≤ 1,000,000)
- **출력**
  - 한 줄에 하나씩, 증가하는 순서대로 소수를 출력한다.

예제 입력 1 복사

```
3 16
```

예제 출력 1 복사

```
3
5
7
11
13
```

에라토스테네스의 체 | 02 예제 : 백준 1929번 소수 구하기

- **구현**

Line 11

is_prime[n] = (n이 소수이면 is_prime[n]은 true의 값을 가진다)

Line 14 ~ 20

에라토스테네스의 체를 구현한 부분이다.

2부터 시작하여, 그것의 배수들을 전부 제거해 나간다.

- i가 이미 제거되었다면, 그것의 배수는 이미 제거되었으므로 더 이상 계산해줄 필요가 없다.
- (최적화) 어떤 수 i에 대해 i*i부터 제거해 나가면 된다.

Line 22 ~ 24

구한 소수들을 출력한다.

```cpp
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  
5  int M, N;
6  int main(){
7      ios::sync_with_stdio(false);
8  
9      cin >> N >> M;
10 
11     vector<bool> is_prime(10101010, true);
12     is_prime[0] = is_prime[1] = false;
13 
14     for(ll i = 2; i <= M; i++){
15         if(is_prime[i]){
16             for(ll j = i * i; j <= M; j += i){
17                 is_prime[j] = false;
18             }
19         }
20     }
21 
22     for(int i = N; i <= M; i++){
23         if(is_prime[i]) cout << i << "\n";
24     }
25 
26     return 0;
27 }
```

유클리드 호제법

유클리드 호제법 | 01 개요

- 유클리드 호제법은 **두 정수 a와 b의 최대공약수를 구하는 알고리즘**이다.

  $GCD(a,b) = GCD(b, a\ mod\ b)$임을 활용한다. ($a$와 b의 대소관계는 중요하지 않다)

- **증명**

  - 두 수 a, b가 있을 때, $a = bq + r$ 로 표현할 수 있다. (단, q는 몫, r은 나머지 , $0 \le r < b$)

  - 어떤 수 d가 a와 b의 공약수라면, $d \mid a$ 그리고 $d \mid b$를 만족한다.

  - $a\ mod\ d = (bq + r)\ mod\ d \rightarrow 0 = (bq\ mod\ d) + (r\ mod\ d) \rightarrow 0 = 0 + (r\ mod\ d) \rightarrow r\ mod\ d = 0$ 이므로 $d \mid r$

  - d는 a와 b의 공약수이지만, b와 r의 공약수 이기도 한 것이다.

    결론적으로, $\boldsymbol{a}$와 $\boldsymbol{b}$의 최대공약수는 $\boldsymbol{b}$와 $\boldsymbol{r}$의 최대공약수와 동일하다.

  - 이 과정을 반복하면, 결국 나머지가 0이 되는 순간($a\ mod\ b$가 0인 순간)의 b값이 최대공약수가 된다.

    $ex)\ \mathrm{gcd}(12, 6) = \mathrm{gcd}(6, 12\ mod\ 6)$

유클리드 호제법 | 01 개요

- **구현** (재귀)

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

- **구현** (반복문)

```
int gcd_iterative(int a, int b) {
    while (b != 0) {
        int r = a % b;
        a = b;
        b = r;
    }
    return a;
}
```

유클리드 호제법 | 02 예제 : 2609번 GCD와 LCD

- **문제 요약**
  - 두 개의 자연수를 입력받아 최대 공약수와 최소 공배수를 출력하는 프로그램을 작성하시오.
- **입력**
  - 첫째 줄에는 두 개의 자연수가 주어진다. 이 둘은 10,000이하의 자연수이며 사이에 한 칸의 공백이 주어진다.
- **출력**
  - 첫째 줄에는 입력으로 주어진 두 수의 최대공약수를, 둘째 줄에는 입력으로 주어진 두 수의 최소 공배수를 출력한다.

예제 입력 1 복사

```
24 18
```

예제 출력 1 복사

```
6
72
```

유클리드 호제법 | 02 예제 : 2609번 GCD와 LCD

- **풀이**

```cpp
#include <bits/stdc++.h>
using namespace std;

int A, B;

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

int main(){
    cin >> A >> B;
    int ret = gcd(A, B);
    cout << ret << "\n";
    cout << (A * B) / ret << "\n";
    return 0;
}
```

거듭제곱

거듭제곱 | 01 개요

- **일반적인 거듭제곱 계산 방법**
  - $a^n = a \ * a \ * a \ \ldots * a$
  - 이 방식대로 계산하려면 O(N)

```
long long power(long long base, long long exponent) {
    long long result = 1;
    for (long long i = 0; i < exponent; i++) {
        result *= base;
    }
    return result;
}
```

- 더 빠른 방법 없을까?

- **빠른 거듭제곱 계산 방법**
  - a^8을 계산하고 싶으면 a^4를 계산하면 된다. $a^8 = (a^4)^2$에서 a^4만 계산하면 되기 때문이다.
  - 지수가 홀수여도 문제없다. a^7을 계산하고 싶으면 a^3을 계산하면되고 $a^7 = a \times (a^3)^2$을 계산하면 된다.
  - $a^n = \begin{cases} (a^{n/2})^2 \ (n\text{은 짝수}) \\ a \times (a^{(n-1)/2})^2 \ (n\text{은 홀수}) \end{cases}$
  - 이 방식대로 계산하면 $O(\log_2 n)$이다!
  - 참고) 정수 2개를 피연산자로 받는 / 연산자는 몫 연산을 리턴해주므로 1을 안 빼도 결과는 동일하다.
  - 후에서는 동일한 부분 문제임을 강조하기 위해 1을 안 빼서 표기하는 것으로 하겠다.

거듭제곱 | 01 개요

- **빠른 거듭제곱 계산 방법 - 구현**

```
long long fastPower(long long base, long long exponent) {
    if (exponent == 0) return 1; // a^0 = 1
    long long half = fastPower(base, exponent / 2);
    if (exponent % 2 == 0)
        return half * half; // 짝수일 경우
    else
        return base * half * half; // 홀수일 경우
}
```

참고) 반복문으로도 구현이 가능하다.

거듭제곱 | 01 개요

- **큰 수의 거듭제곱**
  - C++은 BigInt를 지원하지 않는다. // 파이썬은 지원함
  - a^n 계산에서 INT64_MAX를 초과하는 값이 발생하면 오버플로우가 발생한다.
  - 그래서 **매우 큰 수의 거듭제곱을 구하라는 문제**에서는 **나머지**를 출력하라고 한다.
    - 예) a^b 의 결과 값을 c로 나눈 나머지를 출력하시오.

  - 모듈러 연산의 특징을 활용하라는 것이다.
    - $(a \times b) \, mod \, M = [(a \, mod \, M) \times (b \, mod \, M)] \, mod \, M$
    - $b = a^{N-1}$ 라고 한다면, $a^N \, mod \, M = (a \times a^{N-1}) \, mod \, M = [(a \, mod \, M) \times (a^{N-1} \, mod \, M)] \, mod \, M$
    - 이렇게 짜면 오버플로우가 발생하지 않으므로 올바른 값을 출력할 것이다. 하지만, 지수가 커질 경우에는 TLE이다.

거듭제곱 | 01 개요

- **큰 수의 거듭제곱**
  - 빠른 거듭제곱 + 모듈러 연산 → $O(\log_2 N)$
    - N이 짝수일때, $a^N \, mod \, M \; = \left(a^{\frac{N}{2}} \times a^{\frac{N}{2}}\right) mod \, M = \left[(a^{\frac{N}{2}} \, mod \, M) \; \times \; (a^{\frac{N}{2}} \, mod \, M)\right] \, mod \, M$
    - N이 홀수일때, $a^N \, mod \, M \; = \left(a \times a^{\frac{N}{2}} \times a^{\frac{N}{2}}\right) mod \, M = \left[(a \, mod \, M) \; \times \; (a^{\frac{N}{2}} \, mod \, M) \; \times \; (a^{\frac{N}{2}} \, mod \, M)\right] \, mod \, M$

  - 구현 방법은 예제를 다루면서 살펴보겠습니다.

거듭제곱 | 02 예제 : 백준 1629번 빠른 거듭제곱

- **문제요약**

  A^B를 구하는 게 목표인데, 매우 커질 수 있으므로 C로 나눈 나머지를 출력해라.

  A, B, C는 2,147,483,647 이하의 자연수이다.

- **입력**

  A B C가 주어진다.

- **출력**

  첫째 줄에 A를 B번 곱한 수를 C로 나눈 나머지를 출력한다.

예제 입력 1 복사

```
10 11 12
```

예제 출력 1 복사

```
4
```

거듭제곱 | 02 예제 : 백준 1629번 빠른 거듭제곱

- **풀이**
  - $A^B \ mod \ C$를 직접 계산해볼까?
  - A^B 를 계산하고 C로 나눈 나머지를 구하면 되는데…
  - 이 과정에서 오버플로우가 발생하여 틀린 값이 나온다. // 최악의 경우 : A = B = C = INT32_MAX 일 때
    - 오버플로우가 나지 않더라도 곱하는 연산을 20억번 반복하는 것 자체가 시간 초과의 원인이다.

거듭제곱 | 02 예제 : 백준 1629번 빠른 거듭제곱

- **풀이**
  - 오버플로우 방지를 위해 모듈러 연산의 특징을 활용해야 한다.

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

"A와 B를 곱한 후 C로 나눈 나머지는, 각각 A와 B를 C로 나눈 나머지를 곱한 후 다시 C로 나눈 나머지와 같다."

  - 이 연산을 활용하면 큰 수의 연산에서 오버플로우를 방지하는 데 유용하다. 즉, 값이 훼손되지 않는다.

거듭제곱 | 02 예제 : 백준 1629번 빠른 거듭제곱

- **풀이**
  - $A^2\ mod\ C$의 결과로 $(A^2 \times A)\ mod\ C$ 를 하고, 그것의 결과로 $(A^3 \times A)\ mod\ C$ 를 하면

    오버플로우가 나지 않을 것이므로 올바른 $A^B\ mod\ C$ 의 결과값이 나온다.

    하지만, B는 여전히 21억이므로 저 연산을 21억 번 반복하는 건 TLE이다.

거듭제곱 | 02 예제 : 백준 1629번 빠른 거듭제곱

- **풀이**
  - 분할 정복의 아이디어를 활용하면 B번을 전부 하지 않아도 된다. $\log_2 B$번만 하면 된다.
  - 예를 들자면, $A^{17} mod\ C = [(A\ mod\ C) * (A^8\ mod\ C)^2]\ mod\ C$ 로 계산할 수 있다.
  - $A^{17} mod\ C$ 라는 문제를 하위 문제 $A^8 mod\ C$로 **분할**했다.
  - 이 과정을 반복하다보면 $A^0 mod\ C$ 를 계산하게 되는데 1 mod C는 1이다. **기저 조건**인 것이다.
  - **정복** 과정은 간단하다. 구한 $A^8 mod\ C$ 값을 이용해 $A^{17} mod\ C$ 를 계산하면 된다.

  - N이 짝수일때, $a^N\ mod\ M\ = \left(a^{\frac{N}{2}} \times a^{\frac{N}{2}}\right) mod\ M = \left[(a^{\frac{N}{2}}\ mod\ M)\ \times\ (a^{\frac{N}{2}}\ mod\ M)\right]\ mod\ M$
  - N이 홀수일때, $a^N\ mod\ M\ = \left(a\ \times\ a^{\frac{N}{2}}\ \times\ a^{\frac{N}{2}}\right) mod\ M = \left[(a\ mod\ M)\ \times\ (a^{\frac{N}{2}}\ mod\ M)\ \times\ (a^{\frac{N}{2}}\ mod\ M)\right]\ mod\ M$

거듭제곱 | 02 예제 : 백준 1629번 빠른 거듭제곱

- **구현**

Line 7 ~ 8

- $func(b)$는 $a^b \ mod \ c$를 계산해주는 함수라고 믿는다(정의한다).
- $b = 0$인 경우, $a^0 \ mod \ C = 1$이다. 기저조건이다.

Line 9

$func(b/2)$을 계산한다. (= $a^{\frac{b}{2}} \ mod \ C$를 계산한다)

Line 10

- 9행을 마치고 10행으로 실행이 넘어왔을 때는

  이미 $a^{\frac{b}{2}}$ mod C 값이 계산되었다는걸 알아야 한다.

- 편의를 위해 $\left(a^{\frac{b}{2}}\right)^2 \ mod \ C$ 를 먼저 계산해준다.

  b가 홀수이든, 짝수이든 항상 쓰이는 값이기 때문이다.

Line 12 ~ 17

- 짝수일 때, 홀수일 때 값을 계산해준다.

```cpp
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  
5  ll A, B, C;
6  
7  ll func(ll b){
8      if(b == 0) return 1; // a^0은 1이다.
9      ll half = func(b/2); // a^(b/2)을 계산한다.
10     ll all = half * half % C; // (a^(b/2))^2을 계산한다.
11 
12     if(b % 2 == 1){
13         // b가 홀수이면 a^n = a * a^(b-1) = a * (half^2)이다.
14         return (A * all) % C;
15     }
16     // b가 짝수이면 a^b = (a^(b/2))^2 = half^2 이다.
17     return all;
18 }
19 
20 int main(){
21     cin >> A >> B >> C;
22     cout << func(B) << "\n";
23     return 0;
24 }
```

추천 문제집(https://www.acmicpc.net/workbook/view/8174)입니다!

중간고사 끝나고 뵙겠습니다. 감사합니다!