Lecture1

Monday, January 25, 2021 9:43 AM

Algorithm : India @ 600 AD → Alkhawrizmi @ 800 AD → Algebra ← book title → Leonardo of Pisa — Fibonacci @ 1200 AD

Fibonacci Series

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

F(n) : n^{th} term of Fibonacci series

base cases

F(0) = 0

F(1) = 1

Recursive formula

F(n) = F(n-1) + F(n-2)

Algorithm

~~@ memoize~~

Fib1 (n) :

2 Compare

If n = 0, return 0

If n = 1, return 1

return Fib1(n-1) + Fib1(n-2)

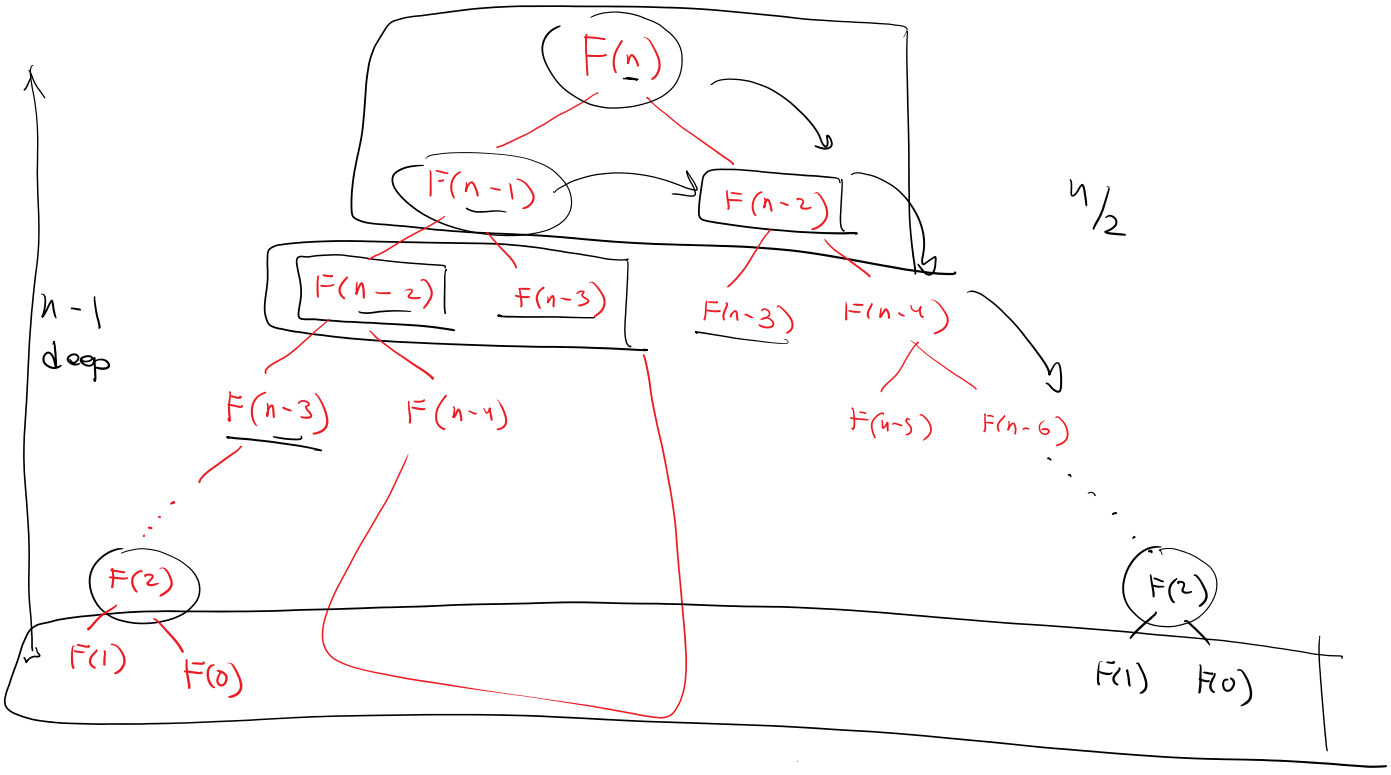
3 add / subtract

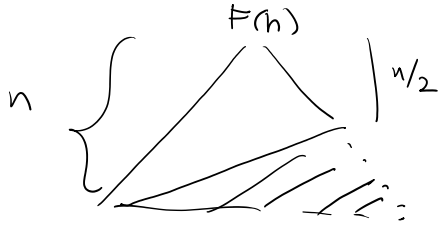
Correct ✓

time ?

5 operations per call $= O(1)$ ← constant

how many calls do we make ? $O(2^n)$ ← exponential





Complete Call tree

$$\underbrace{1 + 2 + 4 + \cdots + 2^{n-1}}_{\text{geometric series}} = \sum_{i=0}^{n-1} 2^i = \frac{2^n - 1}{2-1} \approx 2^n$$

$K = n-1$

$O(2^n)$

$$r^0 + r^1 + r^2 + \cdots + r^k = \sum_{i=0}^{k} r^i$$

r : factor

I) $|r| > 1$,

$$\sum_{i=0}^{k} r^i = \frac{r^{k+1} - 1}{r - 1}$$

II) $|r| < 1$,

$$\sum_{i=0}^{\infty} r^i = \frac{1}{1 - r}$$

II) $|r| < 1$, $\sum_{i=0}^{\infty} r^i = \frac{1}{1-r}$

Can we do better?

Sometimes yes
sometime NO → Sorting → lower bound $O(n \log n)$ Comparison based

Fib2 (n) : memoization (store previously computed value) the reuse

Fi → allocate an array of length n+1

0	1	2	3	4	...	n
0	1	1	2	3	...	F(n)

replace recursion with iteration

F(0) = 0
F(1) = 1

For i = 2 to n ← loop
  F(i) = F(i-1) + F(i-2) → naive assumption O(1)
Return F(n)

n-1 time

3 add/sub

F(n) is O(n) bits
∴ addition is O(n)

n-2+1 # of iterations

n-1 ≈ O(n) ← linear # of steps
O(n) for F(n)
↳ ~~O(1)~~ ← constant time per iteration

O(n) × O(n) = O(n²)
# of iterations, time to add

O(n) × O(1) = O(n) ← linear time? ← sublinear

vs

$O(2^n)$ ← hopeless

Quadratic time

n: denotes the size of the input (usually)

Caveat:

"n" denotes the size of the input (usually)

Caveat: n is the value of the input

$F(n)$ = value increases exponentially $\approx O(2^n)$

$$F(n) = F(n-1) + F(n-2)$$

addition / subtraction → constant time operations

64 bit numbers

size

$F(\underset{\uparrow}{10000}) = 2^{10^5}$ → # of bits to store $O(n)$

input value

Value of $F(10^5)$

64

$2^{64} - 1$

Valu

size's # of bits it take to store that number

↳ $\log_2(2^{10^5}) \approx 10^5$

$F(n)$ take n bits to store

$F(n) = F(n-1) + F(n-2)$

n bit n bit

$O(n)$ time

Recursion cost

Monday, January 25, 2021 12:04 PM