The type of the problem is dynamic programming. This is a new type which I have not learn and practice. To know and master a new type is a good thing for us.

This problem shows that to use a Fibonacci numbers example to illustrate the principles of dynamic programming.

First of all, I need to understand the meaning of this problem. I should know how to find a Fibonacci number using dynamic programming to solve this problem.

In this problem, I can make an array number of two dimension for storing the results. And then, to store the base cases. Class: NumberofFiboCalls. Let x[i][0] and x[i][1] be the number of times '0' and '1', respectively, will be printed during a Fibonacci(n) call.

Then
x [0][0] = 1;
x [0][1] = 0;
x [1][0] = 0;
x [1][1] = 1.
x values can be calculated by forward dynamic programming (calculate x[n] using known x[j-1] and x[j-2])

NumberofFiboCalls: Tests passed.

In this problem, the difficult thing is that to identify the main observation needed to solve in this problem. And sometimes there will be deviations in thinking. But after correctly understanding the meaning of the topic, I can solve the problem faster.

There are some challenges in understanding and thinking. And there has another method of solving this problem. For Fibonacci algorithms, the

Dynamic programming could slower than fast doubling. This is a very classic question. And through this question, I can extend the learning of other knowledge. Moreover, this exercise also uses Junit from IDEA to test.

https://community.topcoder.com/stat?c=problem_statement&pm=2292

```
import org.junit.Assert;
import org.junit.Test;

public class NumberofFiboCalls {
    public int[] fiboCallsMade(int n){
        int[][] x = new int[n+1][2];
        x[0][0] = 1;
        x[0][1] = 0;
        x[1][0] = 0;
        x[1][1] = 1;
        for (int j = 2 ; j <= n ; ++j) {
            x[j][0] = x[j-1][0] + x[j-2][0];
            x[j][1] = x[j-1][1] + x[j-2][1];
        }
        return x[n];
    }

    @Test
    public void test1() {
        NumberofFiboCalls fibo = new NumberofFiboCalls();
        int[] result = fibo.fiboCallsMade(3);
        Assert.assertArrayEquals(new int[]{1, 2}, result);
    }
```

```
	@Test
	public void test2() {
		NumberofFiboCalls fibo = new NumberofFiboCalls();
		int[] result = fibo.fiboCallsMade(6);
		Assert.assertArrayEquals(new int[]{5, 8}, result);
	}
	@Test
	public void test3() {
		NumberofFiboCalls fibo = new NumberofFiboCalls();
		int[] result = fibo.fiboCallsMade(22);
		Assert.assertArrayEquals(new int[]{10946, 17711}, result);
	}

}
```