



CSE332: Data Abstractions

Lecture 12: Introduction to Sorting

Dan Grossman Spring 2010

### More reasons to sort

General technique in computing:

Preprocess data to make subsequent operations faster

Example: Sort the data so that you can

- Find the  $\mathbf{k}^{\text{th}}$  largest in constant time for any  $\mathbf{k}$
- Perform binary search to find an element in logarithmic time

Whether the performance of the preprocessing matters depends on

- How often the data will change
- How much data there is

### Introduction to sorting

- Stacks, queues, priority queues, and dictionaries all focused on providing one element at a time
- But often we know we want "all the data items" in some order.
  - A huge reason to use computers: an 8-year-old child can sort, but a computer can sort faster
  - Different algorithms have different asymptotic and constantfactor trade-offs
    - Knowing one way to sort just isn't enough

Spring 2010 CSE332: Data Abstractions 2

### The main problem, stated carefully

For now we will assume we have *n* comparable elements in an array and we want to rearrange them to be in increasing order

### Input:

- An array A of data records
- A key value in each data record
- A comparison function (consistent and total)

#### Effect:

- Reorganize the elements of A such that for any i and j,
  if i < j then A[i] ≤ A[j]</li>
- (A must have all the same data it started with)

An algorithm doing this is a comparison sort

Spring 2010 CSE332: Data Abstractions 3 Spring 2010 CSE332: Data Abstractions

## Variations on the basic problem

- 1. Maybe elements are in a linked list (could convert to array and back in linear time, but some algorithms needn't do so)
- 2. Maybe ties need to be resolved by "original array position"
  - Sorts that do this naturally are called stable sorts
  - Others could tag each item with its original position and adjust comparisons accordingly (non-trivial constant factors)
- 3. Maybe we must not use more than O(1) "auxiliary space"
  - Sorts meeting this requirement are called in-place sorts

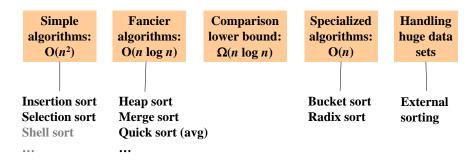
5

- 4. Maybe we can do more with elements than just compare
  - Sometimes leads to faster algorithms
- 5. Maybe we have too much data to fit in memory
  - Use an "external sorting" algorithm

Spring 2010 CSE332: Data Abstractions

# The Big Picture

Surprising amount of juicy computer science: 2-3 lectures...



Spring 2010 CSE332: Data Abstractions 6