# Multiplicación de matrices

# Multiplicación de matrices

- Hay dos matrices A y B que son compatibles en el sentido de que el número de columnas de A es igual al número de filas de B.
- En general, una expresión que contiene un producto matricial **A B** siempre se supone que implica que las matrices **A** y **B** son compatibles.
  - Si $\mathbf{A = (a_{ik})}$ es una matriz ***m x n***
  - $\mathbf{B = (b_{kj})}$ es una matriz ***n x p***, entonces su producto matricial **C = A B** es
  - $\mathbf{C = (a_{ij})}$ es una matriz ***m x p*** donde:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Para **i** = 1, 2, ..., m
y **j** = 1, 2, ..., p

# El algoritmo de Strassen para la multiplicación de matrices

- Si **A = ($a_{ij}$)** y **B = ($b_{ij}$)** son matrices cuadradas **n x n**, entonces en el producto **C = A • B**, definimos la entrada **$C_{ij}$**, para **i, j = 1, 2, …., n** como:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

- Debemos calcular **$n^2$** entradas de la matriz, y cada una es la suma de **n** valores.
- El siguiente procedimiento toma **n x n** matrices **A** y **B** y las multiplica, devolviendo el producto **C** de **n x n** .
- Suponemos que cada matriz tiene el atributo filas, dando el número de filas en la matriz.

El procedimiento SQUARE-MATRIX-MULTIPLY trabaja de la siguiente manera:

```
SQUARE-MATRIX-MULTIPLY(A, B)
1  n = A.rows
2  let C be a new n × n matrix
3  for i = 1 to n
4      for j = 1 to n
5          c_ij = 0
6          for k = 1 to n
7              c_ij = c_ij + a_ik · b_kj
8  return C
```

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

Ecuación

- El bucle for de las líneas 3-7 calcula las entradas de cada fila **i**, y dentro de una fila **i** dada, el ciclo for de las líneas 4-7 calcula cada una de las entradas **Cij**, para cada columna **j**.
- La línea 5 inicializa **Cij** = 0 cuando comenzamos a calcular la suma dada en la ecuación.
- Cada iteración del bucle for de las líneas 6-7 agrega un término más de la ecuación.

- Debido a que cada uno de los bucles for triplemente anidados ejecutan exactamente n iteraciones, y cada ejecución de la línea 7 lleva un tiempo constante, el procedimiento SQUARE-MATRIX-MULTIPLY toma un tiempo $\boldsymbol{\Theta(n^3)}$.

- Al principio, se podría pensar que cualquier algoritmo de multiplicación de matrices debe tomar un tiempo $\boldsymbol{\Omega(n^3)}$, ya que la definición natural de multiplicación de matrices requiere tantas multiplicaciones.

- Sin embargo, sería incorrecto: tenemos una forma de multiplicar matrices en tiempo $\boldsymbol{o(n^3)}$.

- El algoritmo recursivo de Strassen para multiplicar **n x n** matrices se ejecuta en tiempo $\Theta(n^{\lg 7})$.

- Dado que **lg 7** se encuentra entre **2.80** y **2.81**, el algoritmo de Strassen se ejecuta en un tiempo $O(n^{2.81})$, que es asintóticamente mejor que el procedimiento simple SQUARE-MATRIX-MULTIPLY.

# Un algoritmo simple de divide y vencerás

# Un algoritmo simple de divide y vencerás

- Para simplificar las cosas, cuando usamos un algoritmo de divide y vencerás para calcular el producto matricial $\mathbf{C = A \cdot B}$, asumimos que $\mathbf{n}$ es una potencia exacta de $\mathbf{2}$ en cada uno de las $\mathbf{n \times n}$ matrices.

- Hacemos esta suposición porque en cada paso de la división, dividiremos $\mathbf{n \times n}$ matrices en cuatro $\mathbf{n/2 \times n/2}$ matrices, y suponiendo que $\mathbf{n}$ es una potencia exacta de $\mathbf{2}$, se garantiza que mientras $\mathbf{n \geq 2}$, la dimensión $\mathbf{n/2}$ es un entero.

- Supongamos que dividimos cada una de **A**, **B** y **C** en cuatro matrices **n/2 x n/2**.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

- De modo que reescribimos la ecuación **C = A • B** como:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

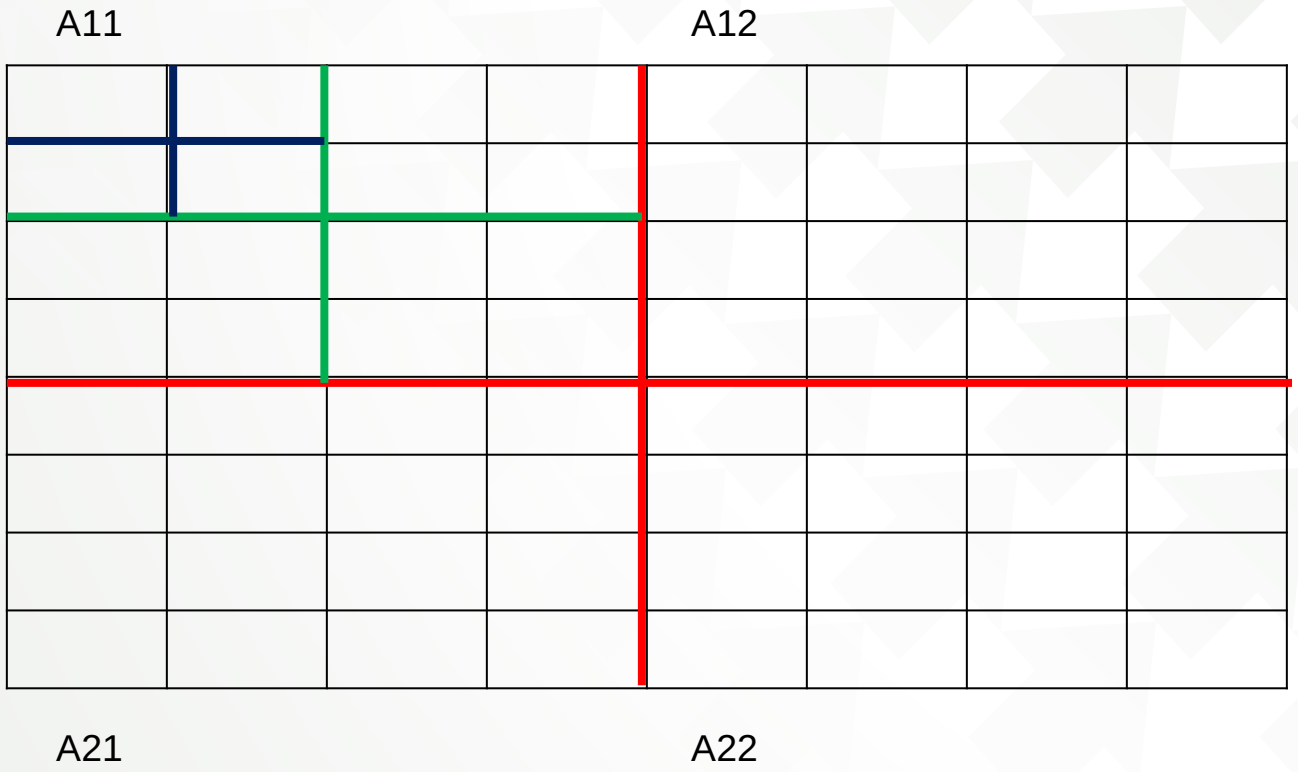
- La ecuación anterior corresponde a las cuatro ecuaciones:

$$\begin{aligned} C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\ C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\ C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\ C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22} \end{aligned}$$

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
 1  n = A.rows
 2  let C be a new n × n matrix
 3  if n == 1
 4      c11 = a11 · b11
 5  else partition A, B, and C as in equations (4.9)
 6      C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
 7      C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
 8      C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
 9      C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
10  return C
```

- Cada una de estas cuatro ecuaciones especifica dos multiplicaciones de **n/2 x n/2** matrices y la suma de sus **n/2 x n/2** productos.
- Podemos usar estas ecuaciones para crear un algoritmo sencillo, recursivo, de divide y vencerás:

N = 8
n/2 = 4



- Este pseudocódigo pasa por alto un detalle de implementación sutil pero importante.

- ¿Cómo dividimos las matrices en la línea 5?
  Si tuviéramos que crear 12 nuevas matrices $\mathbf{n/2 \times n/2}$, gastaríamos un tiempo $\mathbf{\Theta(n^2)}$ copiando entradas.

- De hecho, podemos particionar las matrices sin copiar entradas.

- El truco consiste en utilizar cálculos de índices.
  Se identifica una submatriz mediante un rango de índices de fila y un rango de índices de columna de la matriz original.

- Terminamos representando una submatriz un poco diferente de cómo representamos la matriz original, que es la sutileza que estamos pasando por alto.

- La ventaja es que, dado que podemos especificar submatrices mediante cálculos de índice, ejecutar la línea 5 toma sólo toma un tiempo $\Theta(1)$.

- (aunque veremos que no se diferencia asintóticamente al tiempo de ejecución total si copiamos o particionamos en su lugar).

- Ahora, derivamos una recurrencia para caracterizar el tiempo de ejecución de SQUARE-MATRIX-MULTIPLY-RECURSIVE.

- Sea **T(n)** el tiempo para multiplicar dos matrices de **n x n** utilizando este procedimiento.

- En el caso base, cuando **n = 1**, realizamos solo una multiplicación escalar en la línea 4, y así:

$$T(1) = \Theta(1)$$

- El caso recursivo ocurre cuando n > 1.
- Dividir las matrices en la línea 5 toma un tiempo **Θ(1)**, usando cálculos de índice.
- En las líneas 6 a 9, llamamos de forma recursiva SQUARE-MATRIX-MULTIPLY-RECURSIVE un total de ocho veces.
- Porque cada llamada recursiva multiplica dos matrices **n/2 x n/2**, contribuyendo así **T(n/2)** a el tiempo de ejecución total, el tiempo que toman las ocho llamadas recursivas es **8T(n/2)**.
- Se debe tener en cuenta las cuatro adiciones de la matriz en las líneas 6 a 9.

- Cada una de estas matrices contiene $\mathbf{n^2/4}$ entradas, por lo que cada una de las cuatro adiciones de la matriz toma un tiempo $\mathbf{\Theta(n^2)}$.

- Dado que el número de sumas de matrices es una constante, el tiempo total empleado en sumar matrices en las líneas 6 a 9 es $\mathbf{\Theta(n^2)}$.

- Nuevamente, usamos cálculos de índices para colocar los resultados de las adiciones de la matriz en las posiciones correctas de la matriz $\mathbf{C}$, con una sobrecarga de un tiempo por entrada $\mathbf{\Theta(1)}$.

- El tiempo total para el caso recursivo, por lo tanto, es la suma del tiempo de partición, el tiempo para todas las llamadas recursivas y el tiempo para agregar las matrices resultantes de las llamadas recursivas:

$$\begin{aligned} T(n) &= \Theta(1) + 8T(n/2) + \Theta(n^2) \\ &= 8T(n/2) + \Theta(n^2) \,. \end{aligned}$$

- Se debe tener en cuenta que si se implementa la partición copiando matrices, costaría un tiempo $\mathbf{\Theta(n^2)}$, la recurrencia no cambiaría y, por lo tanto, el tiempo de ejecución general aumentaría solo en un factor constante.

- La combinación de las siguientes ecuaciones (a) y (b) nos da la recurrencia para el tiempo de ejecución de SQUARE-MATRIX-MULTIPLY-RECURSIVE:

$$T(1) = \Theta(1) \quad \textbf{(a)}$$

$$\begin{aligned} T(n) &= \Theta(1) + 8T(n/2) + \Theta(n^2) \\ &= 8T(n/2) + \Theta(n^2) \,. \end{aligned} \quad \textbf{(b)}$$

**(c)** $$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \,, \\ 8T(n/2) + \Theta(n^2) & \text{if } n > 1 \,. \end{cases}$$

- La recurrencia (c) tiene la solución $\mathbf{T(n) = \Theta(n^3)}$.
- Por lo tanto, este simple enfoque de divide y vencerás no es más rápido que el procedimiento sencillo SQUARE-MATRIX-MULTIPLY.

# El método Strassen

# El método Strassen

- La clave del método de Strassen es hacer que el árbol de recursividad sea un poco menos tupido.
- Que es, en lugar de realizar ocho multiplicaciones recursivas de $\mathbf{n/2 \times n/2}$ matrices, se realizan solo siete.
- El costo de eliminar una multiplicación de matrices serán varias nuevas adiciones de matrices $\mathbf{n/2 \times n/2}$, pero aún así solo un número constante de adiciones.
- Como antes, el número constante de adiciones de matrices se absorberá por la notación $\Theta$, cuando configuramos la ecuación de recurrencia para caracterizar el tiempo de ejecución.

⇨ El método de Strassen tiene cuatro pasos:

1. Divida las matrices de entrada **A** y **B** y la matriz de salida **C** en submatrices de **n/2 x n/2**, como en la siguiente ecuación.

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

Este paso toma un tiempo **Θ(1)** por el cálculo de índices, como en SQUARE-MATRIX-MULTIPLY-RECURSIVE.

2. Cree 10 matrices $\mathbf{S_1, S_2, ...., S_{10}}$, cada una de las cuales es $\mathbf{n/2 \times n/2}$ y es la suma o diferencia de dos matrices creadas en el paso 1.
   Podemos crear las 10 matrices en un tiempo $\mathbf{\Theta(n^2)}$.

3. Usando las submatrices creadas en el paso 1 y las 10 matrices creadas en el paso 2, calcular recursivamente siete productos matriciales:

   $$\mathbf{P_1, P_2, ...., P_7}$$

   Cada matriz Pi es de $\mathbf{n/2 \times n/2}$.

4. Calcule las submatrices deseadas $C_{11}, C_{12}, C_{21}, C_{22}$ de la matriz de resultados **C** sumando y restando varias combinaciones de las matrices Pi.
   Se pueden calcular las cuatro submatrices en un tiempo $\Theta(n^2)$.

⇨ Ya se tiene suficiente información para establecer una ecuación de recurrencia para el tiempo de ejecución del método de Strassen.

- Supongamos que una vez que el tamaño de la matriz **n** desciende a **1**, realizamos una multiplicación escalar simple, al igual que en la línea 4 de SQUARE-MATRIX-MULTIPLY-RECURSIVE.
- Cuándo **n> 1**, los pasos 1, 2 y 4 toman un tiempo total de $\Theta(n^2)$, y el paso 3 requiere que realicemos siete multiplicaciones de matrices **n/2 x n/2**.
- Por tanto, obtenemos la siguiente recurrencia para el tiempo de ejecución **T(n)** del algoritmo de Strassen:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1 \end{cases}$$

- Se ha cambiado una multiplicación de matrices por un número constante de sumas de matrices.

- Una vez que se entienden las recurrencias y sus soluciones, se verá que en realidad esta compensación conduce a un menor tiempo de ejecución asintótico.

- Por el método maestro la recurrencia anterior tiene la solución $\mathbf{T(n) = \Theta(n^{\lg 7})}$.

- En el paso 2, se crean las siguientes 10 matrices:

$$
\begin{aligned}
S_1 &= B_{12} - B_{22} , \\
S_2 &= A_{11} + A_{12} , \\
S_3 &= A_{21} + A_{22} , \\
S_4 &= B_{21} - B_{11} , \\
S_5 &= A_{11} + A_{22} , \\
S_6 &= B_{11} + B_{22} , \\
S_7 &= A_{12} - A_{22} , \\
S_8 &= B_{21} + B_{22} , \\
S_9 &= A_{11} - A_{21} , \\
S_{10} &= B_{11} + B_{12} .
\end{aligned}
$$

- Dado que solo se debe sumar o restar matrices de $\mathbf{n/2 \times n/2}$ un total de 10 veces, este paso toma un tiempo de $\boldsymbol{\Theta(n^2)}$.

- En el paso 3, se multiplican de forma recursiva matrices de **n/2 x n/2** siete veces para calcular las siguientes matrices de **n/2 x n/2**, cada una de las cuales es la suma o diferencia de los productos de las submatrices A y B:

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1 &&= A_{11} \cdot B_{12} - A_{11} \cdot B_{22} , \\
P_2 &= S_2 \cdot B_{22} &&= A_{11} \cdot B_{22} + A_{12} \cdot B_{22} , \\
P_3 &= S_3 \cdot B_{11} &&= A_{21} \cdot B_{11} + A_{22} \cdot B_{11} , \\
P_4 &= A_{22} \cdot S_4 &&= A_{22} \cdot B_{21} - A_{22} \cdot B_{11} , \\
P_5 &= S_5 \cdot S_6 &&= A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} , \\
P_6 &= S_7 \cdot S_8 &&= A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22} , \\
P_7 &= S_9 \cdot S_{10} &&= A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12} .
\end{aligned}
$$

Tenga en cuenta que las únicas multiplicaciones que se deben realizar son las de la columna de en medio de las ecuaciones anteriores.

La columna de la derecha solo muestra como quedan estos productos en términos de las submatrices originales creadas en el paso 1.

- El paso 4 se suman y restan las matrices **Pi** creadas en el paso 3 para construir las cuatro submatrices **n/2 x n/2** del producto de **C**.
- Comenzamos con: $C_{11} = P_5 + P_4 - P_2 + P_6$

- Expandiendo el lado derecho, con la expansión de cada **Pi** en su propia línea y alineando verticalmente los términos que se cancelan, vemos que $\mathbf{C_{11}}$ es igual a:

$$
\begin{array}{llll}
A_{11}\cdot B_{11} + A_{11}\cdot B_{22} + A_{22}\cdot B_{11} + A_{22}\cdot B_{22} & & & \\
- A_{22}\cdot B_{11} & & + A_{22}\cdot B_{21} & \\
- A_{11}\cdot B_{22} & & & - A_{12}\cdot B_{22} \\
& - A_{22}\cdot B_{22} & - A_{22}\cdot B_{21} & + A_{12}\cdot B_{22} + A_{12}\cdot B_{21} \\
\hline
A_{11}\cdot B_{11} & & & + A_{12}\cdot B_{21}
\end{array}
$$

- Que corresponde a la ecuación: $C_{11} = A_{11}\cdot B_{11} + A_{12}\cdot B_{21}$

- Del mismo modo, establecemos

$$C_{12} = P_1 + P_2$$

- y entonces $\mathbf{C_{12}}$ es igual a:

$$\begin{array}{r} A_{11} \cdot B_{12} - A_{11} \cdot B_{22} \\ + A_{11} \cdot B_{22} + A_{12} \cdot B_{22} \\ \hline A_{11} \cdot B_{12} \qquad\qquad + A_{12} \cdot B_{22} \end{array}$$

- Correspondiente a la ecuación:

$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$

- Ahora se establece que:

$$C_{21} = P_3 + P_4$$

- y entonces $\mathbf{C_{21}}$ es igual a:

$$\begin{array}{l} A_{21} \cdot B_{11} + A_{22} \cdot B_{11} \\ \qquad - A_{22} \cdot B_{11} + A_{22} \cdot B_{21} \\ \hline A_{21} \cdot B_{11} \qquad\qquad + A_{22} \cdot B_{21} \end{array}$$

- Correspondiente a la ecuación:

$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$

⇨ Finalmente establecemos:

$$C_{22} = P_5 + P_1 - P_3 - P_7$$

⇨ Para que $\mathbf{C_{22}}$ sea igual a:

$$\begin{array}{llll} A_{11}\cdot B_{11} + A_{11}\cdot B_{22} + A_{22}\cdot B_{11} + A_{22}\cdot B_{22} & & & \\ \quad - A_{11}\cdot B_{22} & & + A_{11}\cdot B_{12} & \\ & - A_{22}\cdot B_{11} & & - A_{21}\cdot B_{11} \\ - A_{11}\cdot B_{11} & & - A_{11}\cdot B_{12} + A_{21}\cdot B_{11} + A_{21}\cdot B_{12} & \\ \hline & A_{22}\cdot B_{22} & & + A_{21}\cdot B_{12} \end{array}$$

⇨ Correspondiente a la ecuación:

$$C_{22} = A_{21}\cdot B_{12} + A_{22}\cdot B_{22}$$

- En total, sumamos o restamos ocho veces matrices de **n/2 x n/2** en el paso 4, por lo que este paso de hecho toma un tiempo de $\Theta(n^2)$.

- Por lo tanto, vemos que el algoritmo de Strassen, que comprende los pasos 1 a 4, produce el producto correcto de la matriz y que la recurrencia **T(n)** caracteriza su tiempo de ejecución.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 7T(n/2) + \Theta(n^2) & \text{if } n > 1 \end{cases}$$

- Esta recurrencia tiene la solución $\mathbf{T(n) = \Theta(n^{\lg 7})}$.

- El método de Strassen es asintóticamente más rápido que el procedimiento sencillo SQUARE-MATRIX-MULTIPLY.

# Comentarios generales

- El algoritmo de Strassen causó mucho entusiasmo cuando se publicó en 1969.
- Antes de eso, pocos imaginaban la posibilidad de un algoritmo asintóticamente más rápido que el procedimiento básico SQUARE-MATRIX-MULTIPLY.
- El límite superior asintótico para la multiplicación de matrices se ha mejorado desde entonces.

- El algoritmo asintóticamente más eficiente para multiplicar matrices de **n x n** hasta la fecha, es el de Calderero y Winograd, el cuál tiene un tiempo de ejecución de $\mathbf{O(n^{2.376})}$.

- El mejor limite inferior conocido es simplemente es el límite $\mathbf{\Omega(n^2)}$ (obviamente porque debemos completar $\mathbf{n^2}$ elementos del producto de la matriz).

- Desde un punto de vista práctico, el algoritmo de Strassen a menudo no es el método de elección para la multiplicación de matrices, por cuatro razones:

1. El factor constante oculto en el tiempo de ejecución $\Theta(n^{\lg 7})$ del algoritmo de Strassen es mayor que el factor constante en el tiempo $\Theta(n^3)$ del procedimiento SQUARE-MATRIX-MULTIPLY.

2. Cuando las matrices son pequeñas, los métodos diseñados para matrices pequeñas son más rápidos.

3. El algoritmo de Strassen no es tan estable numéricamente como SQUARE-MATRIX.MULTIPLY.
   En otras palabras, debido a la precisión limitada de la aritmética informática en valores no enteros, los errores más grandes se acumulan en el algoritmo de Strassen que en SQUARE-MATRIX-MULTIPLY.

4. Las submatrices formadas en los niveles de recursividad consumen espacio.