Searching and Sorting

- Sorting
  - insertion sort
  - mergesort
- Compare various possible Map implementations

Announcements

- Midterm 2
  - Tue. 4/4, 8am – 9:20am, in THH 101
  - closed book, closed note, no electronic devices,
  - bring USC ID card
- PA4 is due in less than 2 weeks.
- No lab assignment or lab meetings this week.
- Next week:
  - There will be a C++ lab assignment next week – do readings ahead of time.

Sorting

- Input is an array of values:

$$[10, 3, 52, 60, 12, 9, 7, 17]$$

- Output is the same values in the same array, but in order:

$$[3, 7, 9, 10, 12, 17, 52, 60]$$

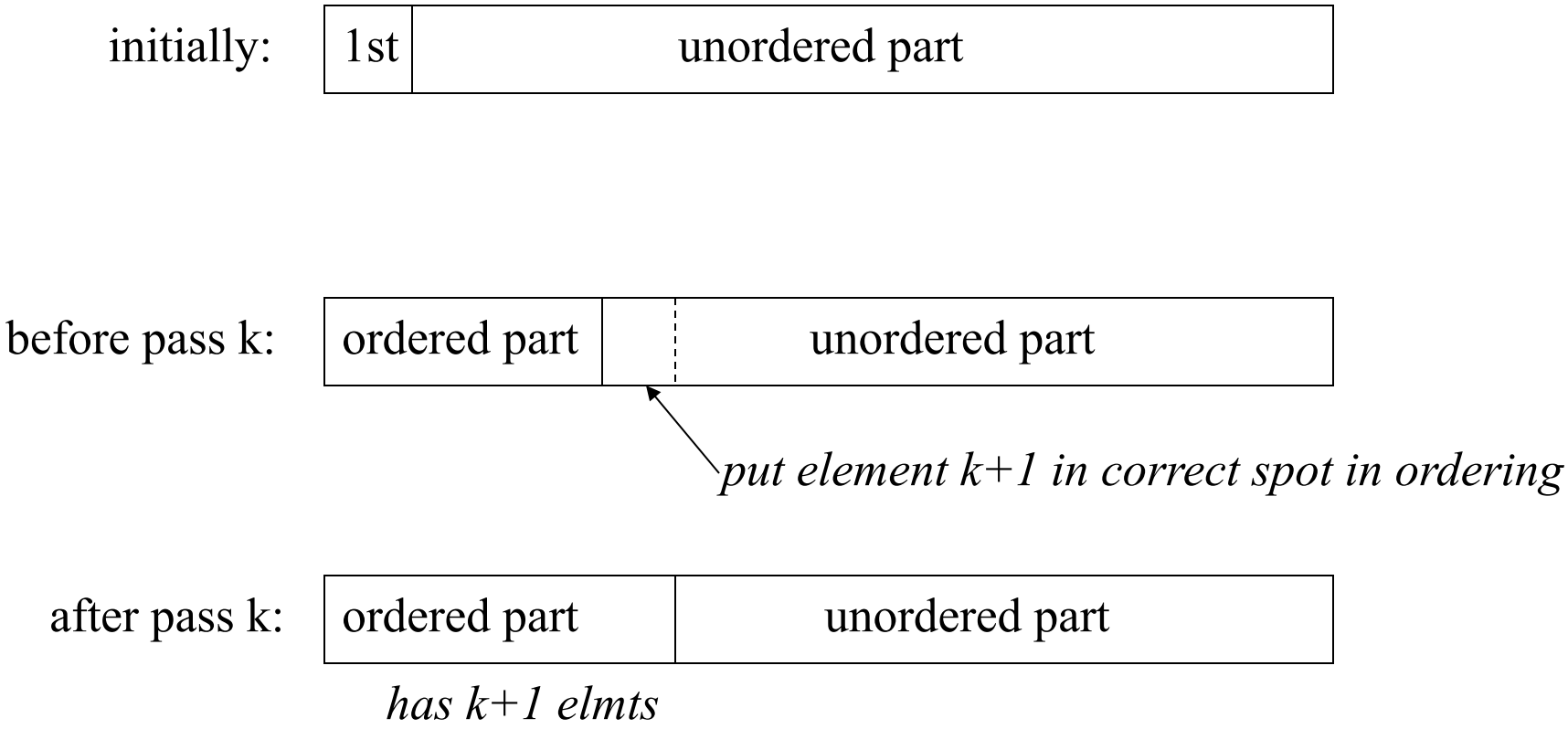
- (some sorts also work on linked lists)
- Many sort algorithms
- We'll discuss just a few today.
- First we'll hear from an expert . . .

One sorting algorithm

- Insertion sort
- based on inserting into a sorted list/array
- e.g., **`Names`** class: repeatedly inserted a value into the sorted array.
- Idea:
  - use linear or binary search to find correct spot
  - shift values over to make room for new value
- How much time (big-O) to create a Names object with *n* elements this way?

Insertion sort

- Insertion sort in place in an array:



Insertion sort example

5 10 3 7 6

Fast sorts

- Other O(n^2)(not fast) sorts:
  selection sort, bubble sort.
- Fastest general purpose sorts are O(***n*log*n***):
  quicksort, mergesort, heapsort
- Let's discuss mergesort in more detail:
- Basic operation is the merge
  - we discussed algorithm in big-O lecture
  - merge of 2 lists of length *n* takes 2*n*

Mergesort

- Think of each element as a sorted list of len 1
- Merge each of them pairwise.
  - Now have n/2 sorted lists of len 2
- Merge each of those pairwise.
  - Now have n/4 sorted lists of len 4

. . .

- Eventually merge 2 sorted lists of len n/2

- Big-O?
  - How much time for all the merges at a level?
  - How many levels total?

Mergesort example

5 10 3 7 26 6 12 8

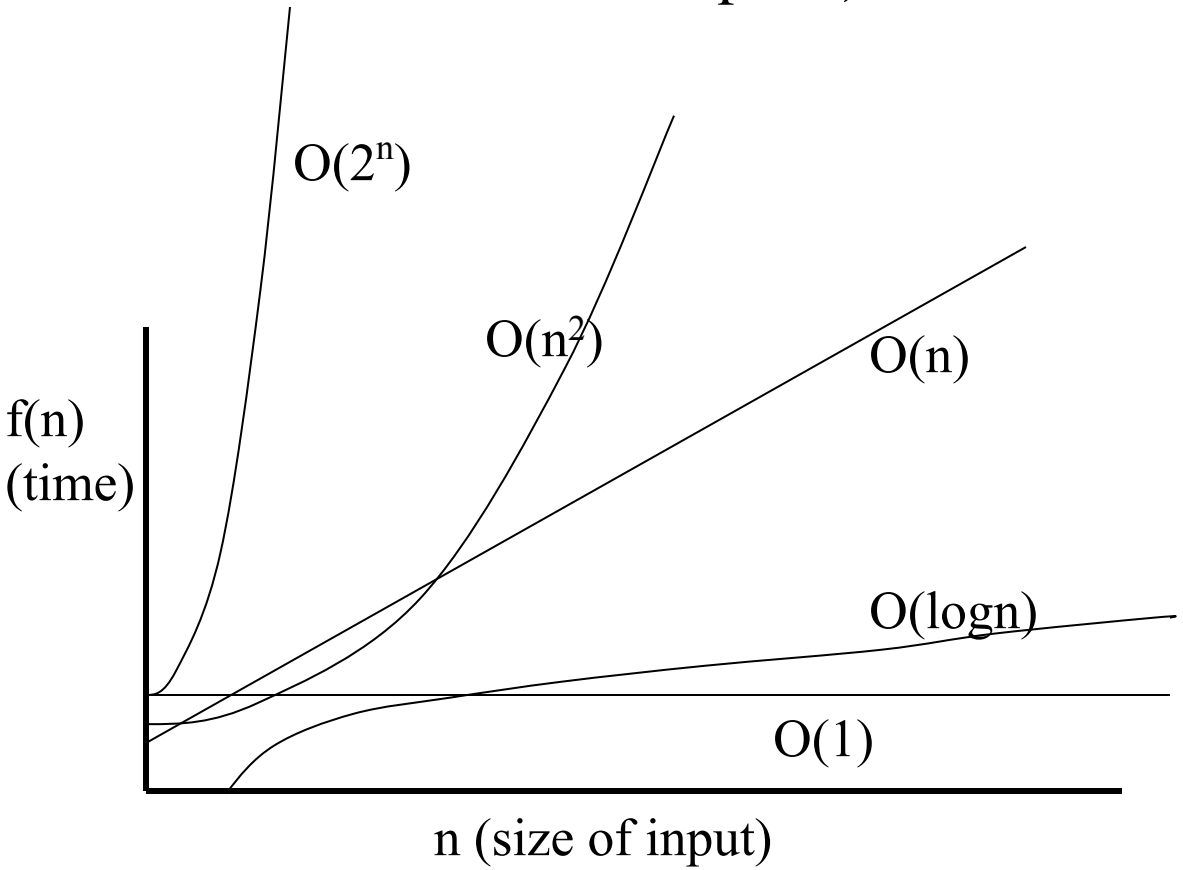
Merge sort: another example of tree recursion

- Recursive solution very short! (similar to tree traversal code)

```
void mergesort(array){
    if (array.length > 1) {
       mergesort(first half);
       mergesort(second half);
       array =
          merge (first half, second half);
   }
}
```

Comparing different time bounds

- Revision of a slide you saw in an earlier lecture. (this is just a hand-drawn estimation of real plots)



Compare Map representations

- Recall Map operations:

  lookup by key

  insert (key, value)

  remove by key

  visit elements in order by key

  or visit elements any order

- What are possible data structures we could use to implement?

Comparing big-O for Map operations

representations

operation	ordered array	ordered list	unordered array	unordered list	balanced search tree	hash table
lookup by key	log n	n	n	n	log n	1
insert (key, value)	~~n + log n~~	n	1	1	log n	1
remove by key	n	n	n	n	log n	1
visit elements in order by key	n	n	n log n	n log n	O(n)	n log n
visit elements any order	n	n	n	n	O(n)	n

Summary

- Usually you don't have to implement binary search, sort, binary search, binary trees
- E.g., Java library methods / classes provide them.
- **Do** need to be able to compare algorithms and representations (complexity)
- Should I use an ArrayList? LinkedList? TreeMap? for my app?