5.1.2

Integer Multiplication

Divide and Conquer

Divide and Conquer

1. Divide the input into subproblems
2. Solve each subproblem
3. Carefully merge the solutions (often $O(n)$)

Merge Sort

Merge-Sort(*L*)

If $|L| = 1$ then Return L

Split L into two halves A, B
A <- Merge-Sort(A)
B <- Merge-Sort(B)
L <- Merge(A,B)

Return L

Runtime:

$T(n)$:= runtime of MS on a list of length n.

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + cn$$

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + c\left(\frac{n}{2}\right)$$

Another D&C Example: Counting Inversions

Input: A list L of distinct numbers $x_1, \ldots, x_n$

An inversion is a pair of indices $i < j$, s.t. $x_i > x_j$.

Output: The # of inversions.

#of inversions is sort of how "unsorted" L is

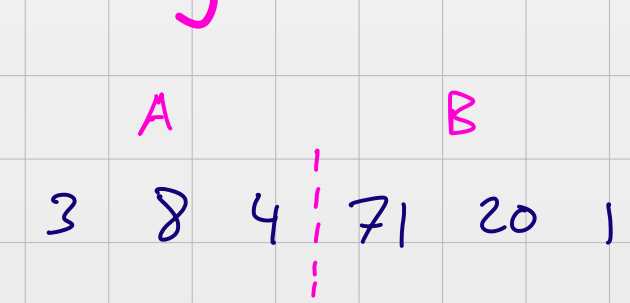
Another D&C Example: Counting Inversions

Input: A list L of distinct numbers $x_1, \ldots, x_n$

An inversion is a pair of indices $i < j$, s.t. $x_i > x_j$.

Output: The # of inversions.

#of inversions is sort of how "unsorted" L is



# of inversions in A: r_A

# of inversions in B: r_B

Obs 1 total # of inversions $r = r_A + r_B + r_{cross}$

Obs 2 r_{cross} doesn't care about the orders within A and B.

Another D&C Example: Counting Inversions

Obs 1 total # of inversions $r = r_A + r_B + r_{cross}$

Obs 2 r_{cross} doesn't come about the orders within A and B.

A B

3 8 4 | 7 1 20 1

3 4 8 | 1 20 7 1

~~3~~ ~~4~~ 8

~~1~~ ~~7~~ 20

output list

1 3 4 7 8 20

count

3 + 1 = 4

MERGE-AND-COUNT

Another D&C Example: Counting Inversions

Sort-and-Count (L)

If $|L| = 1$ then Return 0
Divide L into two halves A, B
    A has first ceil(n/2) elements
    B has last floor(n/2) elements
(r_A , A) <- Sort-and-Count(A)
(r_B , B) <- Sort-and-Count(B)
(r , L) <- Merge-and-Count(A,B)

Return ($r + r_A + r_A$, L)

Another D&C Example: Counting Inversions

Sort-and-Count (L)

If $|L| = 1$ then Return 0
Divide L into two halves A, B
    A has first ceil(n/2) elements
    B has last floor(n/2) elements
(r_A, A) <- Sort-and-Count(A)
(r_B, B) <- Sort-and-Count(B)
(r, L) <- Merge-and-Count(A,B)

Return $(r + r_A + r_A, L)$

Merge-and-Count(A,B)
L <- empty list
Count <- 0 ; i <- 1 ; j <- 1
While $i \leq |A|$ and $j \leq |B|$
    If $a_i > b_j$ then
        Append b_j to L
        Count <- Count + ($|A| - i$)
        j <- j + 1
    else
        Append a_i to L
        i <- i + 1
Return (Count , L)

Runtime:

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

$\Rightarrow O(n \log n)$

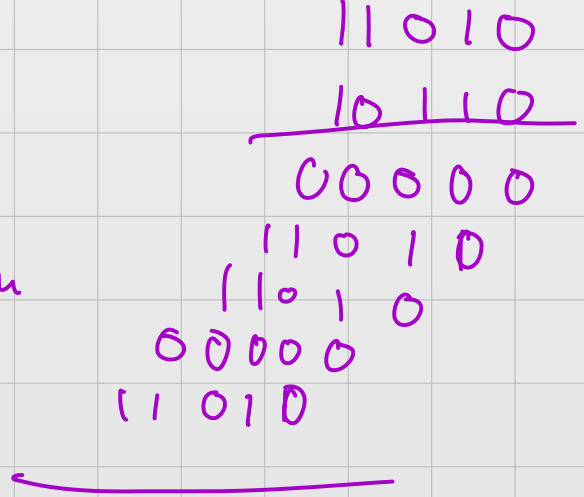
Integer Multiplication

Input : 2 n-bit integers x and y (in binary)

Output : $x \cdot y$ (in binary)

"School book" multiplication

$O(n^2)$



Integer Multiplication

Input : 2 n-bit integers x and y (in binary)
Output : $x \cdot y$ (in binary)

"School book" multiplication $O(n^2)$

Kolmogorov conjectured $O(n^2)$ is tight (1956)

Karatsuba showed $O(n^{\log_3 2}) = O(n^{1.59})$

Harvey + others 2019 $O(n \log n)$

Integer Multiplication

$$x = x_1 2^{n/2} + x_0$$
$$y = y_1 2^{n/2} + y_0$$

$x = 1101$

$y = 1001$

$x_1 = 11 \quad x_0 = 01$

$y_1 = 10 \quad y_0 = 01$

$$x \cdot y = (x_1 2^{n/2} + x_0)(y_1 2^{n/2} + y_0)$$

$$= x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$$

Integer Multiplication

Multiply1(x,y,n)

If $|L| = 1$ then Return $x \cdot y$
$m = n/2$
$x_1 = x / 2^m$; $x_0 = x \bmod 2^m$
$y_1 = y / 2^m$; $y_0 = y \bmod 2^m$
a <- Multiply1(x_1, y_1,m)
b <- Multiply1(x_1, y_0,m)
c <- Multiply1(x_0, y_1,m)
d <- Multiply1(x_0, y_0,m)

Return $a \cdot 2^n + (b + c) \cdot 2^{n/2} + d$

Does this beat $O(n^2)$

$T(n) = 4T\left(\frac{n}{2}\right) + cn \qquad n \geq 2$

Same as mergesort except $2 \to 4$.

$T(n) = O(1) \qquad n = 1$

Master Theorem

Theorem. Let $a \geq 1$, $b \geq 2$, and $c \geq 0$ and suppose that $T(n)$ is a function on the non-negative integers that satisfies the recurrence

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$$

with $T(0) = 0$ and $T(1) = \Theta(1)$. Then

Case 1: If $c > \log_b a$, then $T(n) = \Theta(n^c)$
Case 2: If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$
Case 3: If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$

Apply the Master Theorem

Integer Mult. D&C #1

$T(n) = 4T\left(\frac{n}{2}\right) + dn$

$a = 4$

$b = 2$ $\quad \log_2 4 = 2$

$c = 1$ $\quad \Rightarrow c < \log_b a$

$\Rightarrow T(n) = O\left(n^{\log_b a}\right) = O(n^2)$

Master Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$$

Case 1: If $c > \log_b a$, then $T(n) = \Theta(n^c)$
Case 2: If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$
Case 3: If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$

Apply the Master Theorem

Merge Sort

$T(n) = 2\,T\left(\frac{n}{2}\right) + dn$

$a = 2$

$b = 2$

$c = 1$

$\log_b a = \log_2 2 = 1$

$\Rightarrow c = \log_b a$

$\Rightarrow T(n) = O(n^c \log n) = O(n \log n)$

Master Theorem

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$$

Case 1: If $c > \log_b a$, then $T(n) = \Theta(n^c)$

Case 2: If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$

Case 3: If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$

Can we do better?

Multiply1(x,y,n)

If $|L| = 1$ then Return $x \cdot y$
$m = n/2$
$x_1 = x / 2^m$; $x_0 = x \bmod 2^m$
$y_1 = y / 2^m$; $y_0 = y \bmod 2^m$
a <- Multiply1(x_1, y_1,m)
b <- Multiply1(x_1, y_0,m)
c <- Multiply1(x_0, y_1,m)
d <- Multiply1(x_0, y_0,m)

Return $a \cdot 2^n + (b + c) \cdot 2^{n/2} + d$

$$x \cdot y = (x_1 2^{n/2} + x_0)(y_1 2^{n/2} + y_0)$$
$$= x_1 y_1 2^n + (x_1 y_0 + x_0 y_1) 2^{n/2} + x_0 y_0$$

$$z_1 = x_1 y_0 + x_0 y_1 + x_1 y_1 - x_1 y_1 + x_0 y_0 - x_0 y_0$$
$$= (x_1 + x_0) y_0 + (x_0 + x_1) y_1 - x_1 y_1 - x_0 y_0$$
$$= (x_0 + x_1)(y_0 + y_1) - x_1 y_1 - x_0 y_0$$

3 multiplications: $x_0 y_0$ $x_1 y_1$ $(x_0 + x_1)(y_0 + y_1)$

Can we do better?

$$T(n) = 3\,T\left(\frac{n}{2}\right) + dn$$

$a = 3 \quad b = 2 \quad c = 1 \quad \log_2 3 = 1{,}59$

$$\Rightarrow T(n) = O\left(n^{\log_b a}\right) = O\left(n^{\log_2 3}\right) = O\left(n^{1{,}59}\right)$$

Karatsuba(*x*,*y*,*n*)

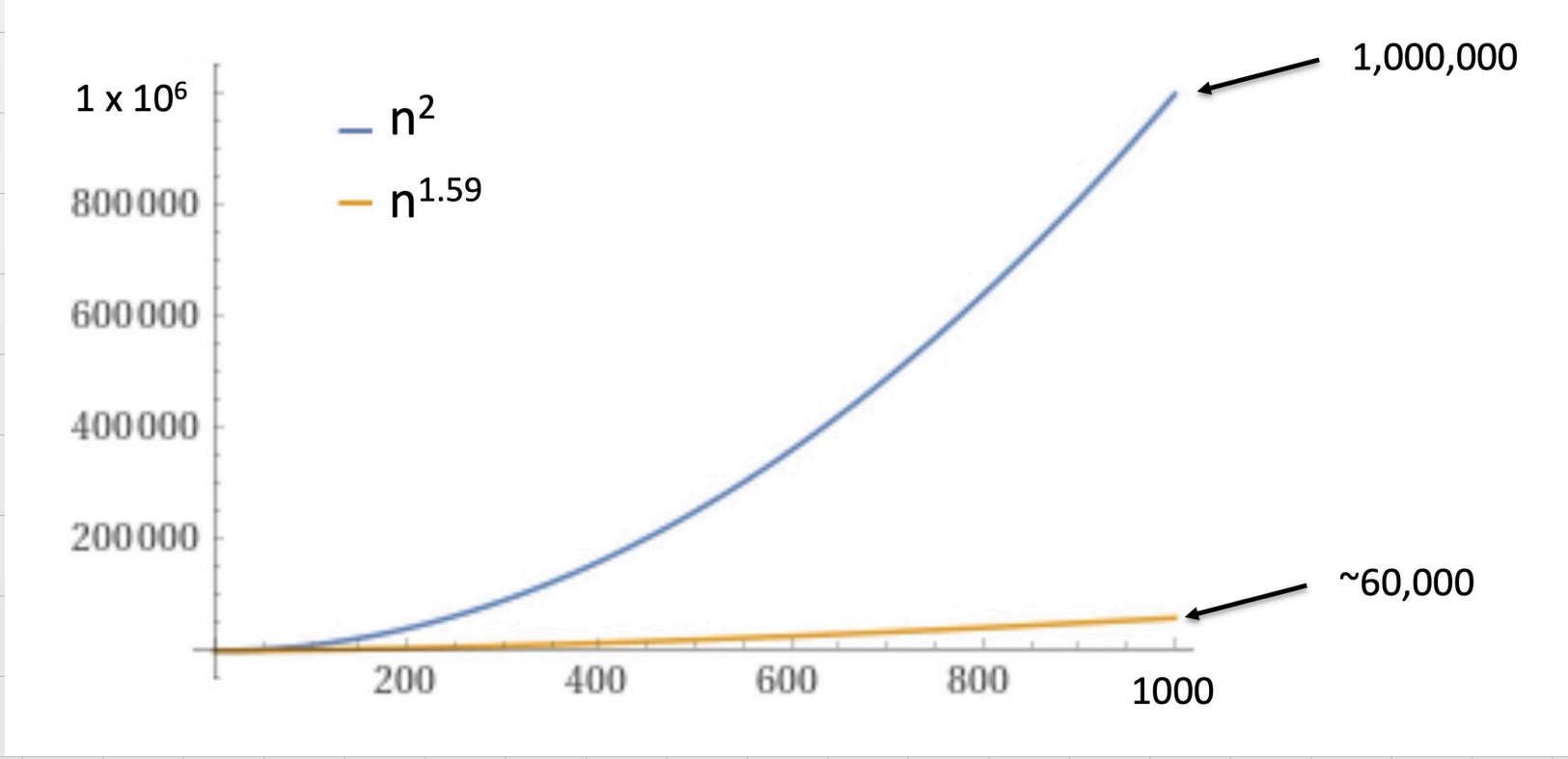
If $|L| = 1$ then Return $x \cdot y$
$m = n/2$
$x_1 = x / 2^m$; $x_0 = x \bmod 2^m$
$y_1 = y / 2^m$; $y_0 = y \bmod 2^m$
a <- Karatsuba(x_1, y_1,m)
b <- Karatsuba($x_1 + x_0$, $y_1 + y_0$,m)
d <- Karatsuba(x_0, y_0,m)

Return $a \cdot 2^n + (b - a - d) \cdot 2^{n/2} + d$

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \Theta(n^c)$$

Case 1: If $c > \log_b a$, then $T(n) = \Theta(n^c)$
Case 2: If $c = \log_b a$, then $T(n) = \Theta(n^c \log n)$
Case 3: If $c < \log_b a$, then $T(n) = \Theta(n^{\log_b a})$

Should we Care?



Integer multiplication algorithms

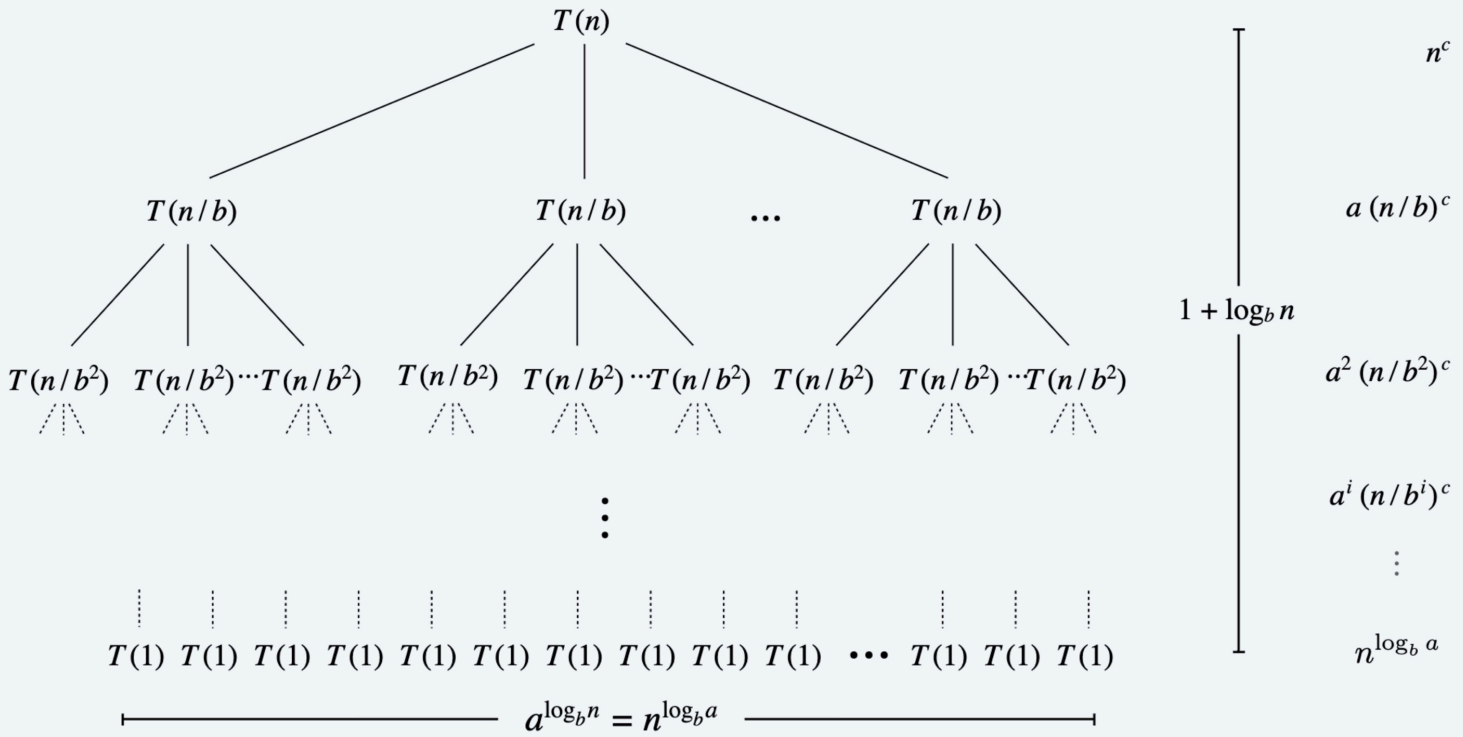
year	algorithm	bit operations
12xx	**grade school**	$O(n^2)$
1962	**Karatsuba–Ofman**	$O(n^{1.585})$
1963	**Toom–3, Toom–4**	$O(n^{1.465})$, $O(n^{1.404})$
1966	**Toom–Cook**	$O(n^{1+\varepsilon})$
1971	**Schönhage–Strassen**	$O(n \log n \cdot \log \log n)$
2007	**Fürer**	$n \log n \, 2^{O(\log^* n)}$
2019	**Harvey–van der Hoeven**	$O(n \log n)$
		$O(n)$

GNU Multiprecision Arithmetic Library (GMP) uses 7 different algorithms, choosing one based on size of integers:
Karatsuba, variants of Toom and Toom-Cook, Schonhage-Strassen (FFT-based method)

Table replicated form https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/05DivideAndConquerII.pdf

Divide-and-conquer recurrences: recursion tree

Suppose $T(n)$ satisfies $T(n) = a\,T(n/b) + n^c$ with $T(1) = 1$, for n a power of b.



$r = a / b^c$ $\qquad T(n) = n^c \sum_{i=0}^{\log_b n} r^i$

Replicated from:
https://www.cs.princeton.edu
tardos/pdf/05DivideAndConq

Divide-and-conquer recurrences: recursion tree analysis

Suppose $T(n)$ satisfies $T(n) = a\,T(n/b) + n^c$ with $T(1) = 1$, for n a power of b.

Let $r = a / b^c$. Note that $r < 1$ iff $c > \log_b a$.

$$T(n) = n^c \sum_{i=0}^{\log_b n} r^i = \begin{cases} \Theta(n^c) & \text{if } r < 1 \quad c > \log_b a \\ \Theta(n^c \log n) & \text{if } r = 1 \quad c = \log_b a \\ \Theta(n^{\log_b a}) & \text{if } r > 1 \quad c < \log_b a \end{cases}$$

← cost dominated by cost of root

← cost evenly distributed in tree

← cost dominated by cost of leaves

Geometric series.

- If $0 < r < 1$, then $1 + r + r^2 + r^3 + \ldots + r^k \le 1 / (1 - r)$.
- If $r = 1$, then $1 + r + r^2 + r^3 + \ldots + r^k = k + 1$.
- If $r > 1$, then $1 + r + r^2 + r^3 + \ldots + r^k = (r^{k+1} - 1) / (r - 1)$.

Replicated from:
https://www.cs.princeto
tardos/pdf/05DivideAnd