CS207 Design and Analysis of Algorithms

Sajith Gopalan

Indian Institute of Technology Guwahati

sajith@iitg.ac.in

January 13, 2022

Divide and Conquer

Divide and Conquer

- solve a problem recursively
  - Divide the problem into a number of smaller instances of the same problem
  - For each of these subproblems, if its size is sufficiently large, then Conquer it recursively, else Conquer it directly
  - Combine the solutions to the subproblems into the solution for the original problem

Multiplication of two n digit numbers

The standard algorithm runs in $O(n^2)$ time

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

Can we do asymptotically faster?

- In 1960, Kolmogorov postulated a $\Omega(n^2)$ lower bound in a seminar
- Karatsuba, then a student, was in the audience
- Within a few days he came up with a surprising algorithm
- His algorithm runs in $O(n^{log_2 3})$ time

Karatsuba Algorithm

Input: x and y, two strings of length n in some constant base B; Assume, wlg, that $n = 2^k$ for some $k \in \mathbb{N}$; otherwise, pad with 0's on the left side.

Algorithm 1 Function Karatsuba (x,y)

if $n == 1$ then **return** $x * y$;
Let $m = n/2$;
Split x and y into halves: $x = \langle x_1, x_0 \rangle$ and $y = \langle y_1, y_0 \rangle$ so that $x = x_1 B^m + x_0$ and $y = y_1 B^m + y_0$;
$z_2 =$ Karatsuba(x_1,y_1); $z_0 =$ Karatsuba(x_0,y_0);
$s_1 = [x_0 - x_1 \geq 0]$; $s_2 = [y_1 - y_0 \geq 0]$; $s = \neg(s_1 \oplus s_2)$;
if s **then**
    $z_1 = z_2 + z_0 +$ Karatsuba$(|x_0 - x_1|, |y_1 - y_0|)$;
else
    $z_1 = z_2 + z_0 -$ Karatsuba$(|x_0 - x_1|, |y_1 - y_0|)$;
end if
return $z_2.B^{2m} + z_1.B^m + z_0$

Correctness

- $z_2 = x_1 * y_1$
- $z_0 = x_0 * y_0$
- As can be seen from the code, $z_1 = (x_0 - x_1) * (y_1 - y_0) + z_2 + z_0$
- i.e., $z_1 = x_0 y_1 + x_1 y_0 - x_0 y_0 - x_1 y_1 + z_2 + z_0 = x_0 y_1 + x_1 y_0$
- The return value is $z_2.B^{2m} + z_1.B^m + z_0$
  $= x_1 y_1.B^{2m} + (x_0 y_1 + x_1 y_0).B^m + x_0 y_0$
  $= (x_1.B^m + x_0) * (y_1.B^m + y_0)$
  $= x * y$

Analysis

- At each level of recursion, there are 3 recursive calls with strings of length $m = n/2$
- There are $O(m)$ additions to perform
- Multiplication by the base B requires only a shift
- Multiplications by B^{2m} and B^m require $O(m)$ shifts
- That is the work outside of the recursive calls amount to $O(m)$
- The recurrence relation for the time complexity function is, therefore, $T(n) = 3T(n/2) + cn$

Analysis

- $T(n) = 3T(\frac{n}{2}) + cn$
  $= 3[3T(\frac{n}{2^2}) + c\frac{n}{2}] + cn = 3^2T(\frac{n}{2^2}) + cn[\frac{3}{2} + 1]$
  $= 3^3T(\frac{n}{2^3}) + cn[(\frac{3}{2})^2 + \frac{3}{2} + 1]$
  $= 3^kT(\frac{n}{2^k}) + cn[(\frac{3}{2})^{k-1} + \ldots + 1]$
  $= 3^{\log n} + 2c.[n^{\log 3} - n]$
  $= O(n^{log 3})$
- All logarithms are of base 2

Examples

- $56 * 12 = 5 * 1 * 100 + ((6 - 5) * (1 - 2) + 5 * 1 + 6 * 2) * 10 + 6 * 2 = 500 + 160 + 12 = 672$
- $78 * 34 = 7 * 3 * 100 + ((8 - 7) * (3 - 4) + 7 * 3 + 8 * 4) * 10 + 8 * 4 = 2100 + 520 + 32 = 2652$
- $22 * 22 = 2 * 2 * 100 + ((2 - 2) * (2 - 2) + 2 * 2 + 2 * 2) * 10 + 2 * 2 = 400 + 80 + 4 = 484$
- $5678 * 1234 = 56 * 12 * 10000 + ((78 - 56) * (12 - 34) + 56 * 12 + 78 * 34) * 100 + 78 * 34 = 6720000 + 284000 + 2652 = 7006652$