Prompt

Difficulty: Easy **Category:** Recursion **Successful Submissions:** 85,260+

Nth Fibonacci

The Fibonacci sequence is defined as follows: the first number of the sequence is `0`, the second number is `1`, and the nth number is the sum of the (n - 1)th and (n - 2)th numbers. Write a function that takes in an integer `n` and returns the nth Fibonacci number.

Important note: the Fibonacci sequence is often defined with its first two numbers as `F0 = 0` and `F1 = 1`. For the purpose of this question, the first Fibonacci number is `F0`; therefore, `getNthFib(1)` is equal to `F0`, `getNthFib(2)` is equal to `F1`, etc..

Sample Input #1

```
n = 2
```

Sample Output #1

```
1 // 0, 1
```

Sample Input #2

```
n = 6
```

Sample Output #2

```
5 // 0, 1, 1, 2, 3, 5
```

Hints

Hint 1

The formula to generate the nth Fibonacci number can be written as follows: F(n) = F(n - 1) + F(n - 2). Think of the case(s) for which this formula doesn't apply (the base case(s)) and try to implement a simple recursive algorithm to find the nth Fibonacci number with this formula.

Hint 2

What are the runtime implications of solving this problem as is described in Hint #1? Can you use memoization (caching) to improve the performance of your algorithm?

Hint 3

Realize that to calculate any single Fibonacci number you only need to have the two previous Fibonacci numbers. Knowing this, can you implement an iterative algorithm to solve this question, storing only the last two Fibonacci numbers at any given time?

Optimal Space & Time Complexity

O(n) time | O(1) space - where n is the input number

Your Solutions

Run Code

Solution 1 Solution 2 Solution 3

```python
'''
O(n) Time | O(n) Space
'''
def getNthFib(n, memoize={1:0, 2:1}):
    if n in memoize:
        return memoize[n]
    else:
        memoize[n] = getNthFib(n-1, memoize) + getNthFib(n-2, memoize)
        return memoize[n]

# Kunal Wadhwa

```

Custom Output | Raw Output | Submit Code

Yay, your code passed all the test cases!

18 / 18 test cases passed.

- Test Case 1 passed!
- Test Case 2 passed!
- Test Case 3 passed!
- Test Case 4 passed!
- Test Case 5 passed!
- Test Case 6 passed!
- Test Case 7 passed!
- Test Case 8 passed!
- Test Case 9 passed!
- Test Case 10 passed!
- Test Case 11 passed!
- Test Case 12 passed!
- Test Case 13 passed!
- Test Case 14 passed!
- Test Case 15 passed!
- Test Case 16 passed!
- Test Case 17 passed!
- Test Case 18 passed!