

## CS-6515/4540 : Intro to Graduate Algos

- \* Instructor : Sahil Singla
- \* TAs - 9
- \* Most communication on Piazza
  - Questions, HWS, Class notes
- \* Office hours to be announced soon
- \* Syllabus on Canvas
- \* Grading : 75% exams + 25% HWs , best 3 out of 4 exams
- \* Waitlist : Name, PhD Program, email id, 9-digit gtid
- Today's plan : Intro to course, Big-O, Start D&C

### What is this Course about ?

Algorithms : Sequence of instructions to accomplish a task while minimizing resources used.

*time, space, communication*

- Goals:
- (1) No single idea for all problems but general approaches applicable to large classes of problems
  - (2) Learn cool math/proof techniques
  - (3) Theoretical lens: Economics, Maths, Physics, ....

### Big-O Notation

$f(n)$  denote amount of resources to solve a problem of size  $n$   $\leftarrow$  integer

*like # operations*

Defn 1:  $f(n) = O(g(n))$  means

$\exists$  constants  $c, x$  s.t.  $f(n) \leq c \cdot g(n)$  for  $n > x$

$\exists$  constants  $c, x$  s.t.  $f(n) \leq c \cdot g(n)$  for  $n > x$

Eg:  $n^2 + 2n = O(n^2)$

$n^2 + 2n = O(n^3)$

Note:  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq c$

Defn 2:  $f(n) \approx \Omega(g(n))$  if  $g(n) = O(f(n))$

Defn 3:  $f(n) = o(g(n))$  (little-o)

Similarly little-w

If  $f(n) = O(g(n))$  but  $g(n) \neq O(f(n))$  i.e.  $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} \rightarrow \infty$

Defn 4:  $f(n) \approx \Theta(g(n))$  if  $f(n) = O(g(n))$  and  $g(n) = O(f(n))$

Eg 1:  $1 = o(\log \log n) = o(\log n) = o(n^{0.1}) = o(n) = o(2^n) = o(2^{2^n})$

Eg 2:  $n! = o(n^n)$  since  $\lim_{n \rightarrow \infty} \frac{n^n}{n!} \rightarrow \infty$

Exercise:  $\log(n!) = \Theta(\log n)$

## Divide and Conquer

- (1) Break pb into smaller subproblems
- (2) Recursively solve each subproblem
- (3) Combine all subprob solns to obtain soln to orig prob

## Merge Sort

Given  $n$  numbers  $a[0], a[1], \dots, a[n-1]$

We will, usually, assume  $n$  is a power of 2 (say by adding 0s)

Goal: Rearrange in increasing order

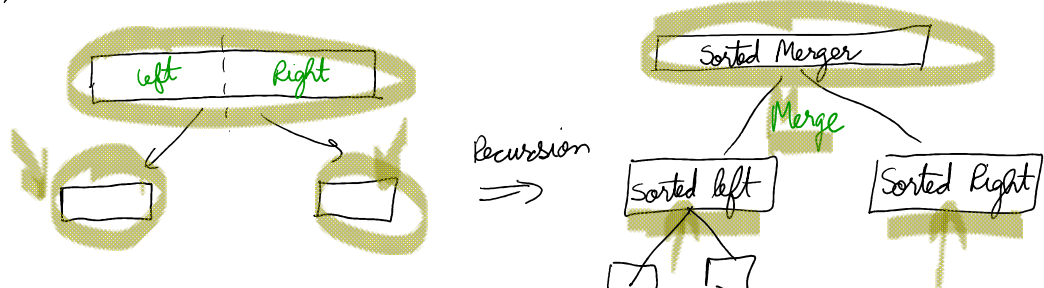
Add, Mult, compare, swap  $\leftarrow \Theta(1)$  time

Algo: (1) Break into  $a[0, \dots, \frac{n}{2}-1]$  and  $a[\frac{n}{2}, \dots, n-1]$

Algo: (1) Break into  $a[0, \dots, \frac{n}{2}-1]$  and  $a[\frac{n}{2}, \dots, n-1]$

(2) Recursively solve each

(3) Combine sorted left & right halves



Exercise: Show how to merge 2 sorted arrays of size  $n$  into a sorted array of size  $2n$  in  $O(n)$  time.

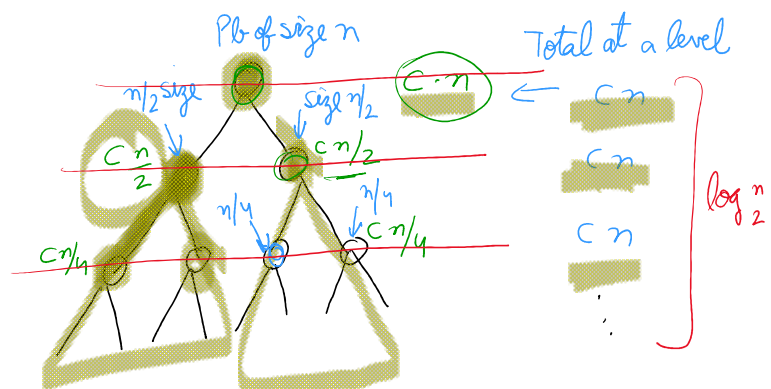
Correctness: Easy to see but formal proof by induction (more on induction next lecture)

Time: 
$$T(n) = \underbrace{2 \cdot T(n/2)}_{\text{Solve subpb}} + \underbrace{C \cdot n}_{\text{Merge}} \quad \text{with } T(1) = 1$$

How to Solve Recursion?

Tree Method:

Draw a tree where nodes denote additional work in merging



Since  $\log_2 n$  levels,  
time in  $O(n \log n)$