|  |  |  |  |
| --- | --- | --- | --- |
| **Academic Year** | **2025 - 26** | **Experiment No.** | **3** |
| **Course & Semester** | **S.E. – Sem. III** | **Subject Name** | **Analysis of Algorithm** |
| **Experiment Type** | **Software Performance** | **Subject Code** | **25PCC12CS05** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of Student:** |  | **Roll No.:** |  |
| **Date of Performance:** |  | **Date of Submission:** |  |
| **LO Mapping** | 25PCC12CS05.1: Analyze the time and space complexity of algorithms.  25PCC12CS05.2: Apply divide and conquer strategy to solve a problem. | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Indicator** | **Poor** | **Average** | **Good** | | Timeline Maintains submission deadline (3) | Submission not done (0) | One or More than One week late (1-2) | Maintains deadline (3) | | Completion and Organization (3) | N/A | Document is just acceptable (1-2) | Completed whole document and neatly organized (3) | | Program Performance (2) | Could not perform at all (0) | Implemented few parts (1) | Full implementation (2) | | Knowledge In depth knowledge of the Experiment (2) | Unable to answer questions (0) | Unable to answer few questions (1) | Able to answer all questions (2) | |
| **Assessment Marks:**   |  |  | | --- | --- | | Timeline |  | | Completion and Organization |  | | Program Performance |  | | Knowledge |  | |
| Total: (Out of 10) |
| Teacher’s Sign: Student Sign: |

**Experiment No. 3**

**AIM:** To Implement and Analyze time and space complexity of multiplying long Integers using divide and conquer strategy.

**THEORY:** The aim of this experiment is to implement and analyze the time and space complexity of multiplying large integers using the Divide and Conquer strategy, specifically with the Karatsuba algorithm.

In traditional multiplication, multiplying two n-digit numbers takes O(n²) time. The Karatsuba algorithm optimizes this by dividing the numbers into smaller parts, reducing the number of multiplications needed from 4 to 3. This is achieved by recursively splitting the numbers and computing the product of their parts using the formula:

Result=ac⋅102m+(a+b)⋅(c+d)−ac−bd⋅10m+bd\text{Result} = ac \cdot 10^{2m} + (a+b) \cdot (c+d) - ac - bd \cdot 10^m + bdResult=ac⋅102m+(a+b)⋅(c+d)−ac−bd⋅10m+bd

This reduces the time complexity to O(n^1.585), a significant improvement over the naive method. The space complexity is O(n), mainly due to recursion and intermediate results.

The Karatsuba algorithm is efficient for large number multiplication and is widely used in applications like cryptography, where large integers are common.

### ****ALGORITHM****:

**1. Start**

* Begin the program.

**2. Define Functions:**

* **num\_len(double n):**
  + This function computes the number of digits (length) in a given number n.
* **karatsuba(double x, double y):**
  + This is the recursive function that performs Karatsuba multiplication.

**3. Input the Numbers:**

* **Read two numbers** n1 and n2 (floating-point numbers).

**4. Base Case for Karatsuba:**

* If both numbers are less than 10 (i.e., single-digit numbers):
  + Return the product of the two numbers directly: x \* y.

**5. Divide the Numbers into Two Parts:**

* Find the length (number of digits) of both x and y using num\_len(x) and num\_len(y).
* Determine the maximum number of digits between the two numbers: n = max(n1, n2).
* Calculate n\_half as the floor value of half of n (i.e., n / 2).

**6. Split the Numbers:**

* Split both x and y into two parts:
  + a is the most significant part of x.
  + b is the least significant part of x.
  + c is the most significant part of y.
  + d is the least significant part of y.

**7. Recursive Steps:**

* Recursively calculate three products:
  + ac = karatsuba(a, c) — the product of the most significant parts.
  + bd = karatsuba(b, d) — the product of the least significant parts.
  + ad\_plus\_bc = karatsuba(a + b, c + d) - ac - bd — the cross-product of the two parts.

**8. Combine the Results:**

* The final result is computed as:
  + result = ac \* 10^(2 \* n\_half) + ad\_plus\_bc \* 10^n\_half + bd

**9. Return the Result:**

* The result is returned by the recursive function.

**10. Display the Result:**

* Print the final result of the multiplication.

**11. End**

* The program ends.

**CODE:**

#include <stdio.h>

#include <math.h>

// Function to calculate number of digits in a number

int num\_len(double n) {

    int len = 0;

    n = fabs(n); // Handle negative numbers

    if (n < 1) return 1;

    while (n >= 1) {

        n /= 10;

        len++;

    }

    return len;

}

// Recursive Karatsuba function

double karatsuba(double x, double y) {

    // Base case

    if (x < 10 || y < 10)

        return x \* y;

    // Find length of both numbers

    int len\_x = num\_len(x);

    int len\_y = num\_len(y);

    int n = (len\_x > len\_y) ? len\_x : len\_y;

    // Split length into half

    int n\_half = n / 2;

    // Power of 10 for splitting

    double power = pow(10, n\_half);

    // Split numbers into parts

    double a = floor(x / power);

    double b = fmod(x, power);

    double c = floor(y / power);

    double d = fmod(y, power);

    // Recursive steps

    double ac = karatsuba(a, c);

    double bd = karatsuba(b, d);

    double ad\_plus\_bc = karatsuba(a + b, c + d) - ac - bd;

    // Combine results

    return ac \* pow(10, 2 \* n\_half) + ad\_plus\_bc \* pow(10, n\_half) + bd;

}

int main() {

    double n1, n2, result;

    // Input numbers

    printf("Enter first number: ");

    scanf("%lf", &n1);

    printf("Enter second number: ");

    scanf("%lf", &n2);

    // Perform Karatsuba multiplication

    result = karatsuba(n1, n2);

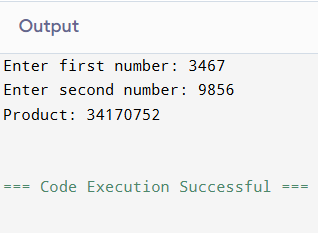
    // Display the result

    printf("Product: %.0lf\n", result);

    return 0;

}

**OUTPUT:**

****

**POST LAB QUESTIONS**

1. How does divide-and-conquer improve performance over the traditional method?

Ans. Divide-and-conquer works by:

* Breaks a large problem into smaller subproblems.
* Reduces the number of multiplications by reusing intermediate results.
* Exploits recursion for faster computation compared to straightforward repeated multiplications.

2. How does the algorithm handle numbers of unequal length?

* Calculates digit lengths of both numbers.
* Uses the **maximum length** as the split size.
* Pads the shorter number implicitly with zeros during splitting to maintain balanced recursion.

3. Compare the time complexity of Karatsuba with the naive approach for inputs of size n = 1024.

**Ans.** Here is the Time complexity comparison (n = 1024):

|  |  |  |
| --- | --- | --- |
| **Approach** | **Time Complexity** | **Approx. Multiplications for n=1024** |
| Naive (Grade-school) | O(n2)O(n^2) | 10242=1,048,5761024^2 = 1,048,576 |
| Karatsuba | O(nlog⁡23)≈O(n1.585)O(n^{\log\_2 3}) ≈ O(n^{1.585}) | ~ **59,700** |

**Result:** Karatsuba performs significantly fewer multiplications, making it much faster for large input sizes.

**CONCLUSION:**

The Karatsuba algorithm leverages the divide-and-conquer strategy to reduce computational complexity from O(n2) to approximately O(n1.585). By efficiently handling numbers of unequal lengths and minimizing the number of multiplications, it offers substantial performance gains for large inputs compared to the traditional multiplication approach.