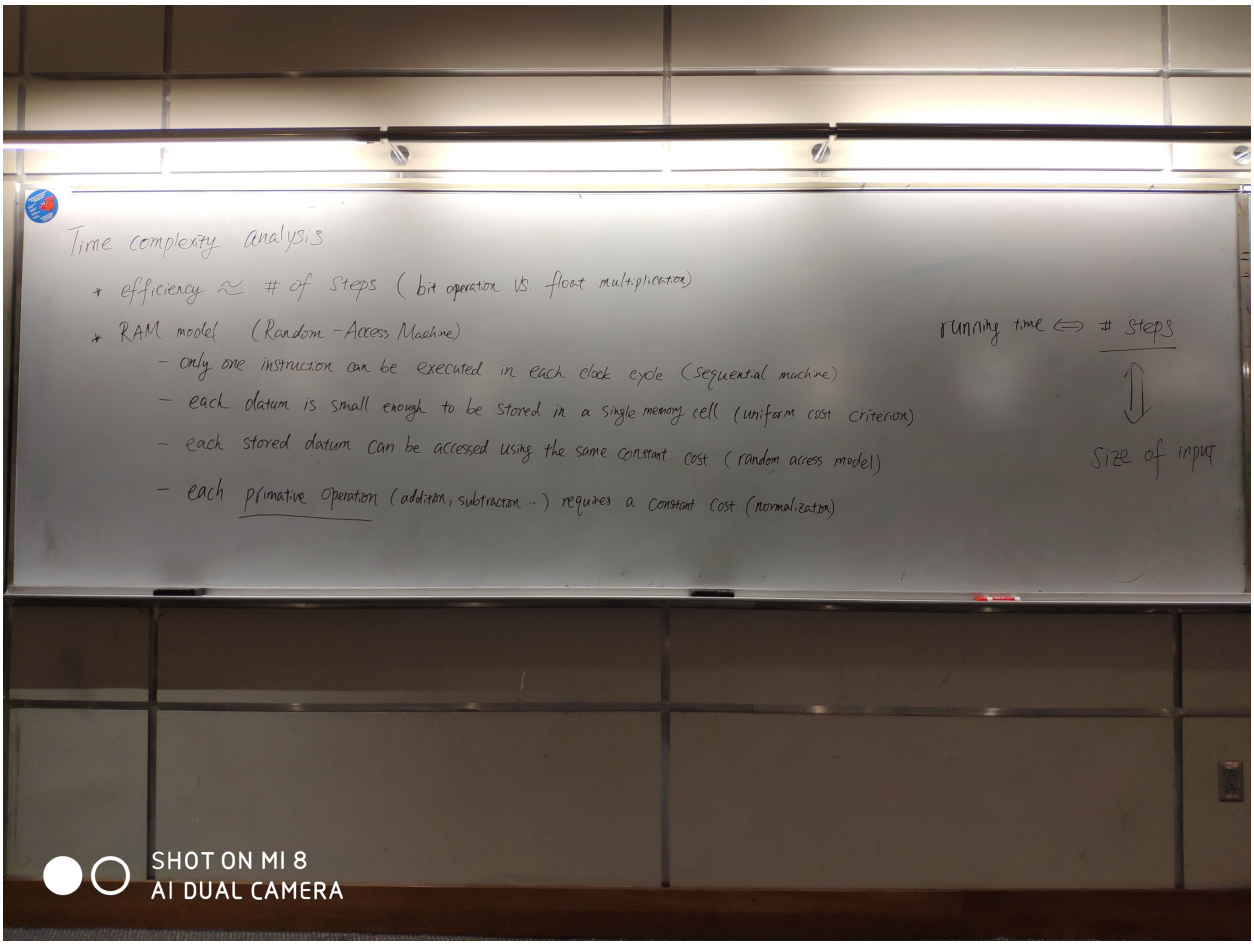
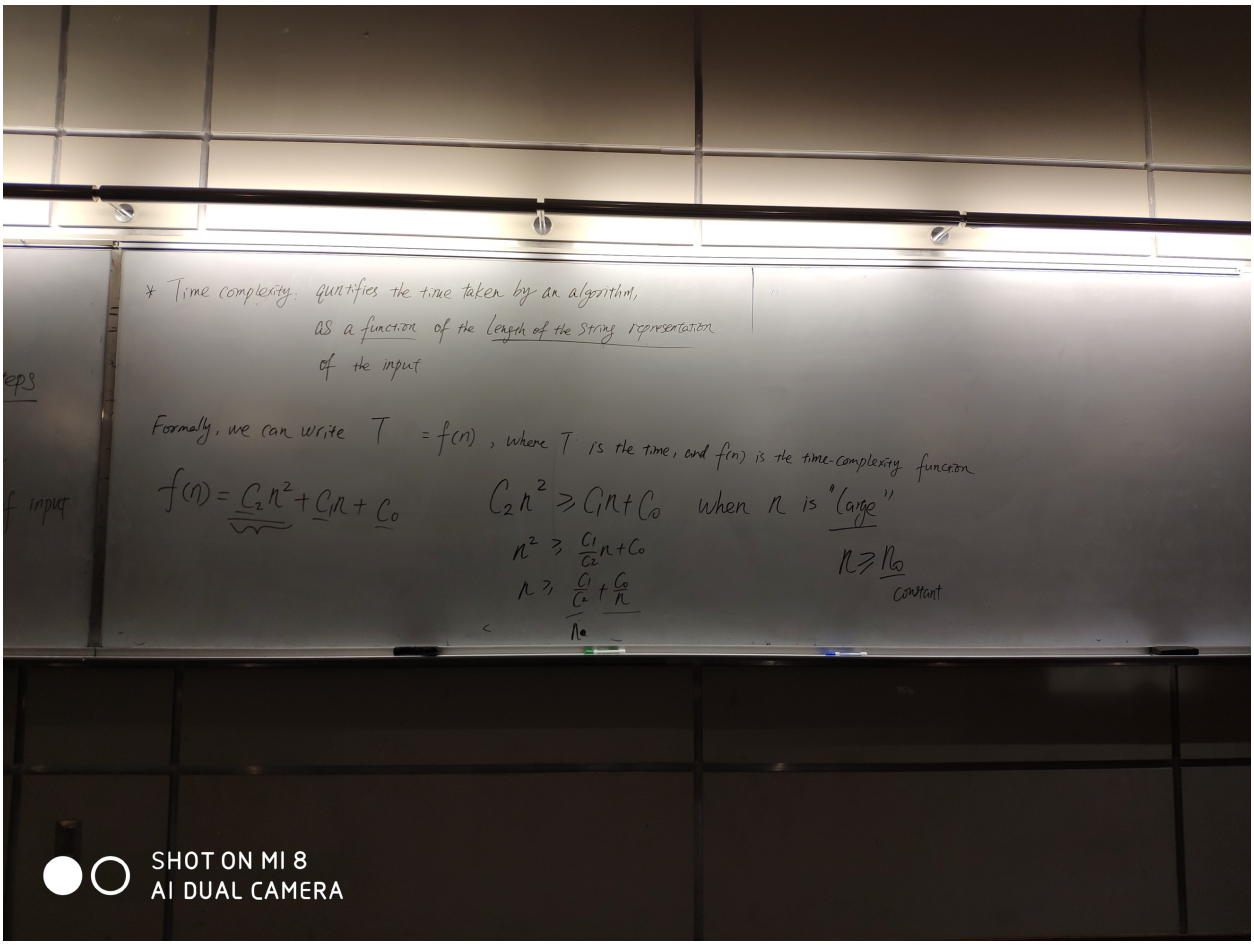
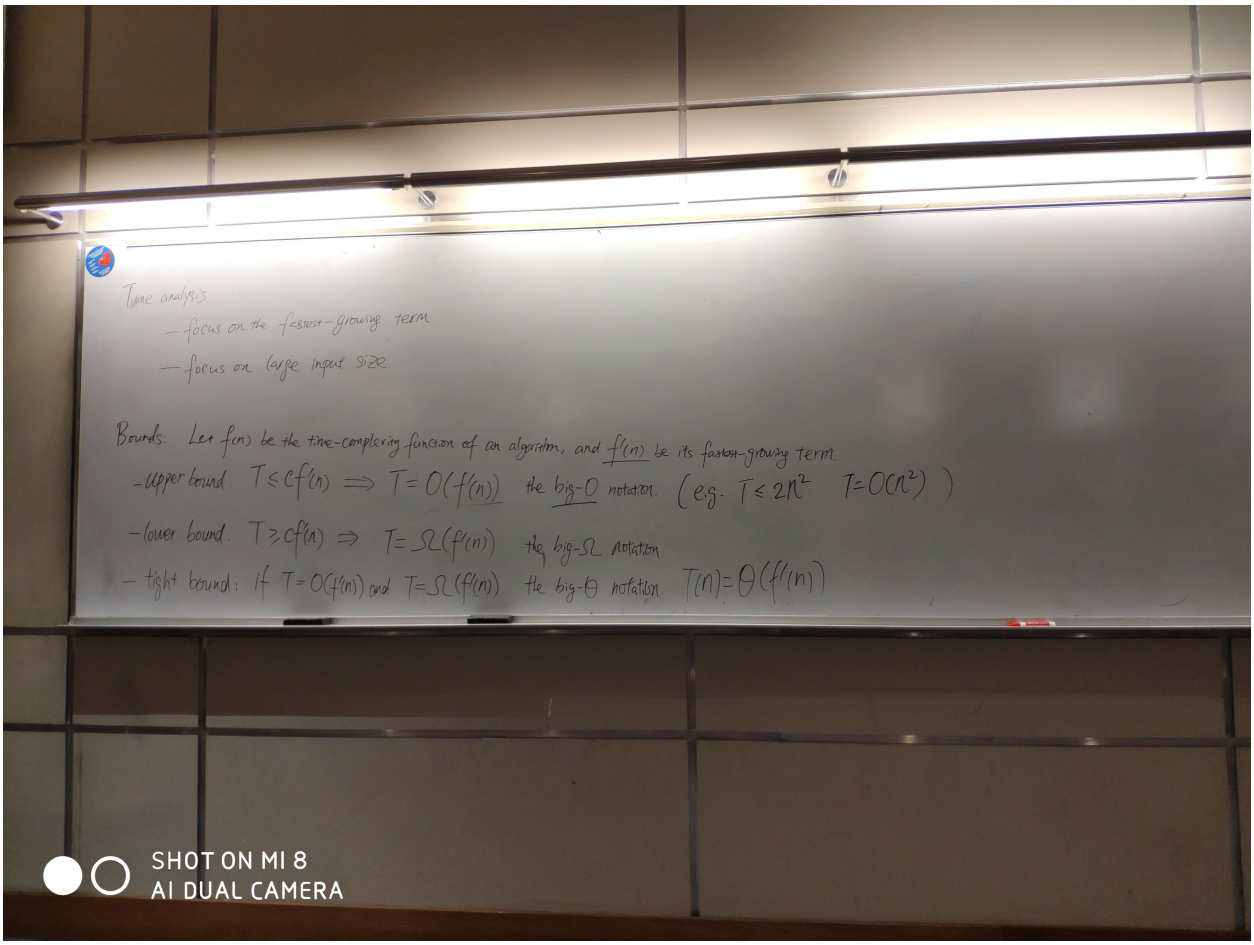
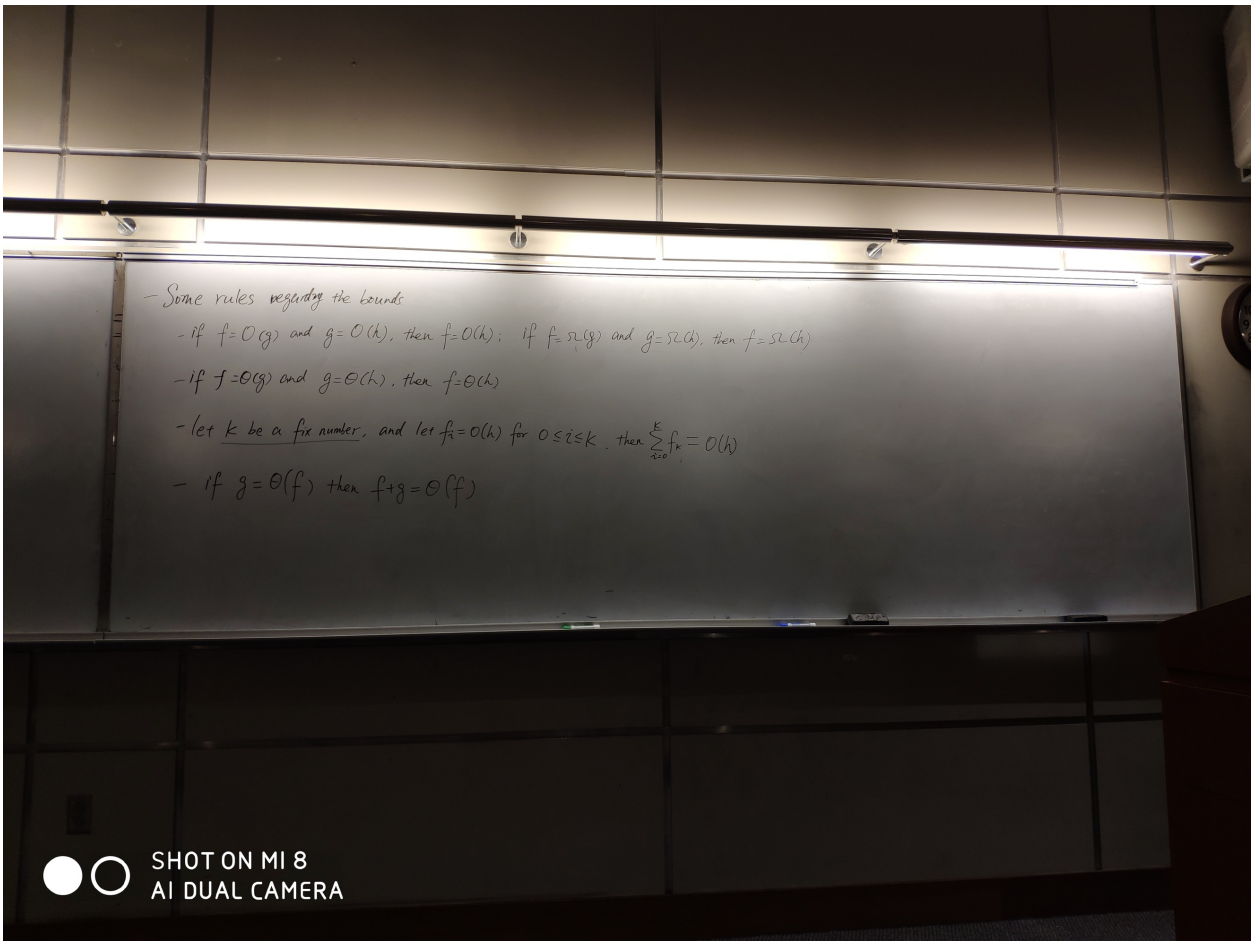
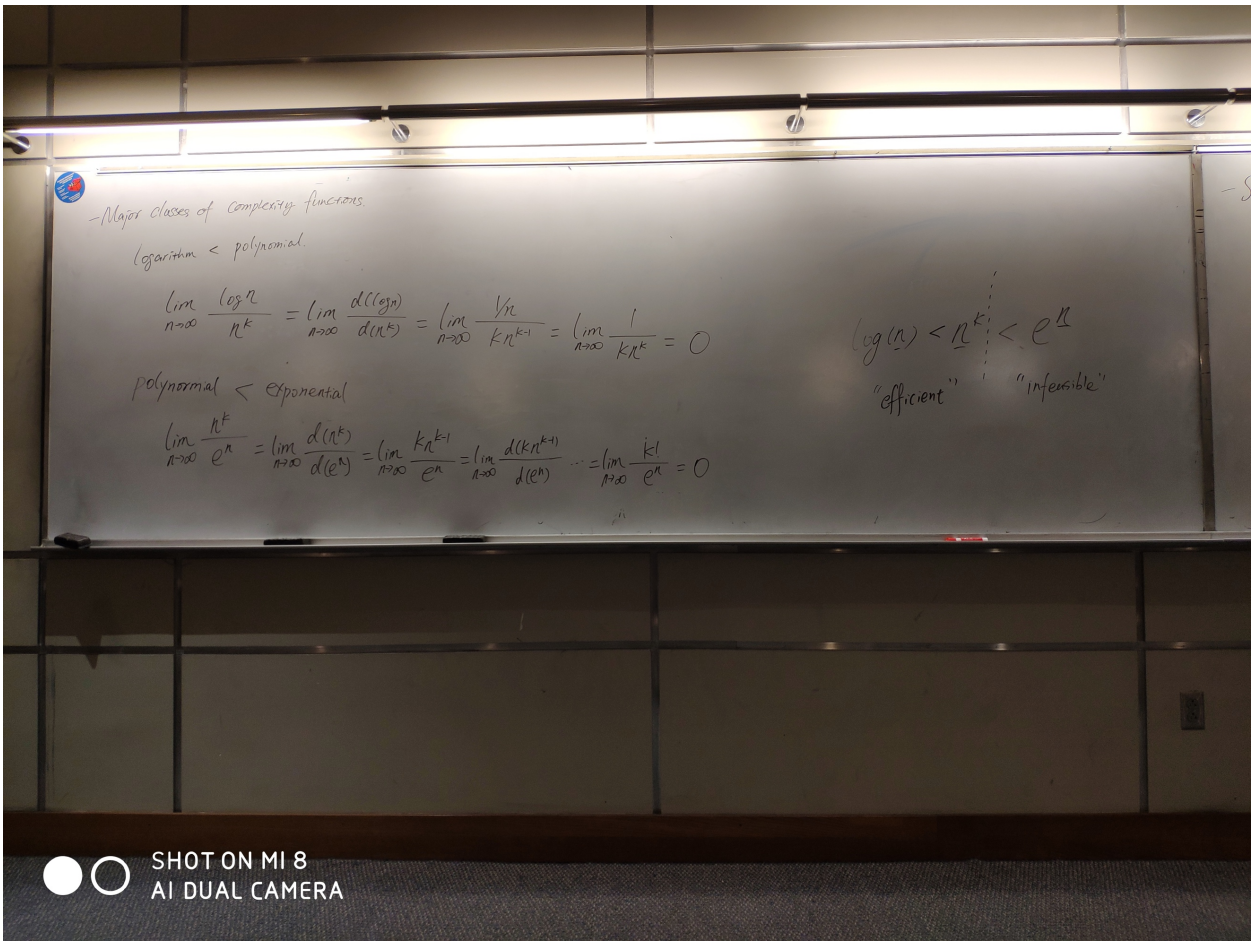
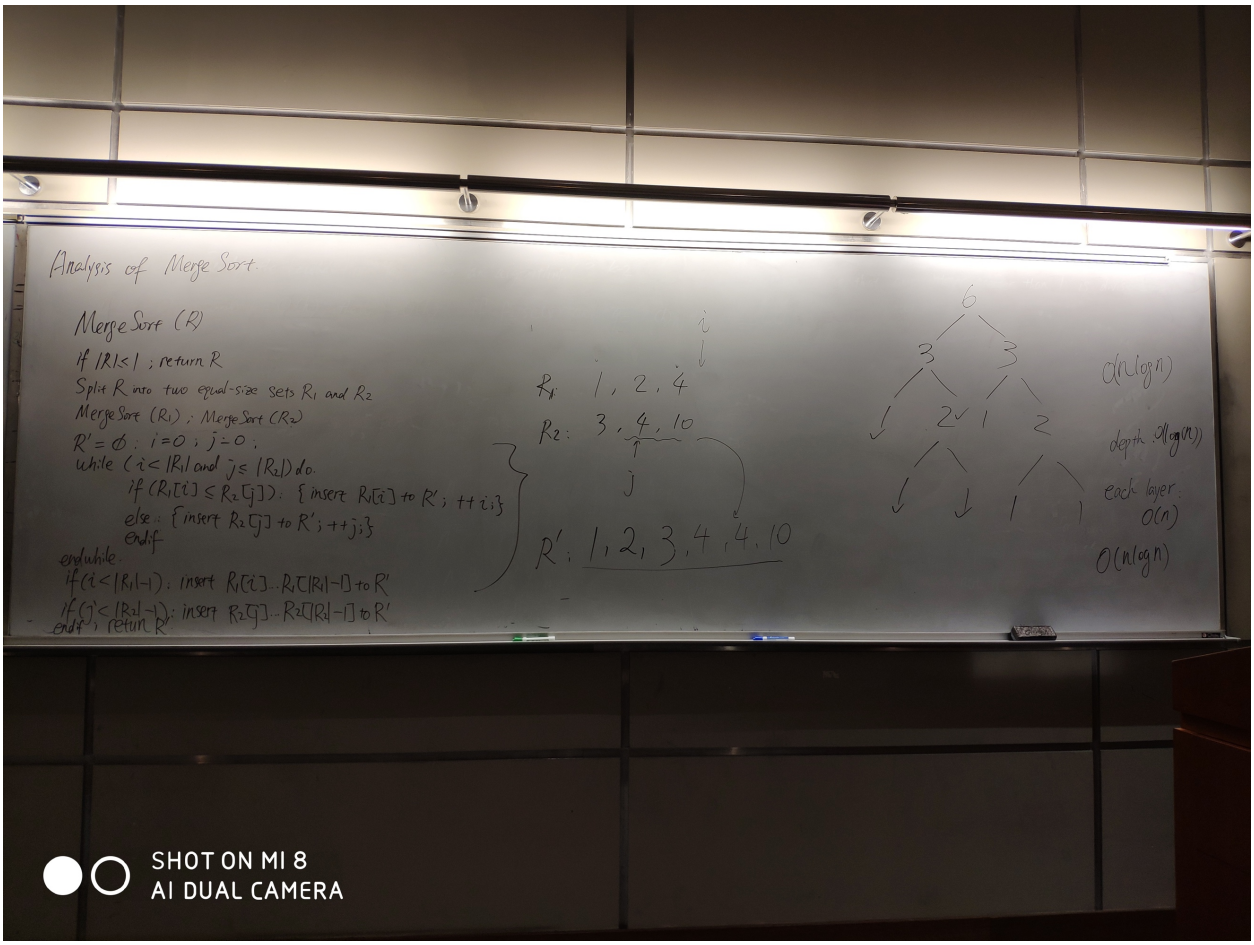

Time complexity analysis
* efficiency ≈ # of steps (bit operation VS. float multiplication)
* RAM model (Random-Access Machine)
- only one instruction can be executed in each clock cycle (sequential machine)
- each datum is small enough to be stored in a single memory cell (uniform cost criterion)
- each stored datum can be accessed using the same constant cost (random access model)
- each primative operation (addition, subtraction ...) requires a constant cost (normalization)
running time ⟺ # steps
Size of input
SHOT ON MI 8
AI DUAL CAMERA


* Time complexity: quantifies the time taken by an algorithm, as a function of the length of the string representation of the input
Formally, we can write T = f(n), where T is the time, and f(n) is the time-complexity function
f(n) = C2n^2 + C1n + C0
C2n^2 ≥ C1n + C0 when n is "large"
n^2 ≥ C1/C2 n + C0
n ≥ C1/C2 + C0/n
n0
n ≥ n0
constant
SHOT ON MI 8
AI DUAL CAMERA


Time analysis
— focus on the fastest-growing term
— focus on large input size
Bounds: Let f(n) be the time-complexity function of an algorithm, and f'(n) be its fastest-growing term.
-upper bound $T \le cf'(n) \Rightarrow T = O(f'(n))$ the big-O notation. (e.g. $T \le 2n^2$ $T = O(n^2)$)
-lower bound $T \ge cf'(n) \Rightarrow T = \Omega(f'(n))$ the big-Ω notation
— tight bound: if $T = O(f'(n))$ and $T = \Omega(f'(n))$ the big-Θ notation. $T(n) = \Theta(f'(n))$
SHOT ON MI 8
AI DUAL CAMERA


- Some rules regarding the bounds
- if $f=O(g)$ and $g=O(h)$, then $f=O(h)$; if $f=\Omega(g)$ and $g=\Omega(h)$, then $f=\Omega(h)$
- if $f=\Theta(g)$ and $g=\Theta(h)$, then $f=\Theta(h)$
- let k be a fix number, and let $f_i=O(h)$ for $0\le i\le k$, then $\sum_{i=0}^{k} f_k = O(h)$
- if $g=\Theta(f)$ then $f+g=\Theta(f)$
SHOT ON MI 8
AI DUAL CAMERA


-Major classes of complexity functions.
logarithm < polynomial
$\lim_{n\to\infty} \frac{\log n}{n^k} = \lim_{n\to\infty} \frac{d(\log n)}{d(n^k)} = \lim_{n\to\infty} \frac{1/n}{kn^{k-1}} = \lim_{n\to\infty} \frac{1}{kn^k} = 0$
polynormial < exponential
$\lim_{n\to\infty} \frac{n^k}{e^n} = \lim_{n\to\infty} \frac{d(n^k)}{d(e^n)} = \lim_{n\to\infty} \frac{kn^{k-1}}{e^n} = \lim_{n\to\infty} \frac{d(kn^{k-1})}{d(e^n)} \cdots = \lim_{n\to\infty} \frac{k!}{e^n} = 0$
$\log(n) < n^k < e^n$
"efficient"
"infeasible"
SHOT ON MI 8
AI DUAL CAMERA


Analysis of Merge Sort
MergeSort (R)
if |R|≤1 ; return R
Split R into two equal-size sets R1 and R2
MergeSort (R1) ; MergeSort (R2)
R' = ∅ ; i=0 ; j=0 ;
while (i<|R1| and j≤|R2|) do
if (R1[i] ≤ R2[j]): {insert R1[i] to R' ; ++i;}
else: {insert R2[j] to R' ; ++j;}
endif
endwhile
if (i<|R1|-1): insert R1[i]...R1[|R1|-1] to R'
if (j<|R2|-1): insert R2[j]...R2[|R2|-1] to R'
endif ; return R'
R1: 1, 2, 4
R2: 3, 4, 10
i
j
R': 1, 2, 3, 4, 4, 10
6
3
3
2
1
2
1
1
O(nlogn)
depth: O(log(n))
each layer: O(n)
O(nlogn)