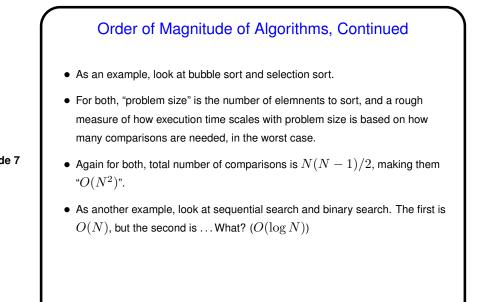


## Order of Magnitude of Algorithms, Continued A key idea — for large enough problem sizes, algorithms with smaller orders of magnitude are faster, though this may not be true for small problem sizes. Another key idea — some orders of magnitude (e.g., O(2<sup>N</sup>)) are sufficiently "big" that solving problems of any non-trivial size is simply not feasible, so "wait until computers get faster" is probably not a good strategy. "Hm!"? Can help rule out algorithms that would not be practical/feasible for large problems. A famous(?) example — "traveling salesperson problem", for which all known algorithms require considering, for N cities, all possible permutations, making them O(N!). Not reasonable! (Worth noting that there apparently *are* practical approximations. Still!)

Slide 6



Slide 7

