CS 2100

Sorting (pt 2)

Recap

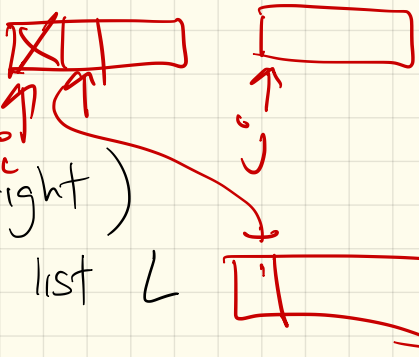
- Lab tomorrow
- HW due Friday
- Midterms almost done!
  (come tommorrow!)
- HW on Lists - up by tomorrow, due Friday the 22nd
- Also: reading due Monday after break
- No class on Friday
- Check blackboard over break

Merge Sort:

if length of A is > 2

divide in half
Mergesort (left)
Mergesort (right)
Merge (left + right)
return list

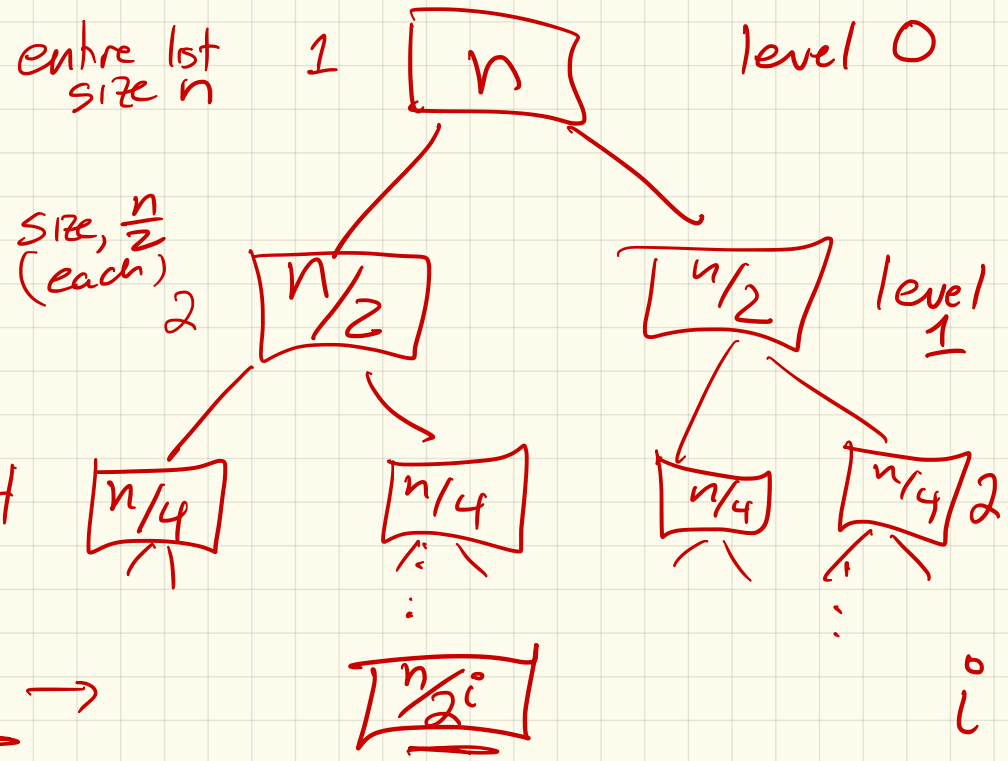
else // (list of length 0 or 1
done.



Merge (left, right)
create empty list L
i ← 1, j ← 1
while (i or j < SIZE)
put smallest one next in L

Runtime: # of comparisons

$$M(n) \leq 2M\left(\frac{n}{2}\right) + n$$



$$\frac{n}{2^\ell} = 1 \Rightarrow n = 2^\ell$$

$$\log_2 n = \ell$$

$$\#\text{ comps} = \sum_{i=0}^{\ell} (\#\text{ nodes on level } i)(\#\text{ comps inside a single node})$$

$$= \sum_{i=0}^{\lceil \log_2 n \rceil} (2^i)\left(\frac{n}{2^i}\right)$$

$$= \sum_{i=0}^{\log_2 n} n$$

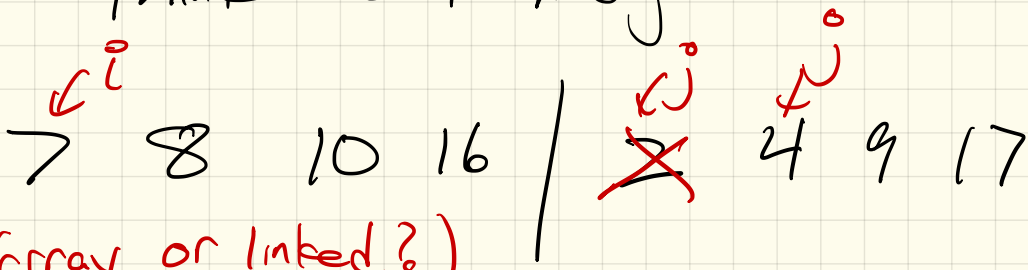
$$= \underbrace{n + n + n + \cdots + n}_{\log_2 n}$$

$$= O(n \log n)$$

Issue: space!

How much do we need?

Think about merge:



(array or linked?)

L: 2

↑

new spot

→ need new array!

linked: insert (before) first item, j

Radix Sort

Consider a list of numbers:

28, 11, 265, 63, 22, 58, 85, 80

"Sort" first by 1's digit:

80 11 22 63 265 85 58 28

0 1 2 3 5 8

Then by 10's:

10 20 36

Then by 100's:

Best application: Strings!

How many "buckets"?

26 buckets

How many phases: length of longest string

Can show this will always work:

- Last round put in order of most significant digit.
- Then 2nd to last round ensured correct order
- etc

Formally - induction

Runtime: digits
(# of ~~bits~~)(n + b)
rounds
"buckets"

Looks linear!
(Is it?)

Largest # = X
$\log_2 X$ bits to write it down

Takeaways

- Sorting is a fundamental CS problem
- Many ways + "optimal" can depend upon the data + the setting
- Low level issues can drastically affect speed
- Other constraints: space, size of each value, etc.

(array vs. linked)