재귀(Recursive)
메모화/분기조작

정의
- 자기 자신을 호출한다고 이해하기 보다는 자기 자신과 똑같은 메서드를 호출한다고 이해하는게 바람직하다. 재귀는 직접 재귀와 간접 재귀로 나뉘는데, 메서드 내부에서 메서드를 호출하는 것이 직접 재귀이고, 이와 달리 메서드 A가 메서드 B를 호출하고 다시 메서드 B가 메서드 A를 호출하는 구조로 이루어진것이 간접 재귀이다.

- 재귀 알로기즘에 알맞은 경우는 '풀어야 할 문제' '계산할 메서드' '처리할 자료구조'가 있다.

유클리드 호제법
- 유클리드 호제법은 최대 공약수 (GCD)를 구하는 알고리즘으로 재귀를 사용하여 최대 공약수를 구한다.

```java
// 유클리드 호제법으로 최대공약수를 구함
import java.util.Scanner;

class EuclidGCD {
    //--- 정수 x, y의 최대공약수를 구하여 반환 ---//
    static int gcd(int x, int y) {
        if (y == 0)
            return x;
        else
            return gcd(y,  y: x % y);
    }

    public static void main(String[] args) {
        Scanner stdIn = new Scanner(System.in);

        System.out.println("두 정수의 최대공약수를 구합니다.");

        System.out.print("정수를 입력하세요 : ");  int x = stdIn.nextInt();
        System.out.print("정수를 입력하세요 : ");  int y = stdIn.nextInt();

        System.out.println("최대공약수는 " + gcd(x, y) + "입니다.");
    }
}
```

- 두 정수 x와 y를 0이 될 때 까지 나머지 연산을 진행하는데, 이 때 1회전 할 때마다 x와 y의 연산 위치가 바뀌는 특징을 가지고 있다.
- 배열의 모든 요소의 최대 공약수를 구하는 방법은 github를 참고하자

재귀 분석
- 재귀는 시작점을 기준으로 하향식 분석 / 상향식 분석으로 나눌 수 있는데 가장 위쪽에 위치

한 상자의 메서드를 호출하는 것부터 계단식으로 조사하는 방식을 하향식 분석 (Top-Down analysis)라고 한다.
- 이 때, 하향식 분석은 같은 메서드를 여러 번 호출 할 수 있기 때문에 메모화와 같은 기법이 사용된다.
- 상향식 분석은 아래쪽부터 쌓아 올리며 분석하는 방법으로 효과적이지만, 가지가 뻗는것을 파악하는데 어려움이 있다.

메모화
- 메모화(memoization) 기법을 사용하면 동일한 계산을 반복하지않아 보다 효율적으로 재귀를 사용 할 수 있게 해준다.

```java
// 재귀 함수를 메모화로 구현
import java.util.Scanner;
class RecurMemo {
    static String[] memo;

    //--- 메모화를 도입한 메서드 recur ---//
    static void recur(int n) {
        if (memo[n + 1] != null)
            System.out.print(memo[n + 1]);  // 메모를 출력
        else {
            if (n > 0) {
                recur( n: n - 1);
                System.out.println(n);
                recur( n: n - 2);
                memo[n + 1] = memo[n] + n + "\n" + memo[n - 1]; // 메모화
            } else {
                memo[n + 1] = ""; // 메모화 : recur(0)과 recur(-1)은 빈 문자열
            }
        }
    }
    public static void main(String[] args) {
        Scanner stdIn = new Scanner(System.in);

        System.out.print("정수를 입력하세요 : ");
        int x = stdIn.nextInt();

        memo = new String[x + 2];
        recur(x);
    }
}
```

- 일반적으로 메모화는 boolean 배열을 사용하거나 int 배열을 사용해 true/false 혹은 1 / 0 으로 체크해 사용 된다.

분기 조작
- 1차원 배열 혹은 정수로 재귀 알고리즘을 사용 할 때는 단순히 이미 계산이 된 영역인지 아

닌지를 판단하며 진행 했지만 2차원 배열 이상의 문제를 해결하기 위해선 메모화를 통한 분기 조작과 분기 한정법이 필요하다. 대표적인 문제로 퀸이 있는데, 행, 열 뿐만 아니라 우측 대각, 좌측 대각까지 체크해야할 필요가 있기 때문에 고난이도의 기술이 필요한 문제이다.