Lecture 5: ArrayList implementation & Complexity

CS 62
Spring 2012
Kim Bruce & Kevin Coogan

Back to ArrayList

- Focus on implementation
- See ArrayIndexList<E>
  - Similar to ArrayList
  - Instance variables:
    - elts: array instance variable,
    - eltsFilled: number of slots filled.
- Some operations very cheap:
  - size, isEmpty, get, set take constant time (no search)
- Others more expensive

Adding Elts in Slot i

- Easy if there is space:
  - At end, just add it
  - If before end, must move all elements at i and beyond to right before inserting
  - *Delete similar*
- What if run out of space
  - Create new array twice as big and copy old elements over before adding.
- How expensive is this?

Complexity of Operations

- Count number of compares and/or moves to accomplish operation.
- Rather than keeping an exact count of operations, use *order of magnitude* count of complexity.
- Ignore differences which are constant
  - e.g., treat n and n/2 as same order of magnitude.
  - Same with 2 n^2 and 1000 n^2

Order of Magnitude

- <u>Definition</u>: We say that *g(n)* is *O(f(n))* if there exist two constants C and k such that $|g(n)| <= C\,|f(n)|$ for all n > k.
- Examples: $2n+1$, n^3-n^2+83, 2^n+n^2
- Used to measure time and space complexity of algorithms on data structures of size n.
- Most common are
  - O(1) - for any constant
  - O(log n), O(n), O(n log n), $O(n^2)$, ..., $O(2^n)$

Use simplest version in O(...)

Complexity

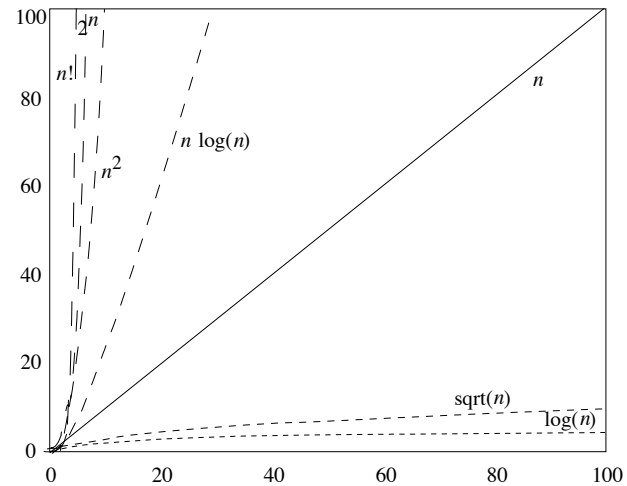

Figure 5.3 Long-range trends of common curves. Compare with Figure 5.2.