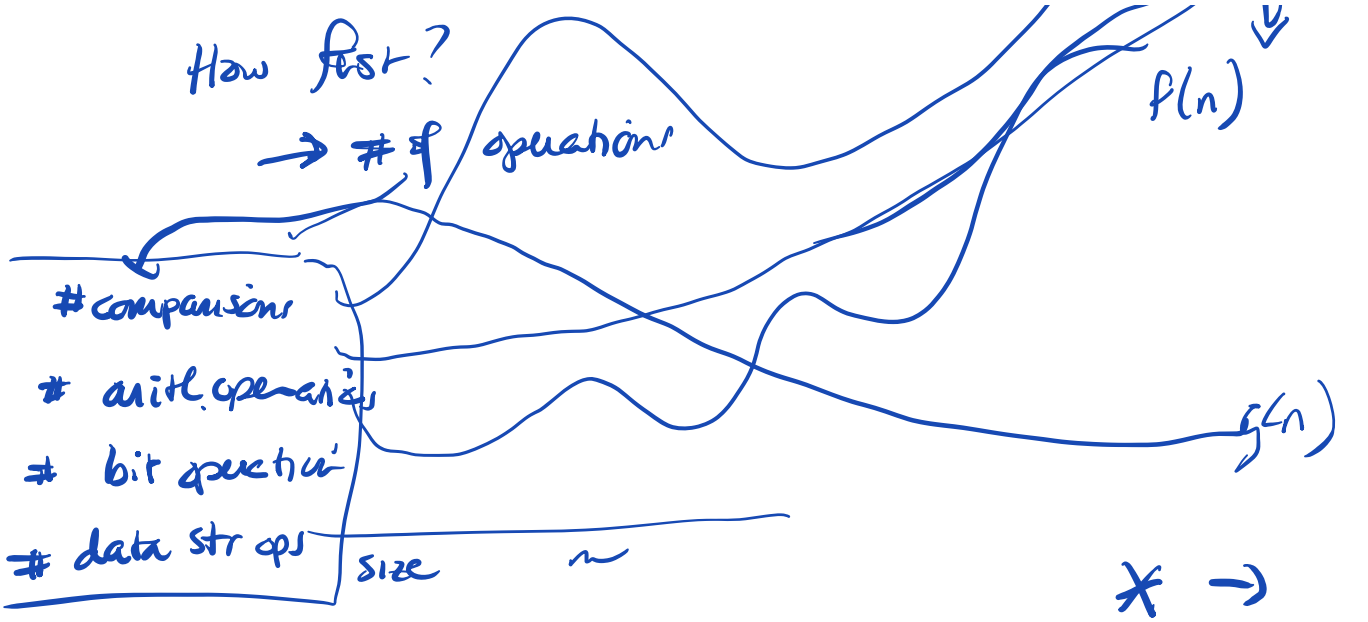
CS 3510 Algorithms

Binary search
DFS
sorting
Addition

Goals: Thinking abstractly
Thinking recursively
Design + Analysis of algorithms

How fast?
Correct?

I Divide and conquer
II Dynamic programming
III Graph algorithms
IV NP-completeness
V optional topics



Big-O notation

Defi $f(n)$ is $O(g(n))$

if $\exists c \geq 0$ s.t. $f(n) \leq c \cdot g(n)$

$\exists n_0$ $\forall n \geq n_0$

I Divide & conquer.

A) Divide the problem into smaller subproblems (of the same type)

B Solve subproblems recursively

C Merge the solutions to solve original

Merge sort

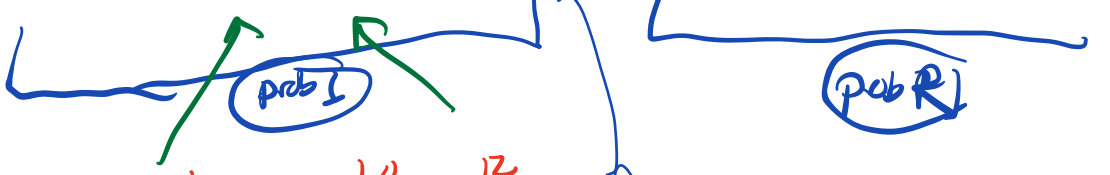
$n = 2^k$ some k

Given $a_1, \ldots, a_n$ ← distinct and we want to output them in order.

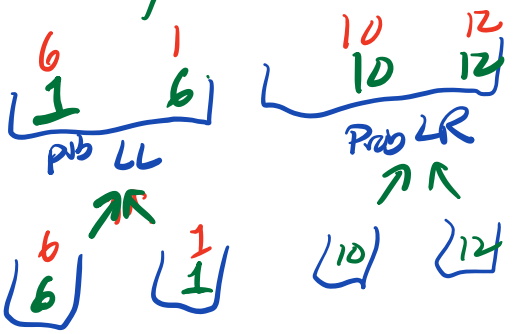
→ operation: is $a_i > a_j$?

sort: 6 1 10 12 3 8 7 2

1. Divide



2. Solve



3. Merge

merge

is 1 < 10 yes

1

is 6 < 10 yes

6

is 12 is next

1, 6, 10, 12 as the answer.

Merge:

merge

Is 1<2?

1

Is 6<2?

2

Is 6<3?

3

Is 6<7?

6

Correct?

sort $a_1 - a_n$

Sorting

correctly

merge the sorted lists

always gives a sorted list of n

How long?

Let $T(n)$ = the total # of operations

D+C step

$$T(n) \le 2\,T\left(\frac{n}{2}\right) + n$$

$$\Rightarrow T(n) \le 2\left(2\,T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

Note

$T(1) = 0$

$$= 4\,T\left(\frac{n}{4}\right) + 2\cdot\frac{n}{2} + n$$

$$= 4\,T\left(\frac{n}{4}\right) + 2n$$

$$\Rightarrow T(n) \le 8\,T\left(\frac{n}{8}\right) + 3n$$

$$\vdots$$

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

$$\Rightarrow T(n) \le \underline{?}\,T(1) + \underline{?}\,n$$

Guess $2^k\,T\left(\frac{n}{2^k}\right) + kn$

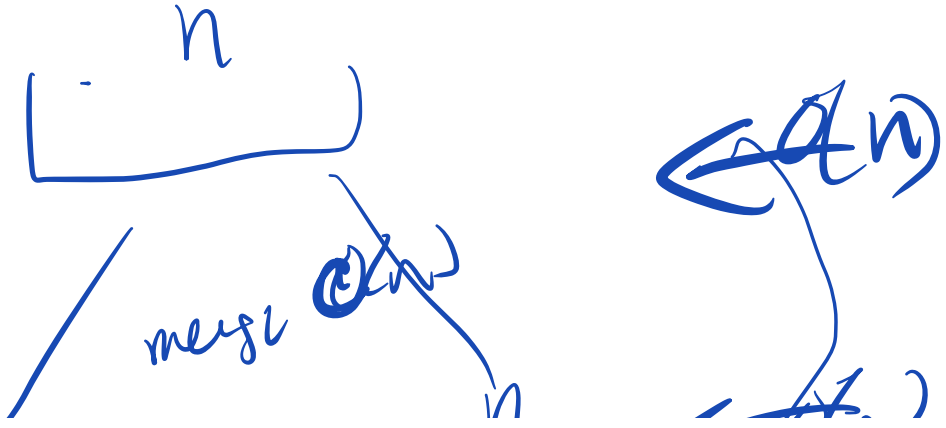
$T(n) = \alpha \quad T(2^k)$

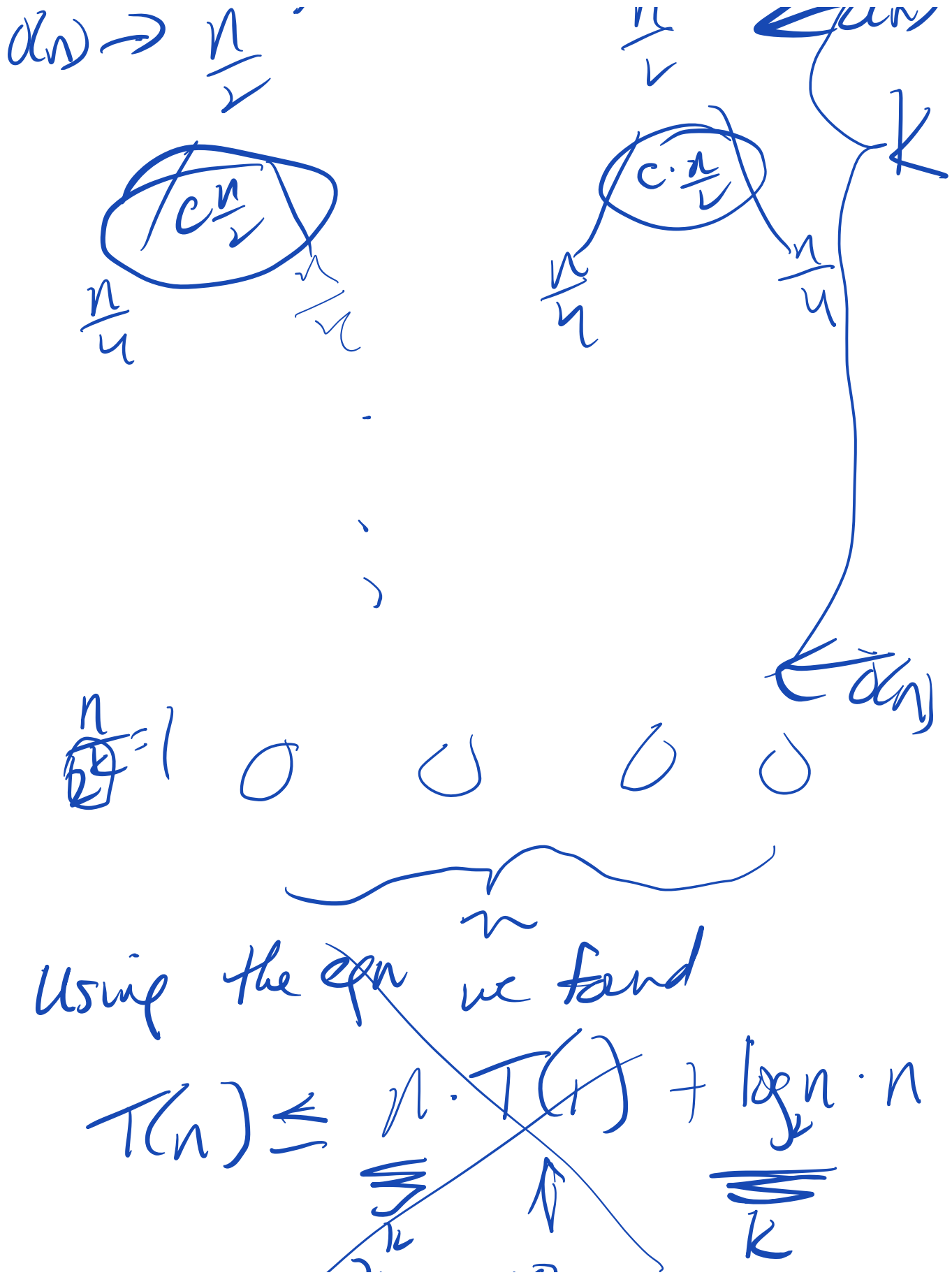
If $T(n) \le 2^k T\left(\frac{n}{2^k}\right) + kn$

then is $T(n) \le 2^{k+1} T\left(\frac{n}{2^{k+1}}\right) + (k+1)n$?

Prove easily with induction!

If we take $k = \log_2 n$




n/k
cn/k
n/k
n/k
n/k
c·n/k
n/k
n/k
k
cn
n/k = 1
n
Using the eqn we found
T(n) ≤ n · T(n/k) + log_k n · n

$= \log_2 n \cdot n$

$= k \cdot n$

↑ height of tree ↖ amt work per level

Technically $O(k \cdot n)$