

Euclidean Algorithms

🕒 작성일시	@2022년 8월 4일 오전 9:32
📌 강의 번호	
📅 Date	@2022년 8월 4일
📎 자료	
☑ 복습	<input type="checkbox"/>

유클리드 알고리즘 Euclidean Algorithms

- 최대 공약수를 구하는 알고리즘
- 최대 공약수는 약수들 중에서 가장 큰수를 말한다.

반복 구조를 이용하는 중요한 알고리즘

최대 공약수

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 공약수는 4이다.

최대 공약수를 구하는 절차

먼저 어떤 복수의 수를 소수의 곱셈 형태로 분해하자. → 소인수 분해

$$8 = 1 * 2 * 2 * 2$$

$$12 = 1 * 2 * 2 * 3$$

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

$$1 * 2 * 2 = 4$$

그러나 이런 절차를 반복하는 것은 상당히 복잡하다.

어떤 수를 소인수 분해하려면 먼저 그 수 이하의 소수들을 모두 구해야 한다.

그리고 그 소수 중 작은 숫자 부터 순서대로 원래의 수를 나누고, 나누어지지 않으면 그 다음 소수의 순서로 계속 계산을 반복해야한다.

따라서 단순해 보이지만 절차는 상당히 복잡해진다.

이러한 복잡성에 비해 매우 간단한 방법으로 최대 공약수를 구하는 것이 바로 '유클리드 알고리즘'이다.

이러한 복잡성에 비해 매우 간단한 방법으로 최대 공약수를 구하는 것이

바로 '유클리드 알고리즘'이다.

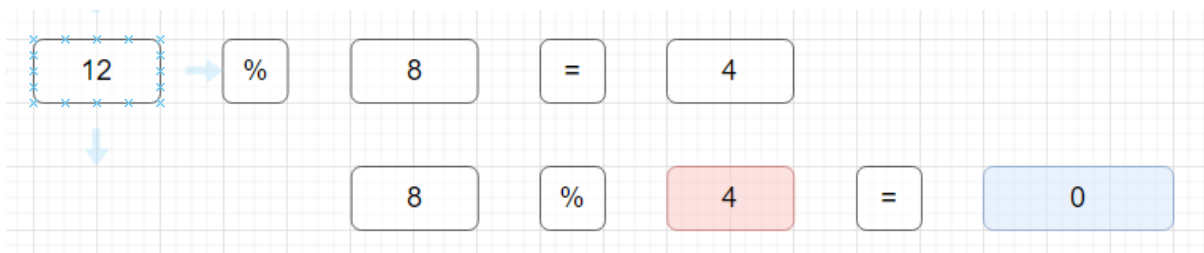
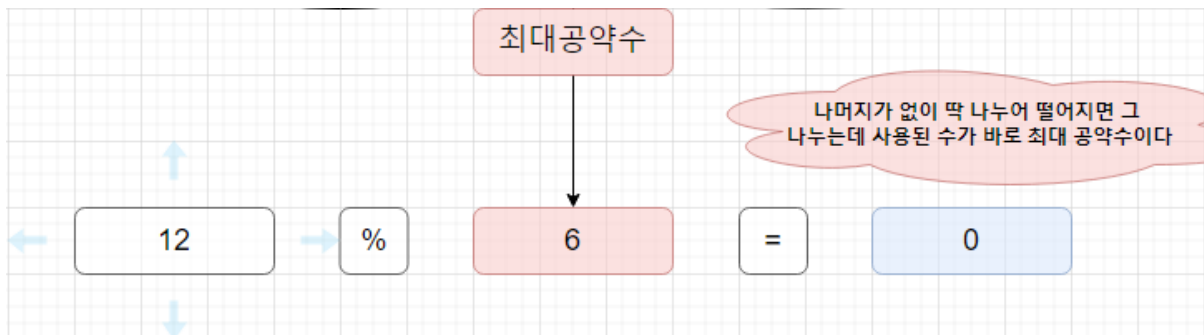
유클리드 알고리즘이란

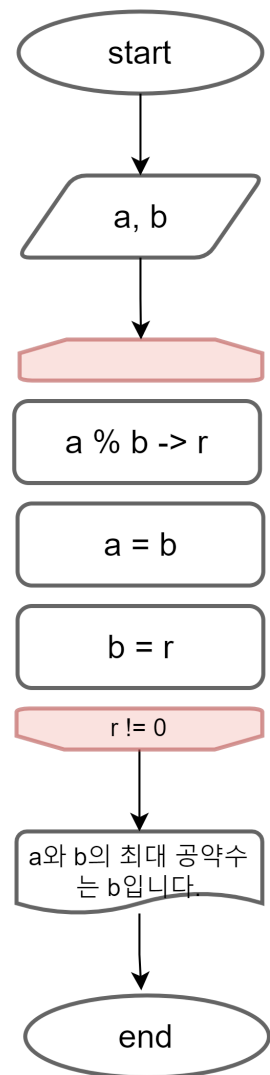
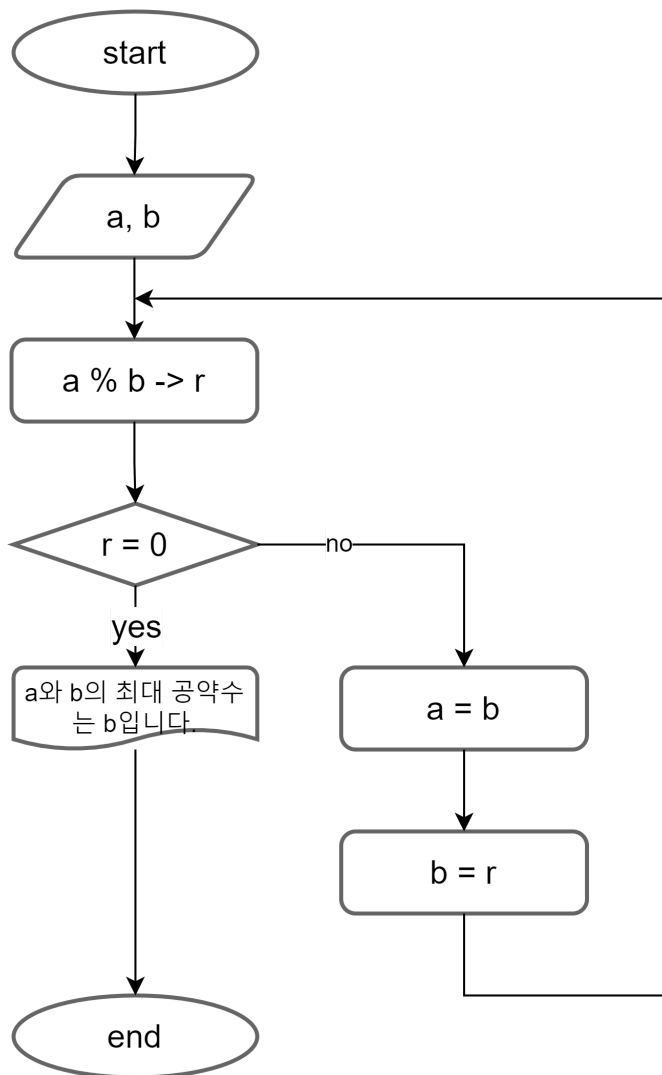
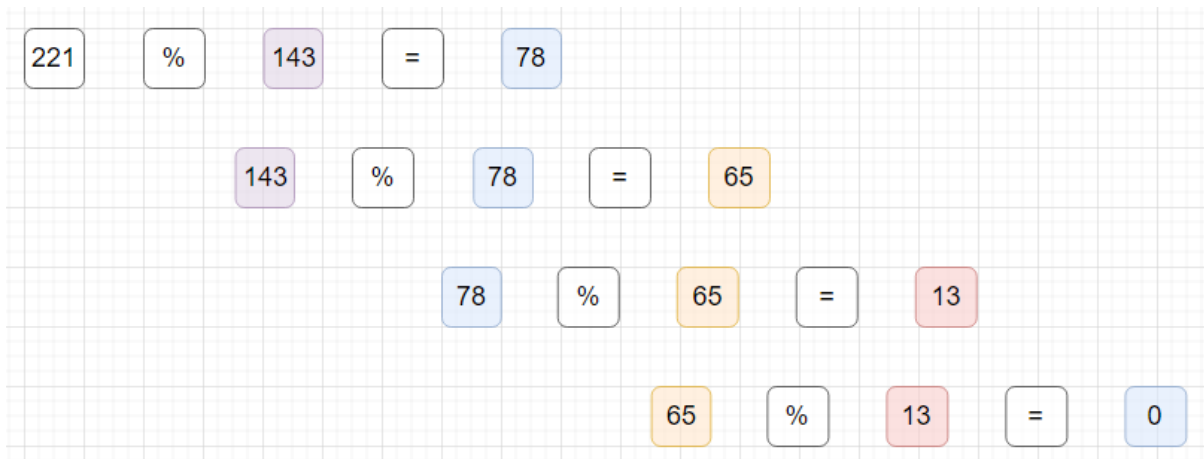
약 2300년 전의 고대 그리스의 수학자로 수많은 수학자 이론을 생각했었다. 그 중 하나가 바로 유클리드 알고리즘이다. 간단히 말하면 두 수의 나눗셈을 반복하여 최대 공약수를 구하는 것이다.

나눗셈 반복하는법

먼저 큰수를 작은 수로 나눈다. 나머지가 나오지 않게 되면 (%) 그 때의 나누는데 사용된 작은 수가 바로 최대 공약수 이다.

예를 들면 12와 6의 최대 공약수는 6이다.





```

package Euclidean;

import java.util.*;

public class Euclidean {

    public static void main(String[] args) {

        int r;
        Scanner sc = new Scanner(System.in);
        System.out.println("숫자를 입력하세요 : " + " ");
        int a = sc.nextInt();
        int b = sc.nextInt();
        r = a % b;
        while (r != 0) {
            if (r == 0) {

            } else {
                a = b;
                b = r;
            }
        }
        break;
    }
    System.out.print(a + " 와 " + b + "최대 공약수는 " + b + "입니다.");

}

}

```