

Lecture 6: ArrayList implementation & Complexity

CS 62
Fall 2017
Kim Bruce & Alexandra Papoutsaki

Lab This Week

- Timing ArrayList operations
 - Encourage working in pairs
 - Stopwatch class: start(), stop(), getTime(), reset()
- Java has Just-In-Time compiler
 - Must “warm-up” before you get accurate timing
 - What can mess up timing?
- Uses Vector from Bailey rather than ArrayList from Java libraries because can change way it increases in size.

Programming Assignment This Week

- Weak AI/Natural Language Processing:
 - Generate text by building frequency lists based on pairs of words. ArrayList of Associations of String (words) and Integer (count of that word).

PostIt App

- Generated javadoc for fun
- See how ArrayList used in methods for PostItApplication
 - findWindowInList, moveToTop, removeWindow
 - Used in mouse-event-handling methods

ArrayList

- Not using Bailey implementation
 - see code on-line for implementation by Tomassia & Goodrich
- Standard Java libraries have lots of extra methods not in our implementation:
 - Many involve working on other collections
 - irrelevant for us at this point.
 - addAll, clear, contains, containsAll, listIterator, removeAll, replaceAll, retainAll, sort, spliterator, sublist, toArray

Back to ArrayList

- Interface is `IndexList<E>`
- See `ArrayIndexList<E>`
 - Similar to `ArrayList`
 - Instance variables:
 - `elts`: array instance variable,
 - `eltsFilled`: number of slots filled.
- Creating new `ArrayList` is weird
 - Can't construct array of variable type!
 - Create array of `Object`, but coerce to believe array of `E`.

ArrayList Implementation

- 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 $2n^2$ and $1000n^2$

Order of Magnitude

- **Definition:** We say that $g(n)$ is $O(f(n))$ if there exist two constants C and k such that $|g(n)| \leq C|f(n)|$ 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 $O(\dots)$*

Complexity

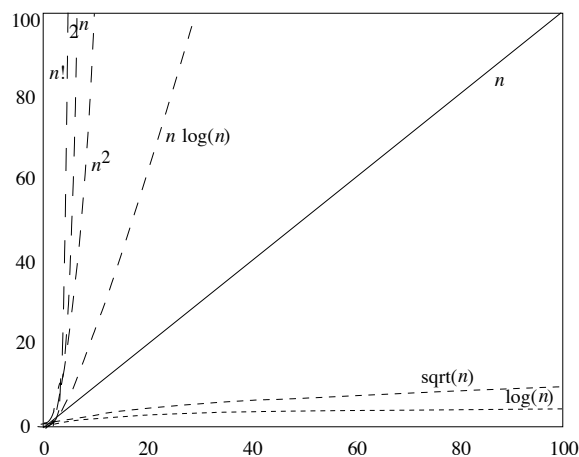


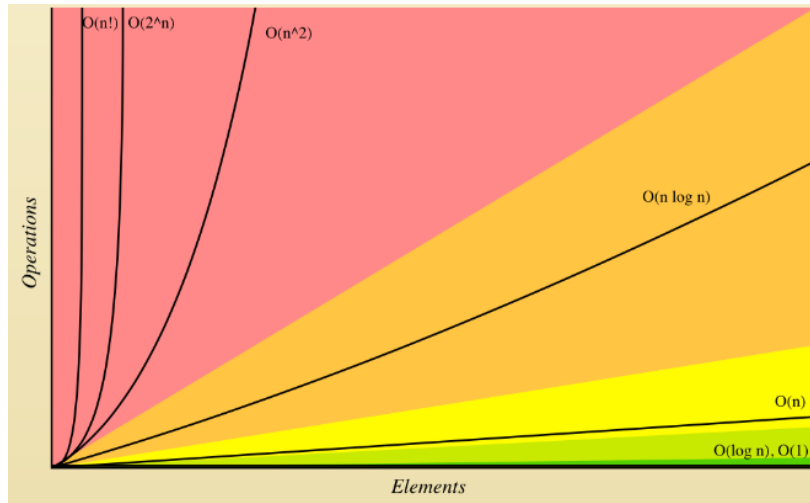
Figure 5.3 Long-range trends of common curves. Compare with Figure 5.2.

Comparing Orders of Magnitude

- Suppose have ops w/complexities given & problem of size n taking time t .
- How long if increase size of problem?

Problem Size:	$10n$	$100n$	$1000n$
$O(\log n)$	$3+t$	$7+t$	$10+t$
$O(n)$	$10t$	$100t$	$1000t$
$O(n \log n)$	$> 10t$	$> 100t$	$> 1000t$
$O(n^2)$	$100t$	$10,000t$	$1,000,000t$
$O(2^n)$	$\sim t^{10}$	$\sim t^{100}$	$\sim t^{1000}$

Complexity 2



Adding to ArrayList

- Suppose n elements in ArrayList and add i .
- If space:
 - Add to end is $O(1)$
 - Add to beginning is $O(n)$
- If not space,
 - What is cost of ensureCapacity?
 - $O(n)$ because n elements in array

EnsureCapacity

- What if only increase in size by i each time?
 - Adding n elements one at a time to end
 - Total cost of copying over arrays: $1+2+3+\dots+(n-1) = n(n-1)/2$
 - Total cost of $O(n^2)$
 - Average cost of each is $O(n)$
- What if double in size each time?
 - Suppose add $n = 2^m$ new elts to end
 - Total cost of copying over arrays: $1+2+4+\dots+n/2 = n-1, O(n)$
 - Average cost of $O(i)$, but “lumpy”

ArrayList Ops

- Worst case
 - $O(1)$: size, isEmpty, get, set
 - $O(n)$: remove, add
- Add to end, on average $O(1)$