LECTURE 5: ARRAYLIST IMPLEMENTATION AND COMPLEXITY

Today

- Reading
  - JS Ch. 3 (Vectors) and Ch. 4 (Generics)
- Objectives
  - Implementation of ArrayList
  - Complexity of ArrayList operations
  - Introduction to Big-O Notation

ArrayList

- Our first example of how to analyze the complexity of a data structure
- java.util.ArrayList<E>
  - Instance variable: array // the actual array
  - Instance variable: capacity // the length of the array
  - Instance variable: size // the number of elts in array
- See ArrayIndexList<E> on course webpage

ArrayList Operations

- Some operations very fast
  - size, isEmpty, get, and set
- Other operations are more complex
  - addLast(E elt) // add elt to end of array
  - addFirst(E elt) // add elt to beginning of array
  - remove(int r) // remove r^{th} elt in array

Complexity

- Count number of operations (e.g. compares, copies, additions) performed
- Express the number of operations in terms of problem size n
- Ignore constants and use *order of magnitude* instead
  - n and n/2 have same order of magnitude
  - $2n^2$ and $1000n^2$ have the same order of magnitude

Big-O Notation (on board)

Big-O Notation and Complexity

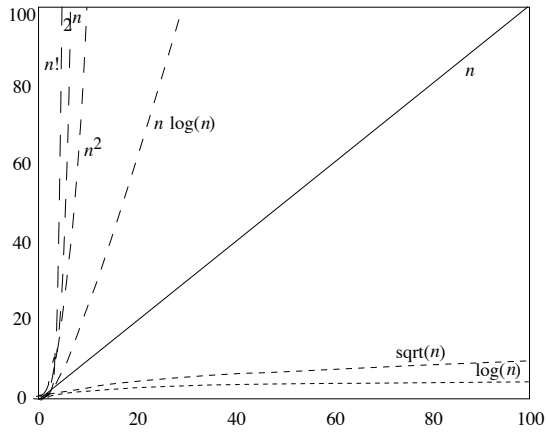

Figure 5.3 Long-range trends of common curves. Compare with Figure 5.2.

Adding to the end

- Best case, there is space
  - Just add element
- Worst case, there is no space
  - Expand array and copy over all N existing elements
- When adding N elements
  - How often do we observe the best case?
  - How often the worst case?

EXTRA SLIDES

Order of Magnitude

- Definition: *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)
  - $O(n^2)$, ..., $O(2^n)$

ArrayListOperations

```
public int size() {
     return size;
}

public void addFirst(E elt) {
for(int i = size; i > 0; i--) {
     array[i] = array[i-1]; // shift elts to
right
}
array[0] = elt;
```