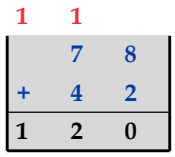
Today's topics

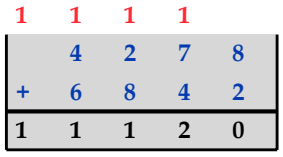
- Algorithms
- Complexity
- Upcoming
  - Security
  - Systems
- Reading
  - *Brookshear 5.6*
  - *Great Ideas* Chapter 13

Linear Growth

- Grade school addition
  - Work is proportional to number of digits N
  - *Linear* growth: kN for some constant k



N = 2



N = 4

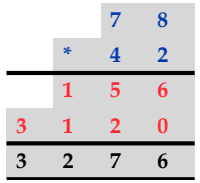
- How many reads? How many writes? How many operations?

New machines vs. new algorithms

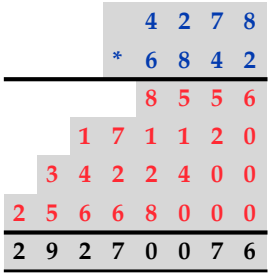
- New machine.
  - Costs $$$ or more.
  - Makes "everything" finish sooner.
  - Incremental quantitative improvements (Moore's Law).
  - May not help much with some problems.
- New algorithm.
  - Costs $ or less.
  - Dramatic qualitative improvements possible! (million times faster)
  - May make the difference, allowing specific problem to be solved.
  - May not help much with some problems.
- Algorithmic Successes
  - N-body Simulation, Discrete Fourier transform, Quantum mechanical simulations, Pixar movies…

Quadratic Growth

- Grade school multiplication
  - Work is proportional to *square* of number of digits N
  - *Quadratic* growth: kN^2 for some constant k



N = 2



N = 4

- How many reads? How many writes? How many operations?

Searching

- Determine the location or existence of an element in a collection of elements of the same type
- Easier to search large collections when the elements are already sorted
  - finding a phone number in the phone book
  - looking up a word in the dictionary
- What if the elements are not sorted?

- Sequential search
  - Worst case
  - Average case

Searching sorted input

- If the input is already sorted, we can search more efficiently than linear time
- Example: "Higher-Lower"
  - think of a number between 1 and 1000
  - have someone try to guess the number
  - if they are wrong, you tell them if the number is higher than their guess or lower
- Strategy?
- How many guesses should we expect to make?

Logarithms Revisited

- Power to which any other number *a* must be raised to produce *n*
  - *a* is called the base of the logarithm
- Frequently used logarithms have special symbols
  - lg n $= \log_2 n$ logarithm base 2
  - ln n $= \log_e n$ natural logarithm (base e)
  - log n $= \log_{10} n$ common logarithm (base 10)
- If we assume n is a power of 2, then the number of times we can recursively divide n numbers in half is lg n

Best Strategy

- Always pick the number in the middle of the range
- Why?
  - you eliminate half of the possibilities with each guess

- We should expect to make at most

  lg1000 ≈ 10 guesses

- Binary search
  - search *n* sorted inputs in logarithmic time

Sequential vs. binary search

- Average-case running time of sequential search is linear
- Average-case running time of binary search is logarithmic
- Number of comparisons:

n	sequential search	binary search
2	1	1
16	8	4
256	128	8
4096	2048	12
65536	32768	16

Why does it matter?

Run time (nanoseconds)		1.3 N^3	10 N^2	47 N $\log_2 N$	48 N
Time to solve a problem of size	1000	1.3 seconds	10 msec	0.4 msec	0.048 msec
	10,000	22 minutes	1 second	6 msec	0.48 msec
	100,000	15 days	1.7 minutes	78 msec	4.8 msec
	million	41 years	2.8 hours	0.94 seconds	48 msec
	10 million	41 millennia	1.7 weeks	11 seconds	0.48 seconds
Max size problem solved in one	second	920	10,000	1 million	21 million
	minute	3,600	77,000	49 million	1.3 billion
	hour	14,000	600,000	2.4 trillion	76 trillion
	day	41,000	2.9 million	50 trillion	1,800 trillion
N multiplied by 10, time multiplied by		1,000	100	10+	10

Sorting

- Given n items, rearrange them so that they are in increasing order
- A key recurring problem
- Many different methods, how do we choose?
- Given a set of cards, describe how you would sort them:

- Given a set of words, describe how you would sort them in alphabetical order?

Orders of Magnitude

Seconds	Equivalent
1	1 second
10	10 seconds
10^2	1.7 minutes
10^3	17 minutes
10^4	2.8 hours
10^5	1.1 days
10^6	1.6 weeks
10^7	3.8 months
10^8	3.1 years
10^9	3.1 decades
10^{10}	3.1 centuries
. . .	forever
10^{21}	age of universe

Meters Per Second	Imperial Units	Example
10^{-10}	1.2 in / decade	Continental drift
10^{-8}	1 ft / year	Hair growing
10^{-6}	3.4 in / day	Glacier
10^{-4}	1.2 ft / hour	Gastro-intestinal tract
10^{-2}	2 ft / minute	Ant
1	2.2 mi / hour	Human walk
10^2	220 mi / hour	Propeller airplane
10^4	370 mi / min	Space shuttle
10^6	620 mi / sec	Earth in galactic orbit
10^8	62,000 mi / sec	1/3 speed of light

Powers of 2		
	2^{10}	thousand
	2^{20}	million
	2^{30}	billion

Comparisons in insertion sort

- **Worst case**
  - element k requires (k-1) comparisons
  - total number of comparisons:

    $0+1+2+ \ \ldots \ + (n-1) \quad = \frac{1}{2}(n)(n-1)$

    $= \frac{1}{2}(n^2-n)$
- **Best case**
  - elements 2 through n each require one comparison
  - total number of comparisons:

    $1+1+1+ \ \ldots \ + 1 \ = n-1$

(n-1) times

Running time of insertion sort

- **Best case running time is *linear***
- **Worst case running time is *quadratic***
- **Average case running time is also *quadratic***
  - on average element k requires (k-1)/2 comparisons
  - total number of comparisons:

    $\frac{1}{2}(0+1+2+ \ \ldots \ + n-1) \quad = \frac{1}{4}(n)(n-1)$

    $= \frac{1}{4}(n^2-n)$

Comparisons in merging

- **Merging two sorted lists of size m requires at least m and at most 2m-1 comparisons**
  - m comparisons if all elements in one list are smaller than all elements in the second list
  - 2m-1 comparisons if the smallest element alternates between lists

Comparisons at each merge

#lists	#elements in each list	#merges	#comparisons per merge	#comparisons total
n	1	n/2	1	n/2
n/2	2	n/4	3	3n/4
n/4	4	n/8	7	7n/8
…	…	…	…	…
2	n/2	1	n-1	n-1

Comparisons in mergesort

- Total number of comparisons is the sum of the number of comparisons made at each merge
  - at most n comparisons at each merge
  - the number of times we can recursively divide n numbers in half is lg n, so there are lg n merges
  - there are at most $n \lg n$ comparisons total

Comparison of sorting algorithms

- Best, worst and average-case running time of mergesort is $\Theta(n \lg n)$
- Compare to average case behavior of insertion sort:

n	Insertion sort	Mergesort
10	25	33
100	2500	664
1000	250000	9965
10000	25000000	132877
100000	2500000000	1660960

Quicksort

- Most commonly used sorting algorithm
- One of the fastest sorts in practice
- Best and average-case running time is $O(n \lg n)$
- Worst-case running time is *quadratic*
- Runs very fast on most computers when implemented correctly

Algorithmic successes

- N-body Simulation
- Discrete Fourier transform
- Quantum mechanical simulations
- Pixar movies…