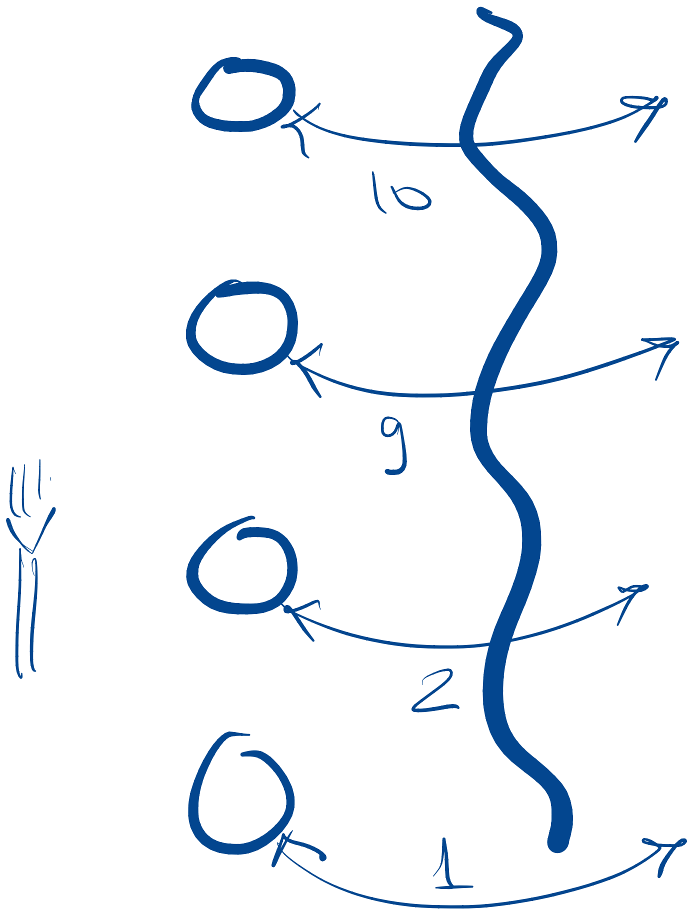


Crossing time

23
25
17

2 can cross at a time → time is for slowest
only with "flashlight"

Fibonacci Number $F_0 = 0$ $F_n = F_{n-1} + F_{n-2}$

$F_1 = 1$ $\forall n \geq 2$

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
0	1	1	2	3	5	8	13	21	34	55

Task: Compute F_n = function of n = input

(1) Math following recursive

Fib(n)

if n=0 return 0
(output)

if n=1 return 1

(~~else~~) return $\boxed{\text{Fib}(n-1)} + \boxed{\text{Fib}(n-2)}$

(A) correct? yes

(B) how ~~fast~~? exponential runtime

$\boxed{\Theta(2^n)}$??

tight asymptote

lower bound $\quad\quad\quad\quad$ upper bound

$\Omega \quad\quad \Theta \quad\quad O$

asymp $\quad\quad$ exact asymptote $\quad\quad$ asymp

ex $\Theta(n^2) = f(n)$

$c_2 \cdot n^2 \leq f(n) \leq c_1 \cdot n^2$

low bound $\quad\quad\quad\quad$ upper bound

2 Array computation Fib(n)

F = array   F[0] = 0 , F[1] = 1

for i = 2 : n

  F[i] = F[i-1] + F[i-2]

return F[n]

C. Storage
A correct
B how fast

LINEAR
$\Theta(n)$

Q: can we store only last 3 values?

③ Matrix-Multiplication-based Fib(n)

$$\underset{2\times 2}{M} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad \text{claim} \quad M^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$$

induction proof

base case $M^1 = M = \begin{bmatrix} F_2 & F_1 \\ F_1 & F_0 \end{bmatrix}$?

inductive step $n \to n+1$

$$\boxed{M^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}} \Longrightarrow M^{n+1} = \begin{bmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{bmatrix}$$

ind hyp   ind conclusion

proof: $M_1^{n+1} = M^n \times M_2 \overset{\text{hyp}}{=} \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

$$= \begin{array}{c} 1 \\ 2 \end{array}\begin{bmatrix} F_{n+1}+F_n & F_{n+1} \\ F_n+F_{n-1} & F_n \end{bmatrix} = \begin{bmatrix} F_{n+2} & F_{n+1} \\ F_{n+1} & F_n \end{bmatrix} \checkmark$$

② Matrix Fib (n) Run Time?

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

$\boxed{\text{compute } M^n = A}$ exponential of a matrix

return $M[1,2]$ ✓

naive $M^n = \underbrace{M \cdot M \cdot M \cdots M}_{n \text{ times}}$

$\Theta(n)$ time

$\Theta(\log n)$

Fast exponentiation (example)

M

$M^2 = M \cdot M$

$M^4 = M^2 \cdot M^2$

$M^8 = M^4 \cdot M^4$

$M^{16} = M^8 \cdot M^8$

$M^{32} = M^{16} \cdot M^{16}$

$M^{64} = M^{32} \cdot M^{32}$

Want $M^{100} = M^{64} \cdot M^{32} \cdot M^4$

$M^{171} = M^{128} \cdot M^{32} \cdot M^8 \cdot M^2 \cdot M^1$

④ Generating Function

$$F(n) = \frac{\phi^n - \varphi^n}{\sqrt{5}}$$

$$\phi = \frac{1+\sqrt{5}}{2}$$

$$\varphi = \frac{1-\sqrt{5}}{2}$$

ϕ real number $\in \mathbb{R}$

math-coprocessor estimate

$\simeq$ constant true.

$\simeq \Theta(1)$

idea for ϕ, φ

guess

$F(n) \simeq a^n$? true

rec. Fib:

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

$$a^{n+1} = a^n + a^{n-1} \quad \Big| \div a^{n-1}$$

$$a^2 = a + 1$$

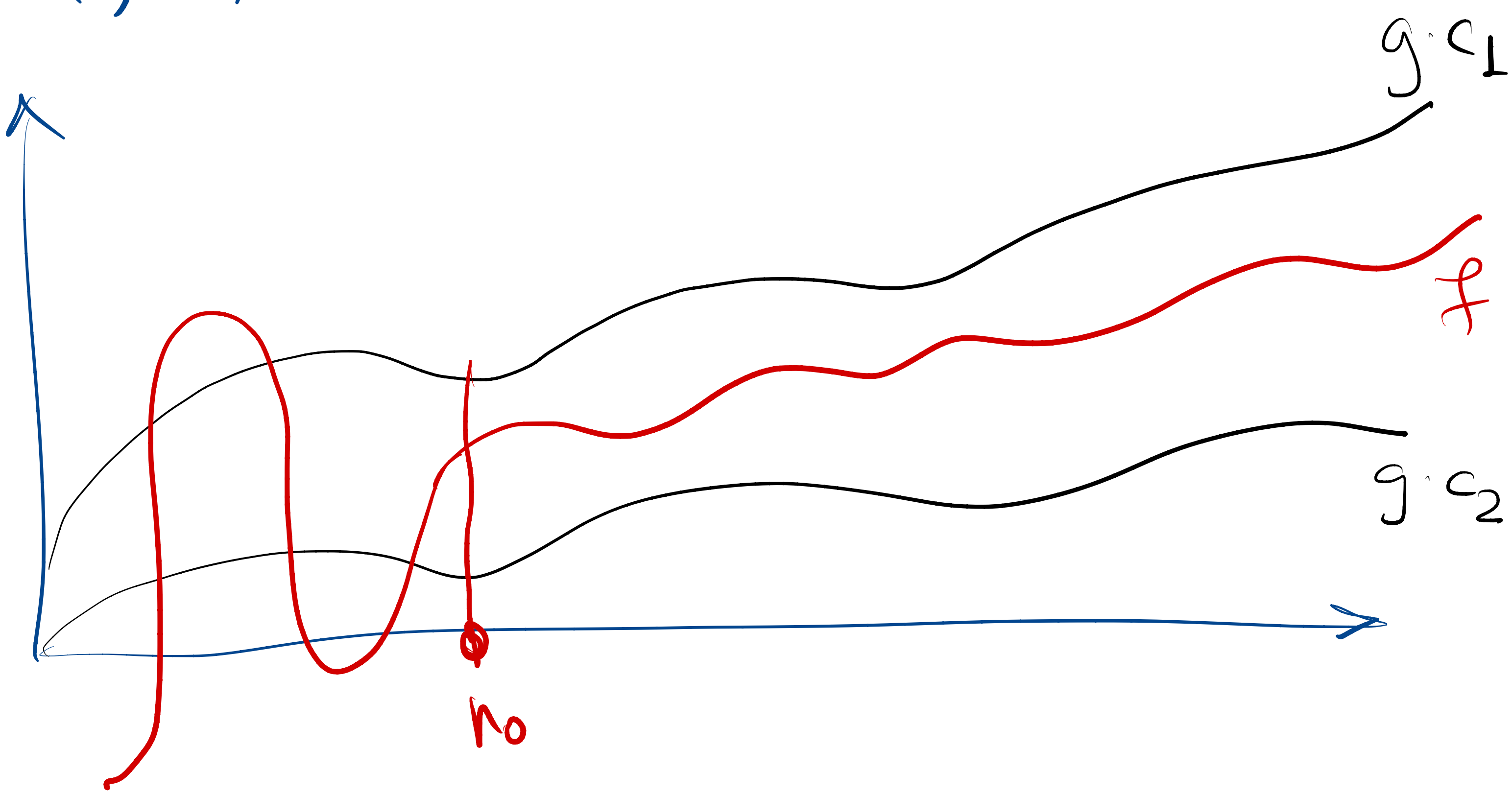
quad eq $\Rightarrow \phi, \varphi$.

big O notation

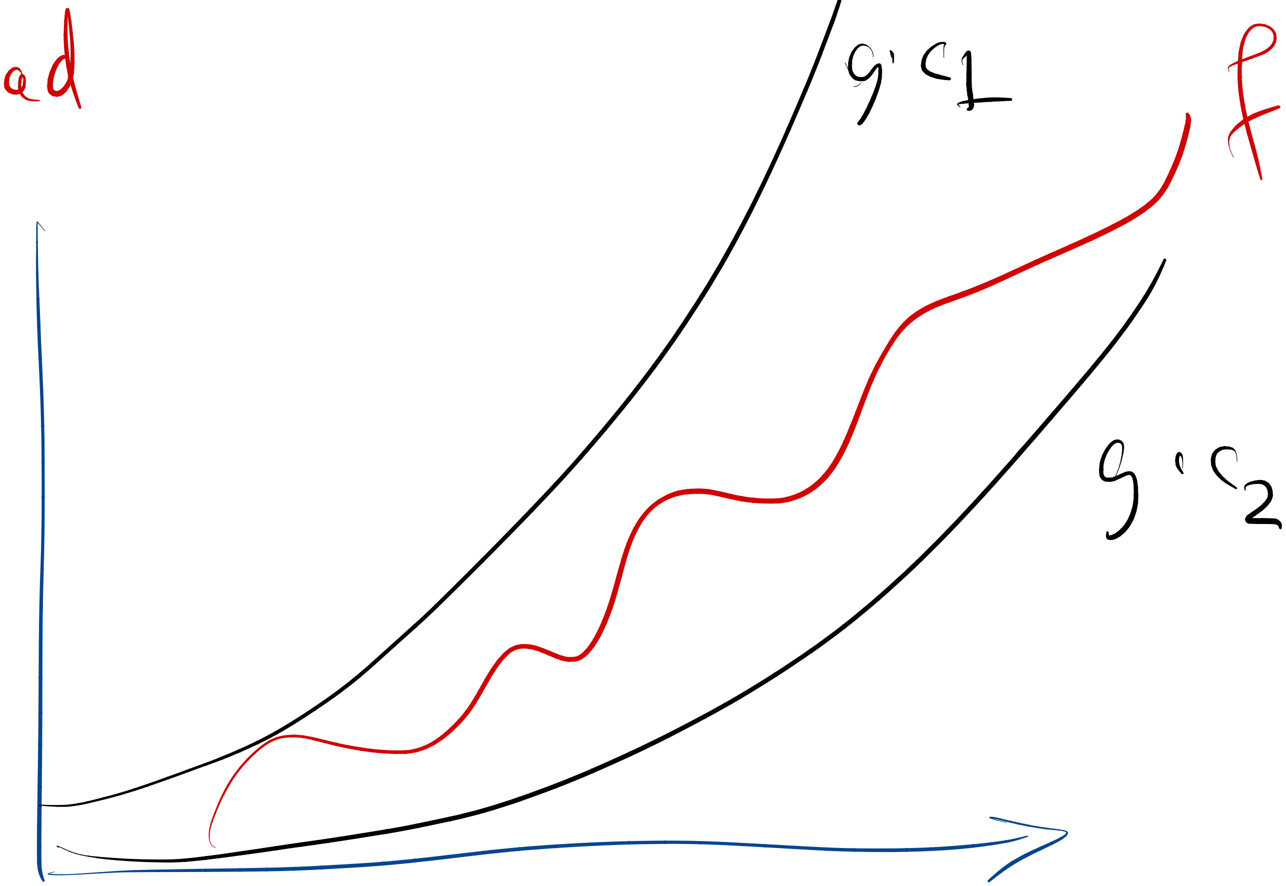
$f = \Theta(g)$

$f(n) = \Theta(g(n))$: $c_1, c_2 > 0$ constants $c_1 > c_2$

$g \cdot c_2 \leq f \leq g \cdot c_1$

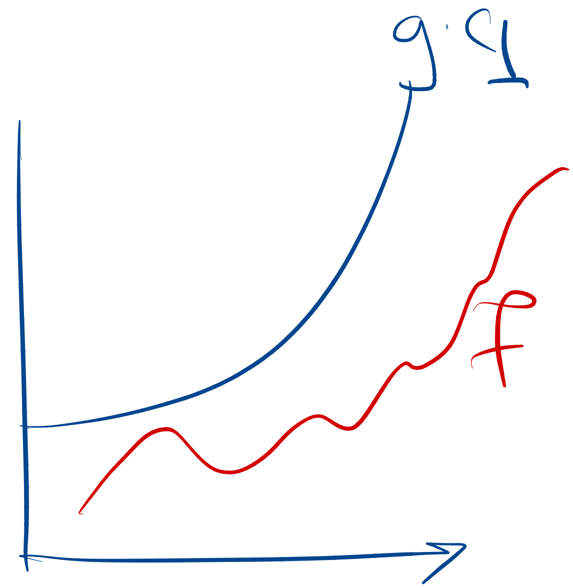


g = quad



Big O = upper bound $f = O(g)$

$\exists c_1 > 0 \quad f \leq g \cdot c_1$



$f(n)$

$2n^2 + 3n + 2$ $\quad\Theta(n^2)$

$n^2 \log n + n^3$ $\quad\Theta(n^3)$

$2^n + 5n^2 - 3$ $\quad\Theta(2^n)$

$3^n + 2^n$ $\quad\Theta(3^n)$

$3^n = \Theta(2^n)$?

$c_2 \cdot 2^n \le 3^n \le \cancel{c_1 \cdot 2^n}$

true

$\frac{3^n}{2^n} \le c_1$

$\left(\frac{3}{2}\right)^n \le c_1$

$\lim (1.5)^n = +\infty$

NO

$\log_a x = b \iff a^b = x$

$f = \log_2 x$

$g = \log_3 x$

$f = \Theta(g)$

$\boxed{c_2 \cdot \log_3(n) \le \log_2(n)} \le c_1 \cdot \log_3(n)$

$\log_3(x) \overset{?}{=} \log_2(x) \cdot \log_3 2$?

$3^{\log_3 x} = 3^{\log_2 x \cdot \log_3 2}$

$x = \left(3^{\log_3 2}\right)^{\log_2 x}$

$c_2 = 1$

$\log_3(n) \le \boxed{\log_2(n) \le c_1 \cdot \log_3(n)}$??

$2^{\log_2 x}$

x

$\log_2(n) \le c_1 \cdot \log_2(n) \cdot \log_3 2$

$1 \le c_1 \cdot \log_3 2$

constant

Binary Search

$A =$

-3	-1	0	5	12	17	21	75	92	103
1	2	3	4	5	6	7	8	9	10

BS(v, A, begin, end)

Find value V = given in A[begin : end] (sorted) (indices)

initially begin = 1, end = 10

- $m = \frac{begin + end}{2}$
- $\boxed{\text{if } V = A[m]}$ ⇒ done, return m, A[m]

if $v < A[m]$ // search A[begin : m]
BS(v, A, begin, m)

//else

if $v > A[m]$ // search A[m : end]
BS(v, A, m, end].

Fix termination

R.T. T(n) = time to search |A| = n (size)

$$T(n) = \Theta(1) + T(n/2)$$

A size | A/2 size

recurrence due to recursive calls

$k=1$ $T(n) = 1 + T(n/2)$ $\forall n$

$k=2$ $= 1 + \left[1 + T(n/4) \right] = 2 + T(n/4)$

$k=3$ $= 2 + \left[1 + T(n/8) \right] = 3 + T(n/8)$

$k=4$ $= 3 + \left[1 + T(n/16) \right] = \boxed{4} + T(n/16)$

pattern

k: $k + T(n/2^k)$

optional (if messy) induction proof

step k $\quad$ step $k+1$

$$k + T(n/2^k) \Longrightarrow k+1 + T(n/2^{k+1})$$

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

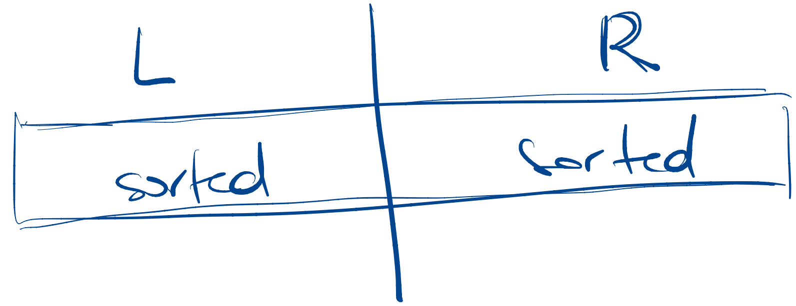
last K : $\frac{n}{2^k} \simeq 1 \Rightarrow T\left(\frac{n}{2^k}\right) \rightarrow T(1) = \text{base}$

$$n \simeq 2^k$$

$$k \simeq \log_2(n)$$

$$T(n) = \log_2(n) + \underbrace{T(1)}_{\text{const}} = \Theta(\log n)$$

Merge Sort $A(b:e)$: sort Array A



$m = \frac{b+e}{2}$

Split in 2 $A[b:m)$

$A[m+1:e]$

$\text{MergeSort}(A[b:m])$ $\quad T(n/2)$

$\text{MergeSort}(A[m+1:e])$ $\quad T(n/2)$

Merge/Combine sorted $A[b:m)$ (L) sorted $A[m+1:e]$ (R)

$i=1$ (L) $\quad j=1$ (R) $\quad t=1$

if $L[i] < R[j]$

$C[t] = L[i]$,

$i = i+1$ $\qquad \Theta(n)$

else

$C[t] = R[j]$

L $\quad j = j+1$

$t = t+1$

until both arrays finished

R.T $\quad T(n) = \underbrace{T(n/2)}_{\text{L-sort}} + \underbrace{T(n/2)}_{\text{R-sort}} + \underbrace{n}_{\text{Merging}}$

$k=1 \quad 2T(n/2) + n$

$k=2 \quad = 2\left[2T(n/4) + n/2\right] + n = 4T(n/4) + 2n$

$k=3 \quad = 4\left[2T(n/8) + n/4\right] + 2n = 8T(n/8) + 3n$

$k=4 \quad = 8\left[2T(n/16) + n/8\right] + 3n = 16T(n/16) + 4n$

k pattern

$$= \boxed{2^k T\left(n/2^k\right) + kn}$$

last k: $\frac{n}{2^k} \simeq 1$ base $\Longleftrightarrow$ $k \simeq \log_2 n$

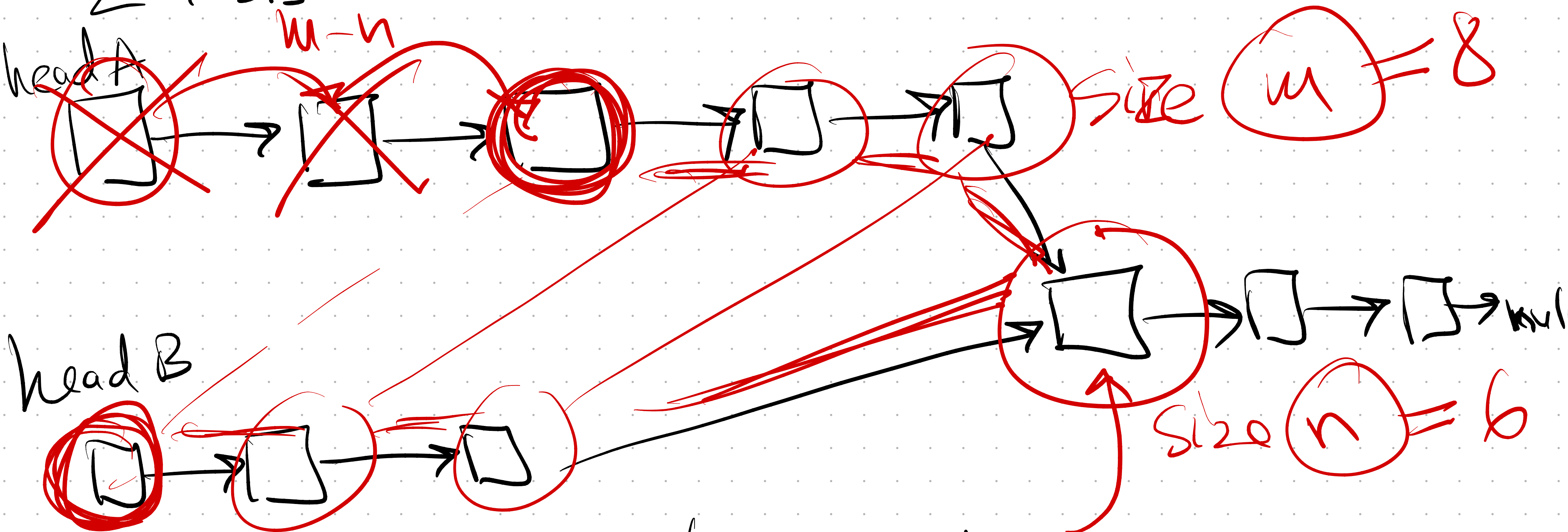
$T(n) = 2^{\log_2 n} T(1) + \log_2(n) \cdot n$

$= n \cdot \text{cnst} + n \log_2 n$

$\Theta(n \log_3 n)$.

Lecture 2 5/12/22

2 lists that intersect



Find the intersection address

naive $\Theta(m\,n)$

$m \geq n$

find m, n $\Theta(m)$

shift head A $\Theta(m)$

better: $\Delta = m - n$

head A advance Δ steps

check $\Theta(n)$

example

$$2^{\ln^2 n} \quad ? \quad n!$$

$$??\ 2^{\ln n \cdot \ln n} \quad ? \quad n! \qquad \overbrace{n \cdot n \cdots n}^{n \text{ time}} \ge n!$$

$$n^{\ln n} \quad ? \quad n!$$

$$\underbrace{n \cdot n \cdot n \cdots n}_{\ln n} \quad ? \quad 1 \cdot 2 \cdot 3 \cdots (n-1) \cdot n$$

$$\le \qquad \underbrace{(n/2) \cdot (n/2) \cdots (n/2)}_{n/2 \text{ time}}$$

$$n^{\log n} \overset{?}{\le} \left(\frac{n}{2}\right)^{n/2} \le n!$$

$$(ab)^x = a^x \cdot b^x$$

$$\left(\frac{n}{2}\right)^{\log n} \cdot 2^{\log n} \overset{\text{want}}{\le} \left(\frac{n}{2}\right)^{n/2}$$

$$n \leq \left(\frac{n}{2}\right)^{n/2 - \log n} \quad ?$$

$$n \leq \left(\frac{n}{2}\right)^{2} \leq \left(\frac{n}{2}\right)^{n/2 - \log n}$$

$$2 \leq n/2 - \log n$$

$$\frac{n}{2} - \log n - 2 \geq 0$$

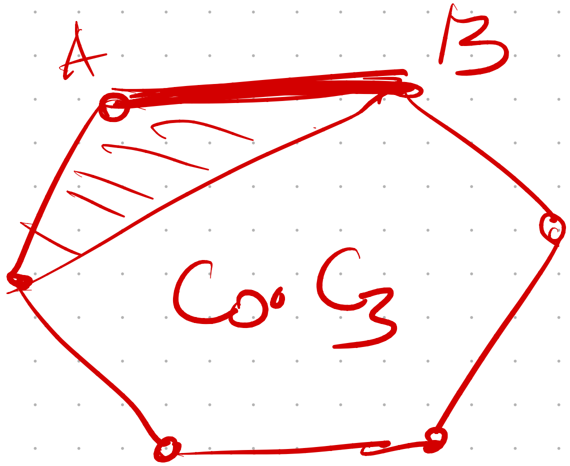
n Δ in n+2 sides polygon How many triangulations?

$n=4 \quad n+2=6$

$C_0=1$

$C_1=1$

C_2 □ $=2$



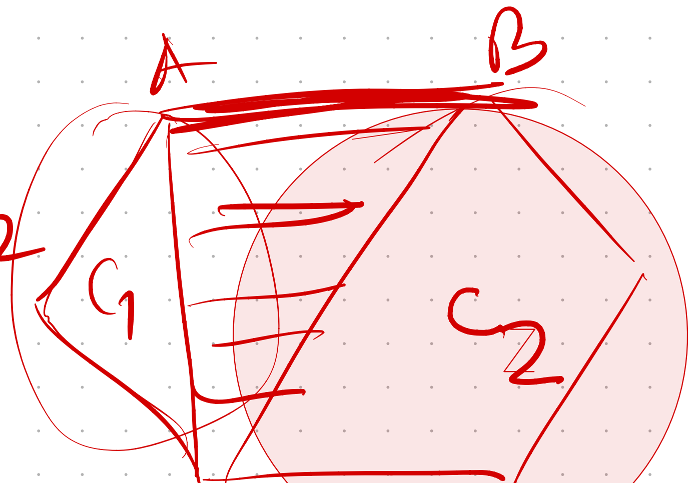
$C_4 = 14$

AB side in Δ

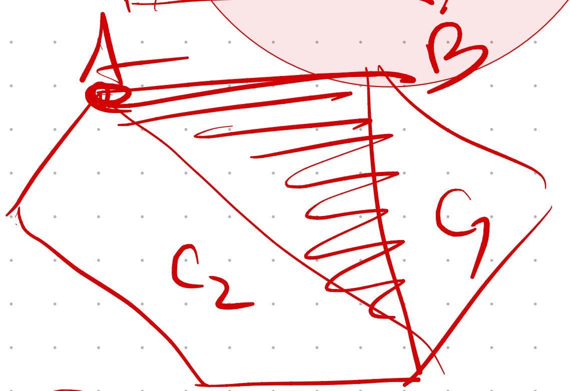
- any triangulation is in one of these 4 cases

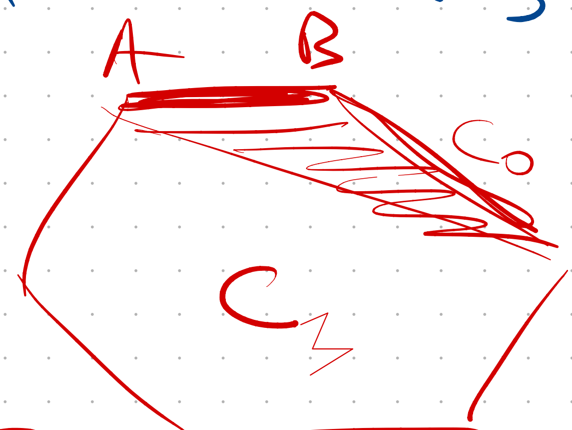
- no double counting
  any Δ is in exactly one of these cases

$C_1 \cdot C_2$



$C_2 \cdot C_1$

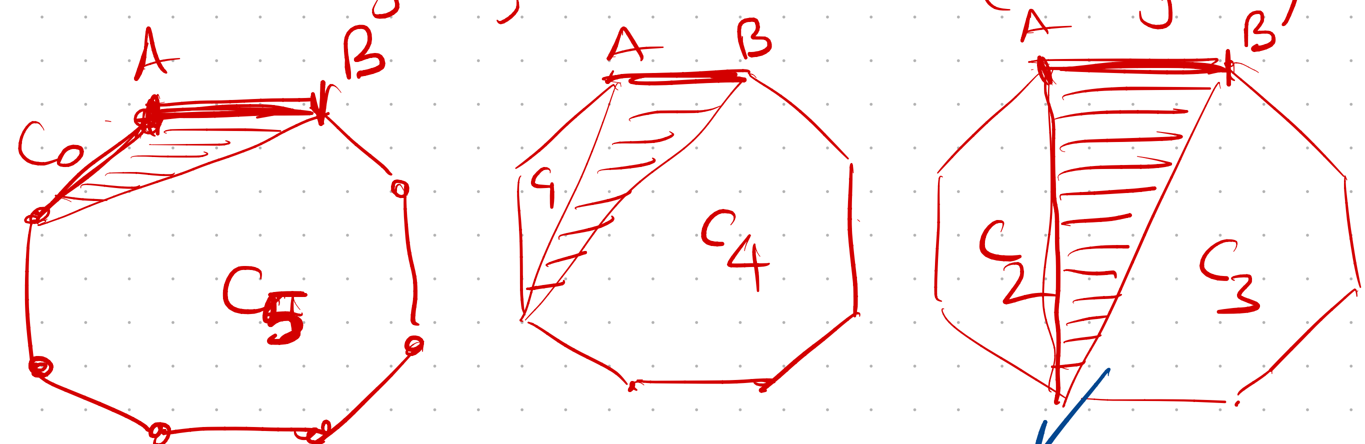




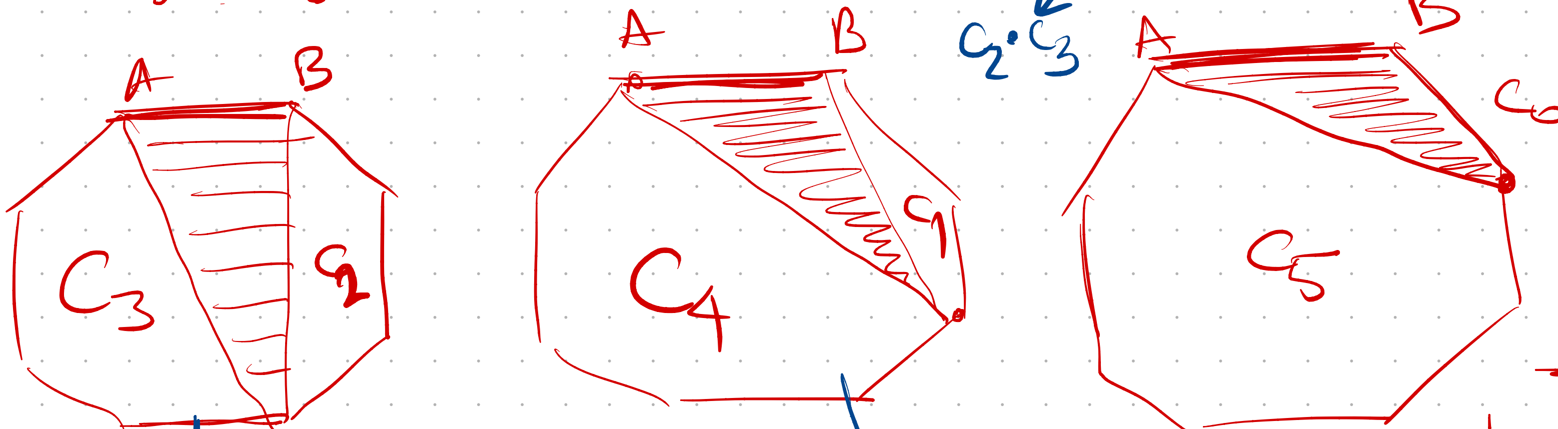
$$C_4 = C_0 \cdot C_3 + C_1 \cdot C_2 + C_2 \cdot C_1 + C_3 \cdot C_0$$

$n = 6$ triangles ; $n+2 = 8$ sides (octogon)

- Fix reference side AB
- that side AB can be part of 6 possible triangles, so 6 cases
  - disjoint cases so we can sum up for total
  - each case is counting possibilities for $\triangle$ on left/right side of ABX triangle



$C_2 \cdot C_3$



triangulations left side C_3 on x on right side C_2

$C_4 \cdot C_1$

$$C_6 = C_0 \cdot C_5 + C_1 \cdot C_4 + C_2 \cdot C_3 + C_3 \cdot C_2 + C_4 \cdot C_1 + C_5 \cdot C_0$$

$$C_n = C_{n-1}C_0 + C_{n-2}C_1 \cdots + C_0 C_{n-1}$$

$$= \sum_{k=0}^{n-1} C_{n-1-k} \cdot C_k$$

Triangulations (n_{max})

$C_0 = 1 \quad C_1 = 1$

For $n = 2 : max$ $\qquad O(n^2)$

$C_n = 0$

For $k = 0 : n-1$

$C_n = C_n + C_k \cdot C_{n-1-k}$

$$C_n = \text{Catalan}\#(n) = \binom{2n}{n} - \binom{2n}{n+1}$$