CS2023 - Data Structures and Algorithms

Take Home Assignment

Week 4

March 23, 2023

By: Sajeev Kugarajah (210554M) – Group IV

Karatsuba algorithm to multiply two largest integers.

Unlike traditional multiplication algorithm, Karatsuba algorithm uses divide and conquer method which reduces the time complexity by a significant amount which makes Karatsuba algorithm more efficient when multiplying larger integers.

For example, here's an implementation of Karatsuba algorithm for 1234*5678 in base 10

$$a_1 = 12 \times 56$$

$$d_1 = 34 \times 78$$

$$e_1 = (12 + 34)(56 + 78) - a_1 - d_1$$

$$now\ 1234 \times 5678 = a_1 10^4 + e_1 10^2 + d_1$$

$$we\ have\ to\ simplify\ a_1, e_1\ and\ d_1\ using\ karatsuba\ algorithm\ again$$

$$a_1 = 12 \times 56$$

$$a_2 = 1 \times 5 = 5$$

$$d_2 = 2 \times 6 = 12$$

$$e_2 = (1 + 2)(5 + 6) - 5 - 12 = 33 - 17 = 16$$

$$a_1 = 12 \times 56 = 5.10^2 + 16.10^1 + 12 = 672$$

$$like\ this\ we\ can\ solve\ that, \qquad d_1 = 2652\ \ and\ \ e_1 = 2840$$

$$so, 1234 \times 5678 = 672.10^4 + 2840.10^2 + 2652 = 7006652$$

Recurrence relation for time complexity

$let\ the\ time\ complexity\ is\ T_{(n)}\ for\ \ two\ n-bit\ numbers$

$since\ this\ algorithm\ is\ dividing\ the\ input\ n-bit\ number\ with\ two\ \frac{n}{2}\ bit\ numbers$

$and\ then\ multiply\ them\ recursively,$

$and\ then\ do\ three\ additional\ multiplications\ over\ \frac{n}{2}\ digits$

$then\ do\ some\ additions\ and\ subtractions\ with\ constant\ time \rightarrow\ O(n)$

$therefore, the\ time\ complexity\ T_{(n)}\ can\ be\ expressed\ as$

$$T_{(n)} = 3.T_{\left(\frac{n}{2}\right)} + O(n)$$

Solving the recurrence relation using master theorem

$master\ theorem\ is\ applicable\ to\ the\ recurrence\ relations\ with\ the\ form\ of$

$$T_{(n)} = a.T_{\left(\frac{n}{b}\right)} + f_{(n)}$$

$where\ a \geq 1, b > 1\ and\ f_{(n)}\ is\ asymptotically\ positive$

$in\ this\ algorithm, a = 3, b = 2\ and\ f_{(n)} = O(n)$

$therefore\ O(n) = O(n^{log_2 3 - 0.585})$

$so, we\ can\ solve\ the\ relation\ using\ the\ first\ case\ of\ master\ theorem\ and,$

$$T(n) = O(n^{log_2 3}) = O(n^{1.585})$$

From this, we can conclude that, Karatsuba algorithm with $O(n^{1.585})$ time complexity is way efficient than the naïve multiplication algorithm with $O(n^2)$ time complexity for multiplying larger integers.

Appendix: Source code for implemented Karatsuba algorithm in C++

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

int karatsuba(int x, int y)
{
    // if x or y is less than 10, then return the product
    if (x < 10 || y < 10)
        return x * y;

    // find the maximum length of x and y
    int n = max(to_string(x).length(), to_string(y).length());

    // find the middle of the number
    int m = n / 2;

    // split the number into two parts
    int a = x / pow(10, m);
    int b = x % (int)pow(10, m);
    int c = y / pow(10, m);
    int d = y % (int)pow(10, m);

    // calculate the product of a and c
    int ac = karatsuba(a, c);

    // calculate the product of b and d
    int bd = karatsuba(b, d);

    // calculate the product of (a+b) and (c+d)
    int adbc = karatsuba(a + b, c + d) - ac - bd;

    // return the product
    return ac * pow(10, 2 * m) + (adbc * pow(10, m)) + bd;
}
int main()
{
    int x, y;
    cout << "Enter two numbers: ";
    cin >> x >> y;
    cout << "Product: " << karatsuba(x, y);
    return 0;
}
```