Searching and Sorting

- Hash tables (cont – from Tue slides)
  - using overriding hashCode/equals (cont.)
  - big-O
  - applications
- Sorting
  - insertion sort
  - mergesort
- Compare various possible Map implementations

Announcements

- MT 2 on Tuesday 4/6 9:30am / 7:30 pm PDT
  - closed book, closed note
  - code handout will be distributed by Mon 4/5 at 8pm
- No lab meetings or lab assignment this week
- Next week: there will be a C++ lab. Do readings ahead of time.

Hashing (cont.)

- See slides from Tues lecture

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:

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

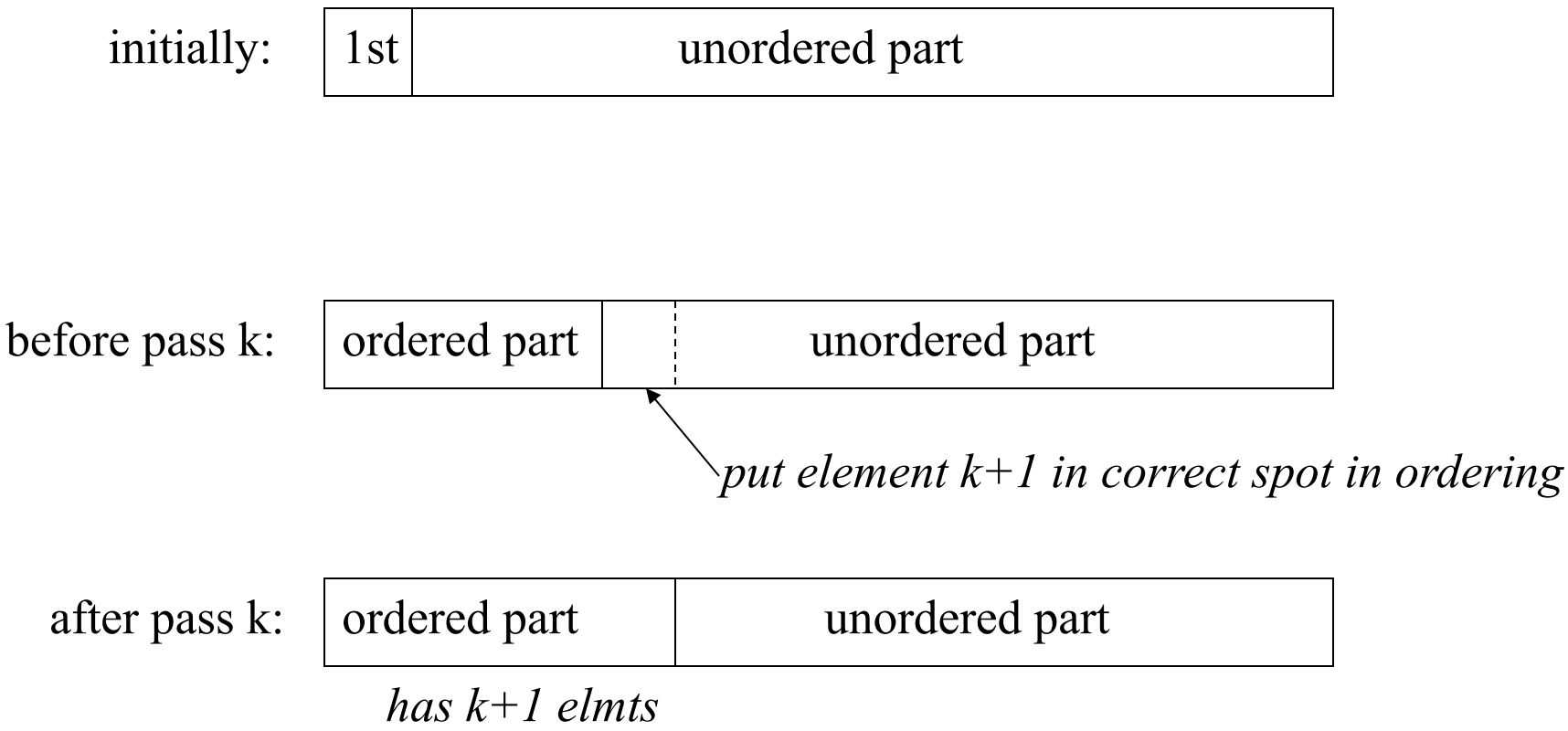
- (some sorts also work on linked lists)
- Many sort algorithms
- We'll discuss just a few today.

One sorting algorithm

- Insertion sort
- based on inserting into a sorted list/array
- e.g., populating a **`Names`** object:
  - repeated calls to **`insert`** method
  - using ordered partially-filled array representation
- insert one element:
  - 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
- We'll discuss mergesort in more detail:
- Basic operation is the merge
  - reminder what merge problem is:

    input1: `[3, 7, 12, 18]` *(ordered list)*

    input2: `[2, 5, 15, 20, 27]` *(ordered list)*

    output: `[2, 3, 5, 7, 12, 15, 18, 20, 27]` *(ordered list)*
  - we discussed fast merge 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

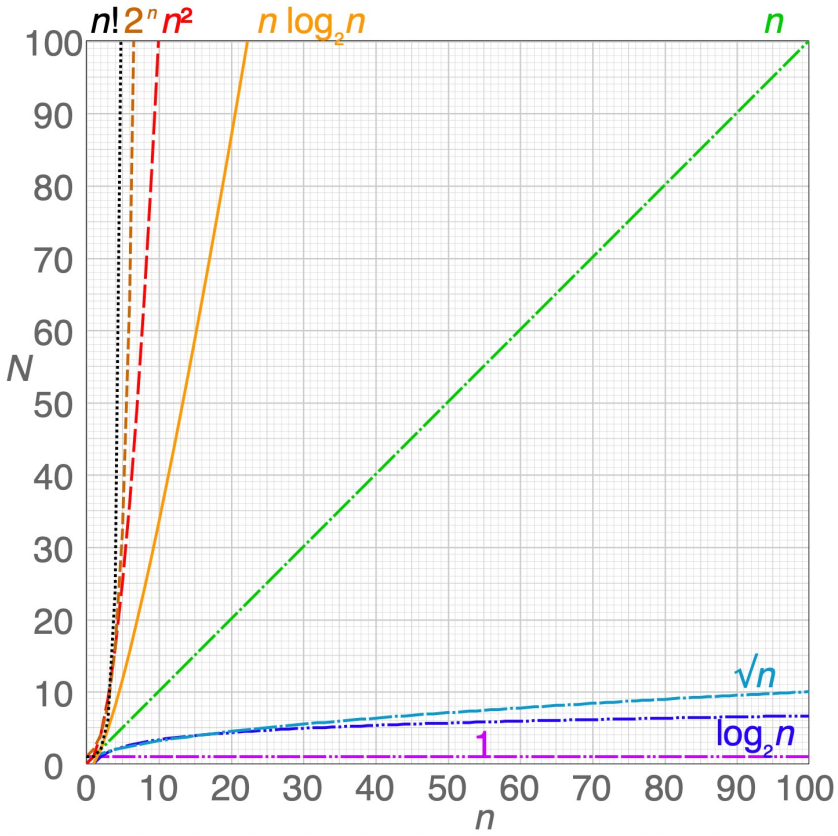
Merge sort: another example of tree recursion

- 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

number of operations



size of input

source: cmglee using CC license.
from Wikipedia page: Computational complexity of mathematical operations

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						
insert (key, value)						
remove by key						
visit elements in order by key					O(n)	
visit elements any order					O(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 array? ArrayList? LinkedList? TreeMap? for my app?