D.COM ALGORITHM STUDY

7강) 수학

2020 - 07 VER.

7강.
수학

오늘의 목표

알고리즘 문제에서 자주 사용하는
수학 개념을 알아보고 구현해봅시다.

=> 최대공약수(GCD), 최소공배수(LCM)
=> 소수(Prime Number)

#어렵고 딱딱한 수학 내용이 아니니 편한 마음으로 임해주세요!

가볍게,
읽을거리

;

링크

일일 커밋의 효용성

"결과가 눈에 보이는 지표를 찾자"

수학

Math

최대공약수, 최소공배수

GCD, LCM

Q.

“사탕 75개, 초콜릿 102개, 풍선껌 153개를 수학 반 학생들에게 똑같이 나누어 주었더니
사탕이 3개, 초콜릿이 6개, 풍선 껌이 9개가 남았다. <u>가능한 수학 반 학생수를 모두 구하여라.”</u>

#놀랍게도 중1 수학입니다.

“가능한 많이 똑같이 나누려면”
어떻게 할까요?

Q.

"사탕 75개, 초콜릿 102개, 풍선껌 153개를 수학 반 학생들에게 똑같이 나누어 주었더니
사탕이 3개, 초콜릿이 6개, 풍선 껌이 9개가 남았다. <u>가능한 수학 반 학생수를 모두 구하여라."</u>

#놀랍게도 중1 수학입니다.

<u>"가능한 많이 똑같이 나누려면"</u>
어떻게 할까요?

=> 최대공약수

ı 최대공약수

"0이 아닌 두 정수나 다항식의 공통되는 약수 중에서 가장 큰 수를 말한다."

ı 최대공약수

“0이 아닌 두 정수나 다항식의 공통되는 약수 중에서 가장 큰 수를 말한다.”

<u>어떻게 구현할 수 있을까요?</u>

I 최대공약수

"0이 아닌 두 정수나 다항식의 공통되는 약수 중에서 가장 큰 수를 말한다."

[1, min(a,b)] 범위에서
두 수 모두의 약수가 되는 값들의 최댓값을 구하자!

I 최대공약수

"0이 아닌 두 정수나 다항식의 공통되는 약수 중에서 가장 큰 수를 말한다."

[1, min(a,b)] 범위에서
<u>두 수 모두의 약수가 되는</u> 값들의 최댓값을 구하자!

#나머지가 0이다.

ı 최대공약수

“0이 아닌 두 정수나 다항식의 공통되는 약수 중에서 가장 큰 수를 말한다.”

[1, min(a,b)] 범위에서
두 수 모두의 약수가 되는 값들의 최댓값을 구하자!

#나머지가 0이다.

위 내용을 코드로 작성해봅시다!

[1, min(a,b)] 범위에서 두 수 모두의 약수가 되는 값들의 최댓값을 구하자!

1) 두 수 a,b중 작은 수를 선택한다.
2) 작은 수에서 1씩 빼면서 둘 다 나누어 떨어지는 수가 있는지 확인한다.
   만약 있다면 그 수가 최대공약수이다.

위 내용을 코드로 작성해봅시다!

ı 최대공약수

```
int gcd_num = 1;
int min_num = min(a,b);
for(int i = min_num; i > 0; i--)
{
    if(a%i == 0 && b%i == 0)
        {
            gcd_num = i;
            break;
        }
}
```

코드를 다음과 같이 작성할 수 있습니다.
다음 알고리즘의 <u>시간복잡도</u>는 어떻게 될까요?
그리고 <u>최악의 경우</u>는 어떻게 될까요?

ı 최대공약수

```
int gcd_num = 1;
int min_num = min(a,b);
for(int i = min_num; i > 0; i--)
{
    if(a%i == 0 && b%i == 0)
        {
            gcd_num = i;
            break;
        }
}
```

최악의 경우는 두 수가 서로소일 때
즉, 두 수의 공약수가 1일 때 입니다.

따라서 a와 b가 엄청 큰 소수라면
과연 원하는 시간 안에 정답을 구할 수 있을까요?

최대공약수

```
int gcd_num = 1;
int min_num = min(a,b);
for(int i = min_num; i > 0; i--)
{
    if(a%i == 0 && b%i == 0)
        {
            gcd_num = i;
            break;
        }
}
```

보다 효율적인 알고리즘은 없을까요?

ㅣ 유클리드 호제법

Euclidean Algorithm

#인류 최초의 알고리즘

위 알고리즘을 이용하면 두 수의 최대공약수를
빠르게 구할 수 있습니다.

| 유클리드 호제법

“2개의 자연수 a,b에 대해서
a를 b로 나눈 나머지를 r이라 하면 (단, a>b)
a와 b의 최대공약수는 b와 r의 최대공약수와 같다.”

알고리즘은 위와 같습니다!

ㅣ 유클리드 호제법

“2개의 자연수 a,b에 대해서
a를 b로 나눈 나머지를 r이라 하면 (단, a〉b)
a와 b의 최대공약수는 b와 r의 최대공약수와 같다.”

<u>좀 더 쉽게 풀어봅시다!</u>

ı 유클리드 호제법

"gcd(a,b) = gcd(b,r)이며, (a 〉 b)
r을 구하는 과정을 반복하여 r이 0이 되었을 때

그 때의 b가 최대 공약수이다."

78696과 19332의 최대공약수를

유클리드 호제법을 이용하여 구해봅시다.

78696과 19332의 최대공약수를
유클리드 호제법을 이용하여 구해봅시다.

1) gcd(78696, 19332) = gcd(19332, 1368)
2) gcd(19332, 1368) = gcd(1368, 180)
3) gcd(1368, 180) = gcd(180, 108)
4) gcd(180, 108) = gcd(108, 72)
5) gcd(108, 72) = gcd(72, 36)
6) gcd(72, 36) = gcd(36, 0)

78696과 19332의 최대공약수를
유클리드 호제법을 이용하여 구해봅시다.

1) gcd(78696, 19332) = gcd(19332, 1368)
2) gcd(19332, 1368) = gcd(1368, 180)
3) gcd(1368, 180) = gcd(180, 108)
4) gcd(180, 108) = gcd(108, 72)
5) gcd(108, 72) = gcd(72, 36)
6) gcd(72, 36) = gcd(36, 0)

따라서 두 수의 최대공약수는 36입니다.

| 유클리드 호제법

"gcd(a,b) = gcd(b,r)이며, (a 〉 b)

r을 구하는 과정을 반복하여 r이 0이 되었을 때

1. "반복" 2. "종료조건"

그 때의 b가 최대 공약수이다."

3. "결과값"

반복, 종료조건, 결과값

위 세가지 단서들을 바탕으로 알고리즘을 구현해 봅시다!

반복, 종료조건, 결과값

재귀를 사용한 방법과
반복문을 사용한 방법이 있습니다.

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

재귀를 사용한 방법

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

반복문을 사용한 방법

ㅣ 최소공배수

"0이 아닌 두 정수나 다항식의 공통되는 배수 중에서 가장 작은 수를 말한다."

ㅣ 최소공배수

최소공배수는 어떻게 구할 수 있을까요?

ㅣ 최소공배수

최소공배수는 어떻게 구할 수 있을까요?

=〉 최대 공약수를 이용하여 구할 수 있습니다!

ㅣ 최소공배수

$$\mathrm{LCM}(A, B) = \frac{A * B}{\mathrm{GCD}(A, B)}$$

*

연습 문제

2609번 최대공약수와 최소공배수

https://www.acmicpc.net/problem/2609

앞서 배운 개념을 활용하여 문제를 풀어봅시다!

*

연습 문제

1934번 최소공배수

https://www.acmicpc.net/problem/1934

앞서 배운 개념을 활용하여 문제를 풀어봅시다!

*

연습 문제

9613번 GCD 합

https://www.acmicpc.net/problem/9613

앞서 배운 개념을 활용하여 문제를 풀어봅시다!

읽을거리

;

링크

유클리드 호제법 시간복잡도

유클리드 호제법이 앞선 방법보다 얼마나 더 빠를까요?

읽을거리

;

링크

재귀 VS 반복문

세련됨 VS 성능
정답은 없습니다.

소수

Prime Number

I 소수

“1과 그 수 이외의 자연수로는 나눌 수 없는 자연수”

어떤 자연수 N이 소수인지 판별하는 알고리즘은
어떻게 작성할 수 있을까요?

어떤 자연수 N이 소수인지 판별하는 알고리즘은
어떻게 작성할 수 있을까요?

“2 ~ N-1까지 1씩 증가시키면서
그 수가 N과 나누어 떨어지는지 확인한다.
이때, 나누어 떨어지는 수가 있으면 합성수이고,
없으면 소수이다.”

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

“2 ~ N-1까지 1씩 증가시키면서
그 수가 N과 나누어 떨어지는지 확인한다.
이때, 나누어 떨어지는 수가 있으면 합성수이고,
없으면 소수이다.”

```
bool prime(int n)
{
    if (n < 2)
    {
        return false;
    }
    for (int i = 2; i <= n - 1; i++)
    {
        if (n % i == 0)
        {
            return false;
        }
    }
    return true;
}
```

이 알고리즘을 좀 더
<u>최적화</u>할 수 있을까요?

앞선 알고리즘을 좀 더

최적화할 수 있을까요?

앞선 알고리즘을 좀 더
최적화할 수 있을까요?

1. 2를 제외한 2의 배수들은 확인할 필요가 없습니다.
2. 주어진 자연수 N이 소수이기 위한 필요충분조건은 N이 N의 제곱근보다 크지 않은 어떤 소수로 나눠지지 않습니다. 따라서 $\sqrt{N}$ 의 수까지만 검사하면 됩니다.

1~2번은 왜 그럴까요? 한번 직접 생각해봅시다!

```
bool prime (int n)
{
    if(n < 2) return false;
    if(n == 2) return true;
    if(n%2==0) return false;
    for(int i=3;i*i<=n;i+=2)
        if(n%i==0) return false;
    return true;
}
```

앞선 조건들을 추가한 알고리즘입니다.

이제 2부터 N까지의 소수를 구하는
알고리즘을 구현해봅시다.

이제 2부터 N까지의 소수를 구하는
알고리즘을 구현해봅시다.

앞서 제작한 소수 판정 알고리즘을 2~N까지
구동하여 풀 수 있을까요?
좀 더 효율적인 방법은 없을까요?

이제 2부터 N까지의 소수를 구하는
알고리즘을 구현해봅시다.

앞서 제작한 소수 판정 알고리즘을 2~N까지
구동하여 풀 수 있을까요?
좀 더 효율적인 방법은 없을까요?
어느 점을 개선할 수 있을까요?

이제 2부터 N까지의 소수를 구하는
알고리즘을 구현해봅시다.

“소수의 배수는 그 소수를 제외하고 소수가 아니다”

이제 2부터 N까지의 소수를 구하는
알고리즘을 구현해봅시다.

"소수의 배수는 그 소수를 제외하고 소수가 아니다"

=> 에라토스테네스의 체

의 기본 아이디어!

에라토스테네스의 체

에라토스테네스의 체

2 ~ N까지의 모든 소수를 구하려면?

1. 2부터 N까지의 모든 수를 써 놓는다.
2. 아직 지워지지 않은 수에서 가장 작은 수를 찾는다.
3. 그 수는 소수이다.
4. 그 수의 배수는 소수가 아니므로 지운다.

읽을거리

;

링크

에라토스테네스의 체

알고리즘 동작 과정을 영상을 통해 이해해봅시다.

에라토스테네스의 체

직접 구현해봅시다!

```cpp
int prime[100];  // 소수 저장
int pn = 0;      // 소수의 개수
bool check[101]; // 소수가 아니면 true
int n = 100;     // 100까지 소수
for (int i = 2; i <= n; i++)
{
    if (check[i] == false)
    {
        prime[pn++] = i;
        for (int j = i * i; j <= n; j += i)
        {
            check[j] = true;
        }
    }
}
```

에라토스테네스의 체는 다음과 같이
구현해 볼 수 있습니다.

The End

수고하셨습니다!