

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ANÁLISIS DE ALGORITMOS**

**PROFESORA: LUZ MARÍA SÁNCHEZ GARCÍA**

**INTEGRANTES:**

**VÁZQUEZ MORENO MARCOS OSWALDO 2016601777**

**DE LOS SANTOS DÍAZ LUIS ALEJANDRO 2017630451**

**PRÁCTICA 8 MULTIPLICACIÓN OPTIMA DE MATRICES**

**3CM2**

**20 DE ABRIL DE 2019**

**Introducción**

Tenemos una secuencia (cadena) de n matrices a ser multiplicadas y queremos calcular el producto: A1A2...An.

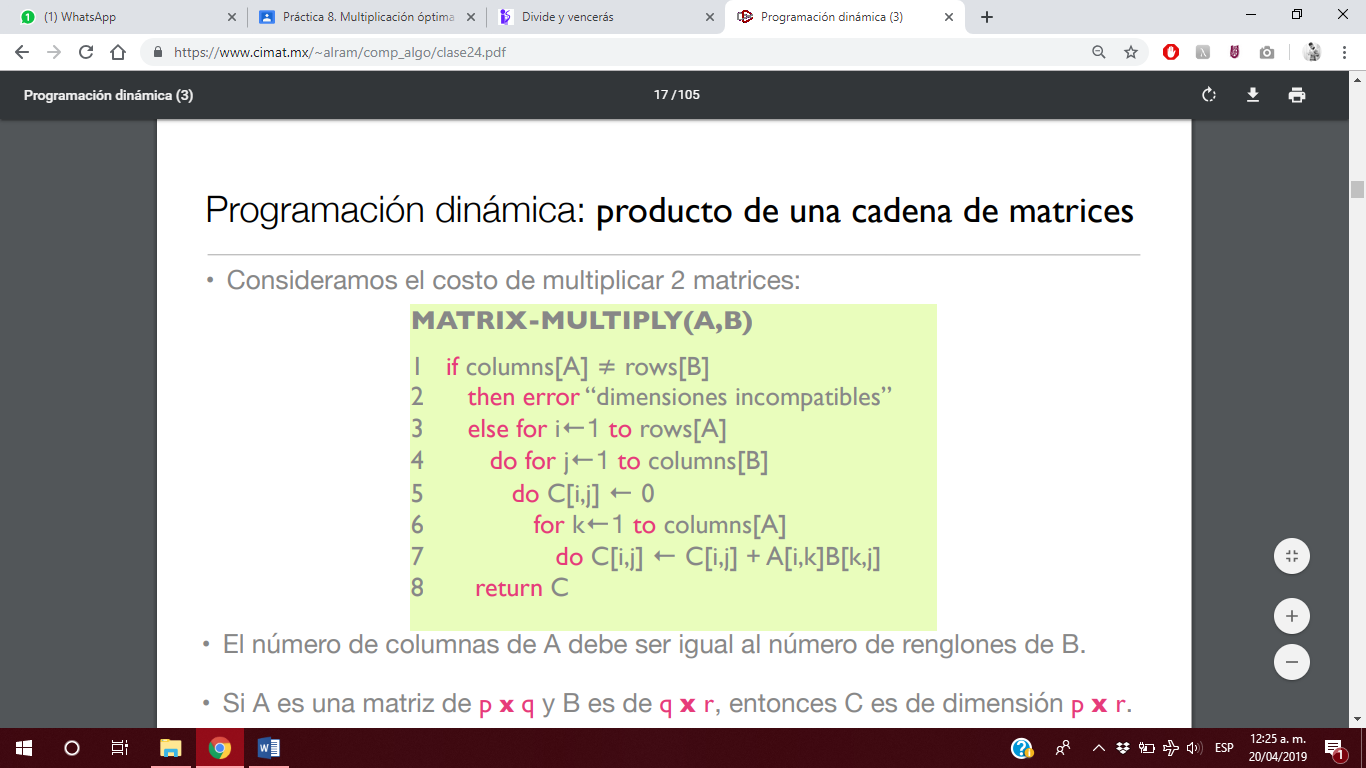
• Esto se puede resolver con una subrutina de multiplicación de pares de matrices una vez que se ha decidido el orden en que serán multiplicadas.

• La multiplicación de matrices es asociativa, así que cualquier orden dará el mismo resultado. (Manzanares, 2017)

• Por ejemplo, si tenemos la cadena , el producto ABCD se puede realizar de 5 formas distintas:

(A(B(CD))), (A((BC)D)), ((AB)(CD)), ((A(BC))D), (((AB)C)D).

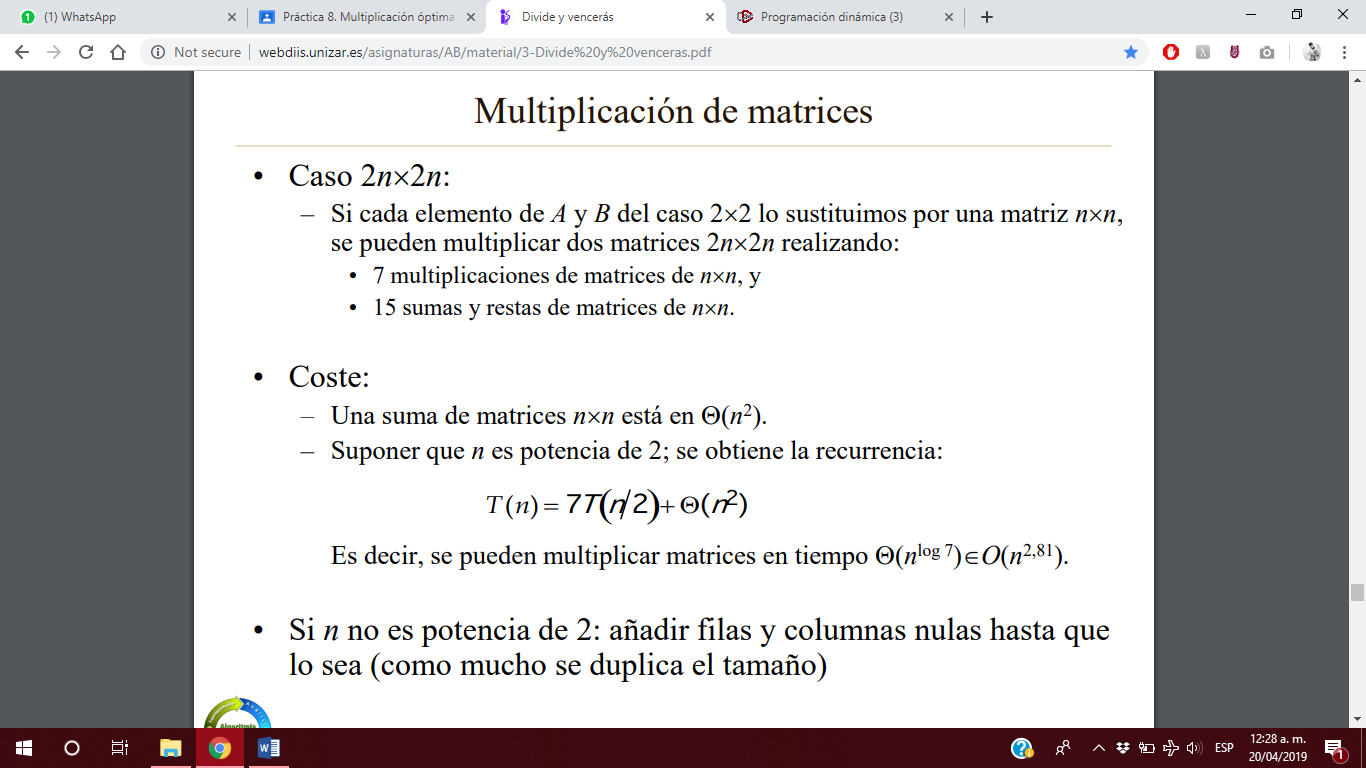
• Consideramos el costo de multiplicar 2 matrices:



El número de columnas de A debe ser igual al número de renglones de B.

• Si A es una matriz de p x q y B es de q x r, entonces C es de dimensión p x r.

• El tiempo de cálculo de C es dominado por la multiplicación de escalares en la línea 7 que es pqr.



(Campos, 2019)

**Planteamiento del problema**

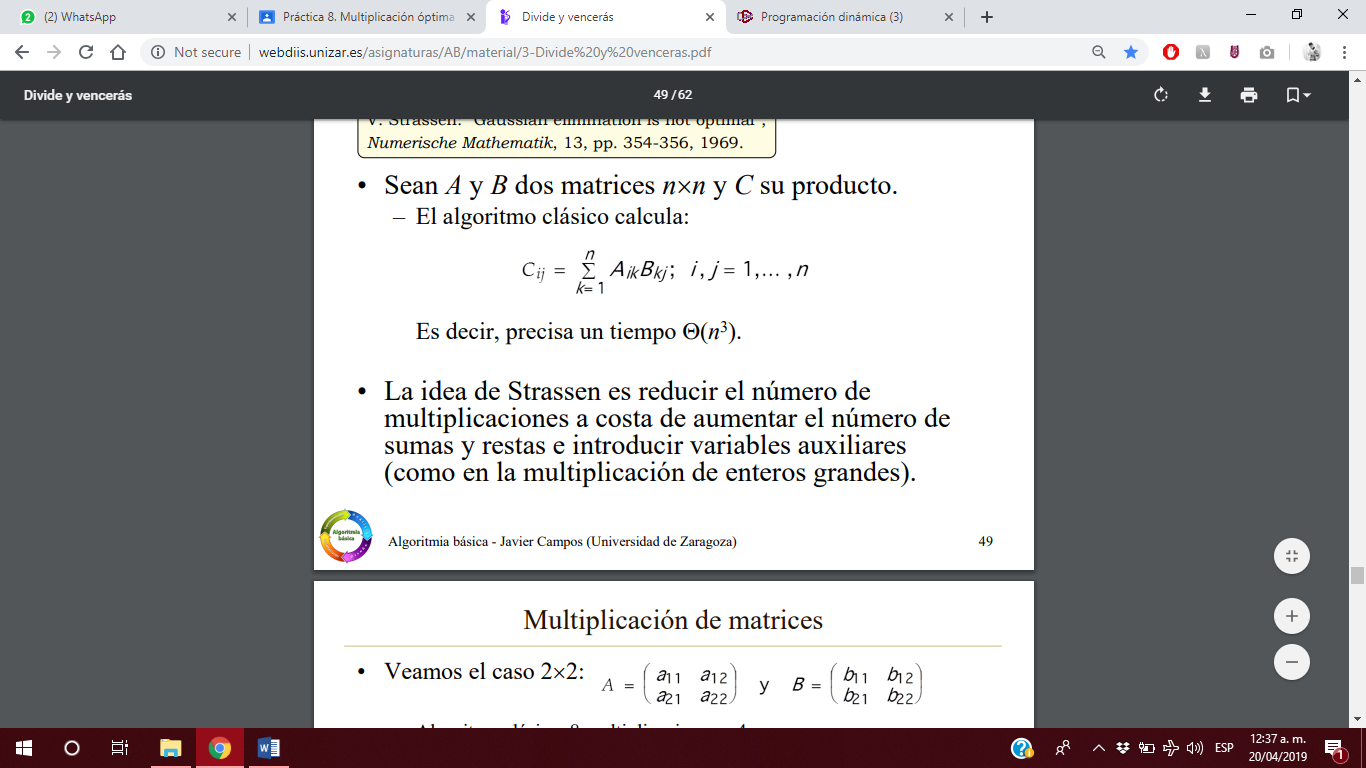
Implementar el algoritmo de multiplicación optima de matrices con programación dinámica en C:

Comprobar las soluciones que arrojan y su optimalidad.

Medir los tiempos de ejecución de las implementaciones para distintos tamaños del problema.

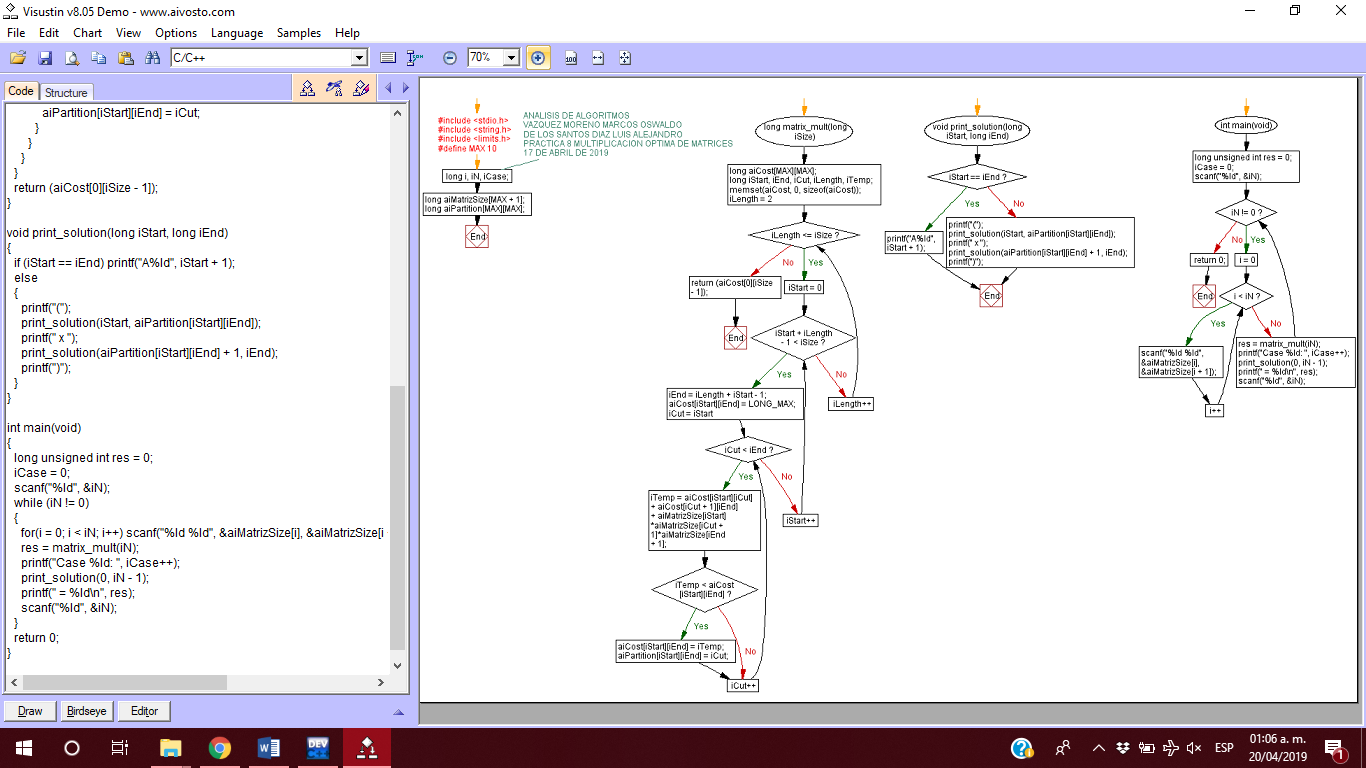
Necesitamos calcular la matriz producto M de n matrices dadas M = M1 M2…Mn minimizando el número total de multiplicaciones escalares a realizar.

**Diseño de la solución**



A continuación, se muestran los diagramas de flujo de nuestra propuesta de solución para el algoritmo de multiplicación óptima de matrices.

Primeramente, se muestra en el diagrama 1.1 el Matrices.c de nuestra propuesta de solución.

  
Diagrama 1.1 Matrices.c

**Implementación de la solución**

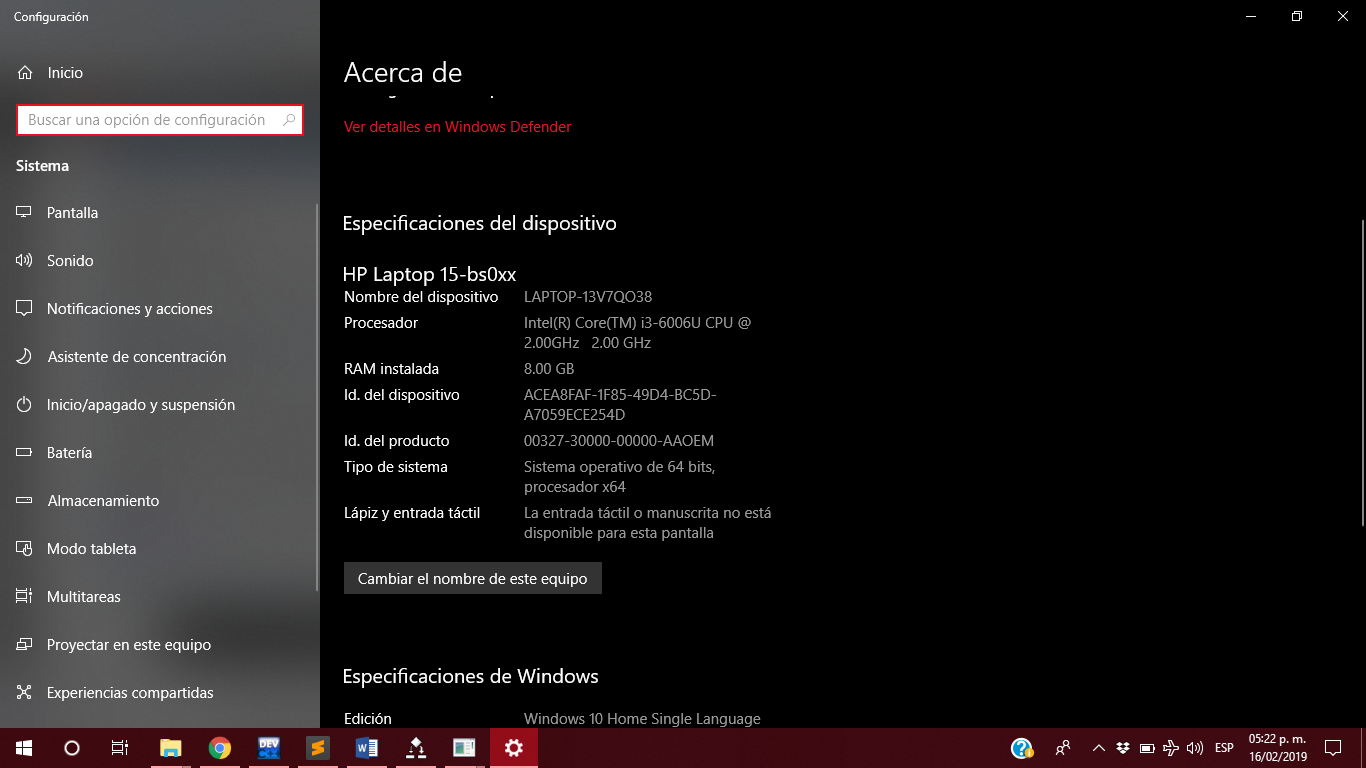
A continuación, se muestra el código de Matrices.c

1. #include <stdio.h>
2. #include <string.h>
3. #include <limits.h>
5. #define MAX 10
7. /\*
8. ANALISIS DE ALGORITMOS
9. VAZQUEZ MORENO MARCOS OSWALDO
10. DE LOS SANTOS DIAZ LUIS ALEJANDRO
12. PRACTICA 8 MULTIPLICACION OPTIMA DE MATRICES
13. 17 DE ABRIL DE 2019
14. \*/
16. **long** i, iN, iCase;
17. **long** aiMatrizSize[MAX + 1];
18. **long** aiPartition[MAX][MAX];
20. **long** matrix\_mult(**long** iSize)
21. {
22. **long** aiCost[MAX][MAX];
23. **long** iStart, iEnd, iCut, iLength, iTemp;
24. memset(aiCost, 0, **sizeof**(aiCost));
26. **for** (iLength = 2; iLength <= iSize; iLength++)
27. {
28. **for**(iStart = 0; iStart + iLength - 1 < iSize; iStart++)
29. {
30. iEnd = iLength + iStart - 1;
31. aiCost[iStart][iEnd] = LONG\_MAX;
32. **for**(iCut = iStart; iCut < iEnd; iCut++)
33. {
34. iTemp = aiCost[iStart][iCut] + aiCost[iCut + 1][iEnd] + aiMatrizSize[iStart]\*aiMatrizSize[iCut + 1]\*aiMatrizSize[iEnd + 1];
35. **if** (iTemp < aiCost[iStart][iEnd])
36. {
37. aiCost[iStart][iEnd] = iTemp;
38. aiPartition[iStart][iEnd] = iCut;
39. }
40. }
41. }
42. }
43. **return** (aiCost[0][iSize - 1]);
44. }
46. **void** print\_solution(**long** iStart, **long** iEnd)
47. {
48. **if** (iStart == iEnd) printf("A%ld", iStart + 1);
49. **else**
50. {
51. printf("(");
52. print\_solution(iStart, aiPartition[iStart][iEnd]);
53. printf(" x ");
54. print\_solution(aiPartition[iStart][iEnd] + 1, iEnd);
55. printf(")");
56. }
57. }
59. **int** main(**void**)
60. {
61. **long** unsigned **int** res = 0;
62. iCase = 0;
63. scanf("%ld", &iN);
64. **while** (iN != 0)
65. {
66. **for**(i = 0; i < iN; i++) scanf("%ld %ld", &aiMatrizSize[i], &aiMatrizSize[i + 1]);
67. res = matrix\_mult(iN);
68. printf("Case %ld: ", iCase++);
69. print\_solution(0, iN - 1);
70. printf(" = %ld\n", res);
71. scanf("%ld", &iN);
72. }
73. **return** 0;
74. }

**Funcionamiento**

**Plataforma experimental**

La ejecución de los algoritmos anteriores se llevó a cabo en una computadora personal que se describe en la siguiente imagen.

  
Imagen 3.1 Plataforma Experimental

El compilador utilizado fue gcc integrado dentro del IDE DevC en un sistema operativo de 64 bits Windows 10.

**Gráfica del comportamiento temporal.**

|  |  |  |
| --- | --- | --- |
| Tamaño de M | Tamaño de N | Tiempo |
| 100 | 100 | 0.000077 |
| 80 | 80 | 0.000076 |
| 60 | 60 | 0.0000732 |
| 40 | 40 | 0.000025 |
| 20 | 20 | 0.000006 |
| 10 | 10 | 0.000003 |
| 8 | 8 | 0.000001 |
| 4 | 4 | 0.000001 |

Tabla 1.1

Gráfica 1.1

**Conclusiones**

En conclusión, según algunos estudios empíricos y debido a la constante multiplicativa, para que se noten mejoras con respecto al algoritmo clásico y no el de Strassen, n debe ser superior a 100 y la matriz densa. Además, es menos estable que el algoritmo clásico ya que para errores similares en los datos produce mayores errores en el resultado.

Es más difícil de implementar que el algoritmo clásico, así como difícilmente paralelizable, mientras que el clásico puede ser fácilmente paralelizable.

Este algoritmo quizá de haberlo visto en cursos anteriores a haber tomado algebra lineal hubiera sido de gran ayuda ya que contamos con un poco más de recursos para llevar a cabo programas con multiplicación de matrices y no solo contando con lo aprendido hasta estructuras de datos.

Por último, importante hay que mencionar que el pseudo código pinta las cosas un poco más sencillo a lo que se espera, sin embargo, es un poco complicado el llevarlo a cabo, una práctica que sin duda nos deja buen sabor de boca, nos llena de alegría poderla entregar tomando en cuenta el reto que para nosotros significo. Terminando no sin antes mencionar que las fuentes citadas en esta práctica fueron de mucha ayuda para poder realizar todo adecuadamente y proporcionando información teórica muy buena.

**Bibliografía**

# Bibliografía

Campos, J. (07 de Abril de 2019). *Algoritmia básica.* Obtenido de Algoritmia básica (Universidad de Zaragoza): http://webdiis.unizar.es/asignaturas/AB/material/3-Divide%20y%20venceras.pdf

Díaz, I. N. (2006). *FIET UNICAUCA*. Obtenido de http://artemisa.unicauca.edu.co/~nediaz/EDDI/cap02.htm

García, L. M. (17 de 04 de 2019). *Classrom.* Obtenido de Práctica 8 Ánalisis de Algoritmos: https://classroom.google.com/c/NzEzNTUxMDU1NVpa/a/NzgxNTE2NTA3MVpa/details

Gurin, S. (Septiembre de 2004). *Algoritmos de ordenación*. Obtenido de Algoritmos de ordenación: http://es.tldp.org/Tutoriales/doc-programacion-algoritmos-ordenacion/alg\_orden.pdf

Manzanares, A. R. (19 de Mayo de 2017). *Computación y Algoritmo.* Obtenido de CITAM: https://www.cimat.mx/~alram/comp\_algo/clase24.pdf