Algorithm 08 GCD (최대 공약수)

```java
public static ArrayList<Integer[]> divideChocolateStick(int M, int N) {
    // TODO:
    ArrayList<Integer[]> result = new ArrayList<>();
    // 최대공약수 구하기
    int GCD = gcd(M, N);
    // 약수는 대칭적이므로 제곱근까지 반복한다.
    int sqrt = (int) Math.floor(Math.sqrt(GCD));

    for (int left = 1; left <= sqrt; left++) {
        if (GCD % left == 0) {
            // 최대공약수의 약수인 경우 중에 제곱근 보다 작은 약수의 경우
            result.add(new Integer[]{left, M / left, N / left});
            if(left * left < GCD) {
                // 제곱근이 아닌 경우(제곱근 보다 작은 경우)
                int right = GCD / left;      // 최대 공약수를 제곱근이 아닌 수로 나누면 제곱근 보다 큰 약수를 구할 수 있다.
                result.add(new Integer[]{right, M / right, N / right});
            }
        }
    }
    // 빼빼로를 받게 되는 직원의 수를 기준으로 오름차순으로 정렬한다.
    Kim-Jihyun1
    Collections.sort(result, new Comparator<Integer[]>() {
        Kim-Jihyun1
        @Override
        public int compare(Integer[] o1, Integer[] o2) { return o1[0].compareTo(o2[0]); }
    });
    return result;
}
// 최대공약수(유클리드 호제법: Euclidean algorithm)
2 usages  Kim-Jihyun1
public static int gcd(int m, int n) {
    if (m % n == 0) return n;
    return gcd(n, n: m % n);
}
```

int M = 4 (1, 2, 4)
int N = 8 (1, 2, 4, 8) → 최대공약수 : 4

→ 최대공약수의 약수인지 판단

첫 번째 배열의 0번째 인덱스와
두 번째 배열의 0번째 인덱스와
비교해 오름차순으로 정렬

② ①

```java
for (int left = 1; left <= GCD; left++) {
    if (GCD % left == 0) {
        // 최대공약수의 약수인 경우 중에 제곱근 보다 작은 약수의 경우
        result.add(new Integer[]{left, M / left, N / left});
    }
}
return result;
```

int M = 4 (1, 2, 4)
int N = 8 (1, 2, 4, 8) → 최대공약수 : 4

1. 최대공약수의 약수일 때마다 나누는 경우

GCD = 4

left	M / left	N / left
1	4	8
2	2	4
3 (F) → 나눌 수 없는 경우		
4	1	2

$4 = 2^2$
$9 = 3^2$
$16 = 4^2$

2. 제곱근을 구하는 경우

Sqrt = 2 → 4의 제곱근

	left	M / left	N / left
1.	1	4	8
3.	2	2	4

right	M / right	N / right
4	1	2

2.