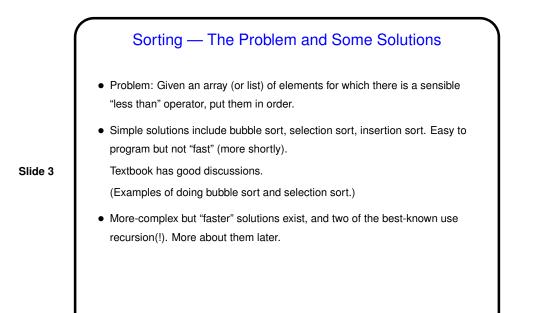
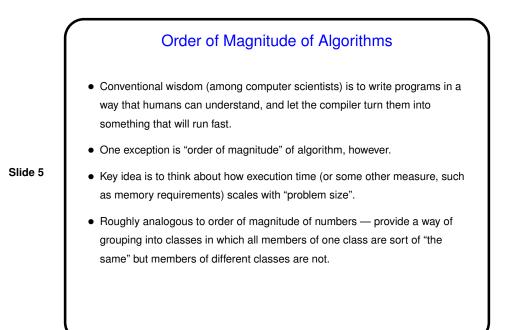


Slide 2

Slide 4



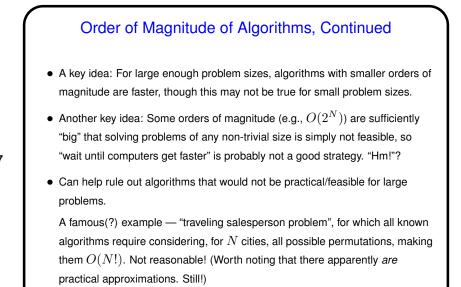
Searching — The Problem and Some Solutions
Problem: Given an array (or list) and an element, search the array for the element.
Simplest solution is sequential search. Easy to program and works for any array but not "fast".
Slightly more-complex solution is binary search. "Faster" but requires array to be in order.





- Typically written using "big-O" notation (e.g., O(N),  $O(N^2)$ , etc.). Formal definition possible, but informally, O(f(N)) means that execution time (or whatever) for problem size N scales as f(N). Examples:
- f(N) = N, f(N) = 10N, and f(N) = N + 1000 are all O(N) ("linear").
- $f(N) = N^2$ ,  $f(N) = 2N^2$ , and  $f(N) = N^2 + 2N + 1$  are all  $O(N^2)$ .
- $f(N) = 2^N$  and  $f(N) = 2^N + N$  are both  $O(2^N)$  ("exponential").
- (Compare using gnuplot.)

Slide 6



Order of Magnitude of Algorithms, Continued
As an example, look at bubble sort and selection sort.
For both, "problem size" is the number of elements to sort, and a rough measure of how execution time scales with problem size is based on how many comparisons are needed, in the worst case.
Again for both, total number of comparisons is N(N - 1)/2, making them "O(N<sup>2</sup>)".
As another example, look at sequential search and binary search. The first is O(N), but the second is ... What? (O(log N))

Slide 8

Slide 7

