Overview

- Describing the Efficiency of Computations
- Calculating the Efficiency of Binary Search

Efficiency of Computations

- A running program consumes resources such as time (seconds) and space (bits). Frequently, we abstract our units, and measure steps and objects, instead of seconds and bits.

- When comparing programs (or algorithms), you should *first* pay attention to *gross* differences in time or space consumed, for example, n^3 versus n^2 steps, rather than $3n$ versus $2n$ steps.

- For a few programs, the cost is fixed and can be calculated by examining the program text. More frequently, however, cost depends on characteristics of the input, such as length.

Order Arithmetic

- When we make gross comparisons of programs, we often refer to the "order-of-magnitude" of the cost. The notation used is sometimes called "Big-Oh," and is always of the form $O(f(n))$, where $f(n)$ is some function over the positive integers.

- The Big-Oh notation simply means that the cost function is bounded by (is less than) some multiple of the function $f(n)$. For example, if we say

$$P = n^3 + O(n^2) \tag{1}$$

we mean that P equals n^3, plus some terms that are "on the order of n^2"—i.e., they don't grow faster than kn^2, where k is some constant term.

Order Arithmetic (cont.)

- More precisely,

  Definition. A function $g(n)$ is said to be $O(f(n))$, written

  $$g(n) = O(f(n)) \tag{2}$$

  if there is a positive integers c and n_0 such that

  $$0 \leq g(n) \leq cf(n) \tag{3}$$

  for all $n \geq n_0$.

- In other words, $O(f(n))$ is the **set** of all functions $h(n)$ such that there exist positive integers c and n_0 such that

  $$0 \leq h(n) \leq cf(n) \tag{4}$$

  for all $n \geq n_0$.

Order Arithmetic (cont.)

- For example,

$$1 + 2 + 3 + ... + n = \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2} \tag{5}$$

$$1 + 2 + 3 + ... + n = \frac{n^2}{2} + O(n) \tag{6}$$

$$1 + 2 + 3 + ... + n = O(n^2) \tag{7}$$

Order Arithmetic (cont.)

- Here are some equivalences that allow you to manipulate equations involving order-of-magnitude quantities:

$$f(n) = O(f(n)) \tag{8}$$

$$K \times O(f(n)) = O(f(n)) \tag{9}$$

$$O(f(n)) + O(f(n)) = O(f(n)) \tag{10}$$

$$O(f(n)) \times O(g(n)) = O(f(n) \times g(n)) \tag{11}$$

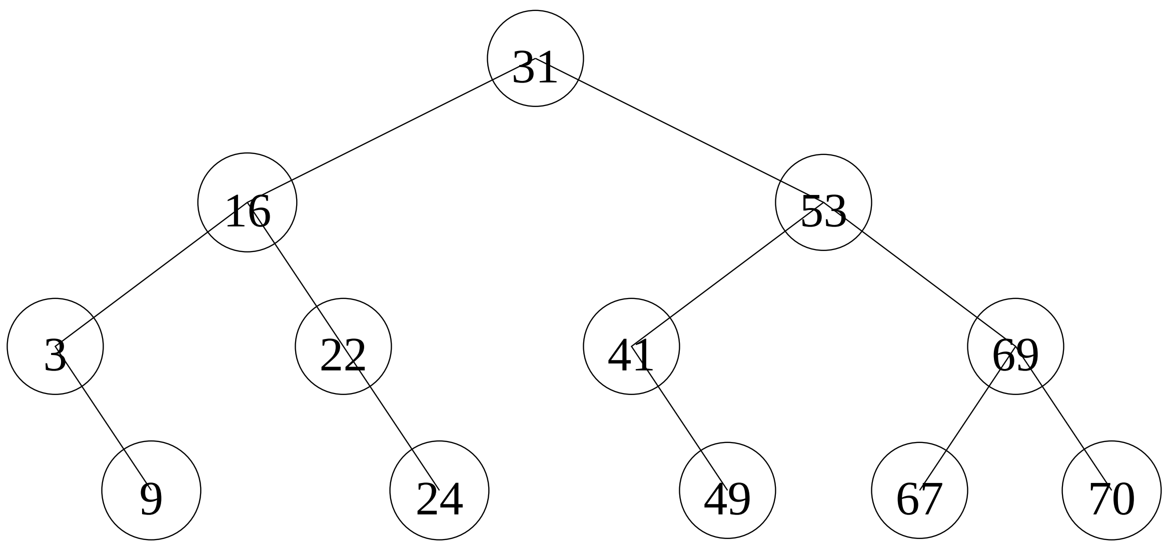
Order Arithmetic (cont.)

- Also, the base to which a logarithm is computed doesn't affect the order of magnitude, because changing the base of the logarithm from 2 to c changes the value by a constant factor of $\log_2 c$.

Calculating the Efficiency of Binary Search

- Consider the following array and its possible traversals by `findIndex`.

3	9	16	22	24	31	41	49	53	67	69	70



- The longest traversal is $\lceil log(n+1) \rceil$ where n is the length of the array.

Can We Do Better Than Binary Search?

- How do you find a number in a phone book?
  - Specifically, if I asked you find "Alan Cox" in the phone book would you start in the middle?

Interpolation Search

- We can rewrite

  ```
  mid = (lo + hi)/2
  ```

  as

  ```
  mid = lo + (hi - lo)/2
  ```

  and replace `(hi - lo)/2` with an expression that places us closer to what we're looking for

  ```
  mid = lo + ((key - a[lo])*(hi - lo))/(a[hi] - a[lo])
  ```