CS1800

12/1 - Fri :)

last lecture :(

Admin

- HW8 are tonight
- HW9 are 12/5
- exam #3 12/5, 30 min exam
  50 mins to compute

Agenda

1. Mergesort - complexity class
2. Binary search - complexity class

} recurrences + solving

0. Logarithm intu

logarithm has a base ... **base 2** $\log_b$

$\log_2$ lg

$\lg n$ ... v. good complexity class

↳ $2^x = n$?

$\lg 8 = 3$

$\lg 16 = 4$

$\lg 32 = 5$

$\lg 64 = 6$

...

$2^{\lg n} = n$

(ex) $n = 16$

$\lg n = 4$

$2^{\lg n} = 2^4 = 16$

$2^k = n$

solve for k? take lg of both sides

$k = \lg n$

(ex) $n = 32$ $k = 5$

$2^5 = 32$

$5 = \lg(32)$

$5 = 5$ ✓

1. mergesort

Divide + conquer (recursive)

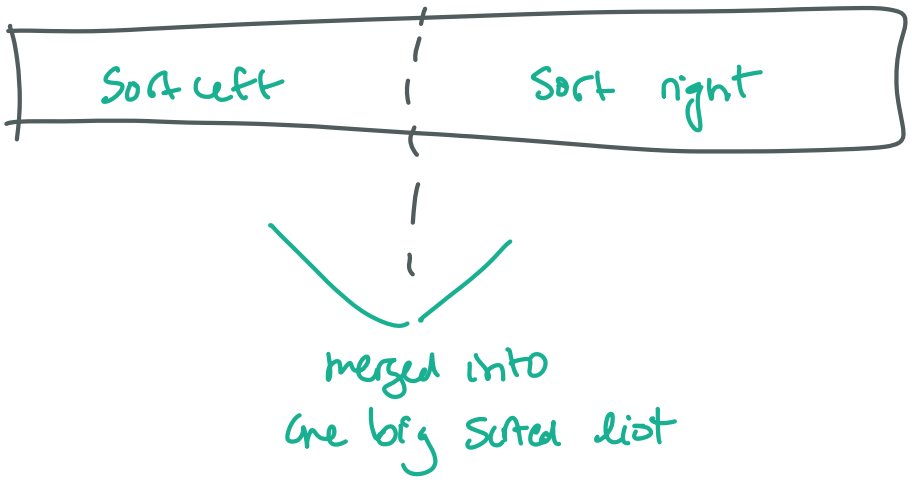
- break list into smaller pieces - cut into left/right halves
- solve the smallest version - n lists of length 1
- merge the sorted sublists - merge sorted

$T(n)$ = # steps on input of size n

= rep with a recurrence: run-time in terms of smaller versions of problem

mergesort from recursive pov.

- split list into left, right halves
- recursively sort the left half, right half
- merge sorted halves



$T(n)$ = # steps on input of size n

$= T(n/2) + T(n/2) + \text{merge}$ — missing info!

Sort left half

Sort right half

- to put into a Complexity Class

We need:

- # steps in merge
- Solve the recurrence — solve for $T(n)$ that does not rely on $T(n)$

steps in merge?

left, sorted

right, sorted

→ create one big, sorted list

output 

roughly n comparisons

(worst case)

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

$$= T(n/2) + T(n/2) + n$$

$T(n/2)$: sort left

$T(n/2)$: sort right

n: merge

$T(n/4) + T(n/4) + n/2$ $\quad$ $T(n/4) + T(n/4) + n/2$

Goal

- get rid of T on right-hand side
- express $T(n)$ in terms of n, constants, coeffs

Solve a recurrence: subs method

- plug in smaller values to $T(n)$
- until we establish a pattern
- express $T(n)$ on its k^{th} iteration
- pick a value for k to get to <u>base case</u> $T(1) = 1$

sort a list of size 1

$T(n) = T(n/2) + T(n/2) + n$ iteration #1

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

$T(n/2) = T(n/4) + T(n/4) + n/2$

Plug it in

$T(n) = 2 \cdot T(n/2) + n$ iteration #2

$= 2 \cdot (2 \cdot T(n/4) + \frac{n}{2}) + n$

$= 4 \cdot T(n/4) + n + n$

$= 4 \cdot T(n/4) + 2n$

$T(n/4) = T(n/8) + T(n/8) + n/4$

Plug in

$T(n) = 4 \cdot T(n/4) + 2n$ interation #3

$= 4 \cdot (2 \cdot T(n/8) + n/4) + 2n$

$= 8 \cdot T(n/8) + n + 2n$

$= 8 \cdot T(n/8) + 3n$

$T(n/8) = T(n/16) + T(n/16) + n/8$

← Plug in

$T(n) = 8 \cdot T(n/8) + 3n$ iteration #4

$= 8 \cdot (2 \cdot T(n/16) + n/8) + 3n$

$= 16 \cdot T(n/16) + 4n$

Find a pattern: kth iteration

$T(n) = \ldots + kn$

↳ powers of 2

$T(n) = 2^k \cdot T(n/2^k) + kn$

- choose any value of k
- get to base case

$T(n/2^k) = T(1) = 1$

$n/2^k = 1$

$n = 2^k$ solve for k

lg both sides

$\lg n = k$

Plug in k

$T(n) = 2^k \cdot T(n/2^k) + kn$

$= 2^{\lg n} \cdot T(n/2^{\lg n}) + \lg n \cdot n$

$= n \cdot T(n/n) + n \cdot \lg n$

$= n \cdot T(1) + n \lg n$

$= n \cdot 1 + n \lg n$

$k = \lg n$

$T(1) = 1$

$O(n \lg n)$

run-time complexity of mergesort!

2. Binary Search

↳ search is done

binary search is go-to on a sorted list

Universe:
- Sort once
- Search over + over in sorted list

Search: determine whether a target value exists in the list

(ex) [3, 5, 6, 9, 12, 13] list in sorted order

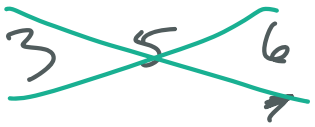
- use the fact that list is sorted
- target: [9]
- cut the list in half, look at value in middle

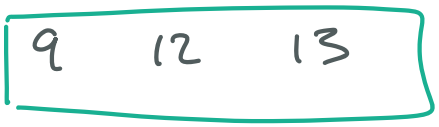
3 5 6 | 9 12 13

↳ middle

- Comparison: 6 vs. 9 $6 < 9$

- anything left of 6 is < 6
- throw away left half of list





keep right half

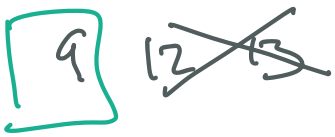
9 12 13

↳ middle

- comparison 9 vs. 12

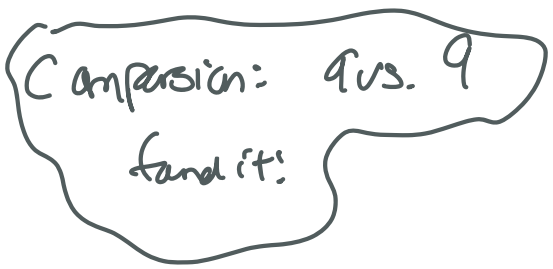
12 > 9

- get right right half



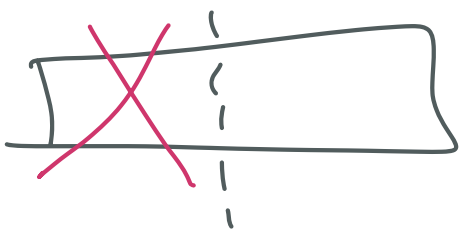
List of size 1

9



List of length 6: look at 3 elements

- cut list in two halves
- throw away one half



Run-time of Binary Search:

$T(n) = T(n/2) + 1$ → recurrence

+1 finding middle doing comparison

need to solve the recurrence $T(1) = 1$ (base)

$T(n) = T(n/2) + 1$

iteration #1

$T(n/2) = T(n/4) + 1$

plug in

$T(n) = T(n/4) + 1 + 1$

$= T(n/4) + 2$

iteration #2

$T(n/4) = T(n/8) + 1$

Plug in

$T(n) = T(n/8) + 1 + 2$

$= T(n/8) + 3$

iteration #3

iteration k... $T(n) =$ ___ $+ k$

$T(n) = T(n/2^k) + k$

Choose value for k to get to base case $T(1) = 1$

$n/2^k = 1$

$k = \lg n$ ... plug in for k

$$T(n) = T(n/2^k) + k$$

$$= T(n/2^{\lg n}) + \lg n$$

$$= T(n/n) + \lg n$$

Boundary cond. $= T(1) + \lg n$

$$= 1 + \lg n$$

complexity class for binary search

$O(\lg n)$

thank you :)