Algorithms

Part 4: More Time Complexity

Logarithmic and Exponential Time

Notes for CSC 100 - The Beauty and Joy of Computing
The University of North Carolina at Greensboro

Reminders

Readings:

Emma - contribute to on-line discussion by Monday!

For Friday:

Pre-Lab work (Lab 7)

Next Week:

Mid-term Exam on Wednesday (study!)

Faster than linear list operations

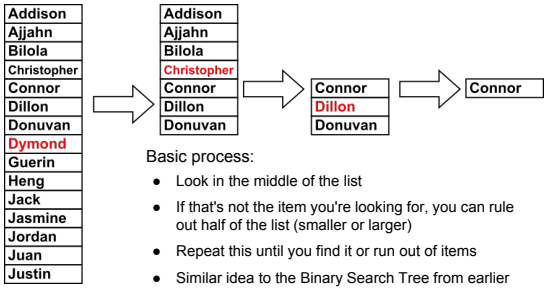
Think about how you find a word in a dictionary:

- From the Webster's web site: *"Webster's Third New International Dictionary*, Unabridged, together with its 1993 Addenda Section, includes some *470,000 entries*."
- If you checked every possible entry to see if it was the one you wanted, it would take way too long.
- How is a dictionary organized in order to make this easier?

Challenge: Describe precisely how to quickly look up a word.

Illustration for a list of students

Problem: Where's Connor? (Like "Where's Waldo?" but without the goofy hat)



Basic process:

- Look in the middle of the list
- If that's not the item you're looking for, you can rule out half of the list (smaller or larger)
- Repeat this until you find it or run out of items
- Similar idea to the Binary Search Tree from earlier

How long does this take?

At beginning: Could be any of n items
After 1 step: Could be any of $n/2$ items
After 2 steps: Could be any of $n/4$ items
After 3 steps: Could be any of $n/8$ items
...
After k steps: Could be any of $n/2^k$ items

To get to one item, need $n=2^k$ - so $k = \log_2 n$

While you're not responsible for knowing or being able to do this derivation, you do need to know about binary search and logarithmic time.

This analysis doesn't require anything beyond high school algebra to understand - so try to understand it!

This is called *logarithmic time*, and gives *very fast* algorithms!

n	$\log_2 n$
1000	10
1,000,000	20
1,000,000,000	30

So can find one item out of a billion in just 30 comparisons!!!

Something worse...

Problem: I have 60 items, each with a value, and want to find a subset with total value as close to some target T as possible.

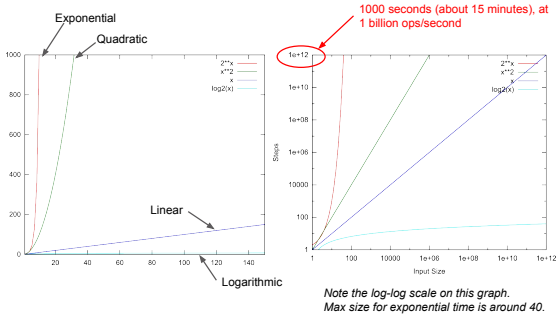
(The Price is Right on steroids...)

Algorithm: List all possible subsets of items
Add up total value of each subset
Find which one is closest

Question: If I have n items, how many subsets of n items are there?

Answer: There are 2^n subsets - this is *exponential time* (and very bad!)

Graphically comparing time complexities



Note the log-log scale on this graph.
Max size for exponential time is around 40.

Comparing with numbers

Different time complexities, by the numbers...

	Time in seconds at 1 billion ops/sec		Largest problem in 1 min at 1 billion ops/sec
	***n*=1,000**	***n*=1,000,000**	
$\log_2 n$	0.00000001	0.00000002	Huge*
n	0.000001	0.001	60,000,000,000
n^2	0.001	1000	244,949
2^n	10^{292}	10^{301029}	35

** Huge means a problem far larger than the number of atoms in the universe*

There is a lot more to this than what we have covered - but this gives a pretty accurate picture of basic algorithm time complexity!

Summary - All 4 Algorithms Lectures

- Algorithm "time complexity" is in basic steps
- Common complexities, from fastest to slowest are logarithmic, linear, quadratic, and exponential
  - A simple loop with constant time operations repeated is linear time
  - A loop containing a linear time loop is quadratic
  - A loop halving the problem size every iteration is logarithmic time
  - A program considering all subsets is exponential time
- Speed depends on algorithm time complexity
  - Logarithmic time is fantastic
  - Linear time is very good
  - Quadratic time is OK
  - Exponential time is awful
- Given time complexity and one actual time, can estimate time for larger inputs