Design Paradigm: Divide and Conquer

- Finding Rank - Merge Sort
- Karatsuba Algorithm for Integers Multiplication
- Counting Inversions
- Finding Closest Pair in Plane

Imdad ullah Khan

Integer Multiplication

Input: A and B (n digit arrays) **Output:** $C = A * B$

$$
\begin{array}{rrrrrr}
 & & & 7 & 5 & 8 \\
 & & \times & 6 & 3 & 2 \\
\hline
 & & 1 & 5 & 1 & 6 \\
 & 2 & 2 & 7 & 4 & \\
4 & 5 & 4 & 8 & & \\
\hline
4 & 7 & 9 & 0 & 5 & 6
\end{array}
$$

Algorithm Long Multiplication algorithm

for $i = 1$ to n **do**
  $c \leftarrow 0$
  for $j = 1$ to n **do**
    $Z[i][j+i-1] \leftarrow (A[j] * B[i] + c) \bmod 10$
    $c \leftarrow (A[j] * B[i] + c)/10$
  $Z[i][i+n] \leftarrow c$
$carry \leftarrow 0$
for $i = 1$ to $2n$ **do**
  $sum \leftarrow carry$
  for $j = 1$ to n **do**
    $sum \leftarrow sum + Z[j][i]$
  $C[i] \leftarrow sum \bmod 10$
  $carry \leftarrow sum/10$
$C[2n+1] \leftarrow carry$

Runtime: $O(n^2)$ single digit arithmetic ops

Multiplying two n digits integers

Input: Two n digits numbers A and B (n-digits arrays)
Output: (integer) $C = A \times B$

Reformulate and apply distributive and associative laws

$$\left(A[0]*10^0 + A[1]*10^1 + A[2]*10^2 + \dots\right) \times \left(B[0]*10^0 + B[1]*10^1 + B[2]*10^2 + \dots\right)$$

```
C ← 0
for i = 1 to n do
    for j = 1 to n do
        C ← C + 10^(i+j) × A[i] * B[j]
```

		×		7	5	8
				6	3	2
			–	–	–	–
		–	–	–	–	
	–	–	–	–		
	4	7	9	0	5	6

Runtime: n^2 single digit multiplications + shifting (multiplying by 10^x)

Divide and Conquer based Multiplication

Compute the product xy from products of '*smaller numbers*'

Assume x and y are $2n$-digits numbers

$$x = 2758 = \begin{array}{cccc} {\scriptstyle 3} & {\scriptstyle 2} & {\scriptstyle 1} & {\scriptstyle 0} \\ \hline \multicolumn{1}{|c|}{2} & \multicolumn{1}{c|}{7} & \multicolumn{1}{c|}{5} & \multicolumn{1}{c|}{8} \\ \hline \end{array}$$

$$x = 2 \times 10^3 + 7 \times 10^2 + 5 \times 10^1 + 8 \times 10^0$$

$$x = 10^2 \times (2 \times 10 + 7) + (5 \times 10 + 8)$$

$$x = 10^2 \times 27 + 58 \implies a = 27, b = 58$$

$$x = \sum_{i=0}^{2n-1} x_i 10^i = \sum_{i=n}^{2n-1} x_i 10^i + \sum_{i=0}^{n-1} x_i 10^i = 10^n \underbrace{\sum_{i=n}^{2n-1} x_i 10^{i-n}}_{a} + \underbrace{\sum_{i=0}^{n-1} x_i 10^i}_{b}$$

Divide and Conquer based Multiplication

Input: x and y ($2n$ digits integers) **Output:** $z = x * y$

$$x = \underbrace{10^n \sum_{i=n}^{2n-1} x_i 10^{i-n}}_{a} + \underbrace{\sum_{i=0}^{n-1} x_i 10^i}_{b} \qquad y = \underbrace{10^n \sum_{i=n}^{2n-1} y_i 10^{i-n}}_{c} + \underbrace{\sum_{i=0}^{n-1} y_i 10^i}_{d}$$

Fact: $(p+q)(r+s) = pr + ps + qr + qs$

$$xy = (10^n a + b)(10^n c + d) = 10^{2n}(ac) + 10^n(ad + bc) + bd$$

- Smaller products (ac, ad, bc, bd) are recursively computed
- Multiplication by 10's and addition do not matter much

$$2758 * 3261 = 10^4(27 * 32) + 10^2(27 * 61 + 58 * 32) + 58 * 61$$

Divide and Conquer based Multiplication

Algorithm Recursive Integer Multiplication

```
function REC-MULTIPLY(x, y, 2n)                ▷ n = 2^k by zero-padding
    if n = 1 then
        return x * y
    else
        x = 10^n a + b,  y = 10^n c + d
        ac ← REC-MULTIPLY(a, c, n)
        ad ← REC-MULTIPLY(a, d, n)
        bc ← REC-MULTIPLY(b, c, n)
        bd ← REC-MULTIPLY(b, d, n)
        return 10^{2n}(ac) + 10^n(ad + bc) + bd
```

$$xy = (10^n a + b)(10^n c + d) = 10^{2n} \underbrace{(ac)}_{\text{1 multiplication}} + 10^n \underbrace{(ad + bc)}_{\text{2 multiplications}} + \underbrace{bd}_{\text{1 multiplication}}$$

$$T(2n) = \begin{cases} 1 & \text{if } n = 1 \\ 4T(n) + 6n & \text{if } n > 1 \end{cases} = O(n^2)$$

No gain

Karatsuba Multiplication Algorithm

$$xy = (10^n a + b)(10^n c + d) = 10^{2n} \underbrace{(ac)}_{\text{1 multiplication}} + 10^n \underbrace{(ad + bc)}_{\text{2 multiplications}} + \underbrace{bd}_{\text{1 multiplication}}$$

$$T(2n) = \begin{cases} 1 & \text{if } n = 1 \\ 4T(n) + 6n & \text{if } n > 1 \end{cases} = O(n^2) \qquad \text{No gain}$$

Karatsuba's Observation: Four multiplications can be reduced to three

$$\begin{aligned} \underline{ad + bc} &= (a+b)(c+d) - ac - bd \\ &= ac + \underline{ad + bc} + bd - ac - bd \end{aligned}$$

- $\underline{ad + bc}$ can be obtained with one additional multiplication

Karatsuba Multiplication

$$xy = (10^n a + b)(10^n c + d) = 10^{2n} \underbrace{(ac)}_{\text{1 multiplication}} + 10^n \underbrace{(ad + bc)}_{\text{2 multiplications}} + \underbrace{bd}_{\text{1 multiplication}}$$

$$\underline{ad + bc} = (a+b)(c+d) - ac - bd = ac + \underline{ad + bc} + bd - ac - bd$$

Algorithm Karatsuba Integer Multiplication

function KARTASUBA-MULTIPLY$(x, y, 2n)$ ▷ $n = 2^k$ by zero-padding
 if $n = 1$ **then return** $x * y$
 else $x = 10^n a + b,\ y = 10^n c + d$
  $ac \leftarrow$ KARTASUBA-MULTIPLY(a, c, n)
  $bd \leftarrow$ KARTASUBA-MULTIPLY(b, d, n)
  $mid \leftarrow$ KARTASUBA-MULTIPLY$(a + b, c + d, n)$
  return $10^{2n}(ac) + 10^n(mid - ac - bd) + bd$

$$T(2n) = \begin{cases} 1 & \text{if } n = 1 \\ 3T(n) + 6n & \text{else } n > 1 \end{cases} = O(n^{1.58})$$

Integer Multiplication

Input: x and y ($2n$ digits integers) **Output:** $z = x * y$

- Repeated Addition (adding x to itself y times) ▷ $O(10^n)$
- Long Multiplication ▷ $O(n^2)$

  Kolmogorov(1960) conjectured: grade-school algorithm is the best possible

- Karatsuba's Algorithm (1960) ▷ $O(n^{1.58})$
- Harvey & van der Hoeven (2019) ▷ $O(n \log n)$
- Can we do better ▷ Not known either way

Karatsuba Multiplication: Summary

- Long multiplication can be implemented recursively
- But runtime is $O(n^2)$ single digit multiplications
- With Karatsuba observation, runtime is reduced to $O(n^{1.58})$
- $n^{1.58} = o(n^2)$