Exercise 1

Alireza Jafartash*[1]

[1]Advanced Algorithms - University of Tehran

September 2024

1 Introduction

In this exercise, I explored various algorithms for both integer and matrix multiplication. For integer multiplication, I analyzed two approaches: the naive algorithm and the more efficient Karatsuba algorithm. Additionally, I investigated naive and Strassen algorithms for matrix multiplication. All algorithms have been implemented and thoroughly tested in Python, demonstrating their performance and effectiveness.

Codes are available on github.

*Email: `jtash@ut.ac.ir`

2 Integer Multiplication

2.1 Algorithms

2.1.1 Naive Algorithm

It iterates over each digit of the multiplier, which leads to a time complexity of $O(n^2)$ where n is the number of digits. There is no explicit use of data structures in this algorithm; therefore, the space complexity is $O(1)$.

Pseudo code:

Algorithm 1 Naive Integer Multiplication

```
function MULTIPLY(a, b)
    result ← 0
    while b >0 do
        result ← result + a
        b ← b - 1
    end while
    return result
end function
```

Complexity:
Time Complexity: $O(n^2)$
Space Complexity: $O(1)$

2.1.2 Karatsuba Algorithm

The Karatsuba algorithm is an efficient method for multiplying large integers by breaking them into smaller parts and combining the results, speeding up the multiplication process.

Pseudo code:

Algorithm 2 Karatsuba Multiplication

```
 1: function KARATSUBA(x, y)
 2:     if $x < 10$ or $y < 10$ then
 3:         return $x * y$
 4:     end if
 5:     n ← max(size of x, size of y)
 6:     m ← n // 2
 7:     $x_1, x_{\leftarrow}$ split($x$ at $m$)
 8:     $y_1, y_0 \leftarrow$ split($y$ at $m$)
 9:     $z_0 \leftarrow$ KARATSUBA($x0, y0$)
10:     $z_1 \leftarrow$ KARATSUBA($(x_1 + x_0), (y_1 + y_0)$)
11:     $z_2 \leftarrow$ KARATSUBA($x_1, y_1$)
12:     return $z_2 \cdot 10^{2m} + (z_1 - z_2 - z_0) \cdot 10^m + z_0$
13: end function
```

Complexity:
Time Complexity: $O(n^{\log_2 3}) \approx O(n^{1.585})$
Space Complexity: $O(n)$

2.2 Comparison

Karatsuba algorithm is more efficient than the naive algorithm both in theory and in practice.

The line chart below demonstrate the elapsed time in multiplying pairs of numbers. The red line shows the karatsuba algorithm and the blue line shows the naive algorithm.

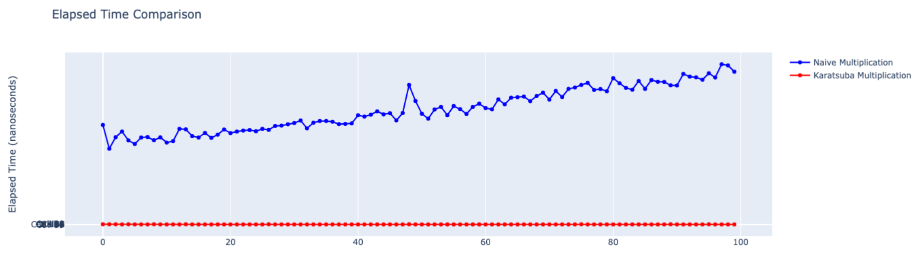

Figure 1: Comparison of Integer Multiplication Algorithms

3 Matrix Multiplication

3.1 Algorithms

3.1.1 Naive Matrix Multiplication

The Strassen algorithm is an efficient method for matrix multiplication that reduces the computational complexity by dividing matrices into smaller submatrices. It performs fewer multiplications than traditional methods, allowing for faster multiplication of large matrices, making it particularly useful in computational applications.

Pseudo code:

Algorithm 3 Naive Matrix Multiplication

```
1: function MATRIX_MULTIPLY(A, B)
2:     for i = 1  to  n do
3:         for j = 1  to  n do C[i][j] ← 0
4:             for k = 1  to  n do C[i][j] ← C[i][j] + A[i][k] * B[k][j]
5:             end for
6:         end for
7:     end for
8:     return C
9: end function
```

Complexity:

Time Complexity: $O(n^3)$
Space Complexity: $O(n^2)$

3.1.2 Strassen Algorithm

The Strassen algorithm is an optimized method for matrix multiplication that reduces the number of multiplications required compared to the naive approach. By dividing matrices into smaller submatrices and using a set of recursive formulas, it significantly improves computational efficiency for large matrices.

Pseudo code:

Algorithm 4 Strassen Matrix Multiplication

```
function STRASSEN(A, B)
    if n == 1 then
        return A * B
    else
        divide A and B into submatrices: A11, A12, A21, A22, B11, B12, B21, B22
        $M1 \leftarrow$ STRASSEN(A11 + A22, B11 + B22)
        $M2 \leftarrow$ STRASSEN(A21 + A22, B11)
        $M3 \leftarrow$ STRASSEN(A11, B12 - B22)
        $M4 \leftarrow$ STRASSEN(A22, B21 - B11)
        $M5 \leftarrow$ STRASSEN(A11 + A12, B22)
        $M6 \leftarrow$ STRASSEN(A21 - A11, B11 + B12)
        $M7 \leftarrow$ STRASSEN(A12 - A22, B21 + B22)
        return COMBINE(M1, M2, M3, M4, M5, M6, M7)
    end if
end function
```

Complexity:

Time Complexity: $O(n^{\log_2 7}) \approx O(n^{2.81})$
Space Complexity: $O(n^2)$

3.2 Comparison

Strassen algorithm is better theorically but in practice it's not good even for matrices with size about 100. As I tested programmatically the reason is the recursion overhead. I used libraries that behind the scene use compiled C codes for efficiency but even that was not enough.

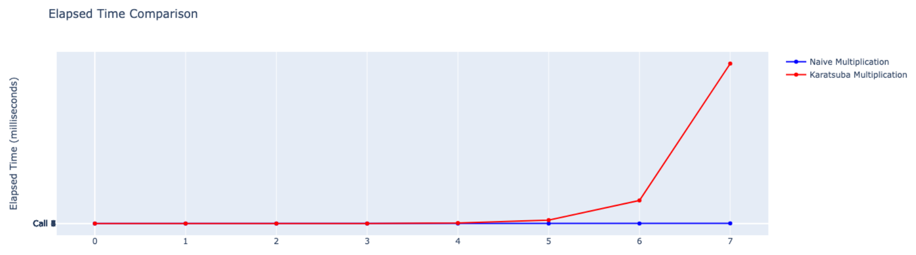

Figure 2: Comparison of Matrix Multiplication Algorithms

4 About Me

I earned my bachelor's degree in Computer Engineering, driven by a long-standing curiosity for mathematics that began in high school, where I participated in math olympiads. During my time at university, I explored a wide range of topics in technology and science, ultimately gravitating towards artificial intelligence, with a particular emphasis on reinforcement learning. I engaged in research and studied several papers on machine learning and RL.
To enhance my skills as a software engineer, I gained practical experience by working at various companies and startups. My goal was to deepen my coding expertise while addressing real-world challenges. Moving forward, I am committed to advancing my knowledge in artificial intelligence.