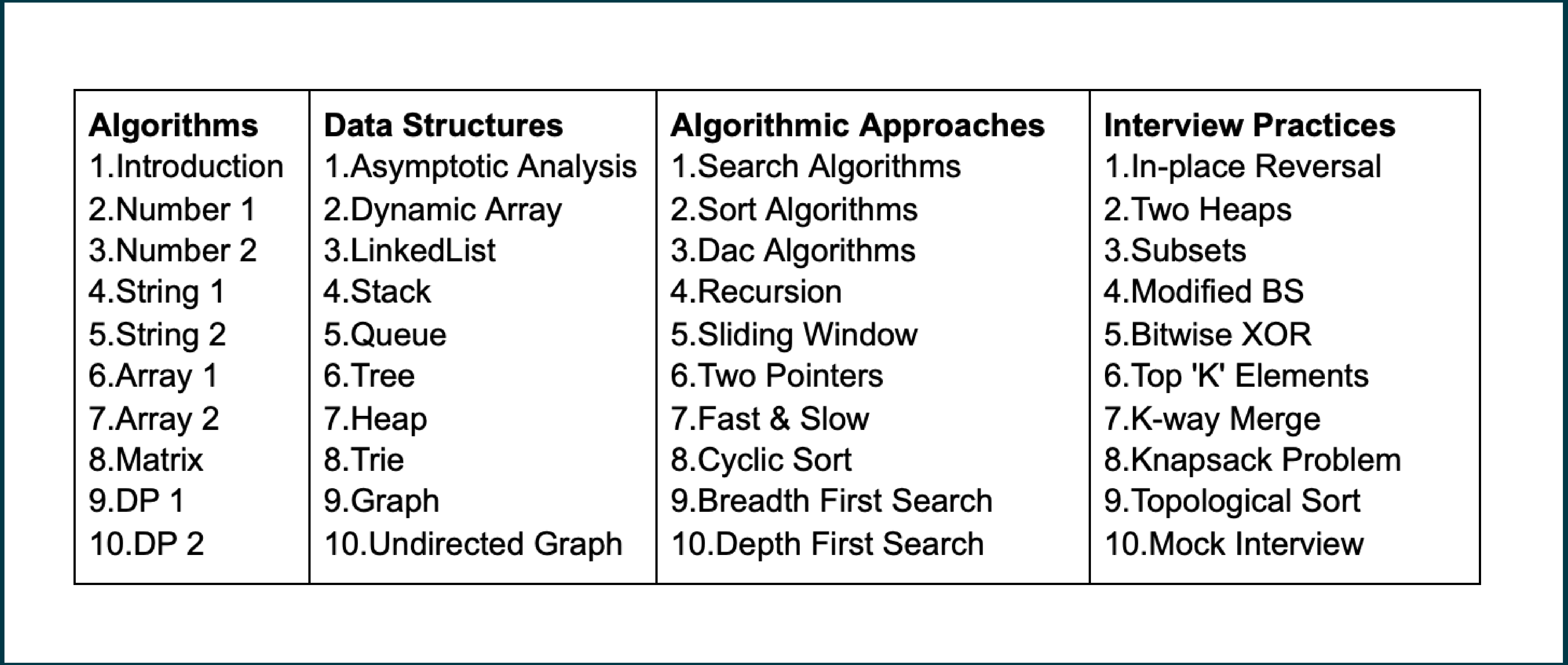
PDP ONLINE

DSA - Algorithms

Dynamic Programming

1

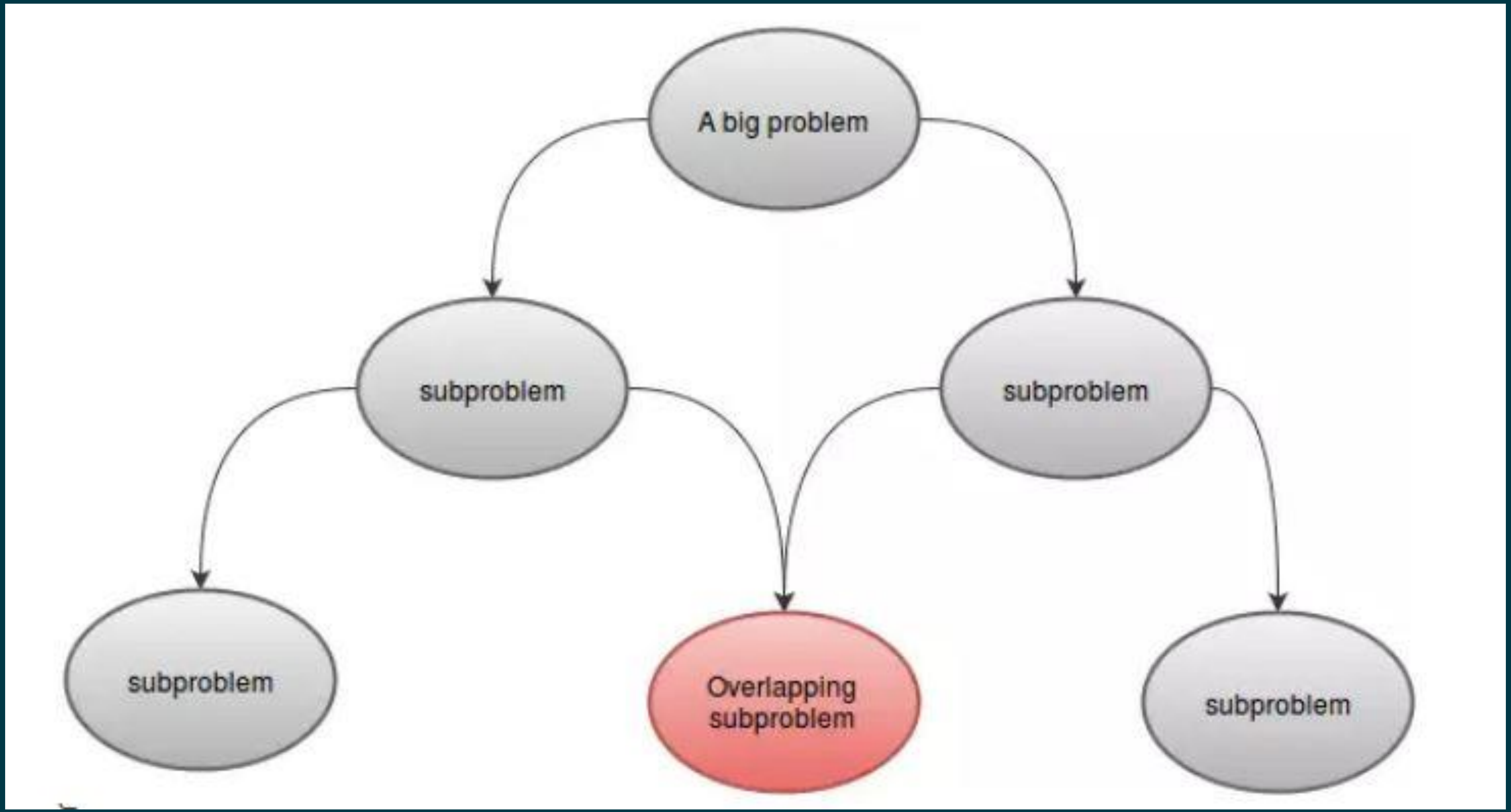
Course Planning



Algorithms	Data Structures	Algorithmic Approaches	Interview Practices
1.Introduction	1.Asymptotic Analysis	1.Search Algorithms	1.In-place Reversal
2.Number 1	2.Dynamic Array	2.Sort Algorithms	2.Two Heaps
3.Number 2	3.LinkedList	3.Dac Algorithms	3.Subsets
4.String 1	4.Stack	4.Recursion	4.Modified BS
5.String 2	5.Queue	5.Sliding Window	5.Bitwise XOR
6.Array 1	6.Tree	6.Two Pointers	6.Top 'K' Elements
7.Array 2	7.Heap	7.Fast & Slow	7.K-way Merge
8.Matrix	8.Trie	8.Cyclic Sort	8.Knapsack Problem
9.DP 1	9.Graph	9.Breadth First Search	9.Topological Sort
10.DP 2	10.Undirected Graph	10.Depth First Search	10.Mock Interview

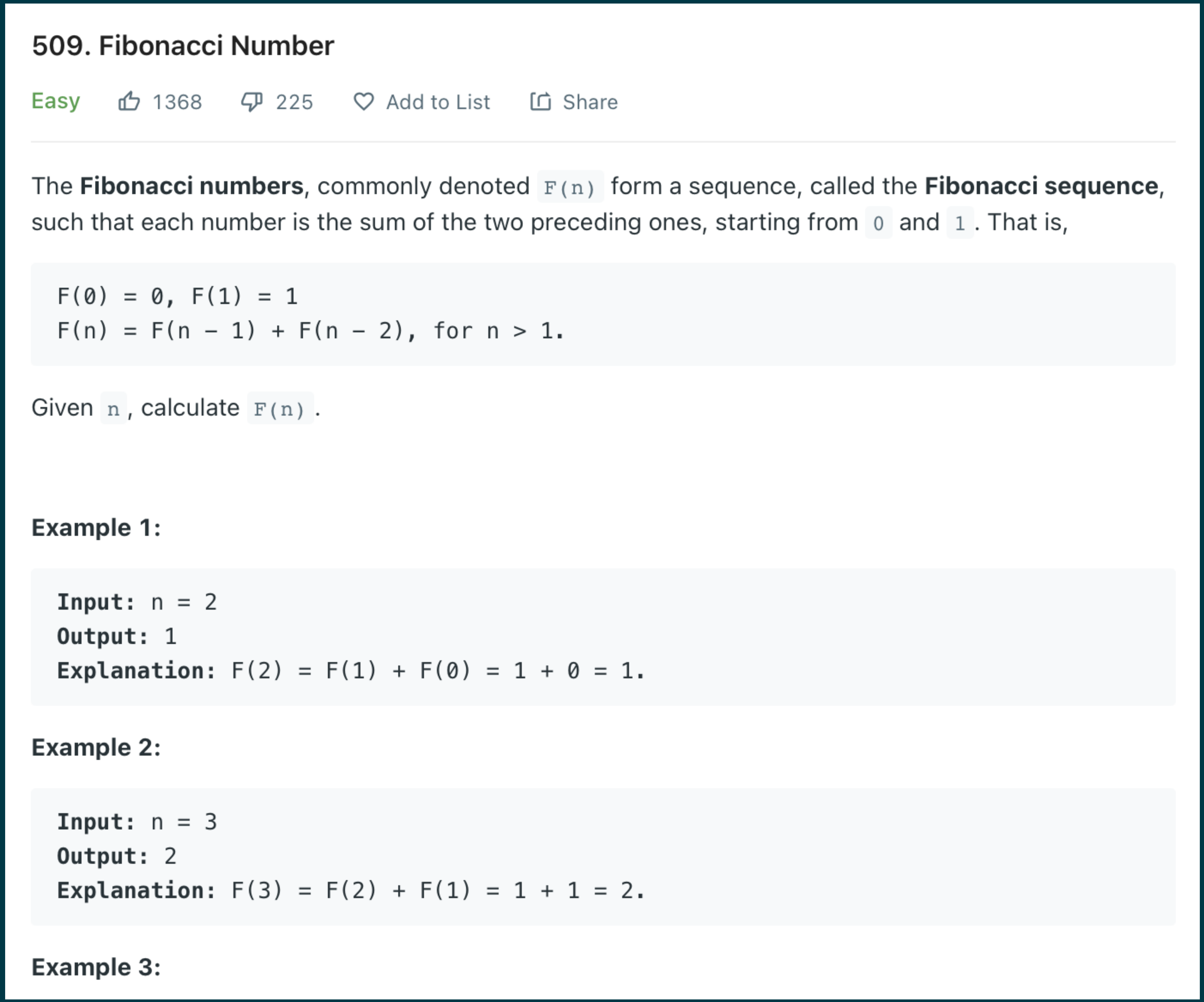
https://kurbanovxurshid.medium.com/data-structures-and-algorithms-course-plan-1ab912ec1e17

Dynamic Programming



Those who can`t remember the past are condemned to repeat it. – Dynamic Programming

Explanation



509. Fibonacci Number

The **Fibonacci numbers**, commonly denoted `F(n)` form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from `0` and `1`. That is,

```
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
```

Given `n`, calculate `F(n)`.

Example 1:

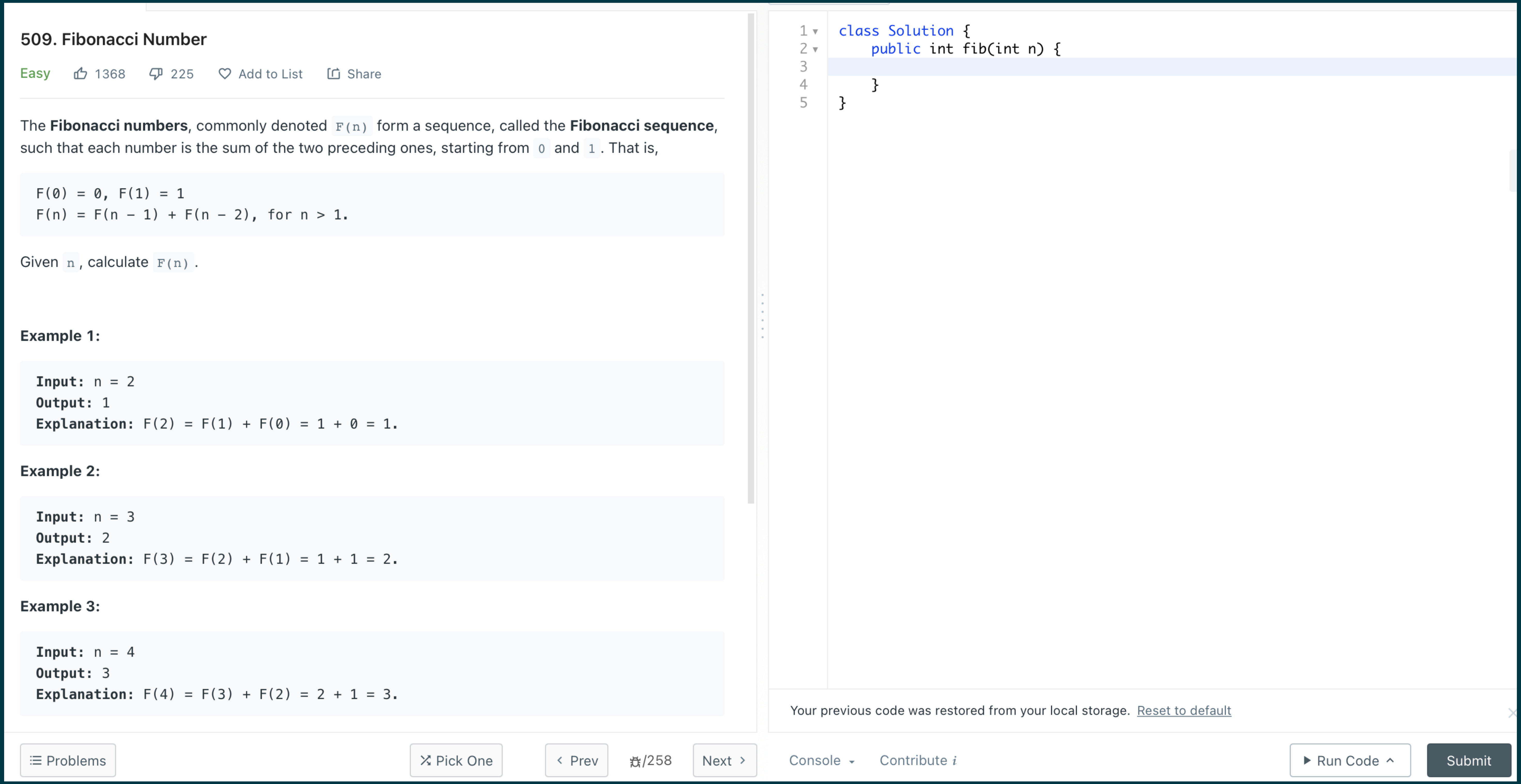
```
Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
```

Example 2:

```
Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
```

Example 3:

Fibonacci Number



509. Fibonacci Number

The **Fibonacci numbers**, commonly denoted `F(n)` form a sequence, called the **Fibonacci sequence**, such that each number is the sum of the two preceding ones, starting from `0` and `1`. That is,

```
F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.
```

Given `n`, calculate `F(n)`.

Example 1:

```
Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.
```

Example 2:

```
Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.
```

Example 3:

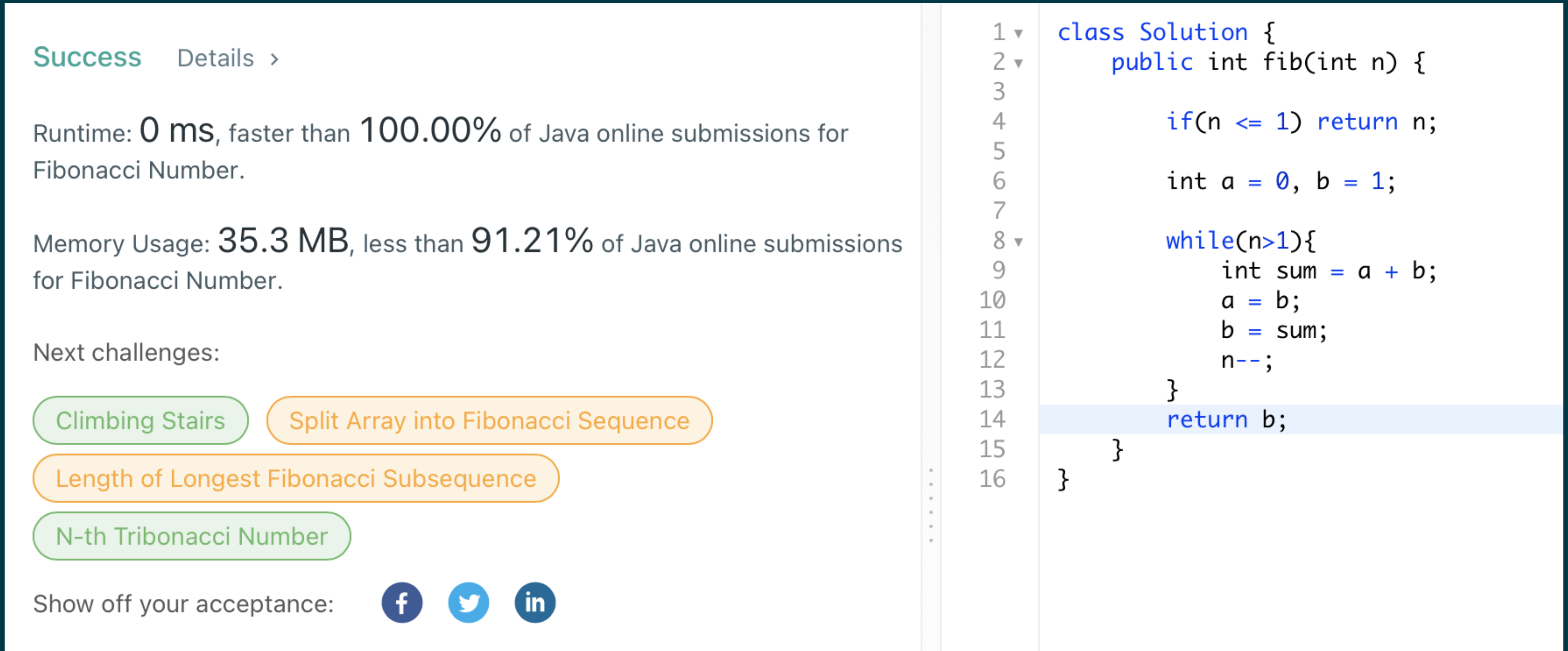
```
Input: n = 4
Output: 3
Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.
```

```java
class Solution {
    public int fib(int n) {

    }
}
```

https://leetcode.com/problems/fibonacci-number/

Iterative Method



Success Details ›

Runtime: 0 ms, faster than 100.00% of Java online submissions for Fibonacci Number.

Memory Usage: 35.3 MB, less than 91.21% of Java online submissions for Fibonacci Number.

Next challenges:

Climbing Stairs

Split Array into Fibonacci Sequence

Length of Longest Fibonacci Subsequence

N-th Tribonacci Number

Show off your acceptance:

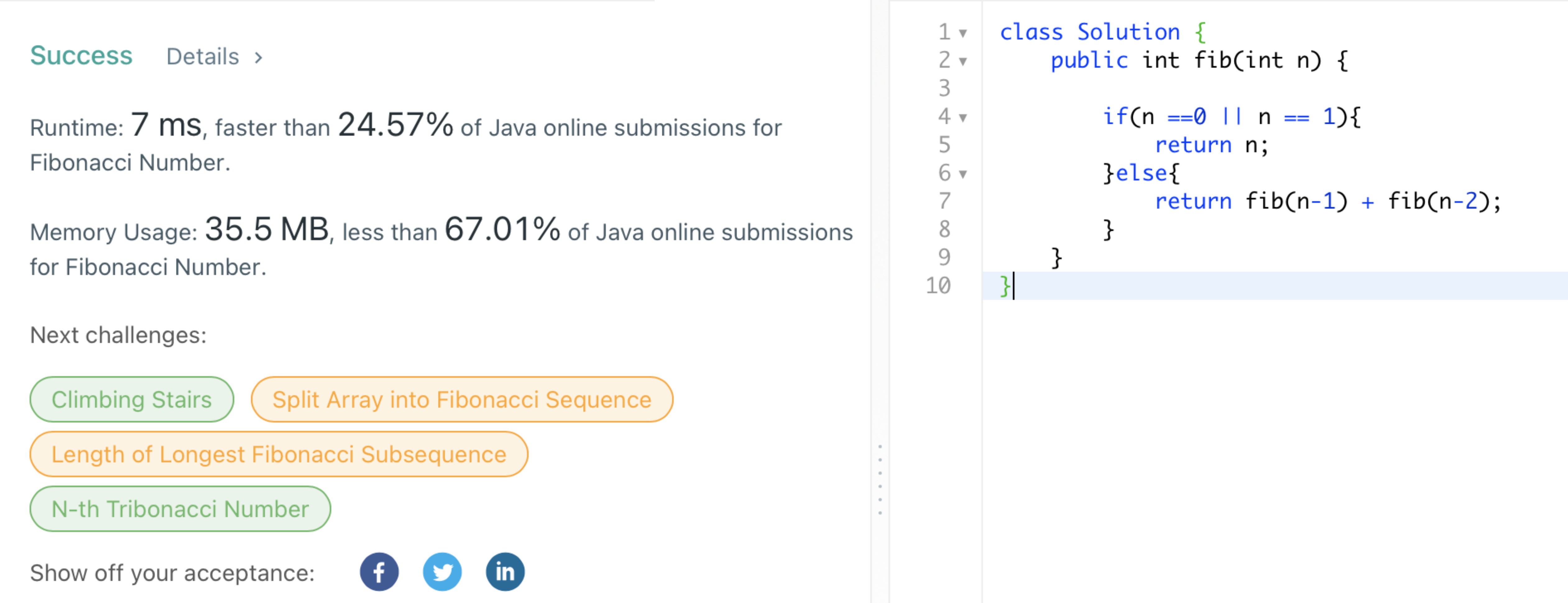
```java
class Solution {
    public int fib(int n) {

        if(n <= 1) return n;

        int a = 0, b = 1;

        while(n>1){
            int sum = a + b;
            a = b;
            b = sum;
            n--;
        }
        return b;
    }
}
```

Recursive Method



Success Details ›

Runtime: 7 ms, faster than 24.57% of Java online submissions for Fibonacci Number.

Memory Usage: 35.5 MB, less than 67.01% of Java online submissions for Fibonacci Number.

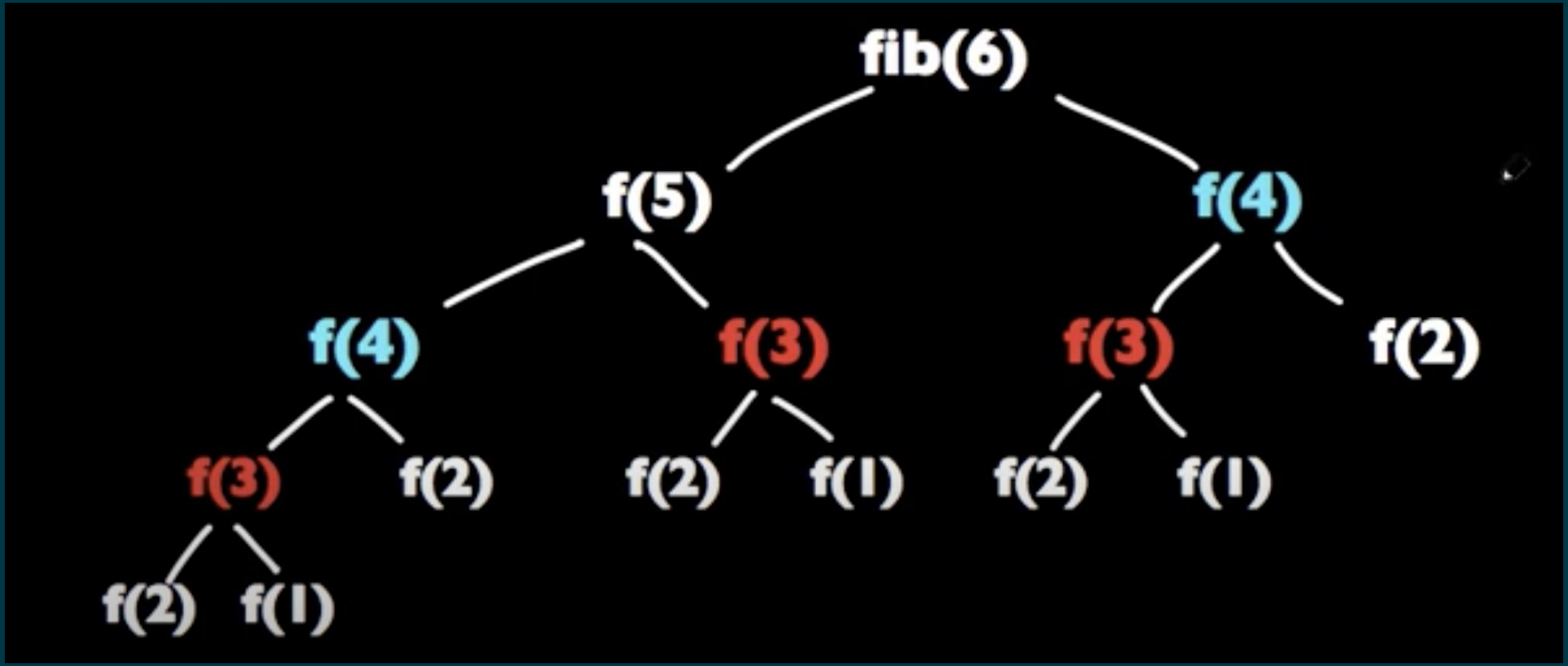
Next challenges:

Climbing Stairs
Split Array into Fibonacci Sequence
Length of Longest Fibonacci Subsequence
N-th Tribonacci Number
Show off your acceptance:

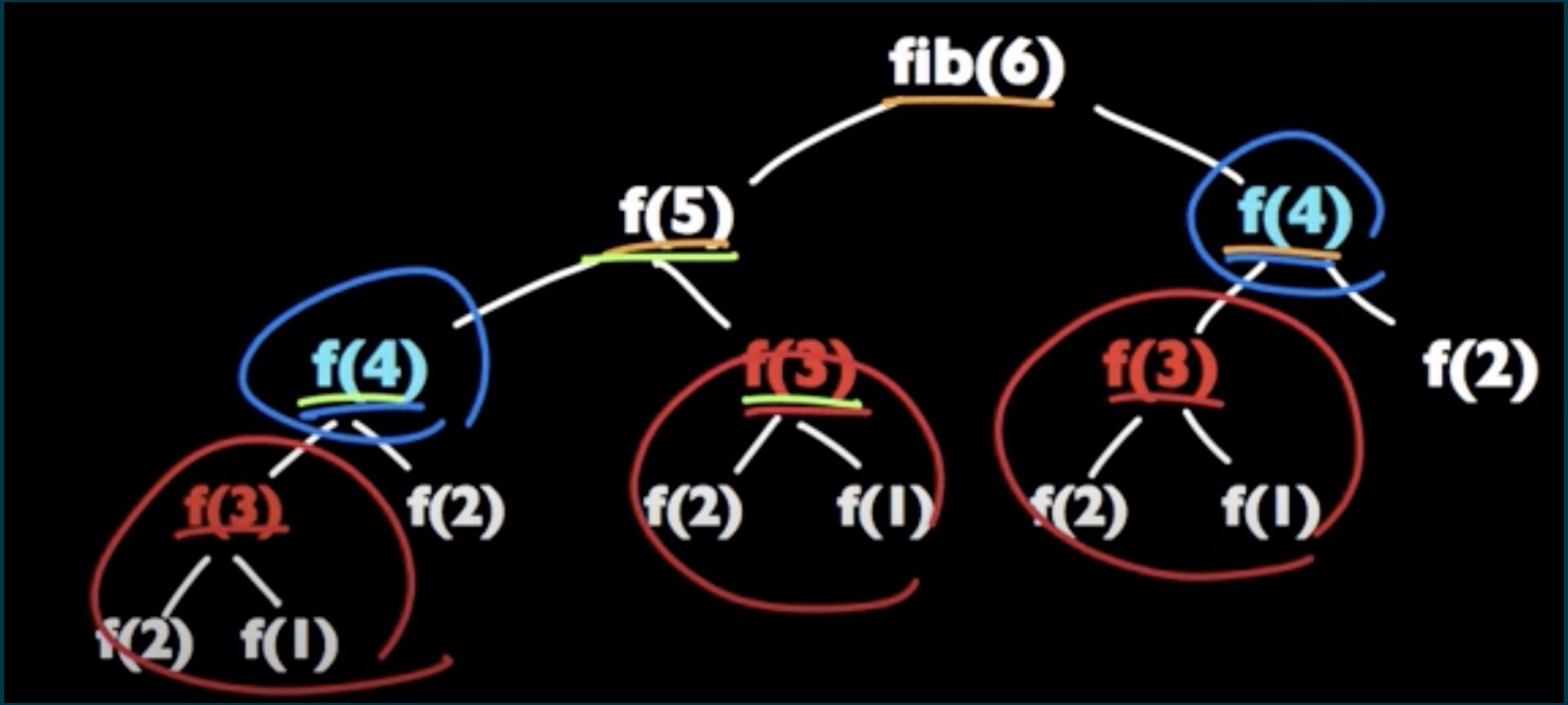
```java
class Solution {
    public int fib(int n) {

        if(n ==0 || n == 1){
            return n;
        }else{
            return fib(n-1) + fib(n-2);
        }
    }
}
```

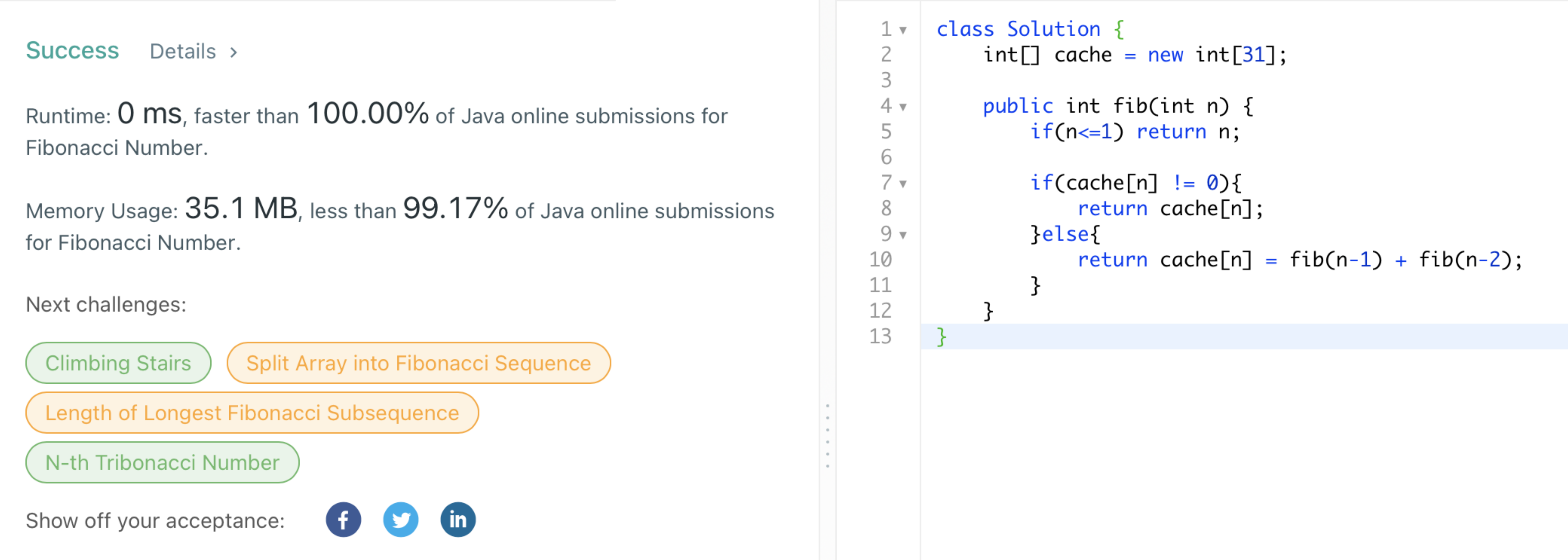
Recursive - Make it work



DP - Make it efficient



Memoization - Top Down



Runtime: 0 ms, faster than 100.00% of Java online submissions for Fibonacci Number.

Memory Usage: 35.1 MB, less than 99.17% of Java online submissions for Fibonacci Number.

Next challenges:

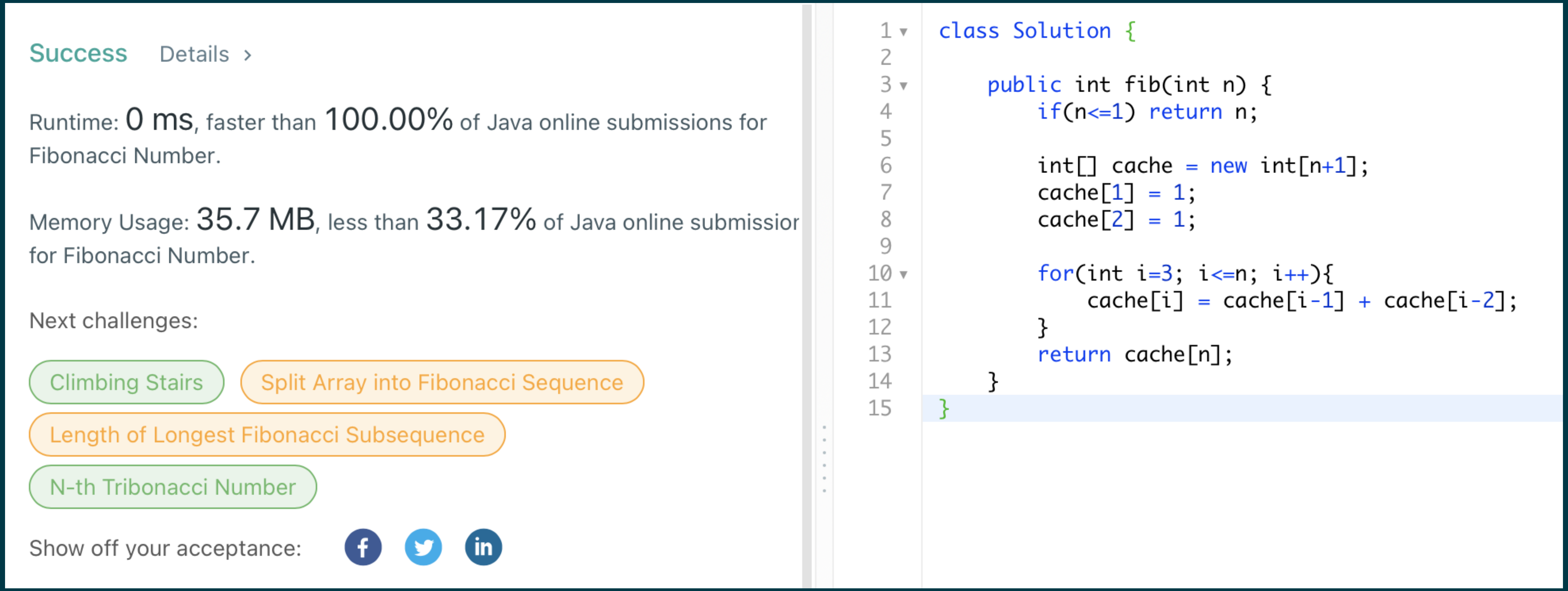
Climbing Stairs
Split Array into Fibonacci Sequence
Length of Longest Fibonacci Subsequence
N-th Tribonacci Number
Show off your acceptance:

```java
class Solution {
    int[] cache = new int[31];

    public int fib(int n) {
        if(n<=1) return n;

        if(cache[n] != 0){
            return cache[n];
        }else{
            return cache[n] = fib(n-1) + fib(n-2);
        }
    }
}
```

Tabulation - Bottom Up



Runtime: 0 ms, faster than 100.00% of Java online submissions for Fibonacci Number.

Memory Usage: 35.7 MB, less than 33.17% of Java online submissior for Fibonacci Number.

Next challenges:

Climbing Stairs
Split Array into Fibonacci Sequence
Length of Longest Fibonacci Subsequence
N-th Tribonacci Number
Show off your acceptance:

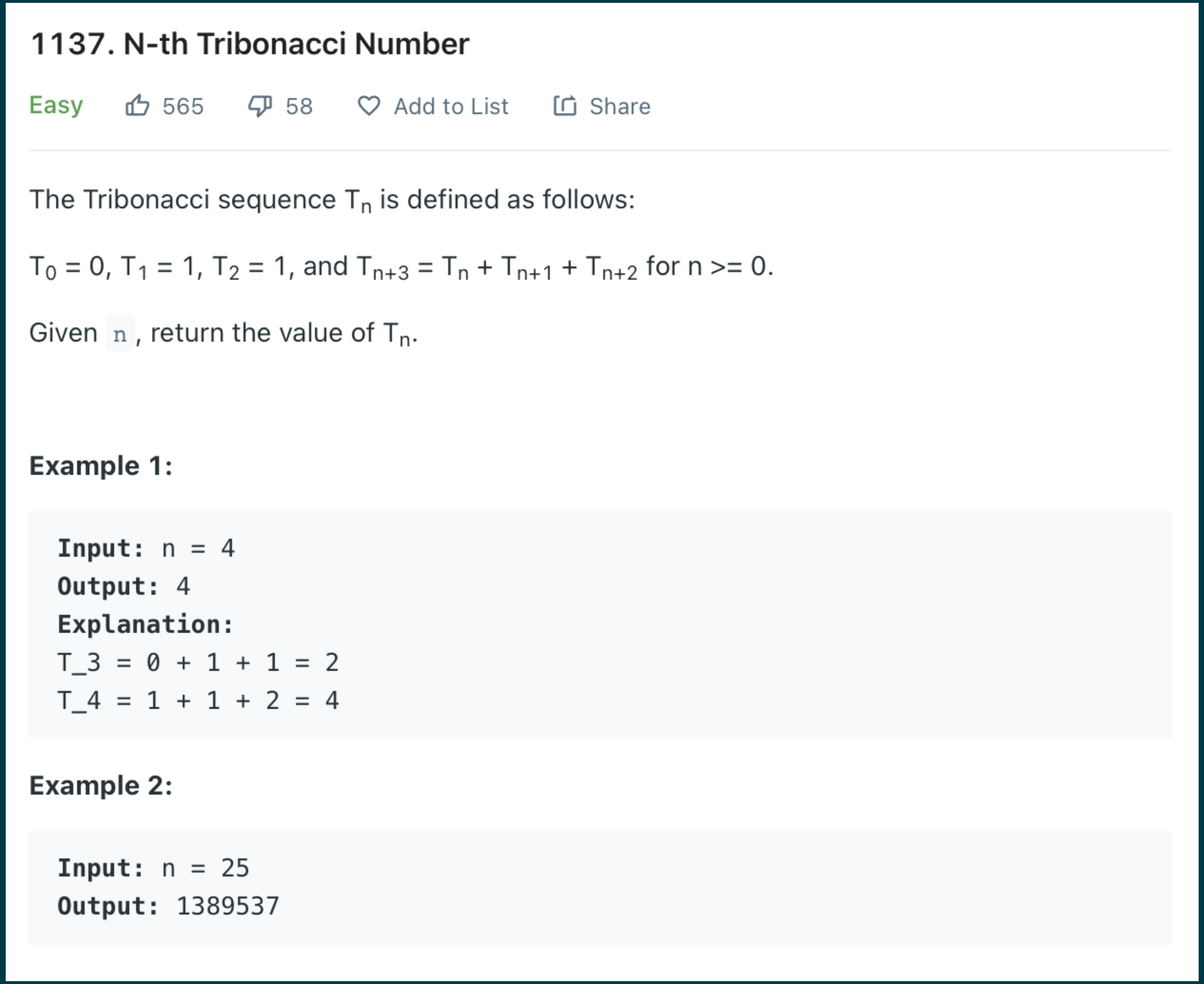
```java
class Solution {

    public int fib(int n) {
        if(n<=1) return n;

        int[] cache = new int[n+1];
        cache[1] = 1;
        cache[2] = 1;

        for(int i=3; i<=n; i++){
            cache[i] = cache[i-1] + cache[i-2];
        }
        return cache[n];
    }
}
```

Task 1 – Tribonacci Number (Iterative)



1137. N-th Tribonacci Number

The Tribonacci sequence T_n is defined as follows:

$T_0 = 0$, $T_1 = 1$, $T_2 = 1$, and $T_{n+3} = T_n + T_{n+1} + T_{n+2}$ for n >= 0.

Given `n`, return the value of T_n.

Example 1:

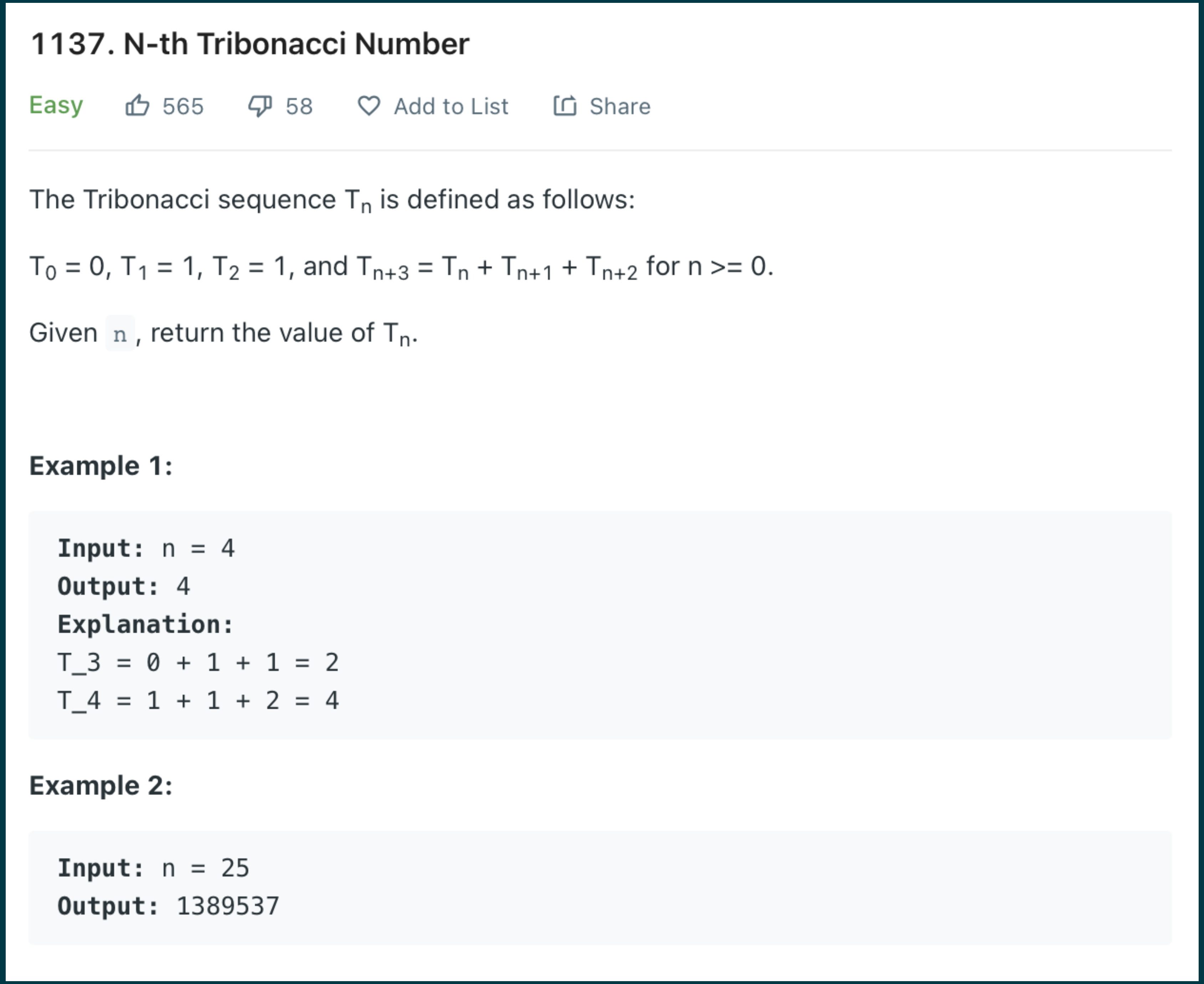
```
Input: n = 4
Output: 4
Explanation:
T_3 = 0 + 1 + 1 = 2
T_4 = 1 + 1 + 2 = 4
```

Example 2:

```
Input: n = 25
Output: 1389537
```

https://leetcode.com/problems/n-th-tribonacci-number/

Task 2 – Tribonacci Number (DP)



1137. N-th Tribonacci Number

The Tribonacci sequence T_n is defined as follows:

$T_0 = 0$, $T_1 = 1$, $T_2 = 1$, and $T_{n+3} = T_n + T_{n+1} + T_{n+2}$ for n >= 0.

Given `n`, return the value of T_n.

Example 1:

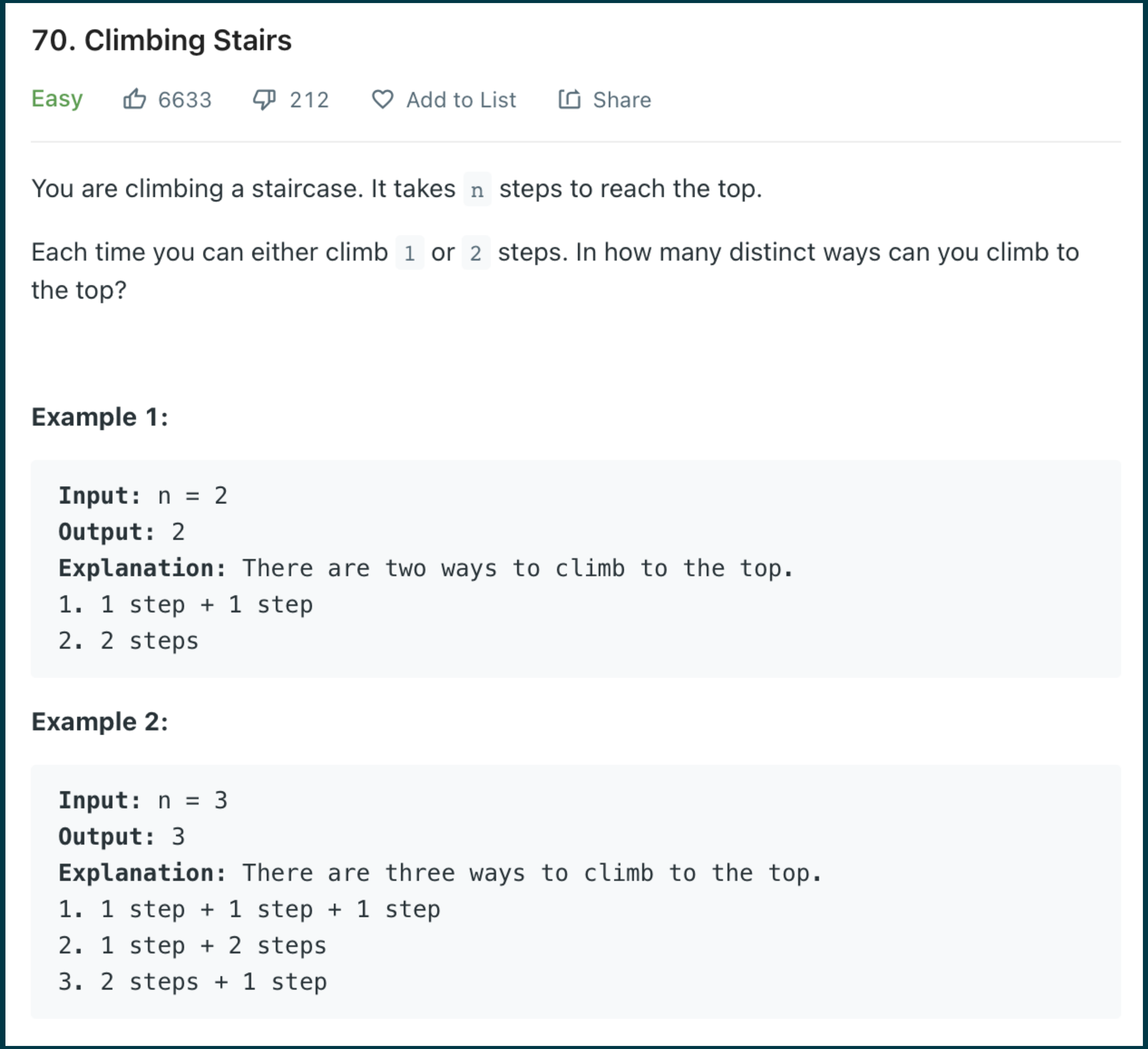
```
Input: n = 4
Output: 4
Explanation:
T_3 = 0 + 1 + 1 = 2
T_4 = 1 + 1 + 2 = 4
```

Example 2:

```
Input: n = 25
Output: 1389537
```

https://leetcode.com/problems/n-th-tribonacci-number/

Task 3 – Climbing Stairs



70. Climbing Stairs

You are climbing a staircase. It takes `n` steps to reach the top.

Each time you can either climb `1` or `2` steps. In how many distinct ways can you climb to the top?

Example 1:

```
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps
```

Example 2:

```
Input: n = 3
Output: 3
Explanation: There are three ways to climb to the top.
1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step
```

https://leetcode.com/problems/climbing-stairs/