INSTITUTO POLITÉCNICO NACIONAL

"ESCOM" (ESCUELA SUPERIOR DE CÓMPUTO)

DOCENTE: CECILIA ALBORTANTE MORATO

INTEGRANTES:

MONTEALEGRE ROSALES DAVID URIEL

VÁZQUEZ BLANCAS CÉSAR SAID

U.A.: ANÁLISIS Y DISEÑO DE ALGORITMOS

GRUPO: 3CM6

PRÁCTICA 4: Divide y vencerás

(Multiplicación de enteros largos usando el truco de Gauss).

Objetivo de la práctica.

Implementar en el lenguaje de programación de su preferencia un algoritmo el cuál implemente el algoritmo “divide y vencerás” para la resolución de multiplicaciones de enteros largos.

Introducción.

La multiplicación de enteros largos es una operación fundamental en matemáticas y en ciencias de la computación. A medida que lidiamos con números cada vez más grandes, la multiplicación puede volverse una tarea intensiva en términos de tiempo y recursos computacionales. Sin embargo, a lo largo de la historia, matemáticos brillantes han ideado métodos ingeniosos para acelerar este proceso. Uno de estos métodos es el "truco de Gauss", que lleva el nombre del eminente matemático Carl Friedrich Gauss.

Gauss una vez notó que, aunque el producto de dos números complejos

$$(a + bi)(c + di) = ac - bd + (bc + ad)i$$

Parece implicar cuatro multiplicaciones de números reales, de hecho, se puede hacer con solo tres: ac, bd y (a + b) (c + d), ya que:

$$bc + ad = (a + b)(c + d) - ac - bd.$$

Con este pequeño cambio podemos lograr bajar la complejidad de nuestro algoritmo.

Carl Friedrich Gauss.

Desarrollo de la práctica.

En esta práctica se implementó un algoritmo en **“Python”**, el cuál se diseñó con el propósito de resolver multiplicaciones de dígitos grandes con una velocidad lo más pequeña posible, implementando el método de Gauss.

El código realizado es el siguiente:

```
def enteroslargos(x, y):
    if x < 10 or y < 10:  # Caso base: Si los números son pequeños, simplemente multiplicarlos
        return x * y
    n = max(len(str(x)), len(str(y))) #calcula el numero de digitos y el mayor de los 2 numeros
    n2 = n // 2 #divide entre 2 el maximo numero



    xi = x // 10 ** n2 #divide entre la potencia de los tamaños de los numeros y el resultado lo pone, ejemplo si es 1234 pondria 12
    xd = x % (10 ** n2)#hace lo mismo pero saca el residuo por lo que quedaria 34
    yi = y // 10 ** n2
    yd = y % (10 ** n2)


    # parte recursiva
    xiyi = enteroslargos(xi, yi)
    xdyd = enteroslargos(xd, yd)
    aux = enteroslargos(xi + xd, yi + yd) - xiyi - xdyd


    # combinacion
    return (xiyi * (10 ** (2 * n2))) + (aux * (10 ** n2)) + xdyd


x = int(input("Escribe el primer numero: "))
y =int(input("Escribe el segundo numero: "))
z = enteroslargos(x, y)
print("El resultado de la multiplicación es:", z)
```

- El código comienza tomando dos números enteros largos, x e y, como entrada del usuario.
- Luego, verifica si x o y son menores que 10. Si alguno de los números es pequeño, se realiza una multiplicación simple y se retorna el resultado. Este es el caso base de la recursión.
- Si los números son más grandes, se procede a descomponerlos en dos partes, una parte izquierda (xi, yi) y una parte derecha (xd, yd). Esta división se realiza de manera que ambas partes tengan aproximadamente la mitad del número de dígitos. Para hacerlo, el código calcula n, que es el número máximo de dígitos entre x e y, y luego divide n entre 2 para obtener n2.
- Luego, se divide x y y en dos partes, xi y xd para x, y yi e yd para y, utilizando la operación de división y módulo con potencias de 10.
- La parte principal del algoritmo es la recursión. El código llama a la función enteroslargos de manera recursiva para multiplicar xi por yi y xd por yd. Luego, calcula una variable aux que representa la multiplicación cruzada de las partes izquierdas e izquierdas y las partes derechas e izquierdas. Resta xiyi y xdyd de aux.
- Finalmente, se realiza la combinación de los resultados parciales. Multiplica xiyi por 10 ^ (2 * n2) para obtener la parte izquierda del resultado, multiplica aux por 10^n2 para obtener la parte intermedia y agrega xdyd para obtener la parte derecha del resultado. Luego, se suman estas tres partes para obtener el resultado final.
- El resultado se almacena en la variable z y se imprime en la pantalla.

En resumen, este código implementa una multiplicación eficiente de números enteros largos dividiendo los números en partes más pequeñas, multiplicando recursivamente esas partes y luego combinando los resultados parciales para obtener el resultado final.

Resultados.

En esta sección se puede observar que el funcionamiento del algoritmo diseñado en lenguaje Python funciona correctamente:

```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.1929 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\alexl\Downloads\enteroslargos.py ==============
Escribe el primer numero: 8738292921
Escribe el segundo numero: 8291817171
El resultado de la multiplicación es: 72456327287575546491
```

```
>>>
============== RESTART: C:\Users\alexl\Downloads\enteroslargos.py =========
Escribe el primer numero: 828191811191
Escribe el segundo numero: 829010191911
El resultado de la multiplicación es: 686579452930627915460001
```

```
>>>
============== RESTART: C:\Users\alexl\Downloads\enteroslargos.py ============
Escribe el primer numero: 6261716151
Escribe el segundo numero: 728276161699
El resultado de la multiplicación es: 4560258604098915900549
```

```
>>>
============== RESTART: C:\Users\alexl\Downloads\enteroslargos.py ============
Escribe el primer numero: 8922929181
Escribe el segundo numero: 5262515628
El resultado de la multiplicación es: 46957054262549740668
```

Conclusiones.

Podemos decir que la implementación del truco de Gauss para resolver multiplicaciones enteras de gran tamaño también es de gran ayuda en la programación, ya que permite que se realizan en menor tiempo, ayudando que la complejidad de nuestro algoritmo se reduzca a comparación del algoritmo base utilizado para el mismo propósito (resolver multiplicaciones de números enteros grandes).