1003 - 피보나치 함수

난이도	실버3

문제

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.25 초 (추가 시간 없음)	128 MB	143020	39664	30982	30.839%

문제

다음 소스는 N번째 피보나치 수를 구하는 C++ 함수이다.

```
int fibonacci(int n) {
    if (n == 0) {
        printf("0");
        return 0;
    } else if (n == 1) {
        printf("1");
        return 1;
    } else {
        return fibonacci(n-1) + fibonacci(n-2);
    }
}
```

`fibonacci(3)` 을 호출하면 다음과 같은 일이 일어난다.

- `fibonacci(3)` 은 `fibonacci(2)` 와 `fibonacci(1)` (첫 번째 호출)을 호출한다.
- `fibonacci(2)` 는 `fibonacci(1)` (두 번째 호출)과 `fibonacci(0)` 을 호출한다.
- 두 번째 호출한 `fibonacci(1)` 은 1을 출력하고 1을 리턴한다.
- `fibonacci(0)` 은 0을 출력하고, 0을 리턴한다.
- `fibonacci(2)` 는 `fibonacci(1)` 과 `fibonacci(0)` 의 결과를 얻고, 1을 리턴한다.
- 첫 번째 호출한 `fibonacci(1)` 은 1을 출력하고, 1을 리턴한다.
- `fibonacci(3)` 은 `fibonacci(2)` 와 `fibonacci(1)` 의 결과를 얻고, 2를 리턴한다.

1은 2번 출력되고, 0은 1번 출력된다. N이 주어졌을 때, `fibonacci(N)` 을 호출했을 때, 0과 1이 각각 몇 번 출력되는지 구하는 프로그램을 작성하시오.

입력

첫째 줄에 테스트 케이스의 개수 T가 주어진다.

각 테스트 케이스는 한 줄로 이루어져 있고, N이 주어진다. N은 40보다 작거나 같은 자연수 또는 0이다.

출력

각 테스트 케이스마다 0이 출력되는 횟수와 1이 출력되는 횟수를 공백으로 구분해서 출력한다.

예제 입력 1 복사

```
3
0
1
3
```

예제 출력 1 복사

```
1 0
0 1
1 2
```

예제 입력 2 복사

```
2
6
22
```

예제 출력 2 복사

```
5 8
10946 17711
```

풀이

DP의 메모이제이션 기법을 활용해 풀었다.

Bottom-up 방식으로 풀이했다.

1. 구하려는 답은?

N까지의 피보나치 수열 중 0과 1이 출력되는 횟수

2. 점화식

문제에서 주어진 코드를 다시 한 번 살펴보자.

```
int fibonacci(int n) {
    if (n == 0) {
        printf("0");
        return 0;
    } else if (n == 1) {
        printf("1");
        return 1;
    } else {
        return fibonacci(n-1) + fibonacci(n-2);
    }
}
```

N = 0, N = 1인 경우만 제외하고 `fibonacci(n-1) + fibonacci(n-2)` 를 실행하고 있다. 여기서, N번째 피보나치 수에서 0과 1이 출력되는 횟수가 N-1번째와 N-2번째의 각 횟수를 더한 것임에 대한 힌트를 얻을 수 있다.

실제로 그렇게 되는지 몇 개만 실험해보자.

- fib(3) = fib(2) + fib(1) = fib(1) + fib(0) + fib(1)
- fib(4) = fib(3) + fib(2)

  = fib(1) + fib(0) + fib(1) + fib(1) + fib(0)

N번째 피보나치 수 / 횟수	0	1
0 (fib(0))	1	0
1 (fib(1))	0	1
2 (fib(1) + fib(0))	1	1
3 (fib(2) + fib(1))	1	2
4 (fib(3) + fib(2))	2	3

따라서 점화식은,

F(N) = F(N-1) + F(N-2) → 0과 1의 변수마다 각각 실행

소스코드

각 수까지의 피보나치 수열 중 0과 1이 나타나는 횟수를 저장해줄 AppearNum이라는 클래스를 따로 만들어주었다.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

class AppearNum {
    int zero;
    int one;

    AppearNum(int z, int o) {
        this.zero = z;
        this.one = o;
    }
}

public class Main {

    static int T, N;
    static AppearNum[] cache;
    public static void main(String[] args) throws Exception{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        T = Integer.parseInt(br.readLine());

        for(int i=0 ; i<T ; i++){
            N = Integer.parseInt(br.readLine());
            AppearNum tmp = dp(N);
            System.out.println(tmp.zero + " " + tmp.one);
        }
    }

    static AppearNum dp(int num) {
        if(num == 0) return new AppearNum(1,0);
        else if(num == 1) return new AppearNum(0,1);

        int zero = 0, one = 0;
        cache = new AppearNum[num+1];

        // N = 0과 N = 1인 경우는 미리 저장 -> 가장 작은 두 단위
        cache[0] = new AppearNum(1, 0);    // N = 0
        cache[1] = new AppearNum(0, 1);    // N = 1

        for(int i=2 ; i<num+1 ; i++){
            cache[i] = new AppearNum(0, 0);

            // 점화식대로 실행
            cache[i].zero = cache[i-1].zero + cache[i-2].zero;
            cache[i].one = cache[i-1].one + cache[i-2].one;
        }

        return cache[num];

    }
}
```