Intro to Algorithms

Today: using Fibonnacci sequence

- Time complexity
- Bit Complexity
- Start Big O notation.

F.B.: 0 1 1 2 3 5 8 13 ......

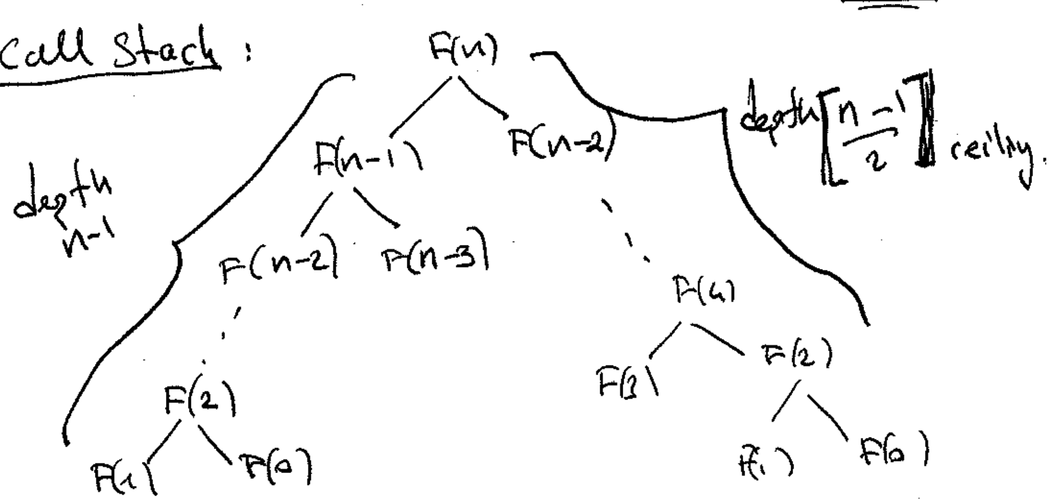
$F_0 = 0$ } base case.
$F_1 = 1$

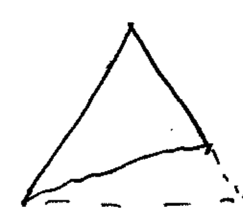
n^{th} element of the seq: $F_n = F_{n-1} + F_{n-2}$ } recursive formula

```
FIB1(n)            n is an integer > 0
    if n = 0    return 0
    if n = 1    return 1
    return   FIB1(n-1) + FIB1(n-2)
```

① Correctness   ② efficiency.
time   space

Call Stack:





$$\underbrace{\underset{\substack{n-1\\ \text{bits}}}{F(n-1)} + \underset{\substack{n-2\\ \text{bits}}}{F(n-2)}}_{n \text{ operations.}} \Rightarrow (n-2+1)\cdot n = n^2 - n \cong n^2$$

$n = 2^{k}$ bits to represent. } deal with large #s.

Claim: $F_n \geq 2^{0.5n}$ for $n \geq 6$.

Proof: (by induction).

base: $F_6 = 8 \geq 2^{(0.5)6} = 2^{6/2} = 8$ ✓

Inductive Step: for $n \geq 6$

$$F_{n+1} = F_n + F_{n-1} \geq 2^{n/2} + 2^{(n-1)/2} = 2^{(n-1)/2} \cdot (2^{1/2} + 1)$$
$$\geq 2^{(n-1)/2} \cdot 2 \geq 2^{(n+1)/2}$$ ✓

$F(n) \approx 2^{0.5 \cdot n} \approx 2^n$

next: big $O(\ )$

Lecture 2

Today: - finish basic analysis
- Asymptotic time analysis $\quad O(.), \Omega(.), \Theta(.)$

Last time:

$F(n) \cong 2^{c.n}$ $\quad c \cong 0.7$

$0.7 \cdot n = O(n)$

$F(n) \sim 2^n$

$N = 2^K$

$\Rightarrow$ nth Fib. # requires n-bits

Fib2

F: [| | | | | .. |]

n times and each time one addition of n-bit #s

$$\left[\begin{array}{l} \text{for } i = 2 .. n \\ F[i] = \underbrace{F[i-1]}_{n} + \underbrace{F[i-2]}_{n} \end{array} \right]$$

$O(n^2)$ bit complexity

FIB3(n) based on Matrix multiplication.

$F_0 = 0$

$F_1 = 1$

$F_n = F_{n-1} + F_{n-2}$.

$$\begin{bmatrix} F_1 \\ F_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{M} \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

$F_1 = 0 . F_0 + 1 . F_1$

$1 F_0 + 1 F_1 = F_0 + F_1 = F_2$ $\quad$ for the general case.

$$\begin{bmatrix} F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

$$\begin{bmatrix} F_2 \\ F_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}}_{M^2} \begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$$

$\begin{bmatrix} F_0 \\ F_1 \end{bmatrix}$

$$\begin{bmatrix} F_n \\ F_{n+1} \end{bmatrix} = M^n \cdot \begin{bmatrix} F_0 \\ F_1 \end{bmatrix} = M^n \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} a & b \\ c & d \end{bmatrix}}_{M^n} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix} \Rightarrow F_n = b$$

FIB3(n)

```
    if n = 0    return 0
    if n = 1    return 1
    M = [0 1]       M' = [0 1]
        [1 1]            [1 1]
    for i = 2 ... n
        M' = M' x M
        M' = [a b]
             [c d]
    return b
```

n times

$O(n^2)$

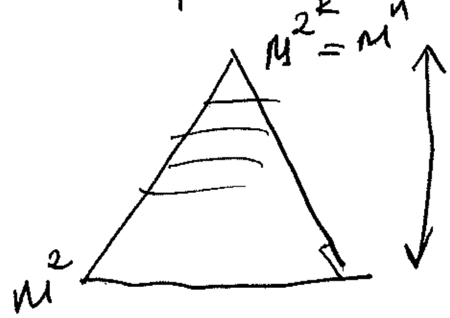
$\Rightarrow O(n^3)$

- let's consider a better analysis

Q: how many matrix multiplications does it take to compute M^n

$n = 2^k$

$M^n = M^{2^k}$ we can compute recursively



$M^{2^2} \cdot M^{2^2} = M^8$

$M^2 = M^{2^1} \cdot M^{2^1}$

!

Note: every squaring of M. we are doubling the exponent

$k = \log n$

$$M^n = \begin{cases} \left(M^{\lfloor n/2 \rfloor}\right)^2 & \text{if } n \text{ is even} \\ M\left(M^{\lfloor n/2 \rfloor}\right)^2 & n \text{ is odd.} \end{cases}$$

$\Rightarrow \log n$ squares, $\log n$ multiplications by M.

[general rule: if the problem size is halfed at each iteration then it takes $O(\log n)$ steps.]

Formally we write a recurrence formula for the running time of an algorithm.

$$\begin{tikzcd} & M^n \arrow[dl, dash] \arrow[dr, dash] & \\ M^{n/2} & & M^{n/2} \end{tikzcd}$$

$$\underbrace{T(n)}_{\text{Fib3}(n)} = T\left(\frac{n}{2}\right) + \underbrace{\text{mul.}}\left(\frac{n}{2}\right)$$

$$= \underbrace{T\left(\frac{n}{2}\right)} + \left(\frac{n}{2}\right)^2$$

$$= \left[T\left(\frac{n}{4}\right) + O\left(\frac{n}{4}\right)^2\right] + \frac{n^2}{4}$$

$$= T\left(\frac{n}{4}\right) + O\left(\frac{n^2}{16}\right) + O\left(\frac{n^2}{4}\right)$$

$$\vdots$$

$$= \frac{n^2}{4} + \frac{n^2}{16} + \frac{n^2}{64} \cdots$$

$$= \frac{n^2}{4}\underbrace{\left(1 + \frac{1}{4} + \frac{1}{16} \cdots\right)}_{\text{geometric series}} \leq 2$$

$$T(n) \leq \frac{n^2}{4} \cdot 2$$

$$\leq \frac{n^2}{2} \Rightarrow \boxed{O(n^2)}$$

Fib3(n): $O(n^3) \rightarrow O(n^2 \log n) \rightarrow O(n^2)$

but: we can do faster matrix multiplication

$O(n^{1.6}) \approx \boxed{O(n \cdot \sqrt{n})}$ Fib3

Fib2 $O(n^2)$

Fib4(n)

$$F_n = \frac{1}{\sqrt{5}}\left(\lambda_1^n - \lambda_2^n\right) \quad (*)$$

λ_1 and λ_2 are the eigen values of matrix M

7

$\lambda_1 = \frac{1+\sqrt{5}}{2}$ $\quad \lambda_2 = \frac{1-\sqrt{5}}{2}$

$M = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

eigenvalues and vectors to express a matrix

$$\boxed{M = \underbrace{V}_{\text{eigen vector}} \underbrace{\Lambda}_{\text{eigenvalues are at the diagonal}} V^{-1}}$$

$$\boxed{M \cdot v = \lambda v}$$

for any value of λ that solves this equation is an eigen value of M, and corresponding v is the eigen vector.

$$M = \begin{bmatrix} \begin{bmatrix} v_1 \\ \vdots \end{bmatrix} & \begin{bmatrix} v_2 \\ \vdots \end{bmatrix} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} & \end{bmatrix}^{-1}$$

$$M^n = \begin{bmatrix} v_1 & v_2 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}^n \begin{bmatrix} & \end{bmatrix}^{-1} \quad (*)$$

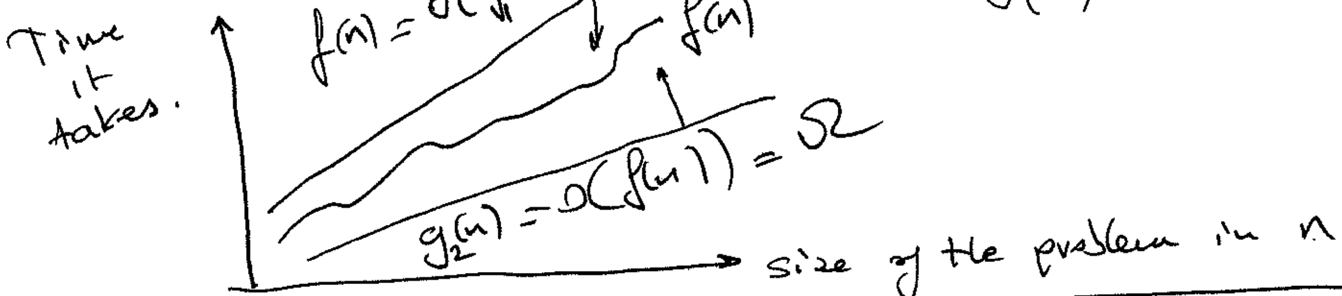
we write characteristic equation

$$|M - \lambda I| = 0$$

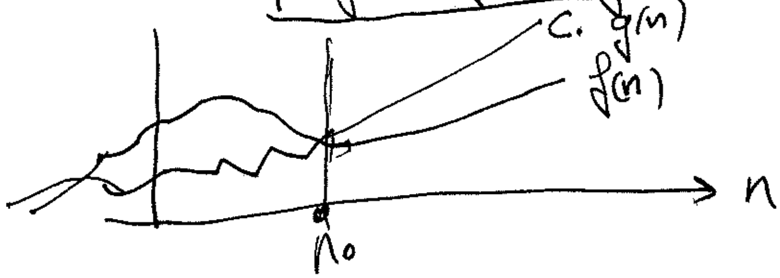
solve this

Asymptotic Time Analysis
with big-Oh notation.

$\Theta(n) = \Omega(n) \Rightarrow \Theta(n)$



<u>Definition:</u> we say that $f(n) = O(g(n))$ if there exists a constant $c > 0$ such that for all $n > n_0$
$f(n) \leq c \cdot g(n)$



$$\underbrace{2n^2}_{g} \quad \text{vs.} \quad \underbrace{5n^2 + 10^6 n}_{f} + 5 \qquad n > 0$$

$$g = O(f)$$

$$g \leq c \cdot f.$$

$1000\, n^2 \quad$ v.s. $\quad n^3$

Asymptotic analysis using limits - order of growth.

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 : \text{case I } f(n) \text{ has a smaller order of growth} \\ c : \text{case II } f(n) \text{ has the same order of growth as } g(n) \\ \infty : \text{case III } f(n) \text{ has larger order of growth w.r.t. } g(n) \end{cases}$$

if case I and case II are true then $f(n) \in O(g(n))$

case II and case III are true then $f(n) \in \Omega(g(n))$

case II $f(n) \in \Theta(g(n))$.

<u>example</u>: $f(n): n^2 + 5n$ $\quad g(n): 10n^2 + 2n + 100$

$$\lim_{n\to\infty} \frac{n^2+5n}{10n^2+2n+100} = \frac{1}{10} = 0.1 \text{ constant}$$

$f(n) \in \Theta(g(n))$ ✓