유클리드 호제법

1. 개요

- 유클리드 호제법 : 자연수 A, B의 최대공약수를 G 라고 하고, A ÷ B 연산의 나머지를 R 이라고 할 때, B와 R의 최대공약수는 G 이다.

<증명>
A = aG, B=bG (a, b는 서로소)
A = B × Q + R 이므로, R = A - B × Q = G(a - bQ)

*b*와 *(a - bQ)*가 서로소 라면, B와 R의 최대공약수는 G 이다.

귀류법 : 명제의 결론을 부정하였을 때, 가정이 모순됨을 보여 간접적으로 명제가 성립함을 증명하는 방법.

귀류법 적용 : *b*와 *(a - bQ)*는 서로소가 아니다. (1이 아닌 최대공약수 p가 존재한다.)
b = mp --- ①
a - bQ = np --- ②
②에서 *a = np + bQ = np + mpQ = p(n + mQ)*

이때, a와 b는 p를 공유하고 있기 때문에, a와 b는 서로소라는 최초의 가정에 위배된다.
따라서, b와 (a - bQ)는 서로소이며, B와 R의 최대공약수는 G 이다.

2. 활용

유클리드 호제법은 최대공약수를 구할 때 사용할 수 있다.

<알고리즘>
① x와 y를 나누었을 때, 나머지가 0인가?
② 0이라면, 최대공약수는 y
③ 0이아니라면, y와 (x ÷ y의 나머지)를 나누는 것을 반복한다. (나머지가 0이 나올때까지)

3. java 코드로 구현하기

```java
static int gcd (int x, int y) {
    // x = q * y + r
    // x = y, y = r
    while (y != 0) {
        int tmp = y;
        y = x % y;
        x = tmp;
    }
    return x;
}
```

소수찾기 알고리즘

1. 개요

- 소수란? : 자신과 1 이외의 정수로는 나누어 떨어지지 않는 정수

2. 소수인지 확인하기

- 어떤 정수 n에 대하여 2부터 n-1에 대하여 2부터 n-1까지의 어떤 정수로도 나누어 떨어지지 않는다면 정수 n은 소수이다. (시간복잡도 : O(n))
- 이때, sqrt (n) 보다 큰 정수에 대하여 나누어 떨어지지 않는 것을 확인하는 것은 중복이므로, 2부터 sqrt (n)에 대하여 나누어 떨어지지 않는지 확인함으로써, n이 소수인지 판별할 수 있다. (시간복잡도 : O(log(n)))

2-2. java 코드로 구현하기

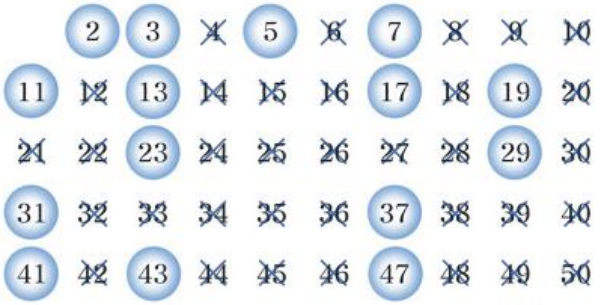
isPrime

```java
static boolean isPrime(int n) {
    for (int i = 2; i <= Math.sqrt(n); i++) {
        if (n % i == 0) return false;
    }

    return true;
}
```

3. 에라토스테네스의 체

① 2부터 시작하여 n까지의 자연수를 차례로 쓴다. (boolean 타입의 배열 선언)
② 2 이외의 2의 배수를 지운다. (n의 제곱근까지 반복)
③ 체로 걸러진 것처럼 남는 수가 소수이다.



<네이버 지식백과>

3-2. java 코드로 구현하기

nPrime

```java
static List<Integer> nPrime(int n) {
    List<Integer> list = new ArrayList<>();
    boolean[] arr = new boolean[n+1];
    Arrays.fill(arr, true);

    for (int i = 2; i <= Math.sqrt(n); i++) {
        if (arr[i]) {
            for (int j = 2; i * j <= n; j++) {
                arr[i * j] = false; // i * j is not prime number
            }
        }
    }

    for (int i = 2; i <= n; i++) {
        if (arr[i]) list.add(i);
    }

    return list;
}
```

팩토리얼 알고리즘

1. 개요

- n 팩토리얼 (n!) :1부터 어떤 양의 정수 n까지의 정수를 모두 곱한 것

2. 재귀적 방법을 이용

- 점화식 : An = A(n-1) × n (종료조건 : A1 = 1)

① factorial (n) 내부에서 factorial (n-1) 반환한 결과값과 n을 곱한 결과를 반환한다.

② n = 1일 때, 1을 반환한다.

2-1. java 코드로 구현하기

```java
static int factorial (int n) {
    if (n == 1) return 1;
    return factorial (n-1) * n;
}
```

3. 비재귀적 방법을 이용

- 반복문 활용

3-1. java 코드로 구현하기

```java
static int factorial (int n) {
    int result = 1;

    for (int i = 2; i <= n; i++) {
        result *= i;
    }
    return result;
}
```