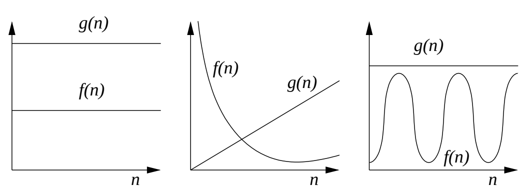
Computer Science 136

Data Structures

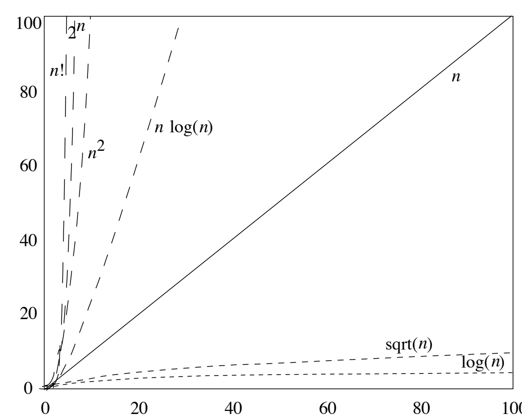
Lecture #6 (September 22, 2021)

1. Vectors.
   (a) Abstract concept: the extensible array.
      i. Vectors start “empty,” though some array has usually been allocated.
      ii. Grows (and, possibly, shrinks) as needed.
      iii. Efficient in time and space.
   (b) Uses methods `get/set/add/remove`, not square-bracket indexing.
   (c) Reshaping occurs through `add(position,value)` and `remove(position)`.
   (d) Utility methods: `isEmpty` and `size`.
   (e) Implementing extensibility:
      i. Keep track of two lengths: the array length and the vector length.
      ii. Allow some guidance by user.
      iii. Keys to efficiency:
         A. double array length when necessary (why is this efficient?)
         B. details encapsulated in protected `ensureCapacity`
         C. shrinking is not automatic (use non-ideal `trimToSize` explicitly). But: it could be.
2. Complexity.
   (a) Formal definition of what it means for $f(x)$ to be $O(g(x))$. A definition worth remembering (see p. 98). Write it here:

      i. $f(x)$ is bounded above by some constant times $g(x)$.



      ii. $f(x)$ need not ever be equal to $c \cdot g(x)$. The bound need not be *tight*.
      iii. It only needs to be bounded above to the right of some x_0.
      iv. Really, we're only concerned about the magnitude of $f(x)$. In practice, $f(x) \geq 0$ since f is often a measure of time or space utilization.
   (b) E.g.: `Vector`s that extend to size n by 1 takes $O(n^2)$ time.
   When you double the `Vector`*'s length, the time to expand increases by 4.*
   (c) E.g.: `Vector`s that extend to size n by doubling takes $O(n)$ time.
   When you double the `Vector`*'s length, the time to expand increases by 2.*



   (d) A similar definition of what it means for $f(x)$ to be $\Omega(g(x))$, a *lower bound*.
   (e) Any function that is both $O(g(x))$ and $\Omega(g(x))$ is $\Theta(g(x))$.
   (f) Little versions, $o(x)$ and $\omega(x)$, are *asymptotic* bounds. They hold true for *arbitrary* positive constants, c.