CS1800

11/29 - Tues.

Admin - home stretch!

- Hw8 due 12/1
- Hw9 due 12/5
- this week - rec 10
- exam #3 [12/5] 50 min total 30 min exam
  - ↳ last day!! :)

Agenda

1. Search and sort
2. Counting comparisons ~ run-time f(n)
3. Divide + conquer sorting (recursive)
4. TRACE !!

1. Search and Sort

- growth of functions
- upper bounds $O(n)$
- complexity classes

$f(n)$ → # steps an algo needs to solve a problem
↳ size of input

What are the steps the algo is taking?

- key, fundamental CS algorithms

→ language-agnostic
Search/Sort any type of object
steps independent of prog lang, processor, computer

Any search engine

offline
1. crawl - find all web pages
2. index - each word, alpha order - Sort

real time
3. query - find pages with word/phrase - Search
4. rank - returns results in order - Sort

Sorting

↳ Still area of research in CS

sequence of elements $\{a_1, a_2, \dots a_n\}$

where $a_1 \leq a_2 \leq \ldots \leq a_n$

Sorting Algorithms

iterative	recursive (divide & conquer)
bubblesort	quicksort
insertion sort	mergesort
selection sort	heapsort
shellsort	

Real life:

- excel - quicksort
- mac - quicksort, mergesort
- C/C++ - quicksort
- Python - timsort

2. Counting Comparisons

$f(n)$ ~ # steps an algo needs to solve a problem

↳ Comparing two values

who goes left, who goes right?

Insertion Sort

given: a list of numbers, unsorted

Create: sorted list

At any point: We move one number from unsorted to its correct position in sorted

ex: 6, 9, 6, 2, 12, 7, 12 — Unsorted

Sorted

#1 Put 6 in correct position in sorted list

9, 6, 2, 12, 7, 12 — Unsorted

6 — Sorted

#2 Put 9 in correct position

9 vs 6? 9 is bigger — 1 comparison

6, 2, 12, 7, 12 — unsorted

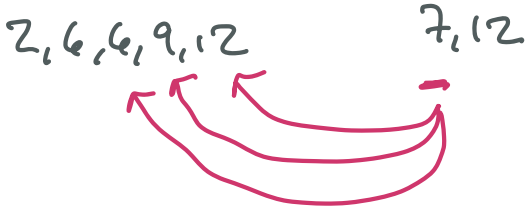
6, 9 — sorted

#3 Put 6 in correct position

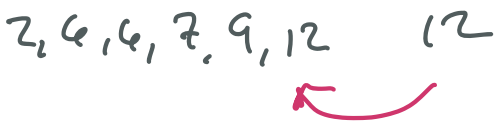
2 comparisons

3 comparisons

1 comparison



3 comparisons



1 comparison

2, 6, 6, 7, 9, 12, 12

Sorted!

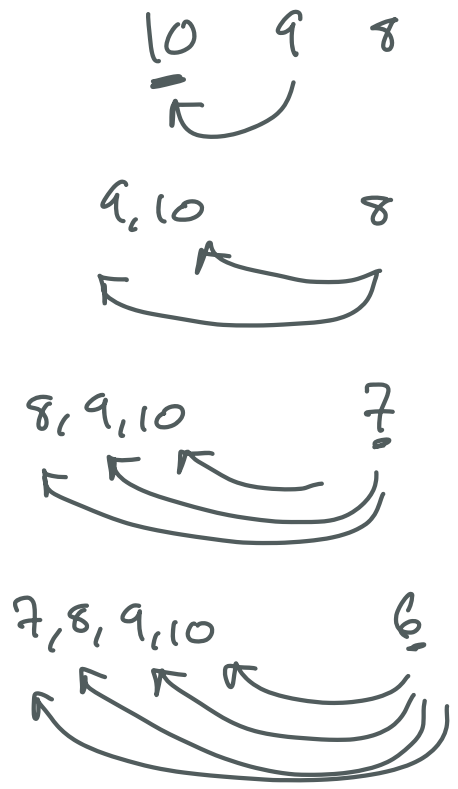
total: 11 comparisons

In general... which complexity class?

Assume worst case scenario

At each step, assume max # of comparisons

Worst: list is reverse-sorted



In general, list of size n:

comparisons: $1, 2, 3, 4, \ldots, n-1$ arithmetic.

$f(n)$ = total comparisons: $1+2+3+4+\ldots+n-1$

Sum of first n terms of arithmetic sequence:

$$\frac{(\#\text{terms})(\text{first} + \text{last})}{2} = \frac{(n-1)(1+n-1)}{2} = \frac{(n-1)(n)}{2}$$

$$= \frac{n^2-n}{2} = \frac{n^2}{2} - \frac{n}{2}$$

$f(n) = \#\text{steps} = \frac{n^2}{2} - \frac{n}{2}$

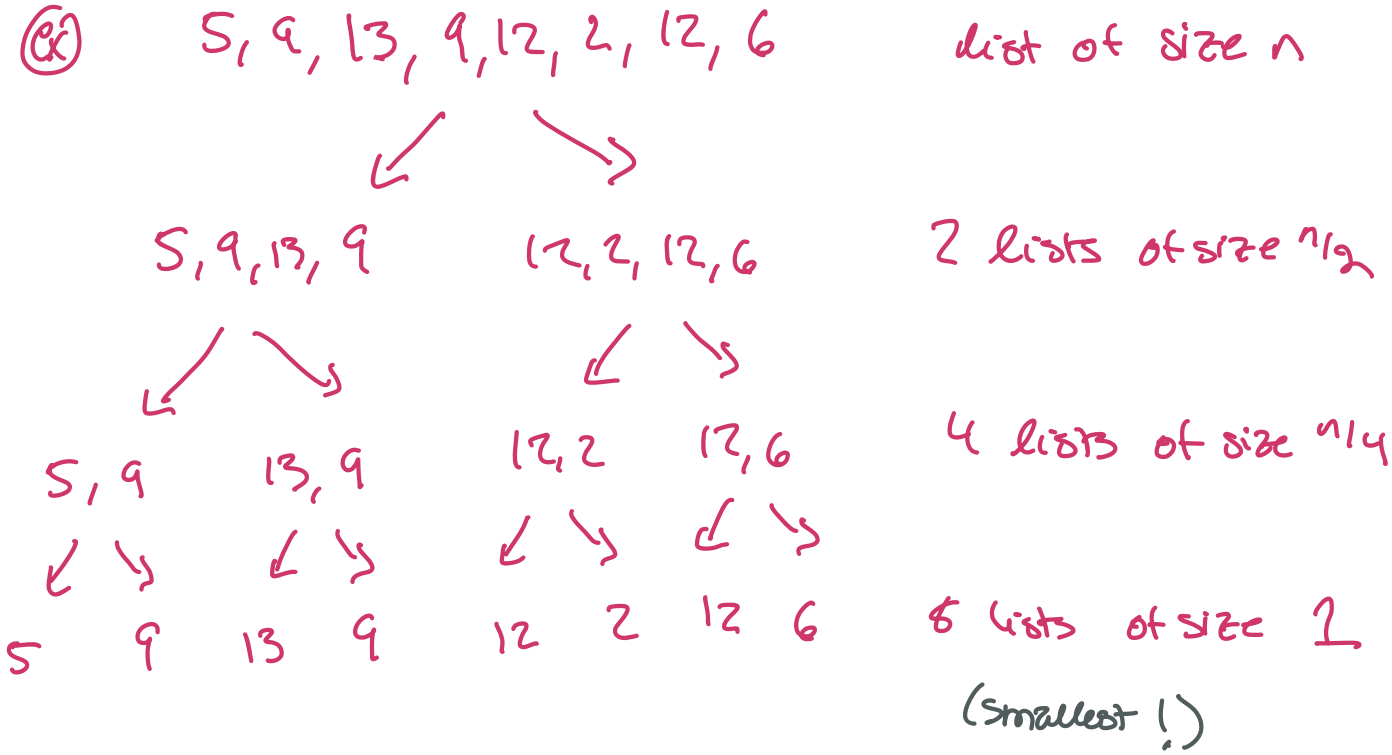
10:50

Drop coeffs, lower-order terms: $O(n^2)$

↳ complexity class

3. Mergesort

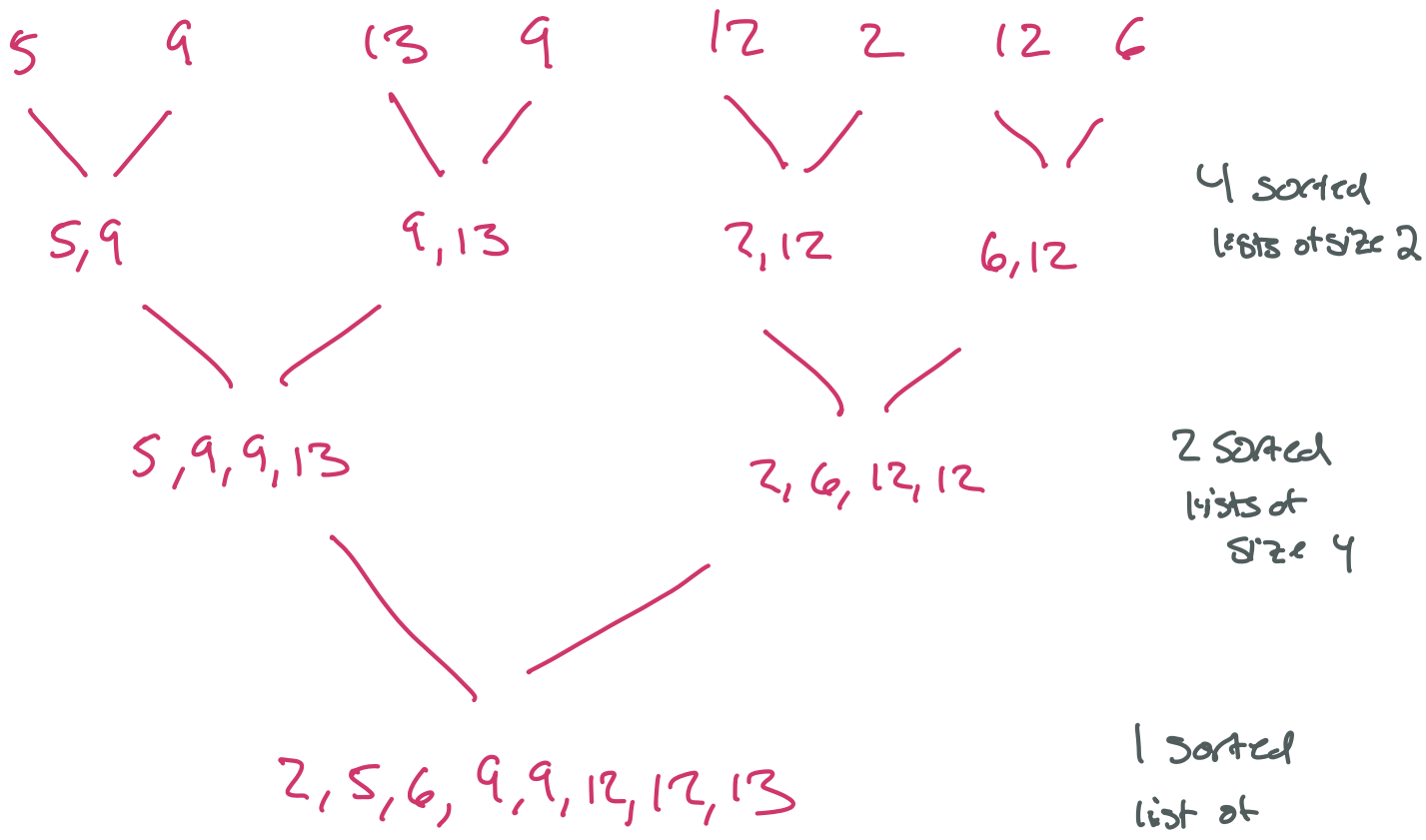
- Divide + Conquer (recursive)
  - ↳ break problem down into smaller parts
  - Solve smallest version of problem
  - Combine small results to solve original problem



- Created smallest version of problem
- Solved smallest version
- We have: 8 sorted lists

next: merge

Combine small sorted lists into bigger sorted lists



Comparisons in merge step (every sublist is sorted) (worst case)

~~2~~, 12 6, 12

2 vs 6

output: 2

12 ~~6~~, 12

output: 2, 6

12 vs 6

12 12

output: 2, 6, 12

12 vs. 12

12

output: 2, 6, 12, 12

12 vs. nothing

5, 9 9, 13 5 (Base case)

9 9, 13 5, 9

9, 13 5, 9, 9, 13

How many steps? ⇝ recurrence $T(n)$ ← # steps

↳ size of list

$T(n)$ = # steps on all smaller versions

TRACE:) 1800/1002

thankyou!