Euclidean Algorithms

▼ 유클리드 알고리즘_Euclidean Algorithms

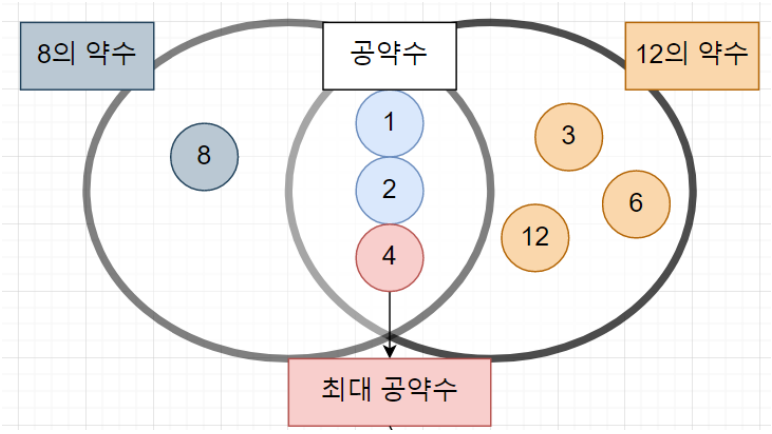
- 최대 공약수를 구하는 알고리즘

  ▼ 최대공약수는 약수들 중에서 가장 큰수를 말한다.

👉 약수
3의 약수 ⇒ 1,3
4의 약수 ⇒ 1,2,4
5의 약수 ⇒ 1,5
6의 약수 ⇒ 1,2,3,6
8의 약수 ⇒ 1,2,4,8,
12의 약수 ⇒ 1,2,3,4,6,12

👉 공약수
8과 12의 공통된 약수 ⇒ 1,2,4

최대 공약수
: 두 수의 공약수 중 최대값
8과 12의 최대 공약수 ⇒ 4



- 반복구조를 이용하는 중요한 알고리즘

▼ 최대공약수를 구하는 절차

1. 먼저 어떤 복수의 수를 소수의 곱셈 형태로 분해하자. ⇒ '소인수분해'

👉 소인수분해
8 ⇒ 1 * 2 * 2 * 2
12 ⇒ 1 * 2 * 2 * 3

2. 이들 중에 공통되는 소수를 서로 곱한 수가 바로 두 수의 최대 공약수이다.

👉 1 * 2 * 2 = 4

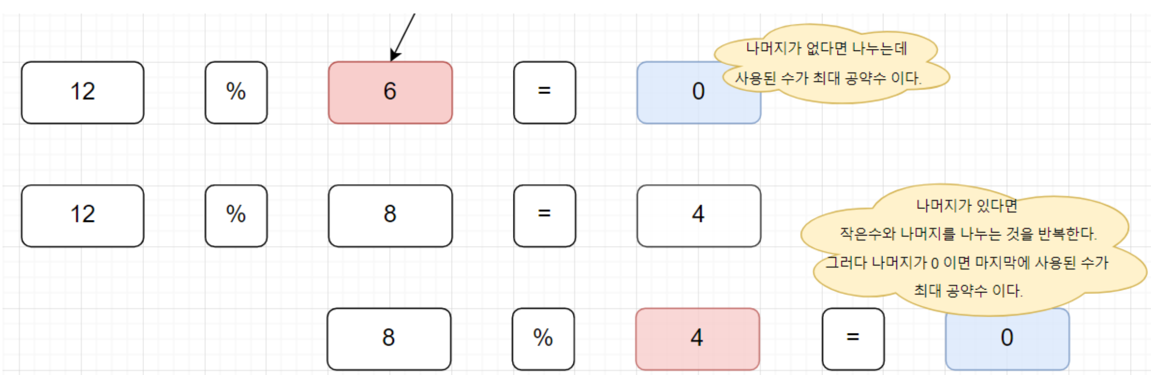
★ 그러나 이런 절차를 반복하는 것은 복잡하다.

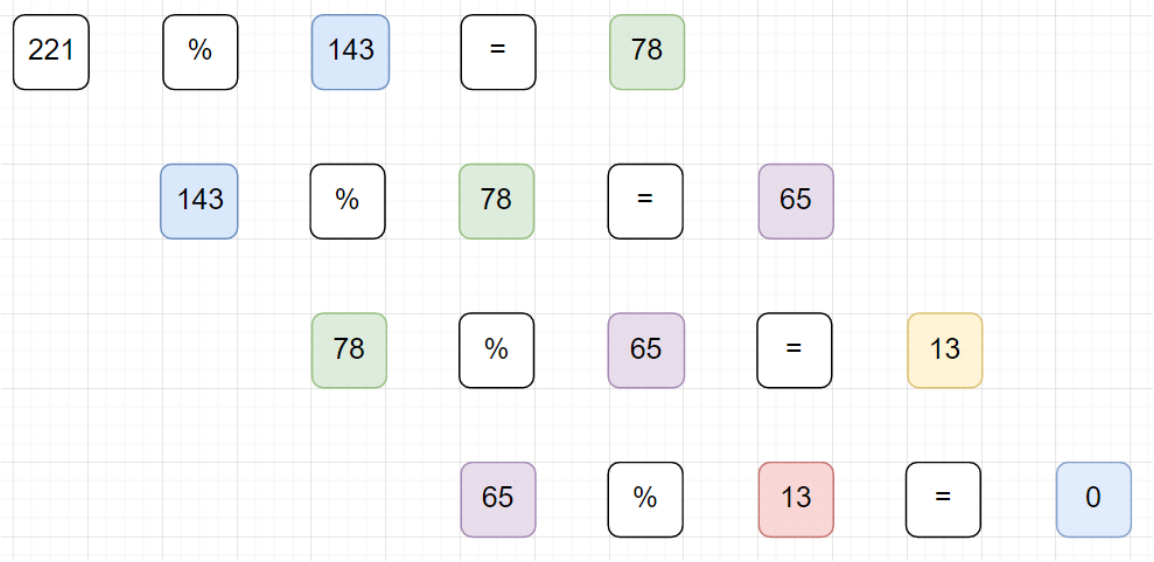
> 어떤 수를 소인수 분해하려면 먼저 그 수의 이하의 소수들을 모두 구해야 한다.
> 그리고 그 소수 중 작은 숫자부터 순서대로 원래의 수를 나누고, 나누어지지 않는다면 그 다음 소수의 순서로 계속 계산을 반복해야 한다.
> 따라서 단순해 보이지만 절차는 상당히 복잡해진다.
> 이러한 복잡성에 비해 매우 간단한 방법으로 최대 공약수를 구하는 것이
> 바로 '유클리드 알고리즘'이다.

▼ Algorithm

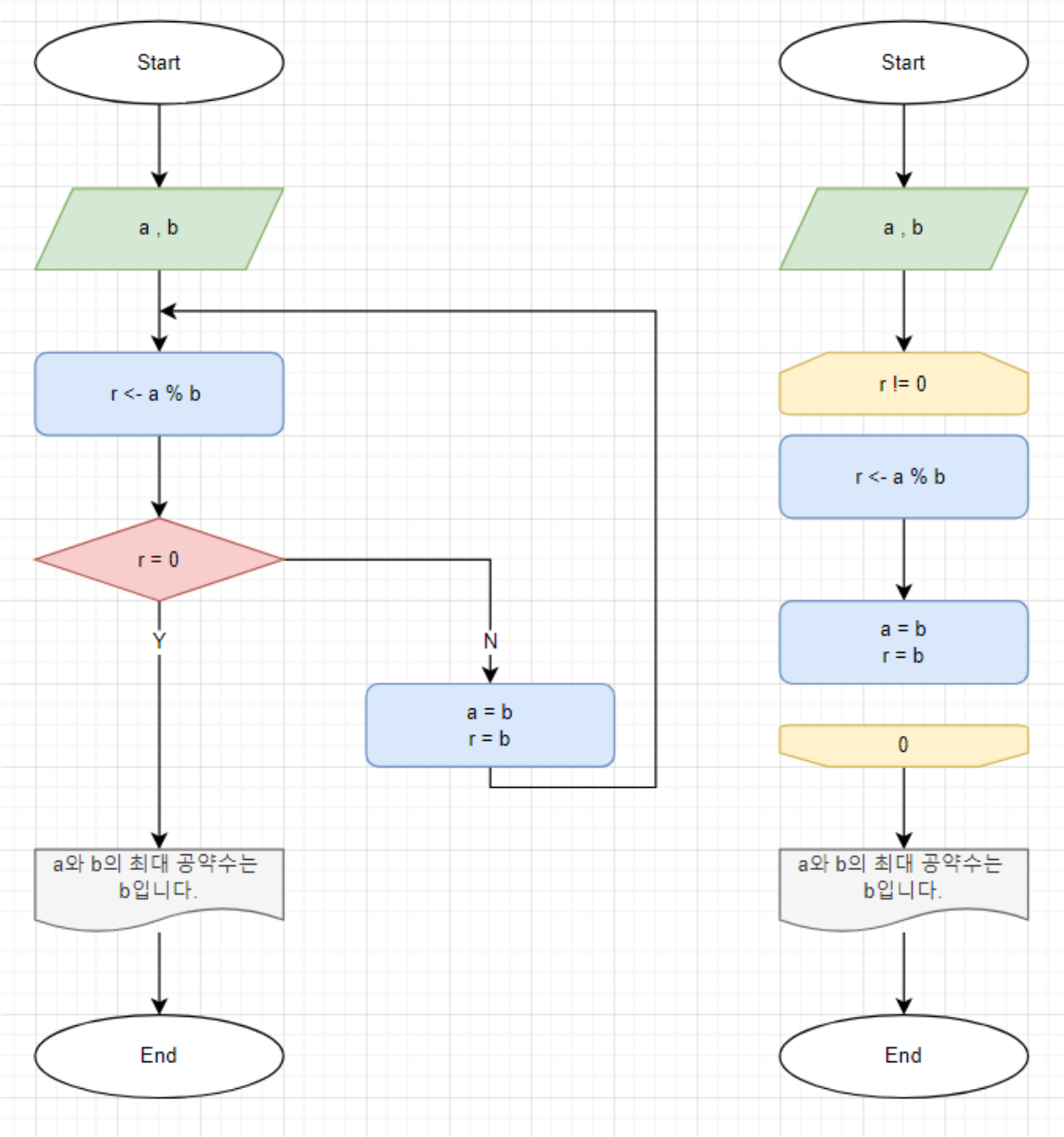
약 2300년 전의 고대 그리스의 수학자로 수많은 수학적 이론을 생각해냈다. 그 중 하나가 바로 유클리드 알고리즘이다. 간단하게 말하면 두 수의 나눗셈을 반복하여 최대공약수를 구하는 것이다.
그러면 어떻게 나눗셈을 반복할까?

> 큰수를 작은 수로 나눈다_(%)
> 나머지를 확인
> Y - 나머지가 있다.
> | 나머지가 없을 때까지 작은수와 나머지를 나눈다_(%)
> -------------------------------------
> N - 나머지가 없다.
> 그 때의 나누는데 사용된 작은 수가 바로 '최대 공약수'이다
> ex) 12로 6을 나누면 나머지는 0, 따라서 최대 공약수 6





▼ Flow chart



▼ Java Code

```
import java.util.*;
public class EuclideanAlgorithms {

  public static void main(String[] args) {
    System.out.print("a를 입력하세요: ");
    Scanner no = new Scanner(System.in);
    int a = no.nextInt();
    System.out.print("b를 입력하세요: ");
    int b = no.nextInt();
    int r = 0;
```

```
    do{
      r = a % b;
      a = b;
      b = r;
    } while (r !=0);
    System.out.printf("a와 b의 최대공약수는 %d입니다.", a);
  }
}
Outputs : ex)a: 221, b: 143
a를 입력하세요: 221
b를 입력하세요: 143
a와 b의 최대공약수는 13입니다.
```