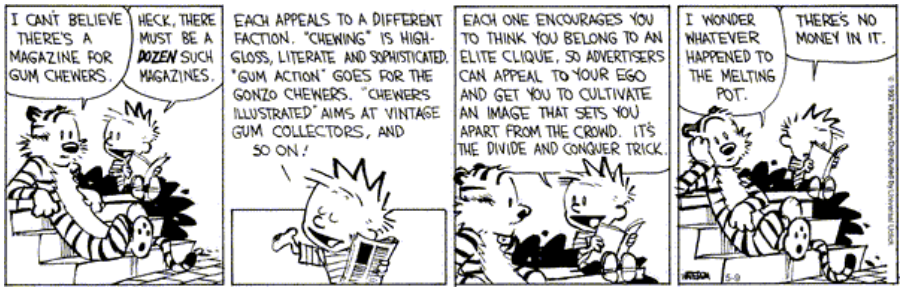
Divide-and-Conquer!



From last time…

- Insertion sort
  - Incremental
  - Loop-invariants
  - Best-/worse- cast running time
  - http://www.sorting-algorithms.com/insertion-sort
- Today
  - Divide-and-conquer
  - Big-Oh

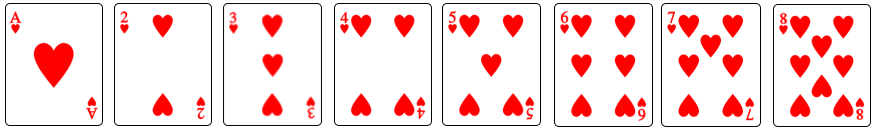
Divide-and-conquer

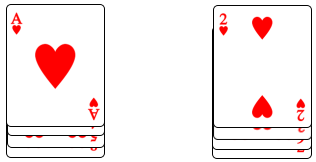
- Last time: insertion sort an example of an incremental algorithm
- Divide-and-conquer another paradigm:
  - **Divide** the problem into subproblems that are smaller instances of the original
  - **Conquer** subproblems by solving them recursively
    - Base-case: small enough problem solved by brute-force
  - **Combine** the subproblem solutions to give a solution to the overall problem

A D&C approach to search

- Lets do it!
- Divide:
- Conquer:
- Combine:

Merge



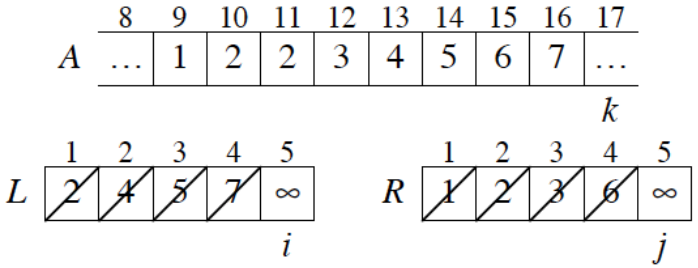


Merge

MERGE(A, p, q, r)

$n_1 = q - p + 1$
$n_2 = r - q$
let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
for $i = 1$ **to** n_1
    $L[i] = A[p + i - 1]$
for $j = 1$ **to** n_2
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ **to** r
    if $L[i] \le R[j]$
        $A[k] = L[i]$
        $i = i + 1$
    else $A[k] = R[j]$
        $j = j + 1$

Merge



Merge

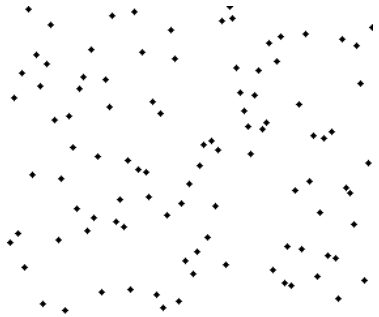
MERGE(A, p, q, r)

$n_1 = q - p + 1$
$n_2 = r - q$
let $L[1..n_1 + 1]$ and $R[1..n_2 + 1]$ be new arrays
for $i = 1$ **to** n_1
    $L[i] = A[p + i - 1]$
for $j = 1$ **to** n_2
    $R[j] = A[q + j]$
$L[n_1 + 1] = \infty$
$R[n_2 + 1] = \infty$
$i = 1$
$j = 1$
for $k = p$ **to** r
    if $L[i] \le R[j]$
        $A[k] = L[i]$
        $i = i + 1$
    else $A[k] = R[j]$
        $j = j + 1$

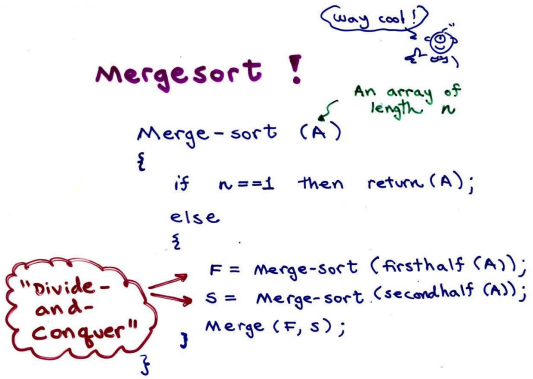
Correct?

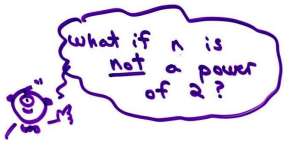
Merge sort in action

6 5 3 1 8 7 2 4



Source: Wikimedia





http://www.sorting-algorithms.com/merge-sort

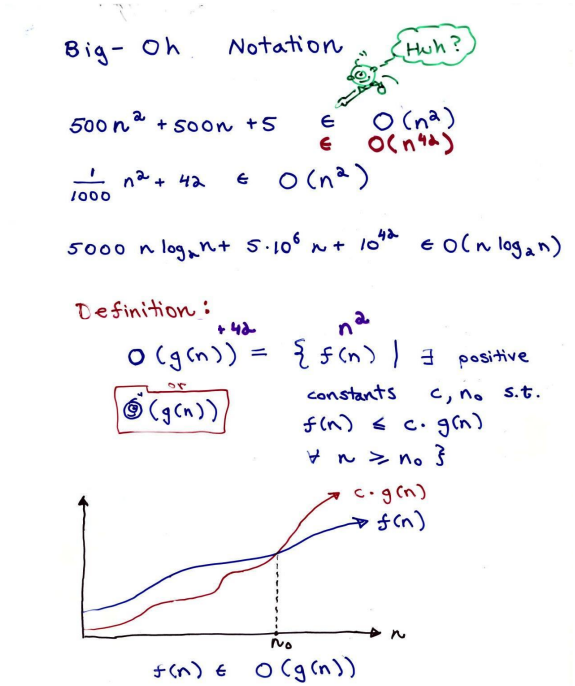
Merge Sort: Running time analysis

Divide-and-conquer: Multiplication

- Addition/subtraction: linear
- Multiplication/division: quadratic
  - Can we do better?

```
>>> 42424242424242424242 * 474747474747474747474747474747
20140801958983777165393939192531374349556167741
```

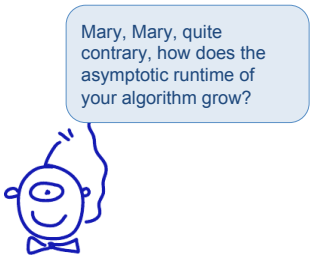
D&C Example: Multiplication



Algorithm Complexity

- General way to describe functions (runtimes, space) in the *limit*
  - I.e., describing the *growth* of the function
- Abstract away unimportant bits (low-order terms, constant factors)

$$O \approx \leq$$
$$\Omega \approx \geq$$
$$\Theta \approx =$$



Worksheet!

- Show that $5n^2 + 8n + 42 \in O(n^2)$
- Prove (BWOC) that $\frac{1}{10} n^3 \notin O(n^2)$
- Is $(1.0001)^n \in O(n^{42})$? (Prove your answer!)

Big-Omega Notation

$\frac{1}{10} n^2 \in \Omega(n^2)$

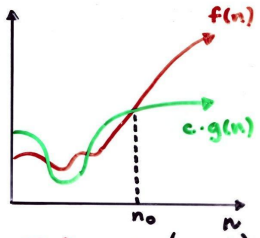
$n^3 \in \Omega(n^2)$

Formally

$\Omega(g(n)) =$

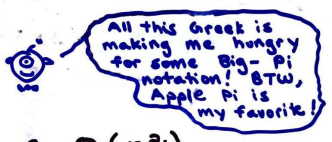
$\{f(n) \mid \exists$ positive constants c, n_0 s.t. $c \cdot g(n) \leq f(n)$ $\forall n \geq n_0.\}$

Graphically



$f(n) \in \Omega(g(n))$

Big-Theta Notation



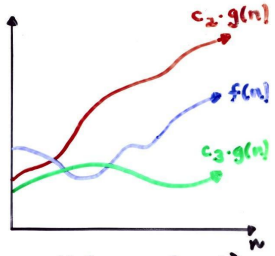
$n^2 + n + 1 \in \Theta(n^2)$

$50n \log_2 n + 500n \in \Theta(n \log_2 n)$

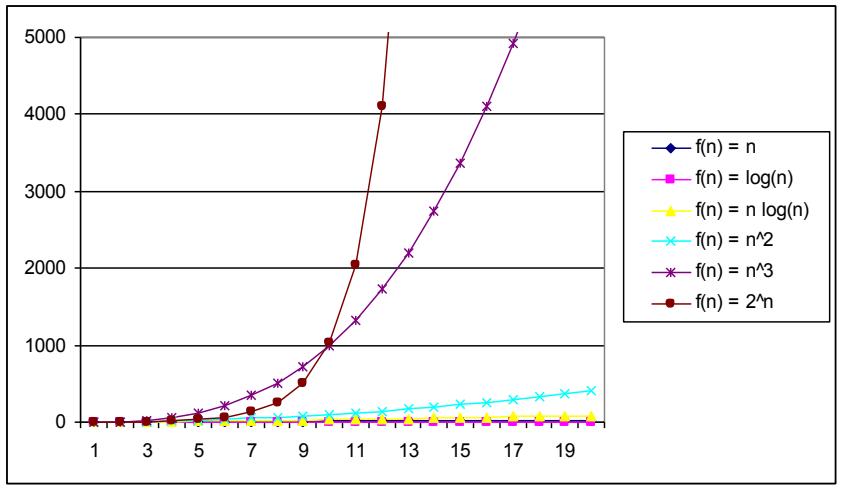
Formally

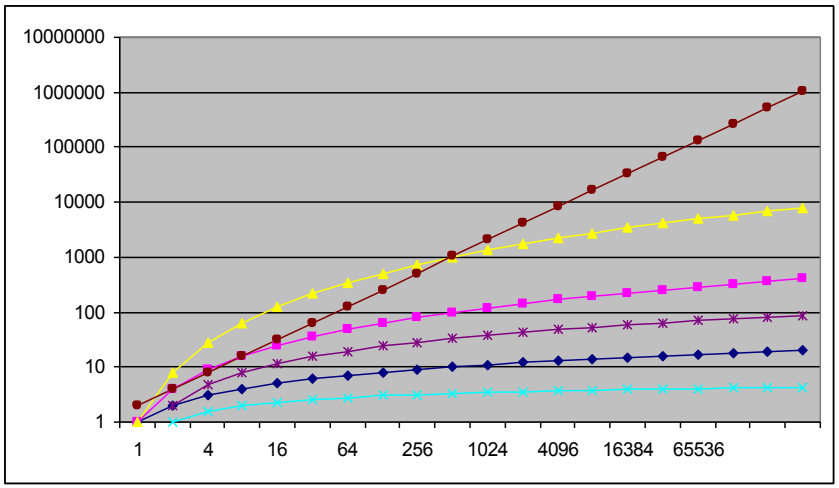
$f(n) \in \Theta(g(n))$ iff $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

Graphically



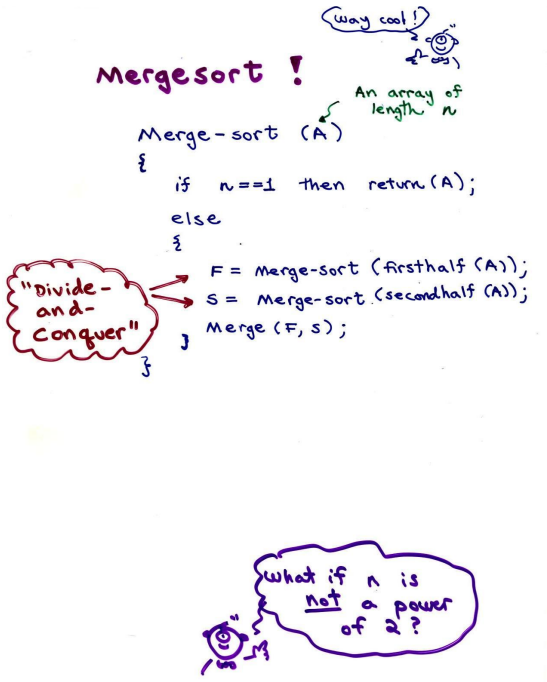
$f(n) \in \Theta(g(n))$





Assumptions

- Random Access Model
  - Non-concurrent
  - Assume each machine-level instruction runs in constant time:
    - Arithmetic: add, subtract, multiply, divide, remainder, floor ceiling , etc.
    - Data movement: load, store, copy
    - Control: (conditional) branches, subroutine calls, returns
- Measure time/space complexity in terms of size of input



Where do our searches fit?

- Insertion Sort: http://www.sorting-algorithms.com/insertion-sort

INSERTION-SORT(A, n)	*cost*	*times*
for $j = 2$ **to** n	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

- Merge sort: http://www.sorting-algorithms.com/merge-sort

Where do our searches fit?

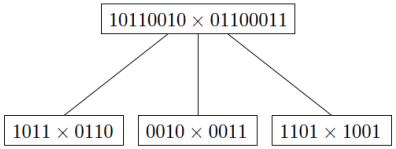
- Insertion Sort:

INSERTION-SORT(A, n)	*cost*	*times*
for $j = 2$ **to** n	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
$i = j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

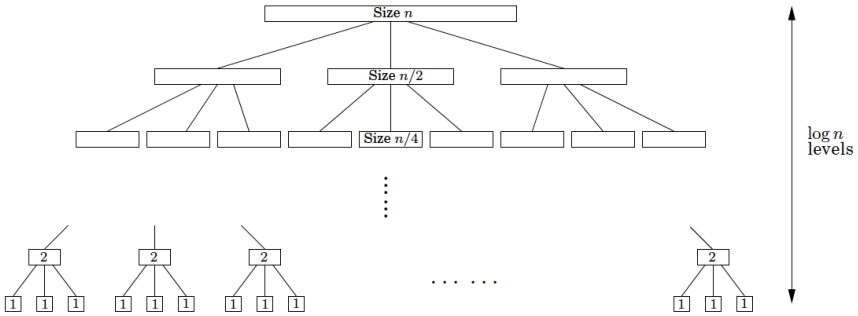
- Merge sort
- What about multiplication?

Figure 2.2 Divide-and-conquer integer multiplication. (a) Each problem is divided into three subproblems. (b) The levels of recursion.

(a)



(b)



Next time

A general framework for analyzing runtime of D&C approaches