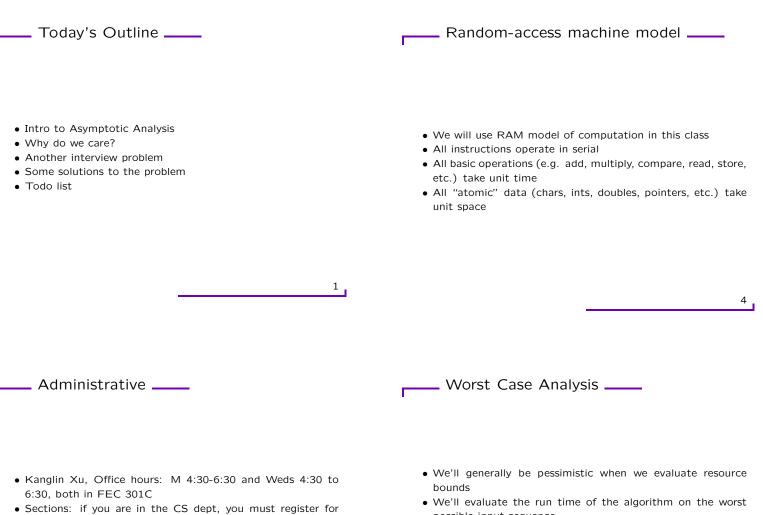
CS 361, Lecture 2

Jared Saia University of New Mexico

- There are several resource bounds we could be concerned about: time, space, communication bandwidth, logic gates, etc.
- However, we are usually most concerned about time
- Recall that algorithms are independent of programming languages and machine types
- Q: So how do we measure resource bounds of algorithms

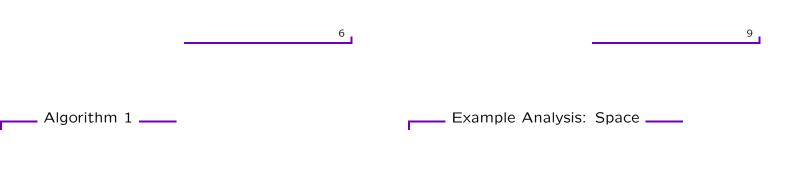


- one of the two sections (Th 3:30-4:20 or F 1:00-1:50)
- Book: "Introduction to Algorithms" by Cormen, Leiserson, Rivest, and Stein
- Pretest due on Tuesday

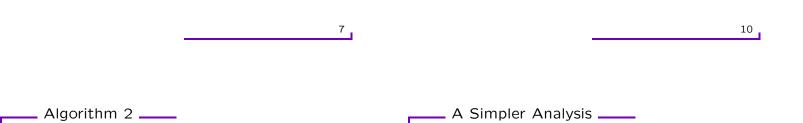
- possible input sequence
- Amazingly, in most cases, we'll still be able to get pretty good bounds
- Justification: The "average case" is often about as bad as the worst case.

3

- Consider the problem discussed last tuesday about finding a redundant element in an array
- Let's consider the more general problem, where the numbers are 1 to n instead of 1 to 1,000,000
- Worst case: Algorithm 1 does 5 * n operations (n inits to 0 in "count" array, n reads of input array, n reads of "count" array (to see if value is 1), n increments, and n stores into count array)
- Worst case: Algorithm 2 does 2 * n + 4 operations (*n* reads of input array, *n* additions to value *S*, 4 computations to determine *x* given *S*)

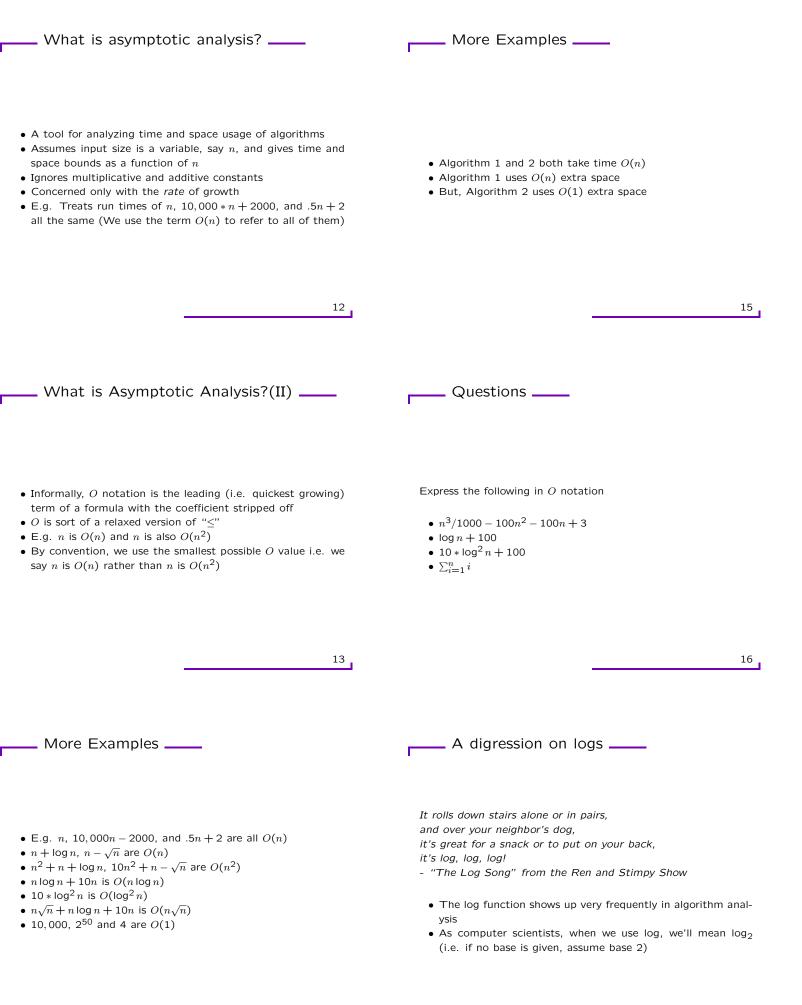


- Create a new "count" array of ints of size *n*, which we'll use to count the occurences of each number. Initialize all entries to 0
- Go through the input array and each time a number is seen, update its count in the "count" array
- As soon as a number is seen in the input array which has already been counted once, return this number
- Worst Case: Algorithm 1 uses *n* additional units of space to store the "count" array
- Worst Case: Algorithm 2 uses 2 additional units of space

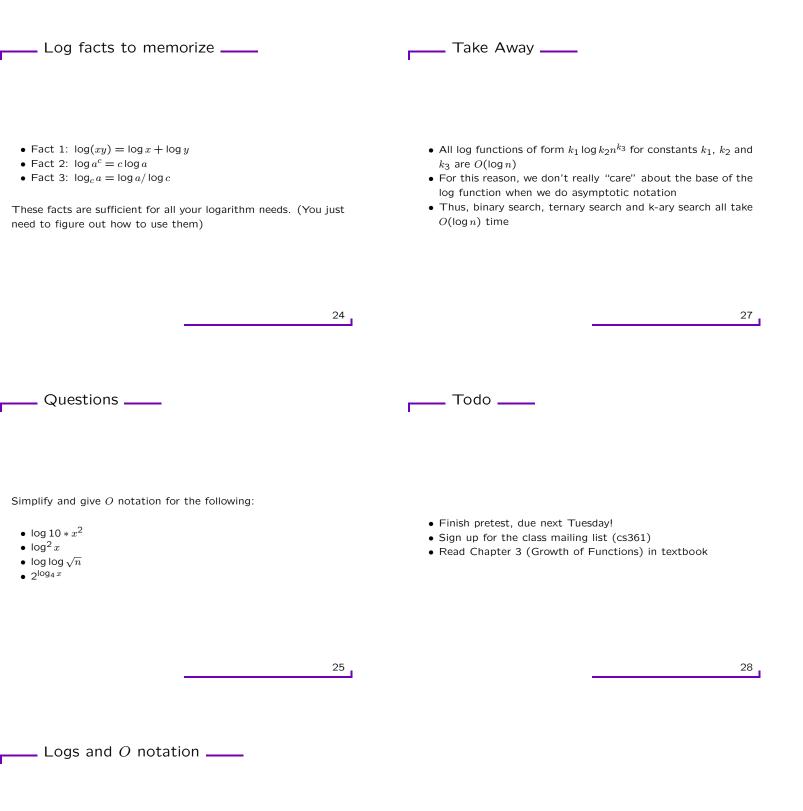


- $\bullet\,$ Iterate through the input array, summing up all the numbers, let S be this sum
- Let x = S (n+1)n/2
- Return x

- Analysis above can be tedious for more complicated algorithms
- In many cases, we don't care about constants. 5n is about the same as 2n + 4 which is about the same as an + b for any constants a and b
- However we do still care about the difference in space: \boldsymbol{n} is very different from 2
- Asymptotic analysis is the solution to removing the tedium but ensuring good analysis







- Note that $\log_8 n = \log n / \log 8$.
- Note that $\log_{600} n^{200} = 200 * \log n / \log 600$.
- Note that $\log_{100000} 30*n^2 = 2*\log n / \log 100000 + \log 30 / \log 100000$.
- Thus, $\log_8 n$, $\log_{600} n^{600}$, and $\log_{100000} 30*n^2$ are all $O(\log n)$
- In general, for any constants k_1 and $k_2, \, \log_{k_1} n^{k_2} = k_2 \log n / \log k_1,$ which is just $O(\log n)$