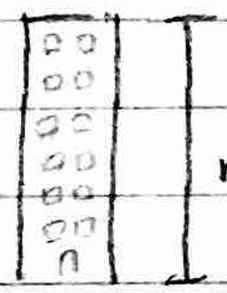
Algo
2/4/19

Recurrences I

Announcements
- email taoj@virginia.edu about write-up questions
- regrades w/n 1 week
- frivolous regrade policy
- TA feedback form

Egg Drop Experiment



Finding which floors it would be okay to drop an egg and it won't crack

Solution: 1/2 the floors each time until optimal floor found

Question of the Day: How to derive the time n log n?
- used in mergesort, binary search, & more

$$T(n) \begin{cases} T(n/2)+1 & n \geq 2 \quad \text{// searching the halves} \\ \underbrace{1}_{\#\text{ of steps}} & \underbrace{n=1}_{\text{input size}} \quad \text{// if given 1 input} \end{cases} \Big\} \text{binary search}$$

$$T(n) \begin{cases} T(n/2)+T(n/2)+n & n \geq 2 \\ 1 & n=1 \end{cases} \Big\} \text{mergesort}$$

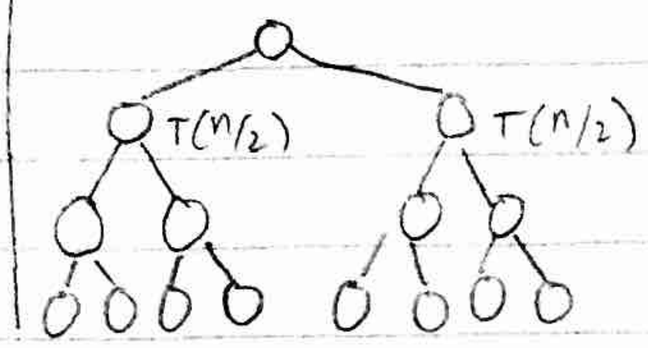
⇓

mergesort code
1) base case : 1
2) sort left } 2T(n/2)
3) sort right }
4) merge : 1 × n = n

Recurrence Relations
def: equation that recursively defines a sequence wher the next term is a function of the previous term
Ex: $T(N) = 2T(n/2) = n$

Recursion Tree - visual representation

Ex: Assuming we are recursing 4 times

iterations (i)	Problem Size (n)	Tree	Keeping Track
0	n		n
1	n/2		2(n/2) = n
2	n/4		4(n/4) = n
3	n/8		8(n/8) = n

Visual → Math (i = 2)

$T(n) = 2T(n/2) + n + \dots$

for $n = n/2$ // i = 1

$T(n/2) = 2T(n/2/2) + n/2 + \dots$

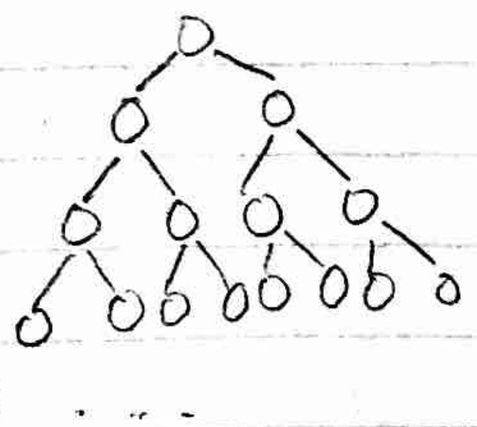
$T(n/2) = 2T(n/4) + n/2 + n$

$T(n) = 4T(n/4) + n + n$ // multiply 2 to each side

$T(n) = 4T(n/4) + 2n$ // i = 2; 2nd level of recursion in tree

↑ 4 nodes/parts ↑ 2 linear combination steps

Recurrence Tree Depth (another way to express the chart above)

i	n	Tree	Keeping Track
0	$n/2^0$ (n)		$2^0(n/2^0) = n$
1	$n/2^1$ (n/2)		$2^1(n/2^1) = n$
2	$n/2^2$ (n/4)		$2^2(n/2^2) = n$
3	$n/2^3$ (n/8)		$2^3(n/2^3) = n$
i	$n/2^i$	...	...

↳ ends when $n/2^i = 1$

$n = 2^i$

$\log_2(n) = i$

Given : $T(N) = 2T(n/2) + n$

$T(N) = (\log_2(n) + i)\, n$ // not too sure what happened here... I didn't copy this down fast enough

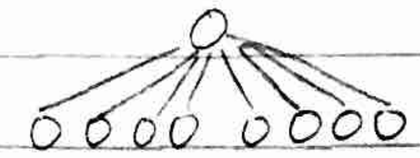
$T(N) = n\log_2(n+i)$

$T(N) \sim n\log_2 n$

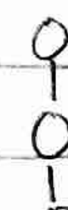
$T(N) = O(n\log_2 n)$

Example Drawings

a) $8T(n/2) + n^3$ (8 sub-parts; n^3 work $(n/2 \cdot 8)^3$; for $i=1$, n^3 is 2)

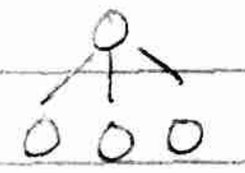
i	n	tree	Keeping Track
0	n		n
1	n/2		8 · n/2
2	n/4	// each node has 8 nodes	64 · n/4

b) $T(n/4) + n$ (1 sub-part)

i	n	tree	Keeping Track
0	n		n
1	n/4		n/4
2	n/16		n/16
3	n/32		n/32

Other Examples

a) Depth? $T(N) \begin{cases} 3T(n/3) + n & n \geq 3 \\ 1 & n < 3 \end{cases}$



$\log_3 n$

b) Worst-case? $T(N) = \begin{cases} 4T(n/2) + n & n \geq 2 \\ 1 & n = 1 \end{cases}$

$n/2^i = 1$ // is when recursion stops

depth: $\log_2(n) + 1$

worst-case: sum of all levels

$$\sum_{i=0}^{\log_2(n)} 2^i n = n \sum_{i=0}^{\log n(n)} 2^i \xrightarrow{\text{formulas we don't need to know}} = n\left(\frac{1-2^{\log_2(n)}}{1-2} + 1\right)$$ // derived from summation formula

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

$$= \frac{n-2n^2}{-1}$$

$$= 2n^2 - n$$

$$= O(n^2)$$