# 알튜비튜

# 정수론

오늘은 각종 수의 성질을 다루는 정수론에 대해 배웁니다.
특히, 최대공약수를 효율적으로 구하는 유클리드 호제법과 소수를 빠른 시간 내에 판별하는
에라토스테네스의 체에 대해 알아봅시다.

# 소개

## 정수론

약수 배수

최대공약수 최소공배수

소수 판별

# 최대공약수

## 최대공약수 구하기

- 소인수분해를 이용하여 구하기 -> 구현이 까다로움
- 두 수 중 작은 수 기준으로 돌리면서 가장 큰 공통 약수 구하기 -> 시간복잡도 O(n)

## 최대공약수 구하기

- 소인수분해를 이용하여 구하기 -> 구현이 까다로움
- 두 수 중 작은 수 기준으로 돌리면서 가장 큰 공통 약수 구하기 -> 시간복잡도 O(n)
- 유클리드 호제법 -> 시간복잡도 O(log(n))

# 유클리드 호제법 기본 원리

#알튜비튜

**A와 B의 최대공약수 = A-B와 B의 최대공약수**

- A = a · G
- B = b · G (a와 b는 서로소)
- A - B = a · G – b · G = (a – b) · G

-> (a – b) 와 b 또한 서로소 이므로 A – B 와 B 의 최대공약수도 G

- GCD(A, B) = GCD(A-B, B) -> A, B의 차이가 크면 오래걸림

# 유클리드 호제법 기본 원리

#알튜비튜

**A와 B의 최대공약수 = A%B와 B의 최대공약수**

- A = a · G
- B = b · G (a와 b는 서로소)
- A = q · B + r (q = A/B 의 몫, r = A%B)
- r (A%B) = a · G – q · b · G = (a – q · b) · G

-> (a – q · b) 와 b 또한 서로소 이므로 A%B 와 B 의 최대공약수도 G

- GCD(A, B) = GCD(A%B, B)

# 유클리드 호제법 기본 원리 - 예시

#알튜비튜

**A와 B의 최대공약수 = A%B와 B의 최대공약수**

- $12 = 3 \cdot 4$
- $8 = 2 \cdot 4$ (2와 3은 서로소)
- $12 = q \cdot 8 + 4$ ($q = 12/8$ 의 몫, $4 = 12\%8$)
- $4\ (12\%8) = 3 \cdot 4 - q \cdot 2 \cdot 4 = (3 - q \cdot 2) \cdot 4$

-> $(3 - q \cdot 2)$ 와 2 또한 서로소 이므로 A%B(4) 와 B(8) 의 최대공약수도 4

- GCD(A, B) = GCD(A%B, B)

# 유클리드 호제법 과정

## 유클리드 호제법

- 두 정수 a, b가 주어짐 (a > b)
- a와 b의 최대공약수는 a%b와 b의 최대공약수와 같음
- a%b를 구한 후, 왼쪽 값 > 오른쪽 값 이어야 하므로 a%b와 b의 위치 바꿈
- b가 0일 때, a의 값이 최대공약수

# 유클리드 호제법 - 반복문 구현

#알튜비튜

```
#include <iostream>

using namespace std;

int main(){

  int a, b;

  while(b){ //b==0, a가 최대공약수

    int t = a % b;
    a = b; //a엔 b 값을
    b = t; //b엔 a % b 값을 저장

  }

  return 0;
}
```

# /<> 2609번 : 최대공약수와 최소공배수 – Silver 5

## 문제

- 두 자연수의 최대공약수와 최소공배수 출력

## 제한 사항

- 입력 범위는 1 <= a,b <= 10,000

## 풀이

- 최소공배수는 A * B = G * L 공식을 이용!

## 예제 입력

```
24 18
```

## 예제 출력

```
6
72
```

# 소개

#알튜비튜

**정수론**

약수 배수
최대공약수 최소공배수
소수 판별

## 소수 구하기

- 2 ~ n 까지 돌리면서 나눠지는 수가 없는지 판단 -> O(n)
- 2 ~ $\sqrt{n}$ 까지 돌리면서 나눠지는 수가 없는지 판단 -> O($\sqrt{n}$ )

-> a = n / ($\sqrt{n}$이상의 수) 라는 a 가 존재한다면, a 는 2 이상 $\sqrt{n}$ 이하다 (n % a = 0)
-> 따라서 $\sqrt{n}$ 까지만 확인을 하면 그 이상은 확인할 필요가 없다
(예) n = 8, 8 = 2 * 4 = 4 * 2 로 대칭

# 소수

## 소수 구하기

- 2 ~ n 까지 돌리면서 나눠지는 수가 없는지 판단 -> O(n)
- 2 ~ $\sqrt{n}$ 까지 돌리면서 나눠지는 수가 없는지 판단 -> O($\sqrt{n}$ )

-> n이 큰 상황에서 한 번에 여러 개의 소수를 구해야 한다면?

# 소수판별법

#알튜비튜

## 에라토스테네스의 체

- 각 수가 소수인지 판단한 여부를 저장하는 배열 사용
- 2부터 시작해서 해당 숫자의 배수에 해당하는 숫자들을 지워나감 (~ $\sqrt{n}$ )

-> 약수가 존재하면 소수가 아니므로
-> 해당 숫자는 소수

- O(nlog(logn))만에 2 ~ n까지 수에 대해 소수 판정 가능

# 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

#알튜비튜

| | | | | | |
|---|---|---|---|---|---|
| 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

#알튜비튜

| | | | | | |
|---|---|---|---|---|---|
| 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

#알튜비튜

| | | | | | |
|---|---|---|---|---|---|
| 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 |

-> 2, 3, 5, 7, 11, 13, 17, 19, 23, 29

# 에라토스테네스의 체



```
void isPrime(int n, vector<bool> &is_prime) {
    //0과 1은 소수가 아니므로 먼저 제거
    is_prime[0] = is_prime[1] = false;
    //2 ~ 제곱근 n까지 검사
    for (int i = 2; i <= sqrt(n); i++) {
        if (is_prime[i]) { //i가 소수라면
            for (int j = i * i; j <= n; j += i) {
                is_prime[j] = false; //i의 배수를 제거
            }
        }
    }
}
```

# 2960번 : 에라토스테네스의 체– Silver 4

## 문제

- 주어진 n 에 대해 에라토스테네스의 체를 적용했을 때, k 번째 지우는 수를 구하는 문제

-> 에라토스테네스의 체 구현하고, 지우는 순서를 카운트하자!

## 제한 사항

- 입력 범위는 1 <= K < N <= 1,000

# 예제

#알튜비튜

| 예제 입력 | 예제 입력 | 예제 입력 |
|---|---|---|
| 7 3 | 15 12 | 10 7 |
| **예제 출력** | **예제 출력** | **예제 출력** |
| 6 | 7 | 9 |

## 16563번 : 어려운 소인수분해 – Gold 4

### 문제

- N 개의 자연수 k 를 소인수분해하여 소인수들을 오름차순으로 출력하는 문제

### 제한 사항

- N의 범위는 $1 <= N <= 1,000,000$
- k의 범위는 $2 <= k <= 5,000,000$

-> 최대 5,000,000까지의 수 중 모든 소수를 구해야 하므로 에라토스테네스의 체 활용

# 예제

#알튜비튜

## 예제 입력

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

## 예제 출력

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

# 몰래 보세요

#알튜비튜

## Hint

1. 에라토스테네스의 체에서 해당 수의 배수를 지워나갈 때 해당 수는 소수였죠…?
2. 구해진 소수를 활용하는 것이 아니라, 이 소수를 구하는 과정을 활용할 수 없을까요?

# 에라토스테네스의 체의 과정 활용?

- (예) K = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| prime | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |

# 에라토스테네스의 체의 과정 활용?



- (예) K = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| prime | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 2 |

  - 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장

# 에라토스테네스의 체의 과정 활용?



- (예) K = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| prime | 0 | 2 | 3 | 2 | 0 | 2 | 0 | 2 |

- 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장

# 에라토스테네스의 체의 과정 활용?



- (예) K = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| prime | 0 | 2 | 3 | 2 | 0 | 2 | 0 | 2 |

- 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장
- 이미 값이 존재한다면 갱신 x

# 에라토스테네스의 체의 과정 활용?



- (예) K = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| prime | 0 | 2 | 3 | 2 | 5 | 2 | 7 | 2 |

- 소수 판단 여부 정보가 아닌 어느 소수의 배수로 지워졌는지 저장
- 이미 값이 존재한다면 갱신 x

# 에라토스테네스의 체의 과정 활용?

- (예) K = 8

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| prime | 0 | 2 | 3 | 2 | 5 | 2 | 7 | 2 |

- 모두 완료하면 역으로 경로 조사 시작
- 8 -> 8/prime[8] -> 4 -> 4/prime[4] -> 2 -> 2 == prime[2] 종료

이때의 prime 값 출력

# 응용 문제

#알튜비튜

## /<> 14490번 : 백대열- Silver 4

### 문제

- n:m 이 주어질 때, 두 수를 최대한 약분하여 a:b형태로 출력하는 문제

-> 최대공약수 문제!

### 제한 사항

- 입력 범위는 1 <= n, m <= 100,000,000

-> 그냥 풀어도 풀리지만, 유클리드 호제법을 사용해보자!
-> 입력을 어떻게 받지?

### 예제 입력

```
100:10
```

### 예제 출력

```
10:1
```

# 몰래 보세요



## Hint

1. String 을 int 로 변환하는 함수가 있는 거 아시나요..?

# 마무리

#알튜비튜

## 정리

- 최대공약수를 빠르게 찾는 유클리드 호제법
- 대량의 소수를 빠르게 판별하는 에라토스테네스의 체
- 에라토스테네스의 체는 $2 \sim \sqrt{n}$까지 검사 (단, 문제에서 어떻게 활용하느냐에 따라 예외 존재)
- String 을 int 로 변환시켜주는 stoi 함수!

# 과제



3문제 이상 선택

- 6588번 : 골드바흐의 추측 – Silver 1
- 9613번 : GCD 합 – Silver 3
- 11050번 : 이항 계수 1 – Bronze 1
- 20302번 : 민트 초코 – Gold 5
- 2841번 : 외계인의 기타 연주 – Silver 1
- 2436번 : 공약수 – Gold 5