03. Basic Math

이화여자대학교 황채원

강의 목차

강사 소개

황채원

- 이화여자대학교 컴퓨터공학부
- AI Security Lab 인턴 (2022.07~Present)

Modular Arithmetic

- 주어진 숫자를 다른 숫자로 나눈 나머지를 구하는 연산
- 연산의 결과가 항상 0 이상 m(모듈러 값) 미만

- 10 % 4 = 2
- 15 % 6 = 3

음수에 대한 Modular Arithmetic?

- 음수 결과에 모듈러 값을 더해 양수 범위로 변환
- -7 % 5 = 3
- -14 % 6 = 4

참고) 음수에 대한 모듈러 연산은 C++과 python에서 결과가 다르다.

C++: -14 % 6 = -2 (피제수의 부호를 반영한다)

Python: -14 % 6 = 4 (결과가 양수가 되도록 한다)

모듈러 합동(modular congruence)

$$a \equiv b(mod\ m)$$

- 모듈러 연산의 개념을 확장한 것으로, 두 숫자가 주어진 모듈러 값에 대해 같은 나머지를 갖는 경우
- a를 m으로 나눈 나머지와 b를 m으로 나눈 나머지가 같다

- $38 \equiv 14 \pmod{12}$
- $-3 \equiv 8 \pmod{11}$

모듈러 연산의 성질

5 10430번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 난이도 기여 | 강의 | 질문 게시판

나머지 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	377493	195232	168722	52.048%

문제

(A+B)%C는 ((A%C) + (B%C))%C 와 같을까?

(A×B)%C는 ((A%C) × (B%C))%C 와 같을까?

세 수 A, B, C가 주어졌을 때, 위의 네 가지 값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 A, B, C가 순서대로 주어진다. (2 ≤ A, B, C ≤ 10000)

출력

첫째 줄에 (A+B)%C, 둘째 줄에 ((A%C) + (B%C))%C, 셋째 줄에 (A×B)%C, 넷째 줄에 ((A%C) × (B%C))%C를 출력한다.

모듈러 연산의 성질

- 모듈러 덧셈: (A+B)%C=(A%C+B%C)%C
- 모듈러 곱셈: (A×B)%C=(A%C×B%C)%C

- 덧셈, 뺄셈, 곱셈에 대해서 모듈러 연산의 분배법칙이 성립한다.
- 단, 나눗셈에 대한 모듈러 연산은

  '모듈러 역원'(modular multiplicative inverse)의 개념을 사용하여 정의

모듈러 연산의 성질

```cpp
#include <iostream>
using namespace std;

int main() {
    int A, B, C;
    cin >> A >> B >> C;

    cout << (A + B) % C << '\n';
    cout << ((A % C) + (B % C)) % C << '\n';
    cout << (A * B) % C << '\n';
    cout << ((A % C) * (B % C)) % C << '\n';

    return 0;
}
```

모듈러 역원

- 모듈러 나눗셈을 계산하기 위해서는 모듈러 역원을 찾는 과정이 필요
- A / B mod C는 A * B의 역원 mod C
- B의 역원은 (B * X) % C = 1을 만족하는 X값을 의미
- (3 * X) % 7 = 1 ?

  (3 * 5) % 7 = 15 % 7 = 1 이므로 X = 5
- 역원이 존재하기 위해서는 숫자 B와 모듈러 값 C가 서로소여야 한다.

  이는 모듈러 값 C가 소수일 때 항상 만족된다.

소수 판정

소수(Prime Number): 1보다 큰 자연수 중 1과 자기 자신만을 약수로 가지는 수

ex) 5는 5*1 또는 1*5로 수를 곱한 결과를 적는 유일한 방법이 그 수 자신을 포함하기 때문에 5는 소수이다.

소수 판정

방법 1 : 2 ~ (N-1)까지 반복하며 나눠보기

- 시간 복잡도 O(n)

```
bool isPrimeBasic(int num) {
    for(int i = 2; i < num; i++) {
        if(num % i == 0) return false;
    }
    return true;
}
```

소수 판정

방법 2 : 2 ~ (N/2)까지 반복하며 나눠보기

- 시간 복잡도 O(n/2) = O(n)
- 방법 1보다는 낫지만, 충분히 효율적이지 못하다!

```
bool isPrimeHalf(int num) {
    for(int i = 2; i <= num / 2; i++) {
        if(num % i == 0) return false;
    }
    return true;
}
```

소수 판정

방법 3 : 2 ~ √n 까지 반복하며 나눠보기

- 시간 복잡도 O(√n)
- 작은 숫자에 대해서는 세 방법 모두 큰 차이가 없지만, 숫자가 커질수록 제곱근 방법의 효율성이 두드러진다.

```
bool isPrimeSqrt(int num) {
    for(int i = 2; i * i <= num; i++) {
        if(num % i == 0) return false;
    }
    return true;
}
```

소수 판정

1978번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 내 제출 난이도 기여 강의 질문 게시판

소수 찾기 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	195606	92454	73889	47.141%

문제

주어진 수 N개 중에서 소수가 몇 개인지 찾아서 출력하는 프로그램을 작성하시오.

입력

첫 줄에 수의 개수 N이 주어진다. N은 100이하이다. 다음으로 N개의 수가 주어지는데 수는 1,000 이하의 자연수이다.

출력

주어진 수들 중 소수의 개수를 출력한다.

예제 입력 1 복사

```
4
1 3 5 7
```

예제 출력 1 복사

```
3
```

소수 판정

```cpp
#include <iostream>
#include <cmath>
using namespace std;

bool isPrime(int num) {
    if(num < 2) return false;
    for(int i = 2; i <= sqrt(num); i++) {
        if(num % i == 0) return false;
    }
    return true;
}

int main() {
    int N, count = 0;
    cin >> N;

    for(int i = 0; i < N; i++) {
        int num;
        cin >> num;
        if(isPrime(num)) count++;
    }

    cout << count << endl;
    return 0;
}
```

여러 개의 수에 대한 소수 판정?

큰 범위 내에 존재하는 모든 소수를 찾아야 할 때는 어떻게 해야 할까?

=> 에라토스테네스의 체 알고리즘

에라토스테네스의 체

- 각 수가 소수인지 판단한 결과를 저장하는 배열을 사용
- 2 ~ √n 까지 반복하며 해당 숫자의 배수에 해당하는 숫자들을 차례대로 지워나간다
- 시간 복잡도 O(N log(log N))

에라토스테네스의 체

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

에라토스테네스의 체

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

에라토스테네스의 체

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

에라토스테네스의 체

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

에라토스테네스의 체

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

에라토스테네스의 체

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

에라토스테네스의 체

```cpp
vector<bool> prime(N + 1, true);
prime[0] = prime[1] = false;

for (int i = 2; i * i <= N; i++) {
    if (prime[i]) {
        for (int j = i * i; j <= N; j += i) {
            prime[j] = false;
        }
    }
}
```

에라토스테네스의 체

3 1929번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 내 제출 난이도 기여 강의 질문 게시판

소수 구하기 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	256 MB	266404	78288	55089	27.412%

문제

M이상 N이하의 소수를 모두 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 M과 N이 빈 칸을 사이에 두고 주어진다. (1 ≤ M ≤ N ≤ 1,000,000) M이상 N이하의 소수가 하나 이상 있는 입력만 주어진다.

출력

한 줄에 하나씩, 증가하는 순서대로 소수를 출력한다.

예제 입력 1 복사

```
3 16
```

예제 출력 1 복사

```
3
5
7
11
13
```

하나씩 판별하는 알고리즘을 사용하면

```cpp
#include <iostream>
#include <cmath>
using namespace std;

bool isPrime(int num) {
    if(num < 2) return false;
    for(int i = 2; i <= sqrt(num); i++) {
        if(num % i == 0) return false;
    }
    return true;
}

int main() {
    int M, N;
    cin >> M >> N;

    for(int i = M; i <= N; i++) {
        if(isPrime(i)) {
            cout << i << endl;
        }
    }
    return 0;
}
```

- 시간초과 발생!

에라토스테네스의 체를 적용하자

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    ios::sync_with_stdio(false); // C++ 입출력 속도 향상
    cin.tie(nullptr);
    int M, N;
    cin >> M >> N;

    vector<bool> prime(N + 1, true);
    prime[0] = prime[1] = false;

    for (int i = 2; i * i <= N; i++) {
        if (prime[i]) {
            for (int j = i * i; j <= N; j += i) {
                prime[j] = false;
            }
        }
    }

    for (int i = M; i <= N; i++) {
        if (prime[i]) {
            cout << i << '\n';
        }
    }

    return 0;
}
```

에라토스테네스의 체

16563번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 내 제출 난이도 기여 질문 게시판

어려운 소인수분해 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	7304	2086	1215	24.786%

문제

지원이는 대회에 출제할 문제에 대해서 고민하다가 소인수분해 문제를 출제해야겠다고 마음을 먹었다. 그러나 그 이야기를 들은 동생의 반응은 지원이의 기분을 상하게 했다.

> "소인수분해? 그거 너무 쉬운 거 아니야?"

지원이는 소인수분해의 어려움을 알려주고자 엄청난 자신감을 가진 동생에게 2와 500만 사이의 자연수 N개를 주고 소인수분해를 시켰다. 그러자 지원이의 동생은 기겁하며 쓰러졌다. 힘들어하는 지원이의 동생을 대신해서 여러분이 이것도 쉽다는 것을 보여주자!

입력

첫째 줄에는 자연수의 개수 N ($1 \le N \le 1{,}000{,}000$)이 주어진다.

둘째 줄에는 자연수 k_i ($2 \le k_i \le 5{,}000{,}000$, $1 \le i \le N$)가 N개 주어진다.

출력

N줄에 걸쳐서 자연수 k_i의 소인수들을 오름차순으로 출력하라.

예제 입력 1 복사

```
5
5 4 45 64 54
```

예제 출력 1 복사

```
5
2 2
3 3 5
2 2 2 2 2 2
2 3 3 3
```

에라토스테네스의 체

소인수 분해

- 1보다 큰 자연수를 소인수(소수인 인수)들만의 곱으로 나타내는 것
- 합성수를 소수의 곱으로 나타내는 방법
- $198 = 2 \times 3 \times 3 \times 11$

에라토스테네스의 체

1. 에라토스테네스의 체를 이용해서 N 이하의 소수들을 모두 구한다.
2. 입력 받은 수를 미리 구해 놓은 소수들을 이용해서 차례대로 나눈다.

에라토스테네스의 체

```cpp
#include <iostream>
#include <vector>
using namespace std;

void findSmallestPrimeFactors(vector<bool>& is_prime, vector<int>& smallest_prime_factor, int max_val) {
    is_prime[0] = is_prime[1] = false;

    for (int i = 2; i * i <= max_val; i++) {
        if (is_prime[i]) {
            smallest_prime_factor[i] = i;
            for (int j = i * i; j <= max_val; j += i) {
                is_prime[j] = false;
                if (smallest_prime_factor[j] == 0) smallest_prime_factor[j] = i;
            }
        }
    }
}
```

```cpp
int main() {
    ios::sync_with_stdio(false); // 입출력 속도 향상
    cin.tie(nullptr);            // cin과 cout의 tie 해제

    const int MAX_VAL = 5000000;
    vector<bool> is_prime(MAX_VAL + 1, true);
    vector<int> smallest_prime_factor(MAX_VAL + 1, 0);

    findSmallestPrimeFactors(is_prime, smallest_prime_factor, MAX_VAL);

    int n;
    cin >> n;
    while (n--) {
        int num;
        cin >> num;

        // 소인수분해 및 출력
        while (num > 1) {
            cout << smallest_prime_factor[num] << ' ';
            num /= smallest_prime_factor[num];
        }
        cout << '\n';
    }

    return 0;
}
```

유클리드 호제법

공약수 (公約數, 영어: common divisor): 두 수 n과 m에 대해, 동시에 n과 m 모두의 약수가 되는 정수

최대공약수(Greatest Common Divisor, GCD) : 두 수 n과 m에 대해, n과 m의 공약수 중 가장 큰 수

공배수(公倍數, 영어: common multiple): 두 수 n과 m에 대해, 동시에 n과 m 모두의 배수가 되는 정수

최소공배수(Lowest Common Multiple, LCM) : 두 수 n과 m에 대해, n과 m의 공배수 중 가장 작은 수

유클리드 호제법

유클리드 호제법을 사용하면 효율적으로 최대공약수를 구할 수 있다.

1. A % B 를 계산한다 (A > B)
2. A = B 로 업데이트, B = A % B 로 업데이트
3. mod 연산의 값이 0이 될 때까지 반복

   마지막 계산의 모듈러 값이 최대공약수가 된다.

유클리드 호제법

예를 들어, 252와 105의 최대공약수를 구한다고 가정해봅시다.

첫 번째 단계: 252 % 105 = 42

두 번째 단계: 105 % 42 = 21

세 번째 단계: 42 % 21 = 0

결과: 세 번째 단계에서 나머지가 0이 되었으므로, 이때의 나누는 수인 21이 최대공약수입니다.

유클리드 호제법

유클리드 호제법의 구현

1. 반복문을 사용하는 방법

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

2. 재귀함수를 사용하는 방법

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

유클리드 호제법

최소공배수도 유클리드 호제법을 이용해 구할 수 있다.

$$LCM(A,B) = \frac{|A \times B|}{GCD(A,B)}$$

```
// 유클리드 호제법을 이용한 최대공약수 계산 함수
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// 최소공배수 계산 함수
int lcm(int a, int b) {
    return a / gcd(a, b) * b; // 먼저 a를 GCD로 나누어 오버플로우 방지
}
```

유클리드 호제법

2609번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 난이도 기여 | 강의 | 질문 게시판

최대공약수와 최소공배수 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	106686	61531	50079	57.867%

문제

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

```cpp
#include <iostream>
using namespace std;

// 유클리드 호제법을 이용한 최대공약수(GCD) 계산 함수
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// 최소공배수(LCM) 계산 함수
int lcm(int a, int b) {
    return a / gcd(a, b) * b;
}

int main() {
    int a, b;
    cin >> a >> b;

    cout << gcd(a, b) << '\n'; // 최대공약수 출력
    cout << lcm(a, b) << '\n'; // 최소공배수 출력

    return 0;
}
```

유클리드 호제법

9613번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점 결과 | 채점 현황 | 내 제출 | 난이도 기여 | 강의 | 질문 게시판

GCD 합 성공 다국어

☆ 한국어

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	34995	14382	11838	41.833%

문제

양의 정수 n개가 주어졌을 때, 가능한 모든 쌍의 GCD의 합을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 t (1 ≤ t ≤ 100)이 주어진다. 각 테스트 케이스는 한 줄로 이루어져 있다. 각 테스트 케이스는 수의 개수 n (1 < n ≤ 100)가 주어지고, 다음에는 n개의 수가 주어진다. 입력으로 주어지는 수는 1,000,000을 넘지 않는다.

출력

각 테스트 케이스마다 가능한 모든 쌍의 GCD의 합을 출력한다.

예제 입력 1 복사

```
3
4 10 20 30 40
3 7 5 12
3 125 15 25
```

예제 출력 1 복사

```
70
3
35
```

유클리드 호제법

```cpp
#include <iostream>
#include <vector>
using namespace std;

// 유클리드 호제법을 이용한 최대공약수(GCD) 계산 함수
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

```cpp
int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        vector<int> numbers(n);
        for (int i = 0; i < n; i++) {
            cin >> numbers[i];
        }

        long long sum = 0;
        for (int i = 0; i < n; i++) {
            for (int j = i + 1; j < n; j++) {
                sum += gcd(numbers[i], numbers[j]);
            }
        }

        cout << sum << '\n';
    }
    return 0;
}
```

문제

필수 문제

- BOJ 2960 (에라토스테네스의 체)
- BOJ 6588 (골드바흐의 추측)
- BOJ 1735 (분수 합)
- BOJ 17087(숨바꼭질 6)
- BOJ 14490 (백대열)

심화 문제

- BOJ 13172 (Σ)
- BOJ 20302 (민트초코)