Divide and Conquer to Multiply and Order

Reading: Chapter 18

Divide-and-conquer is a frequently-useful algorithmic technique tied up in recursion.

We'll see how it is useful in

- SORTING
- MULTIPLICATION

A *divide-and-conquer algorithm has three basic steps...*

1. *Divide problem into smaller versions of the same problem.*
2. *Recursively solve each smaller version.*
3. *Combine solutions to get overall solution.*

Sorting

Problem SORT:

Input: *an array* `A` *of integers.*
Output: *an array with all elements of* `A` *in increasing order.*

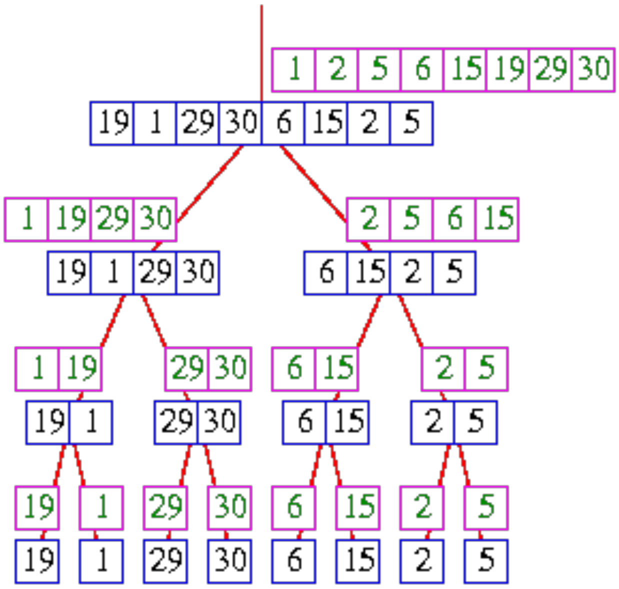
example:

```
         +----+----+----+----+----+----+----+----+
     A = | 19 |  1 | 29 | 30 |  6 | 15 |  2 |  5 |
         +----+----+----+----+----+----+----+----+

         +----+----+----+----+----+----+----+----+
  return |  1 |  2 |  5 |  6 | 15 | 19 | 29 | 30 |
         +----+----+----+----+----+----+----+----+
```

Merge-Sort

> ***Algorithm*** *M-Sort(A)****:***
> ***if*** *a's length > 1,* ***then:***
>   ***return*** *A.*
> ***else:***
>   *Let A_L be the first half of A.*
>   *Let A_R be the second half of A.*
>   ***return*** *Merge(M-Sort(A_L), M-Sort(A_R))*
> ***end of if***



Merging

> ***Algorithm*** *Merge(A, B)**:***
> *Let n_A and n_B be the length of arrays A and B.*
> *// C holds merged array*
> *Let C be an array of length $n_A + n_B$.*
> *// a,b,c are current positions in A,B,C*
> *Let a, b, and c hold 0.*
> ***while*** *$a < n_A$* ***and*** *$b < n_B$,* ***do:***
>   ***if*** *$a < n_A$* ***and*** *$A[a] < B[b]$,* ***then:***
>   *Let C[c] hold A[a].*
>   *Add 1 to a and c.*
>   ***else:***
>   *Let C[c] hold B[b].*
>   *Add 1 to b and c.*
>   ***end of if***
> ***end of loop***
> ***return*** *C.*

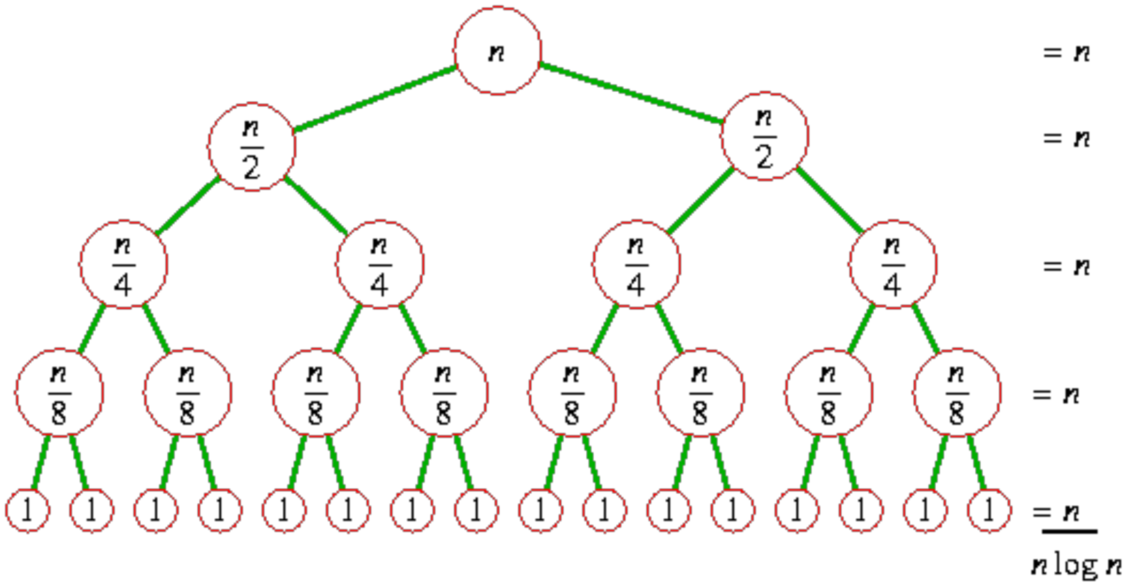
example:

```
+----+----+----+----+
|  1 | 19 | 29 | 30 |
+----+----+----+----+\     +----+----+----+----+----+----+----+----+
                      >-->|  1 |  2 |  5 |  6 | 15 | 19 | 29 | 30 |
+----+----+----+----+/     +----+----+----+----+----+----+----+----+
|  2 |  5 |  6 | 15 |
+----+----+----+----+
```

How Much Time?

Write a recurrence and solve...

```
T(n) = 2 T(n / 2) + cn
```



Multiplication

Problem MULTIPLICATION

Input: *two n-digit integers a and b*
Output: *product of a and b*

example:

```
          1980  = a
x         2315  = b
---------
          9900
         1980
        5940
+ 3960
---------
       4573700  = a x b
```

This is the algorithm you learned in grade school. Notice it takes O(n^2) time.

Dividing the Problem

We divide each integer into two halves.

```
aL = 19   |   80 = aR
          |
bL = 23   |   15 = bR


              aL           aR
  x           bL           bR
-----------------------------
            aL bR        aR bR
```

```
+ aL bL       aR bL
  -----------------------------
  aL bL   aL bR + aR bL   aR bR
```

So our algorithm is to compute aL bL, aL bR, aR bL, and aR bR, and add.

```
T(n) <= 4 T(n / 2) + O(n)

T(n) = O(n^2)
```

Algorithm *Divide-Mult(a,b)***:**
if *a or b has one digit,* **then:**
  *return a * b.*
else:
  Let n be the number of digits in max{a, b}.
  Let a_L and a_R be left and right halves of a.
  Let b_L and b_R be left and right halves of b.
  Let x_1 hold Divide-Mult(a_L, b_L).
  Let x_2 hold Divide-Mult(a_L, b_R).
  Let x_3 hold Divide-Mult(a_R, b_L).
  Let x_4 hold Divide-Mult(a_R, b_R).
  return $x_1*10^n + (x_2 + x_3)*10^{n/2} + x_4$.
end of if

Being Clever

We can actually get away with just three multiplications!

```
x1 = aL bL
x2 = aR bR
x3 = (aL + aR) (bL + bR)

            aL          aR
  x         bL          bR
-----------------------------
aL bL   aL bR + aR bL   aR bR
  x1     x3 - x1 - x2     x2
```

Now our algorithm is to compute x1, x2, and x3, find x3 - x1 - x2, and add.

Algorithm *Karatsuba(a,b)***:**
if *a or b has one digit,* **then:**
  *return a * b.*
else:
  Let n be the number of digits in max{a, b}.
  Let a_L and a_R be left and right halves of a.
  Let b_L and b_R be left and right halves of b.
  Let x_1 hold Karatsuba(a_L, b_L).
  Let x_2 hold Karatsuba($a_L + a_R$, $b_L + b_R$).
  Let x_3 hold Karatsuba(a_R, b_R).
  return $x_1*10^n + (x_2 - x_1 - x_3)*10^{n/2} + x_3$.

end of if

*Ooooh!*

*Aaaah!*

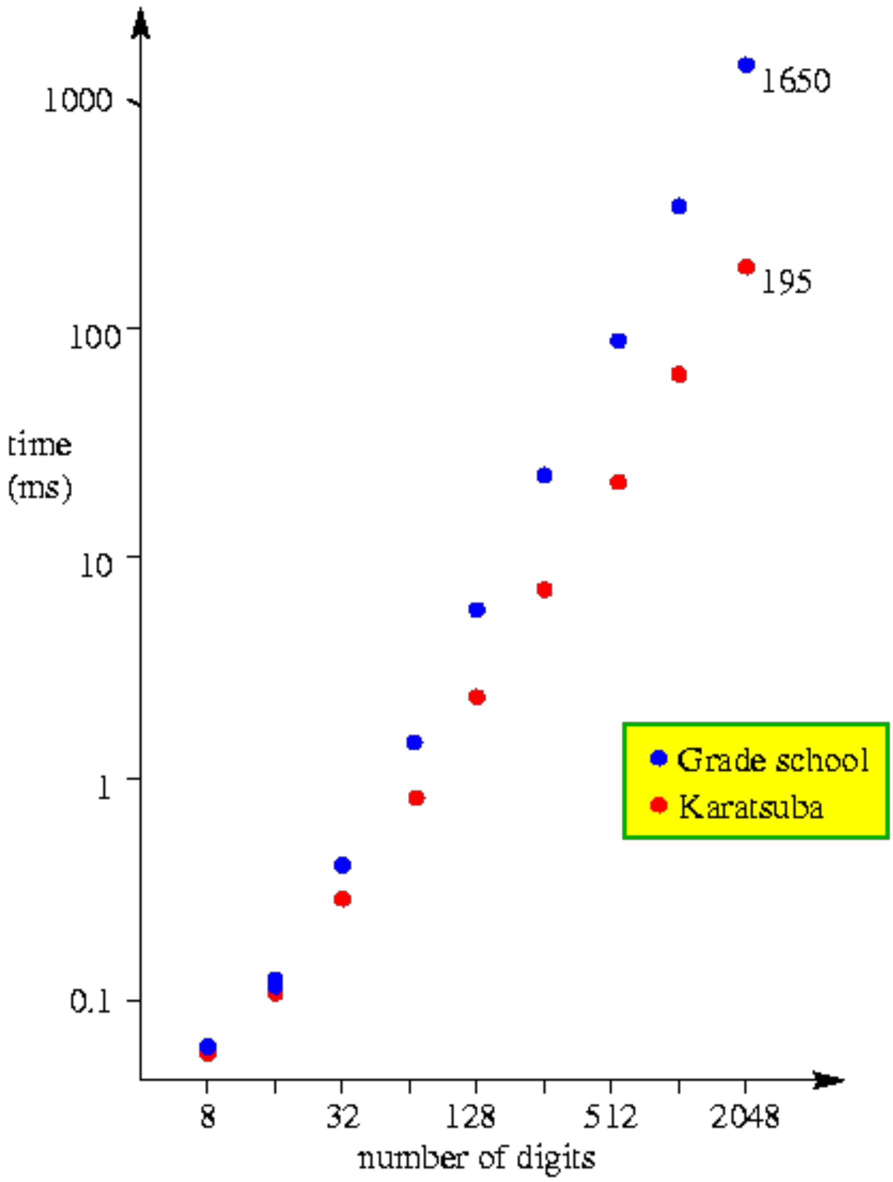
An Example

```
IN Multiply(1980, 2315)
 19 80
 23 15
    IN Multiply(19, 23)
     1 9
     2 3
        IN Multiply(1, 2)
        return 2
        IN Multiply(9, 3)
        return 27
        IN Multiply(10, 5)
         1 0
         0 5
            IN Multiply(1, 0)
            return 0
            IN Multiply(0, 5)
            return 0
            IN Multiply(1, 5)
            return 5
         5 - 0 - 0 = 5
         0
          5
           0
         ---
          50
        return 50
     50 - 27 - 2 = 21
     2
     21
      27
     ---
     437
    return 437
    IN Multiply(80, 15)
     8 0
     1 5
        IN Multiply(8, 1)
        return 8
        IN Multiply(0, 5)
        return 0
        IN Multiply(8, 6)
        return 48
     48 - 8 - 0 = 40
      8
      40
        0
     ----
     1200
    return 1200
```

```
     IN Multiply(99, 38)
      9 9
      3 8
         IN Multiply(9, 3)
         return 27
         IN Multiply(9, 8)
         return 72
         IN Multiply(18, 11)
          1 8
          1 1
             IN Multiply(1, 1)
             return 1
             IN Multiply(8, 1)
             return 8
             IN Multiply(9, 2)
             return 18
          18 - 8 - 1 = 9
          1
           9
            8
          ---
          198
         return 198
       198 - 27 - 72 = 99
       27
        99
         72
       ----
       3762
     return 3762
 3762 - 437 - 1200 = 2125

 437
  2125
    1200
 -------
 4583700
return 4583700
```

How Do Multiplication Algorithms Compare in Practice?


1000
100
10
1
0.1
time
(ms)
1650
195
Grade school
Karatsuba
8
32
128
512
2048
number of digits