CS 218 : Design and Analysis of Algorithms

Lecture 7: Integer Multiplication

Lecturer: *Sundar Vishwanathan*
Computer Science & Engineering Indian Institute of Technology, Bombay

1 Integer Multiplication

1.1 Problem Statement

We wish to multiply two n-bit numbers x and y. The obvious method is to find partial products of x with each bit of y and add them. This algorithm takes $O(n^2)$ time since there are n additions and each takes n time. We will use the divide and conquer paradigm to find a better solution.

First divide x and y into 2 equal parts each, A and B, and, C and D respectively, such that each one is $n/2$ bits long. In other words $x = A.2^{n/2} + B$ and $y = C.2^{n/2} + D$ Assume for simplicity that n is a power of 2. Then,

$$xy = 2^n AC + 2^{\frac{n}{2}} AD + 2^{\frac{n}{2}} BC + BD \tag{1}$$

We get the following reccurence:

$$T(n) = 4T(n/2) + O(n) \tag{2}$$

The solution is $T(n) = O(n^2)$ which is the same as what we had earlier. It is hence indeed surprising that we can do better. This part requires problem dependent fundaes and is something you cultivate by solving problems. In the above method we perform 4 multiplications (recursively) and 3 addition operations. Addition is $O(n)$ and the multiplications are costlier. The question you ask yourself is can the number of recursive calls be reduced? The surprising answer is yes. See if you can figure out how to do it using three recursive calls instead of four before you read ahead.

$$xy = 2^n AC + 2^{n/2}((A + B).(C + D) - AC - BD) + BD \tag{3}$$

Here we perform 3 multiplications and 6 additions (which is $O(n)$) We get the following recurrence

$$T(n) = 3T(n/2) + O(n). \tag{4}$$

The solution to this recurrence is $O(n^{log_2 3})$ which is better than $O(n^2)$