



Can we improve Fibonacci?

Let's recall the Fibonacci sequence

The first Fibonacci numbers are:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

Bases cases:

When n is 0 or 1

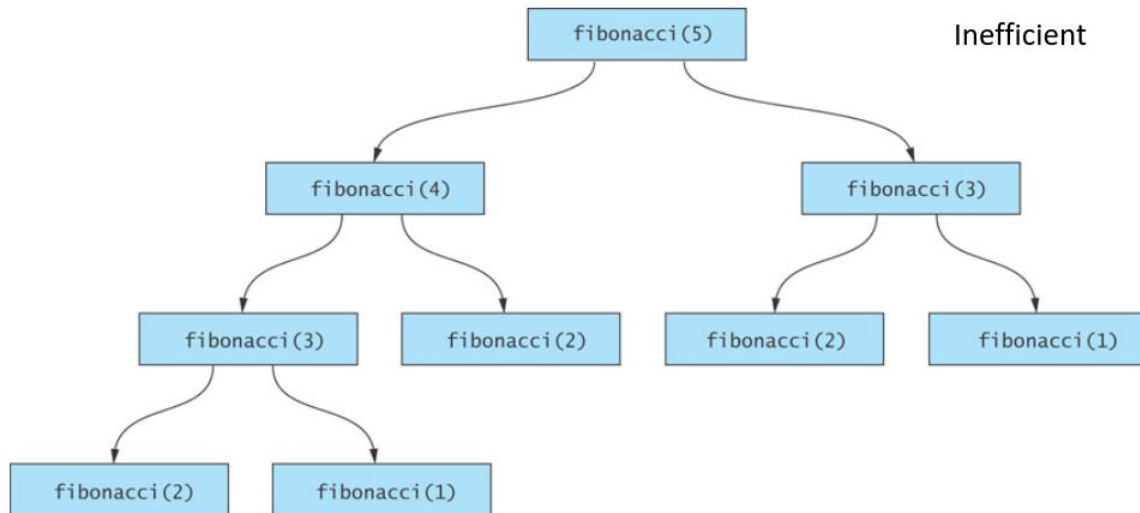
$\text{fib}(0) = 0$ and $\text{fib}(1) = 1$

Recursive case:

$\text{fib}(n-1) + \text{fib}(n-2)$

Time complexity:

$O(2^n)$



Is there a way to improve the recursive Fibonacci?

Yes, we can implement an iterative Fibonacci using tail recursion.

Recall:

In tail recursion, **no other operation is needed** after the successful execution of a recursive function call.

Tail recursion requires to **pass additional parameter to maintain the state of the recursive call.**



Iterative Fibonacci Algorithm

Maintain two variables (**f1 and f2**) to store the last two Fibonacci numbers.
This will allow you to calculate the next value in constant time.

Update these variables in each iteration to compute the next Fibonacci number.

Initial values for these variables f1=1 and f2=0

The method signature should be:

```
public static int fiblter(int n, int f1, int f2) {}
```

Test cases:

fiblter(0, 1, 0) returns 0

fiblter(1, 1, 0) returns 1

fiblter(2, 1, 0) returns 1

fiblter(3, 1, 0) returns 2

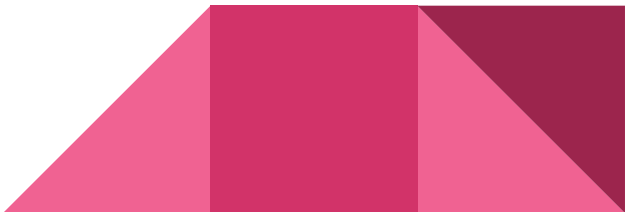
...

Sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, ...

What would be the time complexity?

$O(n)$



Classwork

Save here:

.../APCSA2/apcsa-assignments-YourUsername-spring/classwork/02_02_fibIter/

1. Draw the call stack for fibIter(10, 1, 0) - Submit a txt file with your call stack.
2. Let's implement the iterative Fibonacci algorithm (IterativeFibonacci.java):

```
/*  
 * Preconditions:  
 * n is a non-negative integer  
 * n refers to the nth element in the fib sequence  
 * initial f1 value is 1  
 * initial f2 value is 0  
 */  
public static int fibIter(int n, int f1, int f2){  
    // remember the recursive case calls fibIter once  
}
```