

CS3000
5/15 - Mon

Admin

- Long HW1 due tom. 9pm
- Rec 2 tom.
- Fun recitation Thurs
- Short HW1 grades out later today
(late + see low score: don't freak out!)

Agenda

1. mergesort overview
2. mergesort steps
3. merge steps

Recap

- Binary Search + Karatsuba use what technique?
Divide + Conquer
- Steps of that technique?
Divide, conquer, combine
- Best/worst case runtime of binary search
 \downarrow \uparrow
 $\Theta(n)$ $\Theta(\log n)$ $n = \# \text{ elements in tree}$
- runtime of Karatsuba
 $\Theta(n^{1.59})$ where $n = \# \text{ digits}$

1. Mergesort Overview

- sorting algorithm
- ordering of array elements such that $a_1 \leq a_2 \leq a_3 \leq \dots \leq a_n$

• So far: Selection Sort $O(n^2)$

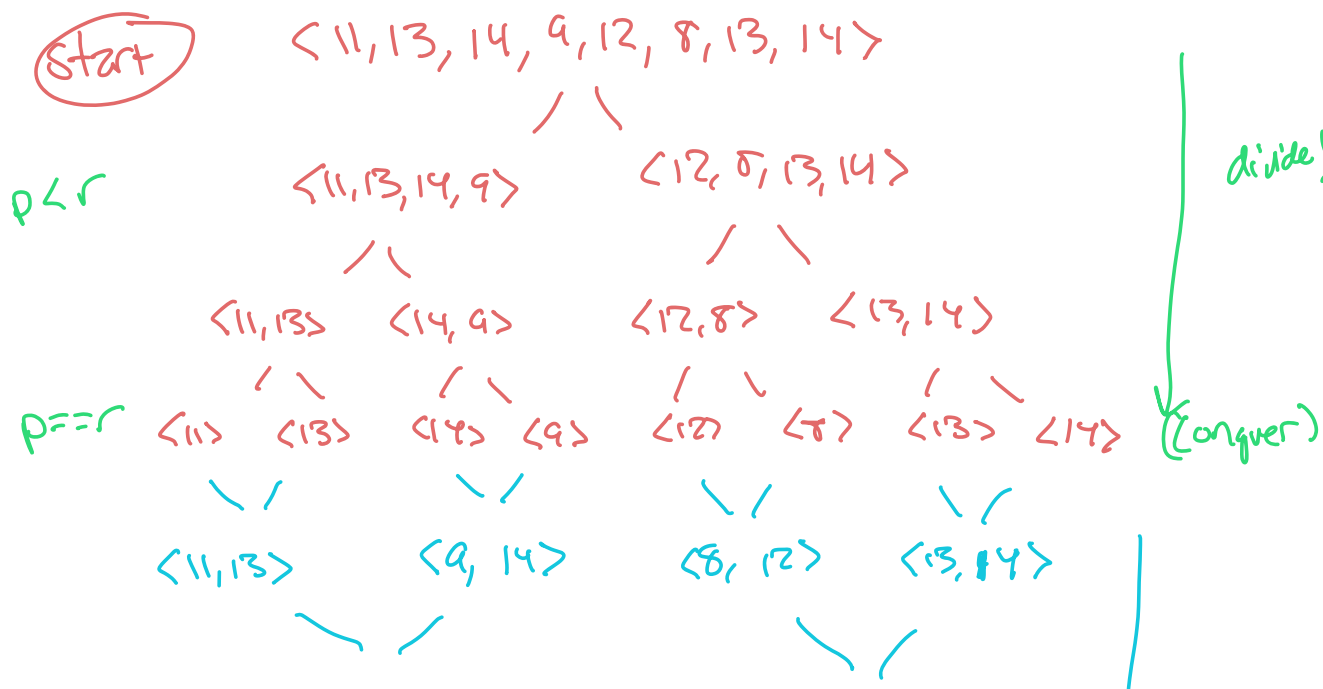
!!!
(zn we do better?)
!!!

↳ apply a known technique
like: Divide + Conquer

input: array A

(ex) Start: $\langle S, K, A, 9, Q, 8, K, A \rangle$

Replace with #s:



$\langle 9, 11, 13, 14 \rangle$ $\langle 8, 12, 13, 14 \rangle$

$\langle 8, 9, 11, 12, 13, 13, 14, 14 \rangle$

merge
(combine)

(merge two sorted
subarrays) →

2. mergesort steps

- mergesort function
- parameters: A, p, r
 ↖ ↗ right index
 ↘ ↙ left index
- returns: nothing (modifies A)
- don't make copies of A (yet 'i')

MERGESORT(A, p, r)

- If not in base case (subarray is length ≥ 1)
- Then, split into left / right halves → ???

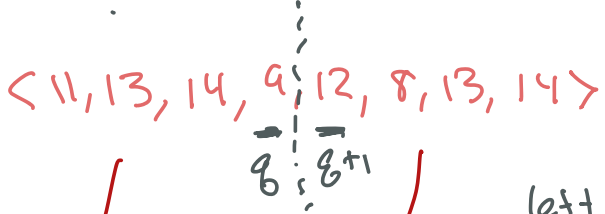
MERGESORT(left half)

MERGESORT(right half)

MERGE(left, right) → ??

A
P } indices
r
l

How do we split into left and right?



p=1
r=8
q=4

left half $A[p..q]$
right half $A[q+1..r]$

<11, 13, 14, 9>

<12, 8, 13, 14>

p=1
r=4
q=2

p=5
r=8
q=6

$q = \lfloor (p+r)/2 \rfloor$

mid point

Follow-up Question

- when do we stop?
- How long is the subarray?
- How do we know?

↳ stop when subarray is length 1

• when $p=r$

$A[p] = A[r] =$ one element

• when $p > r$?

empty array

MERGESORT(A, p, r)

if $p < r$

$$q = \lfloor (p+r)/2 \rfloor$$

divide

MERGESORT(A, p, q)

→ left half

MERGESORT(A, q+1, r)

→ right half

(conquer)

MERGE(A, p, q, r)

→ merge sorted ???

combine

left + right halves

$$T(n) = 2 \cdot T(n/2) + \text{merge?}$$



↳ 2 mergesort calls

on left half, and right half

$$T(1) = 1$$

10:51

3. Merge Step

1. How does the merge step work?

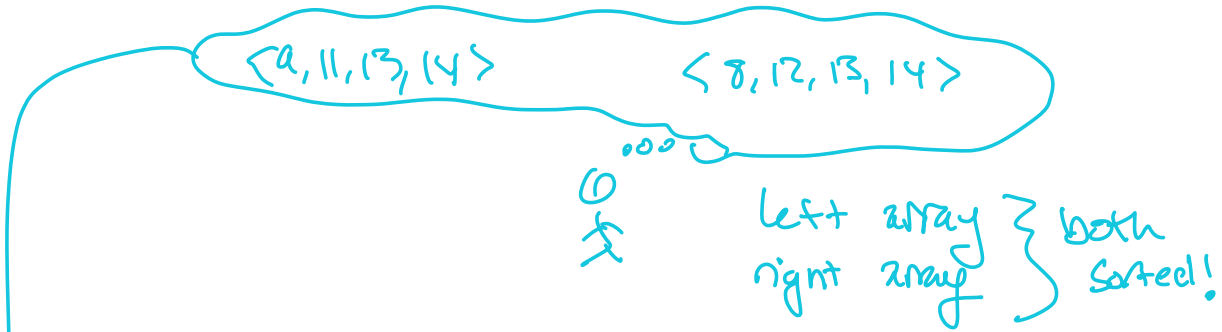
2. How long does the merge step take?

↳ mergesort runtime is

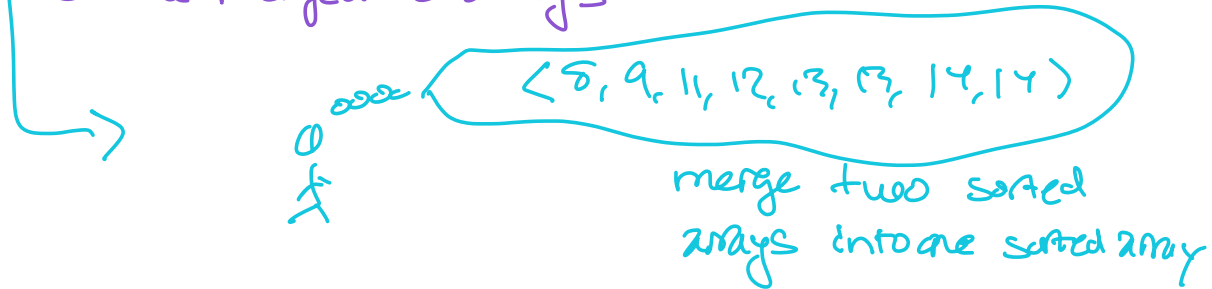
$$T(n) = 2T(n/2) + (\text{merge?})$$

The merge step:

- Given an array (A) left index (p) , midpoint (q) right index (r)



- modify A from p to r , to be the merged subarrays $\rightarrow A[p..r]$ should be sorted



MERGE(A, p, q, r)

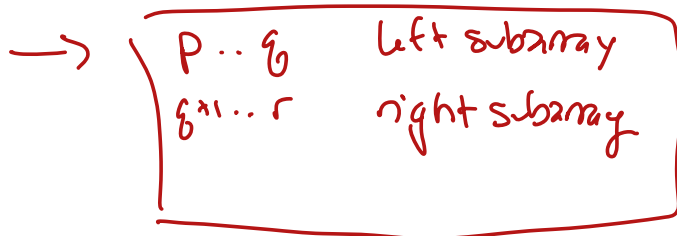
new array $L = \text{copy of } A[p..q]$

new array $R = \text{copy of } A[q+1..r]$

$i = 1$ \rightarrow index into L

$j = 1$ \rightarrow index into R

$k = p$ \rightarrow index into A



(ex) $A = \langle 11, 13, 9, 14 \rangle$
 $p = 1$ $q = 2$ $r = 4$

$L = \langle 11, 13 \rangle$
 $i = 1$ ↑

$R = \langle 9, 14 \rangle$
 $j = 1$ ↑

want

$A = \langle a, 11, 13, 14 \rangle$

MERGE(A, p, l, r)

new array $L = \text{copy of } A[p..l]$ (put ∞ in extra place)

new array $R = \text{copy of } A[l+1..r]$ (put ∞ in extra place)

$i = 1$

$j = 1$

$k = p$

while ...

if $L[i] < R[j]$

$A[k] = L[i]$

$i = i + 1$

$k = k + 1$

→ left value smaller

→ put smaller thing into "next" A position

else → right value is smaller

$A[k] = R[j]$

$j = j + 1$

$k = k + 1$

→ put smaller thing into "next" A position

(ex) $A = \langle 11, 13, 9, 14 \rangle$
 $p = 1$ $r = 4$

$L = \langle 11, 13 \rangle$

$R = \langle 9, 14 \rangle$

$i=1$
 $j=1$
 $k=1$

compare $L[1]$ vs. $R[1]$

- update $A[k] = R[1]$
- $k=2, j=2$

$A = \langle 9, 13, 9, 14 \rangle$

$i=1$
 $j=2$
 $k=2$

compare $L[1]$ vs. $R[2]$

- update $A[k] = L[1]$
- $k=3, i=2$

$A = \langle 9, 11, 9, 14 \rangle$

$i=2$
 $j=2$
 $k=3$

compare $L[2]$ vs. $R[2]$

- update $A[k] = L[2]$
- $k=4, i=3$

$A = \langle 9, 11, 13, 14 \rangle$

$i=3$
 $j=2$
 $k=4$

compare $L[3]$ vs. $R[2]$

- update $A[k] = R[2]$

$A = \langle 9, 11, 13, 14 \rangle$

merge step = $\Theta(n)$

- in the loop, k gets incremented every time
- we iterate over A from p to r

overall mergesort: $T(n) = 2 \cdot T(n/2) + \text{merge}$

v

$$= 2 \cdot T(n/2) + \Theta(n)$$

$$= \Theta(n \lg n)$$

→ average runtime