Design & Analysis of Algorithms

[Divide & Conquer]

Large Integers Multiplication
Strassen's Matrix Multiplication

Soharab Hossain Shaikh
***BML Munjal* University**

1

Multiplication of Large Integers

- Multiplication of two arbitrary length integers may not take constant amount of time.
- Remember that integer numbers are represented in the computer in binary form. For example,

$$(2913774253)_{10} = (10101101101011001010101010101101)_2$$

- Obviously a normal multiplication procedure, for two n-bit integers u and v, multiplies each bit of u with each other bit of v. This takes $\Theta(n^2)$ digit multiplications to compute the product of u and v.
- Using divide and conquer technique, we can reduce this bound of $\Theta(n^2)$ drastically.

2

Multiplication of Large Integers (cont…)

- Let u and v be two n-bit numbers and n is a power of 2.
- Each integer is divided into two parts of size $n/2$ as follows:

$$u = w2^{n/2} + x$$

w	**x**

$$v = y2^{n/2} + z$$

y	**z**

- The product of u and v is:

$$uv = (w2^{n/2} + x)(y2^{n/2} + z) = wy2^n + (wz + xy)2^{n/2} + xz$$

3

Multiplication of Large Integers (cont…)

- **Note:** Multiplying a binary number by 2^n is nothing but shifting n times towards left.
- **Example:** Consider the binary number 11 which is equivalent to its decimal number 3.

$$(11)2^3 = (3)2^3 = 24 = 11000$$

Thus, we only need to shift 11 three times towards left to get the number.

- Shifting a number n times towards left can be done in $\Theta(n)$ time. This means that multiplying a number with 2^n can be done in $\Theta(n)$.
- Apart from multiplications by 2^n, the product

$$uv = (w2^{n/2} + x)(y2^{n/2} + z) = wy2^n + (wz + xy)2^{n/2} + xz$$

contains 4 multiplications and three additions of pair of $n/2$-digit numbers.

4

Multiplication of Large Integers (cont…)

- Thus, if T(n) is the number of multiplications and addition, the total cost leads to the following recurrence relation:

$$T(n) = \begin{cases} d & , \text{if } n = 1 \\ 4T(n/2) + bn, & \text{if } n > 1. \end{cases}$$

for some constants b and $d > 0$.

- The solution to this recurrence is $\Theta(n^2)$.
- The time complexity of $\Theta(n^2)$ can be improved to a lower order complexity.

5

Multiplication of Large Integers (cont…)

- Consider the computation of $wz + xy$ using the following identity:

$$wz + xy = (w + x)(y + z) - wy - xz$$

- This leads to:

$$\begin{aligned} uv &= wy2^n + (wz + xy)2^{n/2} + xz \\ &= wy2^n + ((w + x)(y + z) - wy - xz)2^{n/2} + xz \end{aligned}$$

- The expressions wy and xz will be computed only once.
- Multiplying u and v reduces to three multiplications of integers of size $n/2$ and six additions and subtractions.
- The additions and subtractions cost $\Theta(n)$ time.

6

Multiplication of Large Integers (cont…)

- This method yields the following recurrence:

$$T(n)=\begin{cases} d & , \text{if } n=1 \\ 3T(n/2)+bn, & \text{if } n>1. \end{cases}$$

for some appropriately chosen constants b and $d > 0$.

- This results in

$$T(n)=\Theta\left(n^{\log 3}\right)=O\left(n^{1.59}\right)$$

- **Remark:** This is a significant improvement over the traditional method.

7

Matrix Multiplication

- We show how to apply the divide and conquer strategy to this problem to obtain an efficient algorithm
- Let A and B be two $n \times n$ matrices. We wish to compute their product C=AB.
- **Traditional Algorithm:** Traditionally we can compute each element of C as follows:

$$C(i,j)=\sum_{k=1}^{n} A(i,k)B(k,j).$$

Since C is also an n x n matrix, computing the whole matrix C involves n^3 multiplications. This results in $\Theta(n^3)$ time complexity.

8

Matrix Multiplication (cont..)

- **Divide and Conquer (recursive) Algorithm**
  - For matrix multiplication, we shall divide the problem into smaller sub problems of multiplication.
  - Assume that $n = 2^k, k > 0$. If $n \geq 2$, then A, B and C can be partitioned into four matrices of dimensions $n/2 \times n/2$ each:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}, C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.$$

Note that the matrices a_{ij}, b_{ij} and c_{ij}, $i = 1, 2$, $j = 1, 2$, are of dimensions $n/2 \times n/2$ each.

9

Matrix Multiplication (cont..)

- The divide and conquer solution carries out the matrix multiplication as follows

$$C = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

- This requires 8 multiplications and 4 additions of $n/2 \times n/2$ matrices.
- In order to count the number of scalar operations, let a and m denote the cost of scalar addition and multiplication, respectively.
- If $n = 1$, the total cost is just m since we have only one scalar multiplication.
- The total cost of multiplying two $n \times n$ matrices is governed by the recurrence:

$$T(n) = \begin{cases} m & , \quad if \;\; n = 1 \\ 8T(n/2) + 4(n/2)^2 a, & if \;\; n \geq 2. \end{cases}$$

10

Matrix Multiplication (cont..)

- This recurrence can be simplified as follows:

$$T(n) = \begin{cases} m & , \quad if \ n = 1 \\ 8T(n/2) + an^2, & if \ n \geq 2. \end{cases}$$

- Solving this recurrence relation, using the Master Theorem, we conclude that $T(n) = \Theta(n^3)$.

- This algorithm has the same time complexity as that of the traditional algorithm and therefore, does not result in a more efficient algorithm.

- In fact, it costs more than the traditional method due to the overhead brought by recursion.

11

Strassen's Algorithm

- Let A and B be two $n \times n$ matrices, where $n = 2^k, k \geq 0$. If $n \geq 2$, then A, B can be partitioned into four matrices of dimensions $n/2 \times n/2$ each:

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix},$$

- Multiplying A and B results in another n-dimensional matrix $C = AB$.

- We can express matrix C as follows:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix},$$

  where each of c_{ij}, $i = 1,2, j = 1,2,$ is an $n/2 \times n/2$ matrix.

- Note that all the matrices a_{ij}, b_{ij} and c_{ij}, $i = 1,2,\ j = 1,2,$ are of dimension $n/2 \times n/2$ each.

12

Strassen's Algorithm (cont..)

- We may represent C as follows:

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}.$$

- A common method for getting matrix C is by evaluation of the following matrix :

$$C = \begin{pmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{pmatrix}$$

- In this method of calculating C there are 8 multiplications and 4 additions of matrices of order $n/2 \times n/2$ each.

- By Strassen's algorithm, we can get C in 7 multiplications and 18 additions and subtractions of $n/2 \times n/2$ matrices.

13

Strassen's Algorithm (cont..)

- Let

$$d_1 = (a_{11} + a_{22})(b_{11} + b_{22})$$
$$d_2 = (a_{21} + a_{22})b_{11}$$
$$d_3 = a_{11}(b_{12} - b_{22})$$
$$d_4 = a_{22}(b_{21} - b_{11})$$
$$d_5 = (a_{11} + a_{12})b_{22}$$
$$d_6 = (a_{21} - a_{11})(b_{11} + b_{12})$$
$$d_7 = (a_{12} - a_{22})(b_{21} + b_{22})$$

- Strassen proposed that

$$C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

can be computed as

$$C = \begin{pmatrix} d_1 + d_4 - d_5 + d_7 & d_3 + d_5 \\ d_2 + d_4 & d_1 + d_3 - d_2 + d_6 \end{pmatrix}.$$

- The strassen's formulation has 7 multiplications and 18 additions and subtractions of matrices of order $n/2 \times n/2$ each.

14

Strassen's Algorithm (cont..)

- Let m denote the cost of a single scalar multiplication, and a the cost of a single scalar addition or subtraction.
- The recurrence for the running time is as follows:

$$T(n) = \begin{cases} m & , \quad if \; n = 1 \\ 7T(n/2) + 18(n/2)^2 a, & if \; n \geq 2. \end{cases}$$

This can be simplified to:

$$T(n) = \begin{cases} m & , \quad if \; n = 1 \\ 7T(n/2) + (9a/2)n^2, & if \; n \geq 2. \end{cases}$$

- The solution to this recurrence is $\Theta(n^{\log 7})$.

15

Strassen's Algorithm (cont..)

- The exact solution to a recurrence of the form:

$$f(n) = \begin{cases} d & , \; if \; n = 1 \\ aT(n/c) + bn^x, & if \; n \geq 2, \end{cases}$$

where a and c are nonnegative integers, b, d and x be nonnegative constants, and $n = c^k$, for some nonnegative integer k is given by:

$$f(n) = \begin{cases} bn^x \log_c n + dn^x, & if \; a = c^x \\ f(n) = \left(d + \dfrac{bc^x}{a - c^x} \right) n^{\log_c x} - \left(\dfrac{bc^x}{a - c^x} \right) n^x, & if \; a \neq c^x \end{cases}$$

- Therefore, the exact solution to the Strassen's recurrence is as follows:

$$T(n) = \left(m + \frac{(9a/2)2^2}{7 - 2^2} \right) n^{\log 7} - \left(\frac{(9a/2)2^2}{7 - 2^2} \right) n^2.$$

16

Strassen's Algorithm (cont..)

- Further simplification yields:

$$T(n) = \left(m + \frac{(9a/2)2^2}{7-2^2} \right) n^{\log 7} - \left(\frac{(9a/2)2^2}{7-2^2} \right) n^2$$

$$= mn^{\log 7} + 6an^{\log 7} - 6an^2$$

- So the running time is

$$\Theta(n^{\log 7}) = O(n^{2.81}),$$

which is a good improvement over $\Theta(n^3)$ time complexity of conventional matrix multiplication.

17

Comparison of Algorithms

- A comparative study for the three algorithms of matrix multiplication.
  - Traditional Algorithm
  - Recursive Algorithm
  - Strassen's Algorithm.

- The comparison table:

	Multiplications	Additions	Complexity
Traditional Algorithm	n^3	$n^3 - n^2$	$\Theta(n^3)$
Recursive version	n^3	$n^3 - n^2$	$\Theta(n^3)$
Strassen's Algorithm	$n^{\log 7}$	$6\, n^{\log 7} - 6n^2$	$\Theta(n^{\log 7})$

- **Remark:** Since, $\Theta(n^{\log 7}) = O(n^{2.81}) = o(n^3)$, the Strassen's algorithm has a remarkable improvement over the traditional algorithm.

18

Comparison of Algorithms

- The comparison table for some values of n:

	n	Multiplications	Additions
Traditional Algorithm	100	1,000,000	990,000
Strassen's Algorithm	100	411,822	2,470,334
Traditional Algorithm	1000	1,000,000,000	999,000,000
Strassen's Algorithm	1000	264,280,285	1,579,681,709
Traditional Algorithm	10,000	10^{12}	9.99×10^{11}
Strassen's Algorithm	10,000	0.169×10^{12}	10^{12}

Table: Comparison between Strassen's and Traditional Algorithms.

- **Remark:** The table provides a very clear picture in numeric form. The difference of Strassen's algorithm over the traditional algorithm is clearly visible.

19

End of Lecture

20