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
  $\Theta(1)$ $\Theta(\lg n)$ n = # elements in tree
- run time of Karatsuba
  $\Theta(n^{1.59})$ where n = # digits

1. Mergesort Overview

- sorting algorithm
- ordering of array elements such that $\boxed{a_1 \le a_2 \le a_3 \le \dots \le a_n}$
- So far: Selection Sort $\boxed{\Theta(n^2)}$

Can we do better?

↳ apply a known technique

like: $\boxed{\text{divide + conquer}}$

input: array A

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

Replace with #s:

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

$p < r$ $\langle 11, 13, 14, 9 \rangle$ $\langle 12, 8, 13, 14 \rangle$

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

$p == r$ $\langle 11 \rangle$ $\langle 13 \rangle$ $\langle 14 \rangle$ $\langle 9 \rangle$ $\langle 12 \rangle$ $\langle 8 \rangle$ $\langle 13 \rangle$ $\langle 14 \rangle$ (conquer)

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

divide!

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

merge (combine)

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

(merge two sorted subarrays)

2. Mergesort Steps

- mergesort function
- parameters: A, p, r

p → left index

r → right index

- returns: nothing (modifies A)
- don't make copies of A (yet :))

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

How do we split into left and right?

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

q | q+1

p=1
r=8
q=4

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

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

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

divide $\begin{cases} q = \lfloor (p+r)/2 \rfloor \\ \text{MERGESORT}(A, p, q) \rightarrow \text{left half} \\ \text{MERGESORT}(A, q+1, r) \rightarrow \text{right half} \end{cases}$

(conquer)

combine ← MERGE(A, p, q, r) → merge sorted ??? 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

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

The merge step:

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

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

left array
right array } both sorted!

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

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

merge two sorted arrays into one sorted array

MERGE(A, p, q, r) → $\boxed{\begin{array}{ll} p..q & \text{left subarray} \\ q+1..r & \text{right subarray} \end{array}}$

new array L = copy of $A[p..q]$

new array R = copy of $A[q+1..r]$

$i = 1$ ⟶ index into L

$j = 1$ ⟶ index into R

$k = p$ ⟶ index into A

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

$p = 1$ $q = 2$ $r = 4$

$L = \langle 11, 13 \rangle$ $R = \langle 9, 14 \rangle$

$i = 1$ $j = 1$

want

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

MERGE(A, p, q, r)

new array L = copy of A[p..q] (put ∞ in extra place)

new array R = copy of A[q+1..r] (put ∞ in extra place)

i = 1

j = 1

k = p

while ... ??

  if L[i] < R[j] → left value smaller

    A[k] = L[i] → put smaller thing into "next" A position

    i = i+1

    k = k+1

  else → right value is smaller

    A[k] = R[j] → put smaller thing into "next" A position

    j = j+1

    k = k+1

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

$p = 1$ $r = 4$

$L = \langle \cancel{11}, 13 \rangle$

$R = \langle \cancel{9}, 14 \rangle$

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

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 $\overset{\infty}{L[3]}$ 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: $T(n) = 2 \cdot T(n/2) +$ merge

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

$= \Theta(n \lg n)$

average runtime