Problem Solving

Codility and Leetcode Practitioner

Problem

Where love begins

Problems

Description:

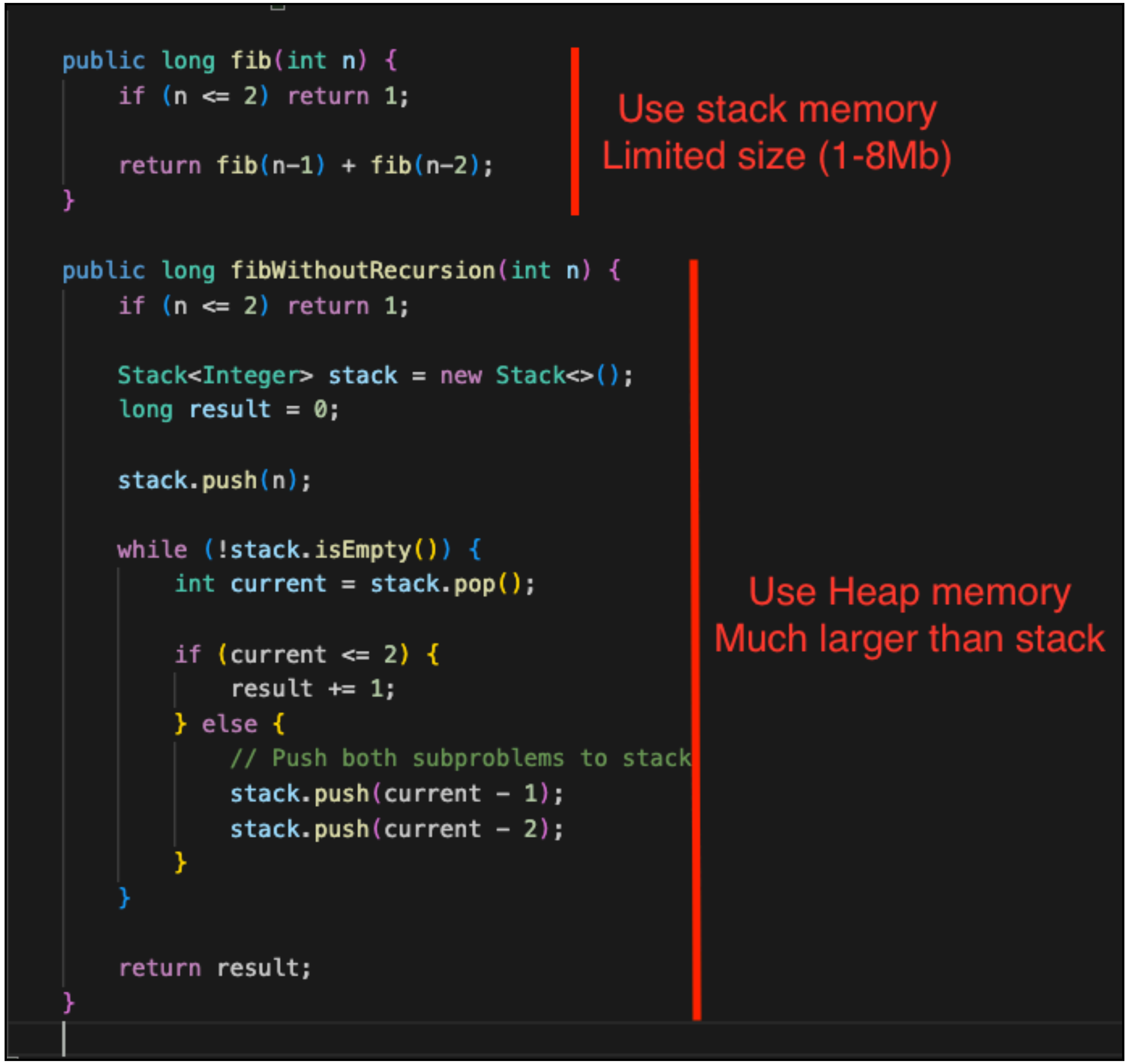
Write a function `fib(n)` that takes in a number as an argument and return the n-th number of the Fibonacci sequence.

Example:

- fib(1) is 1
- fib(2) is 1
- fib(3) is 2
- fib(4) is 3
- fib(n) is fib(n-1)+fib(n-2)

Brute-force

Costly but work



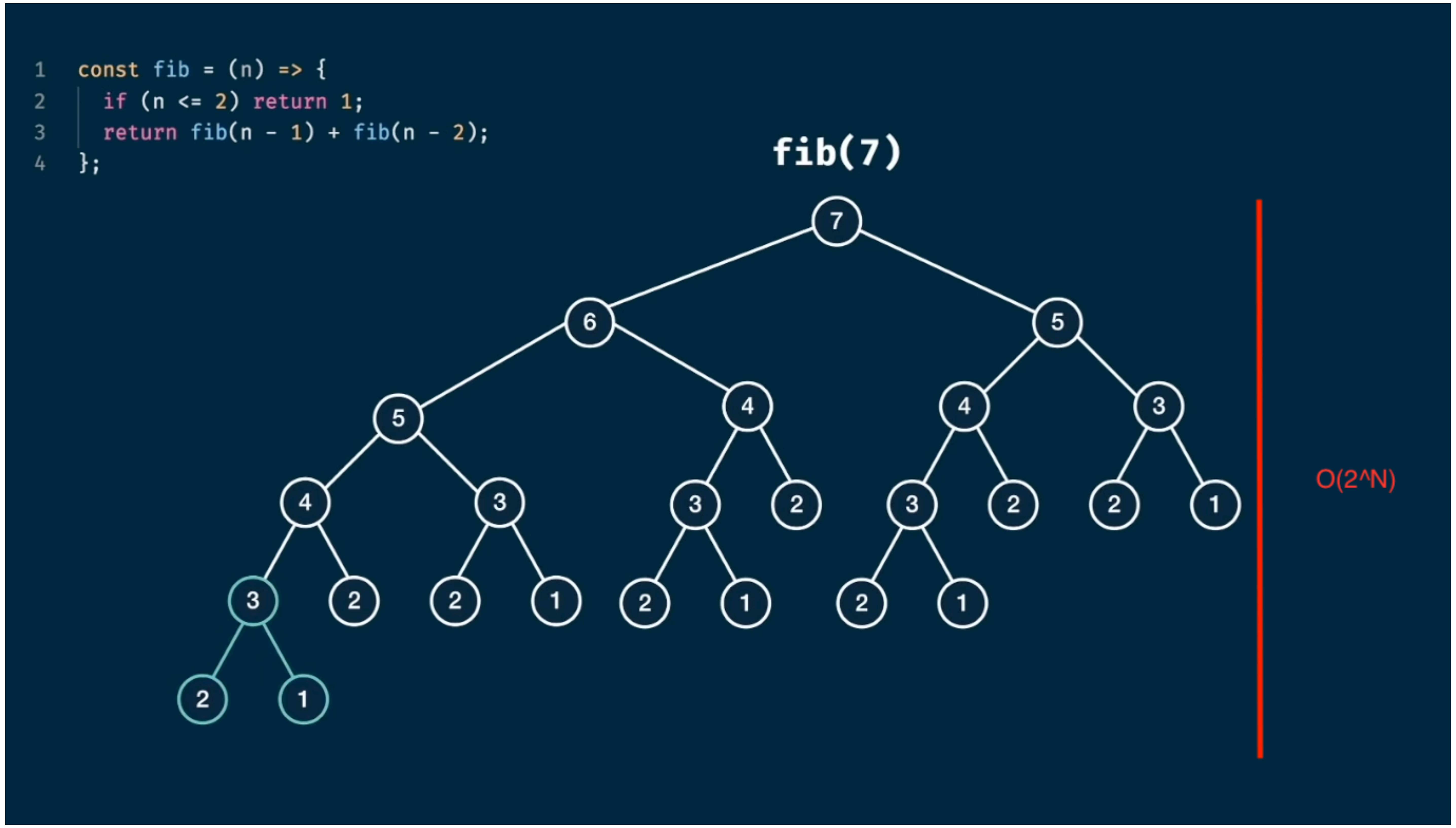
```
public long fib(int n) {
    if (n <= 2) return 1;

    return fib(n-1) + fib(n-2);
}

public long fibWithoutRecursion(int n) {
    if (n <= 2) return 1;

    Stack<Integer> stack = new Stack<>();
    long result = 0;

    stack.push(n);

    while (!stack.isEmpty()) {
        int current = stack.pop();

        if (current <= 2) {
            result += 1;
        } else {
            // Push both subproblems to stack
            stack.push(current - 1);
            stack.push(current - 2);
        }
    }

    return result;
}
```

Use stack memory
Limited size (1-8Mb)

Use Heap memory
Much larger than stack

Brute-force

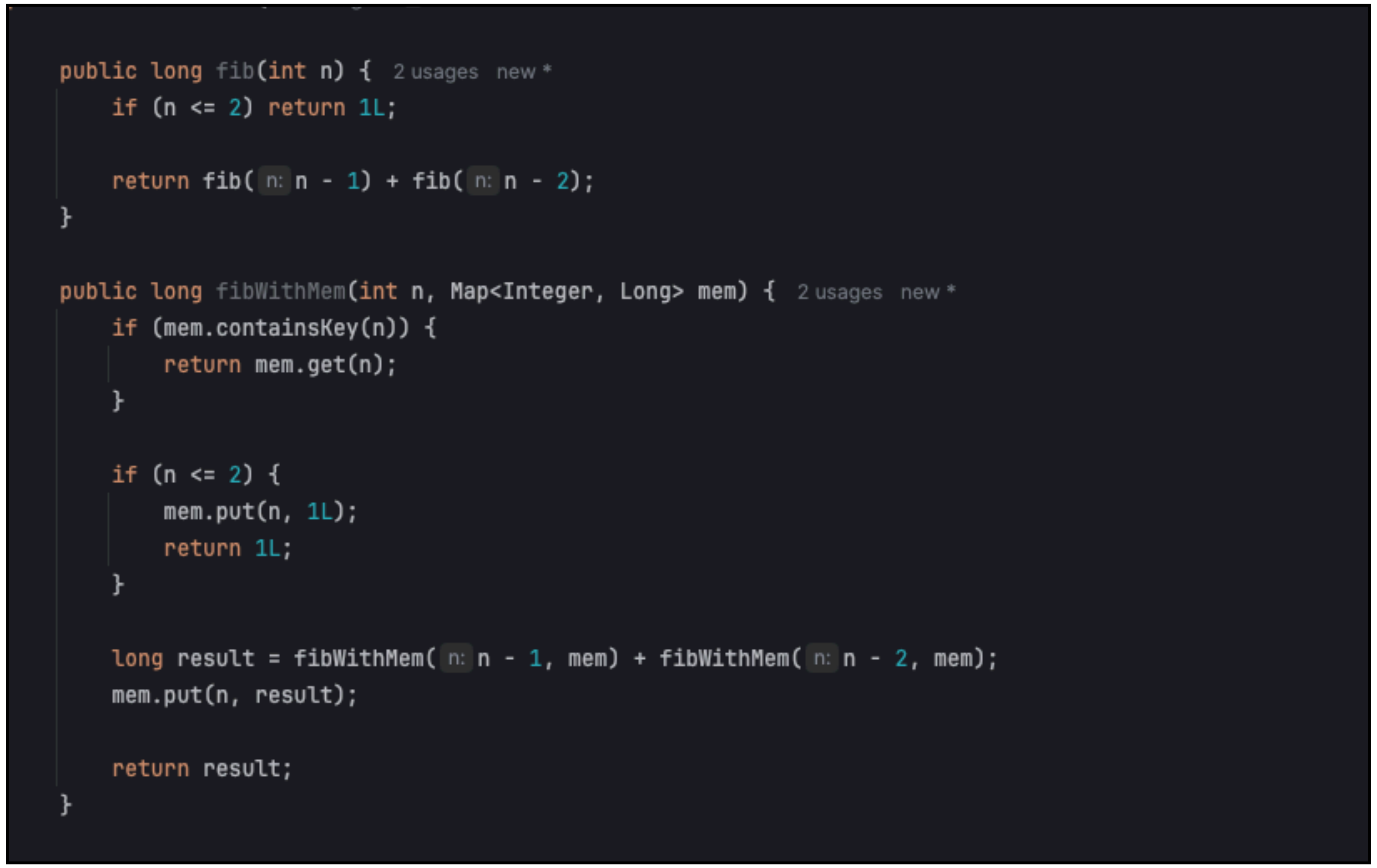
Costly but work



```
const fib = (n) => {
  if (n <= 2) return 1;
  return fib(n - 1) + fib(n - 2);
};
```

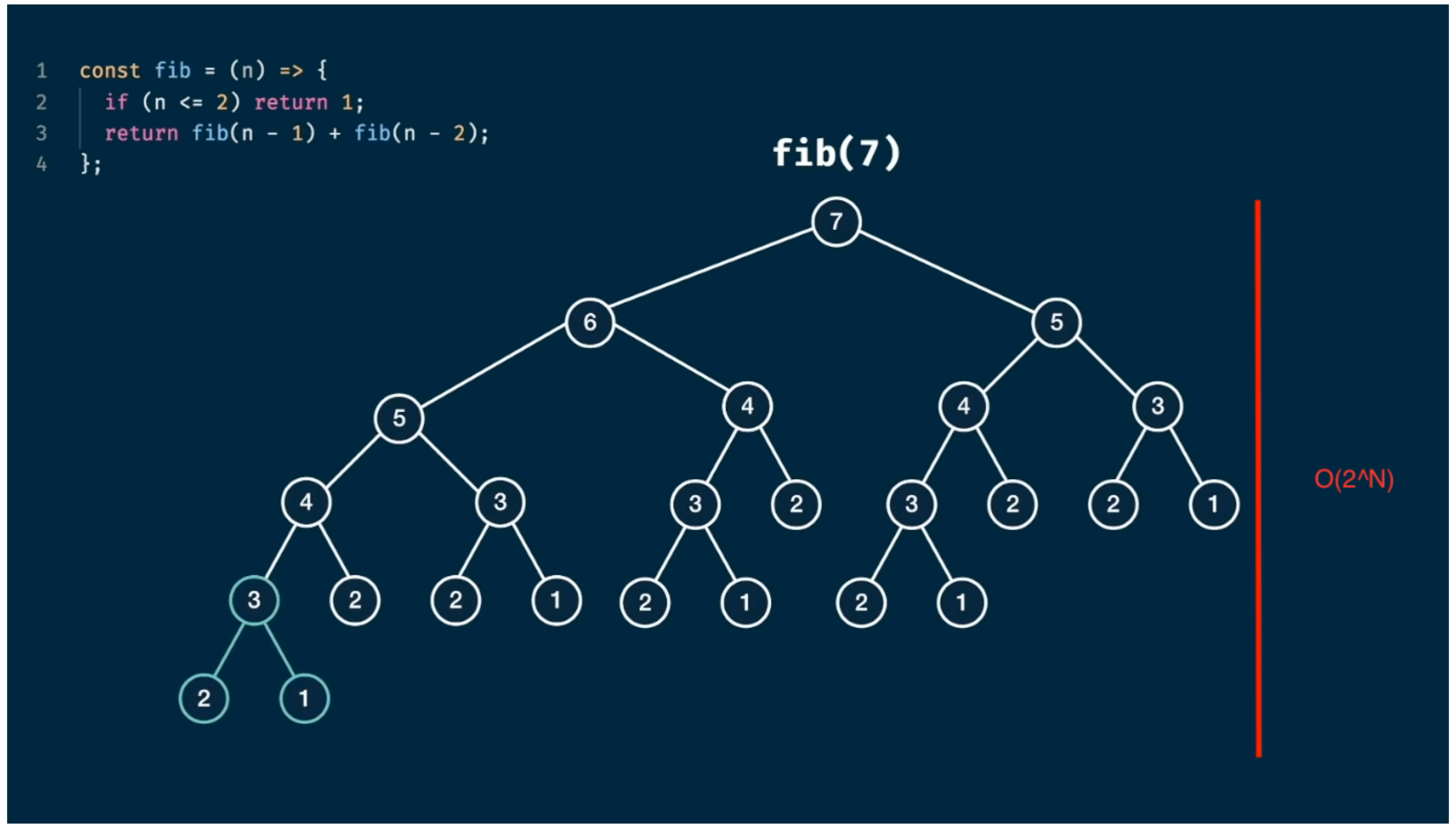
Memoization

Remove duplicate by Caching



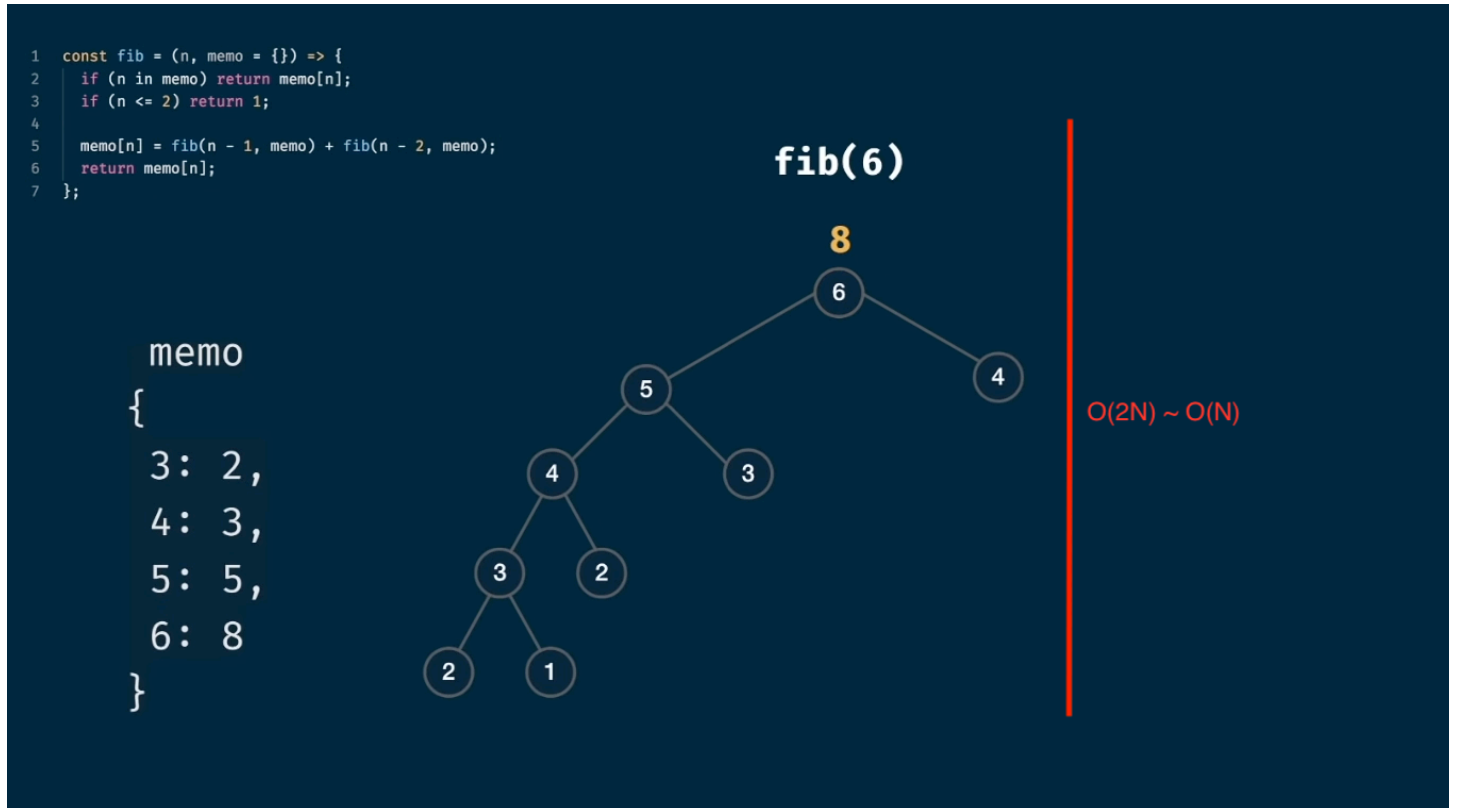
```java
public long fib(int n) {  2 usages  new *
    if (n <= 2) return 1L;

    return fib( n: n - 1) + fib( n: n - 2);
}

public long fibWithMem(int n, Map<Integer, Long> mem) {  2 usages  new *
    if (mem.containsKey(n)) {
        return mem.get(n);
    }

    if (n <= 2) {
        mem.put(n, 1L);
        return 1L;
    }

    long result = fibWithMem( n: n - 1, mem) + fibWithMem( n: n - 2, mem);
    mem.put(n, result);

    return result;
}
```

Brute-force (2^N)



```
const fib = (n) => {
  if (n <= 2) return 1;
  return fib(n - 1) + fib(n - 2);
};
```

Memoization (2N)



```
const fib = (n, memo = {}) => {
  if (n in memo) return memo[n];
  if (n <= 2) return 1;

  memo[n] = fib(n - 1, memo) + fib(n - 2, memo);
  return memo[n];
};
```

```
memo
{
 3: 2,
 4: 3,
 5: 5,
 6: 8
}
```

Thank you

Don’t give up