```
Fibonacci Fast(n)
(1) if n==0 or n==1
(2)     return n
(3) let F[0..n] be a new array
(4) F[0] = 0
(5) F[1] = 1
(6) for j=2 to n
(7)     F[j] = F[j-1] + F[j-2]
(8) return F[n]
```

| Cost | # of times |
|---|---|
| $C_1$ | 1 |
| $C_2$ | 0 ~~or 1~~ |
| $C_3$ | ~~0 or~~ 1 |
| $C_4$ | ~~0 or~~ 1 |
| $C_5$ | ~~0 or~~ 1 |
| $C_6$ | ~~0 or~~ $n$ |
| $C_7$ | ~~0 or~~ $n-1$ |
| $C_8$ | ~~0 or~~ 1 |

If $n=0$ or 1, FibonacciFast returns $n = F_n$

If $n \geq 2$, the if does nothing, we initialize the array $F$, set $F[0]=0$, set $F[1]=1$, then do a for loop. Our loop invariant is the following:

(*) Immediately after the start of the loop, the $0^{th}$ thru $j-1^{st}$ entries of F are the corresponding Fibonacci numbers

Want to prove: For all $n \in \mathbb{N}$, Fibonacci($n$) returns $F_n$.

$P(n)$ = "Fibonacci($n$) returns $F_n$".

Prove $P(n)$ for all $n$ by induction:

Base case $P(0)$: if $n=0$ or $n=1$ return $n$ ⇝ returns $0=F_0$

Induction step: suppose $n \geq 1$ and $P(0), \ldots, P(n-1)$ hold.

If $n=1$, then Fibonacci($n$) returns $1 = F_n$.

If $n \geq 2$, then Fibonacci($n$) calls Fibonacci($n-1$), which returns $F_{n-1}$ by IH, and stores it as $a$, and also stores Fibonacci($n-2$) $= F_{n-2}$ as $b$, then returns $a+b = F_{n-1} + F_{n-2} = F_n$

$P(0)$, $P(1)$
Inductive step:
can assume
$n \geq 2$, IH is
$P(0)$ & $P(1)$ & $\ldots$ & $P(n-1)$

Define $T(n)$ to be the running time of Fibonacci(n). From the above, we see that $T(n) = \begin{cases} c_1 + c_2 & \text{if } n \le 1 \\ c_1 + c_3 + T(n-1) + T(n-2) + c_4 & \text{if } n \ge 2 \end{cases}$

Thus

$$T(0) = T(1) = c_1'$$

$$T(n) = T(n-1) + T(n-2) + c_2' \quad \text{if } n \ge 2.$$

This has solution $T(n) = (c_1' + c_2') \cdot F_{n+1} - c_2'$.

Fact: $F_n = \frac{\phi^n + \bar{\phi}^n}{\sqrt{5}}$, where $\phi = \frac{1+\sqrt{5}}{2} = 1.618\ldots$, $\bar{\phi} = \frac{1-\sqrt{5}}{2} = -.618\ldots$

So $T(n)$ grows <u>exponentially</u>. This is very inefficient!

$$\begin{aligned} Fib(5) &= Fib(4) + Fib(3) \\ &= (Fib(3) + Fib(2)) + [Fib(2) + Fib(1)) \\ &= [(Fib(2) + Fib(1)) + (Fib(1) + Fib(0))] \\ &\quad + [[Fib(1) + Fib(0)) + 1] \\ &= \cdots \end{aligned}$$

```
Fibonacci: Fast (n)
(1) if n==0 or n==1
(2)     return n
(3) let F[0...n] be a new array
(4) F[0] = 0
(5) F[1] = 1
(6) for j=2 to n
(7)     F[j] = F[j-1] + F[j-2]
(8) return F[n]
```

| Cost | # of times for |
|---|---|
| $C_1$ | 1 |
| $C_2$ | 0 ~~of 1~~ |
| $C_3$ | ~~0 of~~ 1 |
| $C_4$ | ~~0 of~~ 1 |
| $C_5$ | ~~0 of~~ 1 |
| $C_6$ | ~~0 of~~ $n$ |
| $C_7$ | ~~0 of~~ $n-1$ |
| $C_8$ | ~~0 of~~ 1 |

If $n=0$ or 1, FibonacciFast returns $n = F_n$

If $n \geq 2$, the if does nothing, we initialize the array $F$, set $F[0]=0$, set $F[1]=1$, then do a for loop. Our loop invariant is the following:

(*) Immediately after the start of the loop, the $0^{th}$ thru $j-1^{st}$ entries of F are the corresponding Fibonacci numbers

$n=4$; $j=2$ $F=[0,1,\ ,\ ,\ ]$

$F[2]=F[1]+F[0]=1+0$

$F=[0,1,1,\ ,\ ]$

$j=3$

$F[3]=F[2]+F[1]=1+1$

$F=[0,1,1,2,\ ]$

$j=4$

$F[4]=F[3]+F[2]=2+1$

$F=[0,1,1,2,3]$

$j=5$ loop ends

$$\lfloor 2x \rfloor = \lfloor x \rfloor + \lfloor x+1/2 \rfloor$$

goes up 2 at integers, half-integers

jumps at integers

jumps at half-integers

If $x \in [k, k+1/2)$, Then

$k \in \mathbb{Z}$

$2k \le 2x < 2k+1 \Rightarrow \lfloor 2x \rfloor = 2$

$k \le x < k+1$ $\lfloor x \rfloor =$

$k \le x < k+1/2$ $\lfloor x+\frac{1}{2} \rfloor$

If $x \in [k+1/2, k+1)$

(Initialization) At the start of the loop, $j=2$, $F=[0, 1, ?, \ldots, ?]$.

This satisfies (*)

(Maintenance) ~~Fix some~~ Assume (*) holds, and run the loop once. For $i=0, \ldots, j-1$, $F[i]=F_i$, we set $F[j]=F[j-1]+F[j-2]$

$=F_{j-1}+F_{j-2}$

$=F_j$,

since $j \geq 2$.

Then we increment $j$. Now for $i=0, \ldots, j+1-1$, $F[i]=F_i$.

~~(Termination)~~

(Termination) At the end of the loop, $j=n+1$, and (*) says that for $i=0, \ldots, n$, $F[i]=F_i$.

Now we return $F[n]=F_n$.

Running time $= c_1 + c_3 + c_4 + c_6 \cdot n + c_4 \cdot (n-1) + c$

$= a + b \cdot n$

This is roughly a constant times

$n$, i.e. $\Theta(n)$.

| | cost | # of times run |
|---|---|---|
| (1) sum = 0 | $c_1$ | 1 |
| (2) for j = 1 to n | $c_2$ | n+1 |
| (3) sum = sum + 1 | $c_3$ | n |
| (4) return sum | $c_4$ | 1 |

Total cost = $c_1 + (n+1)c_2 + n \cdot c_3 + c_4$

LI: sum = m + j - 1

old sum + 1 = m + j + 1 - 1

new sum = m + j - 1

j = 1 ~~check~~ j ≤ n

sum = sum + 1

j = j + 1 (j → 2)

~~check~~

check j ≤ n

sum = sum + 1

j = j + 1 (j = 3)

check j ≤ n

InductiveMult LI: prod = m(j-1)

prod = m

for j = 1 to n

prod = InductiveAdd(prod, m)

return prod