# Divide & Conquer

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

$$n^{\log_b a} = n$$

if $f(n) = n \Rightarrow$ case 2 $\Rightarrow \Theta(n \cdot \log n)$

$(k=0)$

if $f(n) = n \log^k n \Rightarrow$ case 2 $\Rightarrow \Theta(n \log^{k+1} n)$

if $f(n) = \log^c n \Rightarrow$ case 1 $\Rightarrow \Theta(n)$

$f(n) = O(n^{1-\varepsilon})$

if $f(n) = n^c$ for $c > 1$ $\Big\}$ $f(n) = \Omega(n^{1+\varepsilon})$

AND $2f\left(\frac{n}{2}\right) = \frac{2}{2^c} n^c = \frac{1}{2^{c-1}} \cdot f(n)$

... case 3 $\Rightarrow \Theta(f(n))$

$$T(n) = 4T\left(\frac{n}{4}\right) + f(n)$$

SAME

# COMPUTING THE (integer n) POWER OF A NUMBER $x$ : $x^n$

$\underbrace{x \cdot x \cdot x \cdot \cdots x \cdot x}_{n \text{ times}}$ } $\Theta(n)$ time, assuming any multiplication costs $\Theta(1)$

-OR-

for even $n$ : $x^n = \underbrace{x^{n/2}} \cdot \underbrace{x^{n/2}}$ : $T(n) = 2T(\frac{n}{2}) + \Theta(1)$

$\Theta(n)$

# COMPUTING THE (integer $n$) POWER OF A NUMBER $x$ : $x^n$

$\underbrace{x \cdot x \cdot x \cdot \ldots \cdot x \cdot x}_{n \text{ times}}$ } $\Theta(n)$ time , assuming any multiplication costs $\Theta(1)$

-OR-

for even $n$ : $x^n = \underbrace{x^{n/2}} \cdot x^{n/2}$ : $T(n) = 2T\left(\frac{n}{2}\right) + \Theta(1)$

$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$

$O(n) \rightarrow O(\log n)$

# COMPUTING THE (integer $n$) POWER OF A NUMBER $x$ : $x^n$

$$\underbrace{x \cdot x \cdot x \cdot \dots \cdot x \cdot x}_{n \text{ times}} \Big\} \ \Theta(n) \text{ time}$$

, assuming any multiplication costs $\Theta(1)$

-OR-

(1) for even $n$ : $x^n = x^{n/2} \cdot x^{n/2}$ :

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

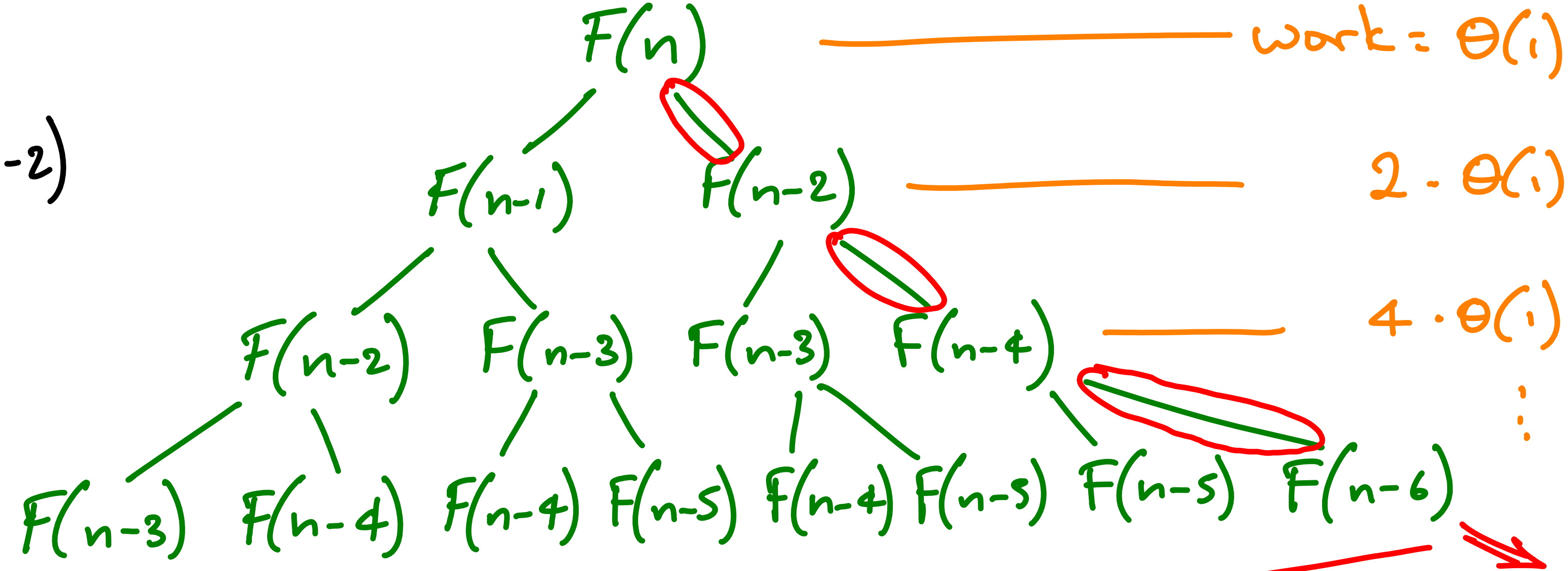
(2) for odd $n$ : $x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x$

$$T(n) = T\left(\frac{n-1}{2}\right) + \Theta(1)$$

$$\boxed{\Theta(\log n)}$$

# Fibonacci Number Computation

$F(0) = 0$
$F(1) = 1$
$F(n) = F(n-1) + F(n-2)$




total work = # nodes

Which root→leaf path is shortest? RIGHTMOST

depth $\sim \frac{n}{2}$

so we have $\frac{n}{2}$ "full" levels: WORK $> 2^{n/2} = \Omega(\sqrt{2}^{\,n})$

Instead, just compute each $F(k)$ iteratively. (once)

$0, 1, F(2)=1, F(3)=2, F(4)=3, F(5)=5, F(6)=8$ etc

$\Theta(n)$ time

Fact: $F(n) = \frac{\Phi^n}{\sqrt{5}}$, rounded to integer

Assume:
1) you can compute & store $\sqrt{5}$
2) you can round numbers quickly
3) you can multiply $\Phi$ powers quickly

i.e quick arithmetic on "difficult" numbers

so this depends on the MODEL OF COMPUTATION

↳ then by our $x^n$ method we get $\Theta(\log n)$ time for $F(n)$

# FIBONACCI NUMBER COMPUTATION

## FINALE

matrix $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$ } true for $n=1$ } $\begin{matrix} F_2 & F_1 \\ F_1 & F_0 \end{matrix} = \begin{matrix} 1 & 1 \\ 1 & 0 \end{matrix}$

General proof      Induction

$$\underbrace{\begin{pmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{pmatrix}}_{\text{assume holds}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{bmatrix} F_n + F_{n-1} & F_n \\ F_{n-1} + F_{n-2} & F_{n-1} \end{bmatrix} = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$$ Q.E.D.

assume holds

i.e $= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^n$ is like $x^n$ : $T(n) \simeq T\left(\frac{n}{2}\right) + \Theta(1)$

$\hookrightarrow F_n$ in $\Theta(\log n)$ time

$n \times n$ SQUARE MATRIX MULTIPLICATION

$C = A \times B \qquad c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \Rightarrow \Theta(n)$ per element $\Rightarrow \Theta(n^3)$ total

Divide & Conquer:

$$\underset{C}{\begin{bmatrix} r & s \\ t & u \end{bmatrix}} = \underset{A}{\begin{bmatrix} a & b \\ c & d \end{bmatrix}} \times \underset{B}{\begin{bmatrix} e & f \\ g & h \end{bmatrix}} \quad \left\{ \begin{array}{l} r = ae + bg \\ s = af + bh \\ t = ce + dg \\ u = cf + dh \end{array} \right\} \begin{array}{l} T(n) = \\ = 8T\left(\frac{n}{2}\right) + \underbrace{\Theta(n^2)}_{\text{4x matrix addition}} \end{array}$$

Master theorem: $f(n) = \Theta(n^2) \qquad n^{\log_b a} = n^3$

so $f(n) = O(n^{3-\varepsilon})$ for $0 < \varepsilon \leqslant 1$

CASE1 $\rightarrow T(n) = \Theta(n^3)$ :(

Trying smaller blocks won't help

# STRASSEN'S ALGORITHM

$$\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right] = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] \times \left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]$$

$$\left\{\begin{array}{ll} r = ae + bg & = p_4 + p_5 + p_6 - p_2 \\ s = af + bh & = p_1 + p_2 \\ t = ce + dg & = p_3 + p_4 \\ u = cf + dh & = p_1 + p_5 - p_3 - p_7 \end{array}\right.$$

note: matrix multip. is not commutative

$$\underbrace{\begin{aligned} p_1 &= a(f-h) \\ p_2 &= (a+b)\cdot h \\ p_3 &= (c+d)\cdot e \\ p_4 &= d\cdot(g-e) \\ p_5 &= (a+d)(e+h) \\ p_6 &= (b-d)(g+h) \\ p_7 &= (a-c)(e+f) \end{aligned}}_{\text{7 multiplications \& } O(1) \text{ additions}}$$

## DIVIDE & CONQUER

$$T(n) = \underbrace{7T\left(\tfrac{n}{2}\right)}_{\substack{\text{multiply} \\ \text{to get } p_i}} + \underbrace{\Theta(n^2)}_{\text{additions to compute } p_i}$$

& to add $p_i$ to get r, s, t, u

still CASE 1

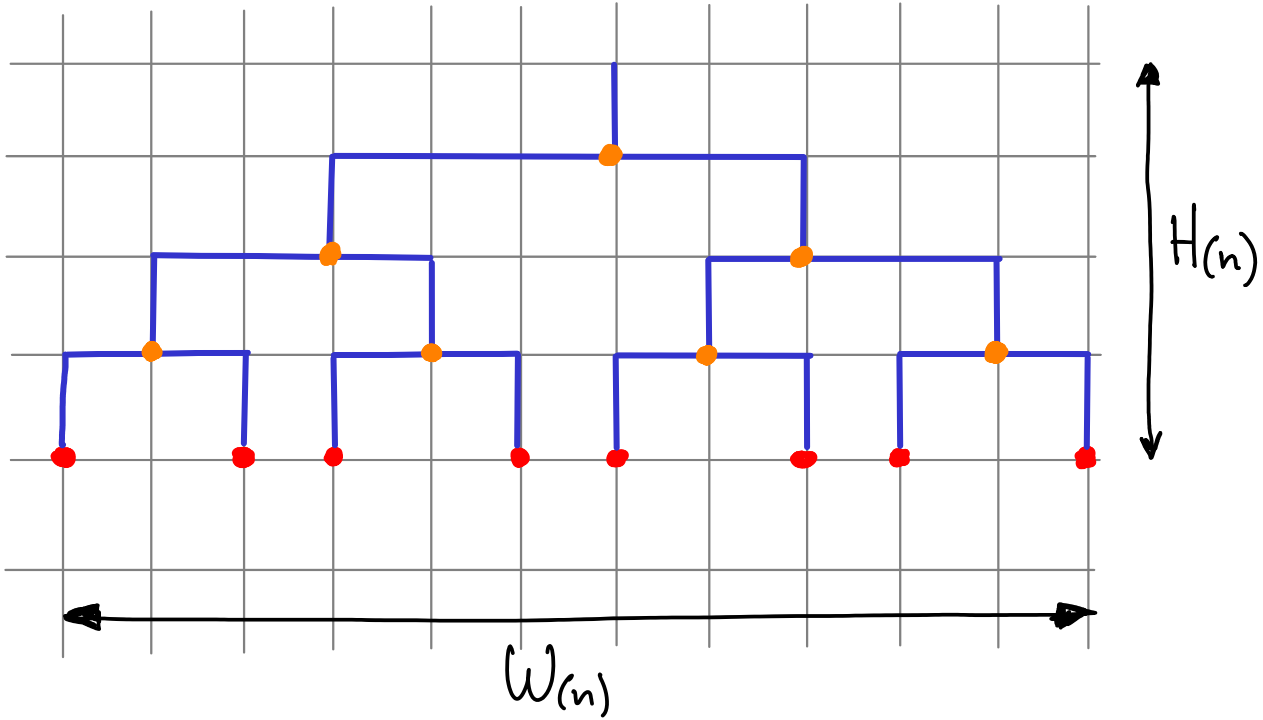
$$T(n) = \Theta\left(n^{\log 7}\right) = O\left(n^{2.81}\right)$$

- worthwhile for $n > 30$
- there is a $n^{2.376}$ algo
  ↳ practically never used.

# VLSI layout [very large scale integrated chips]

e.g.: embed a complete binary tree w/ n leaves on a grid, w/o crossings & MINIMIZE AREA

(edges can be any length, but on grid)

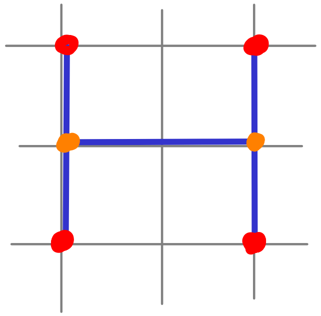


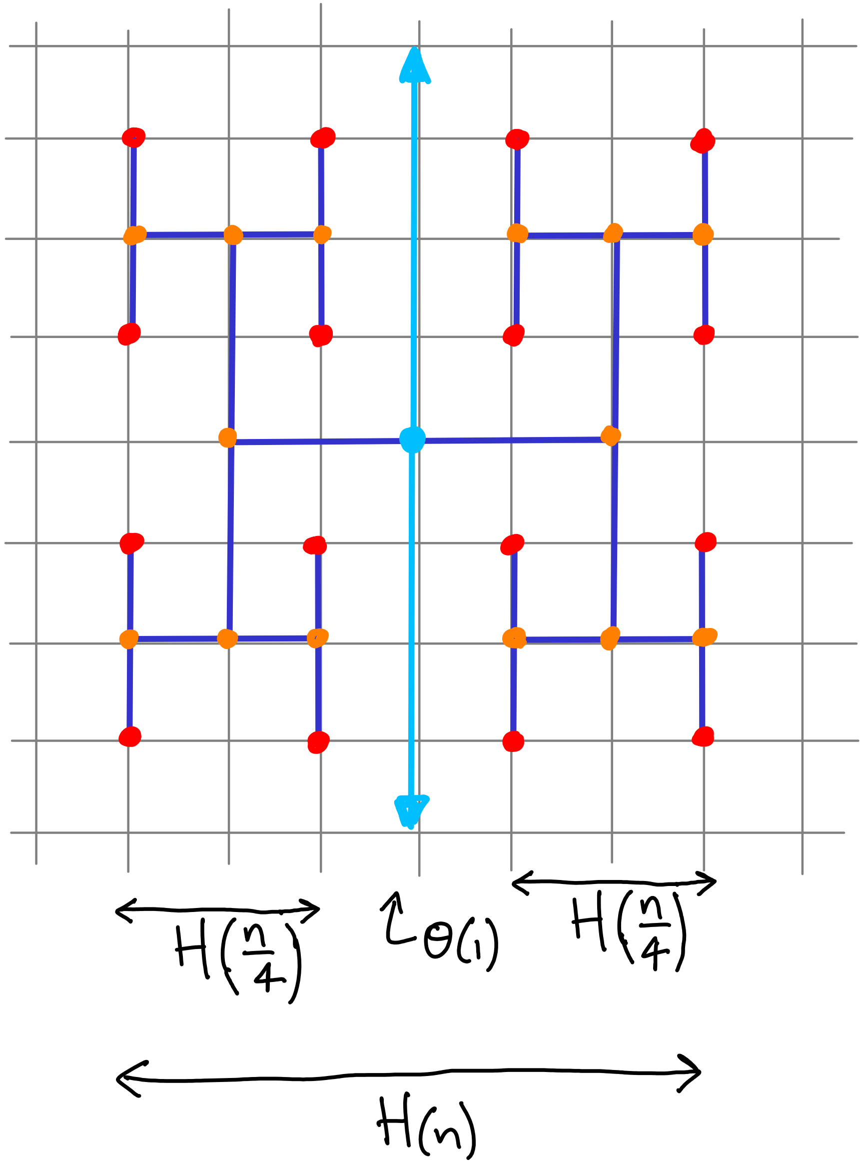

$$H(n) = H\left(\tfrac{n}{2}\right) + \Theta(1) = \Theta(\log n)$$

$$\times$$

$$W(n) = 2W\left(\tfrac{n}{2}\right) + O(1) = \Theta(n)$$

$$\underbrace{\Theta(\log n) \times \Theta(n)}_{\Theta(n \log n)\ \text{area}}$$






$$H(n) = W(n) = 2H\left(\frac{n}{4}\right) + \Theta(1)$$

$$\text{Master: } H(n) = a \cdot H\left(\frac{n}{b}\right) + f(n)$$

$$a = 2 \qquad b = 4$$

$$n^{\log_b a} = n^{1/2}$$

$$f(n) = O\left(n^{1/2 - \varepsilon}\right) \qquad \text{case 1}$$

$$H(n) = \Theta(\sqrt{n})$$

$$\text{Area} = \Theta(n)$$