INFORME DE INGENIERIA
LABORATORIO UNIDAD 1

Presentado por:
Marisol Giraldo Cobo

Presentado al Profesor:
Andres Aristizabal

Universidad ICESI
Santiago de Cali, agosto 20 del 2019

INFORME DE INGENIERÍA

CONTEXTO DEL PROBLEMA

Venus ha estado en guerra con Marte disputando la soberanía territorial de las 53 lunas de Saturno y sus anillos; y con la ayuda de la Tierra necesita crear un plan que les ayude a ganar, para ello, el equipo de implementación utilizará el desarrollo de algoritmos de multiplicación como técnica para localizar y atacar la flota enemiga.

PASO 1: IDENTIFICACIÓN DEL PROBLEMA

IDENTIFICACIÓN DE NECESIDADES Y SINTOMAS:

- Se sabe que las naves de Marte están ubicadas estratégicamente de forma secreta e invisibles a las tropas de Venus.
- Cada nave posee unas coordenadas *x* y *y,* como parte de su posición en el tablero de batalla. La única forma de descifrar su ubicación y destruirlas es utilizando la técnica de multiplicación de matrices para encontrar las coordenadas exactas.
- La matriz de batalla, donde se ven reflejadas las posiciones de las naves de la flota marciana, se encuentra compuesta por números enteros positivos. Si el número presente dentro de alguna celda del tablero es un número primo, en dicha posición se encuentra presente una nave.
- Como la matriz que poseen los altos mandos venusianos tan sólo representa la ubicación de la flota de Marte en guerras pasadas se ha de multiplicar por la matriz de coeficientes que ha podido descubrir el servicio secreto de Venus. Dicha multiplicación dará lugar a una nueva matriz de enteros que mostrará la ubicación de las naves en la actualidad, al igual que su cantidad real.

DEFINICIÓN DEL PROBLEMA

El planeta Marte requiere una aplicación gráfica que mediante la implementación de algoritmos de multiplicación le permita localizar y atacar la flota enemiga del planeta Venus para ganar la guerra y recuperar la soberanía territorial de las 53 lunas de Saturno y sus anillos.

PASO 2: RECOPILACIÓN DE INFORMACIÓN

Algoritmo
Es una secuencia de pasos bien definidos que buscan resolver un problema computacional.

Algoritmo Iterativo
Son algoritmos que se caracterizan por ejecutarse mediante ciclos, es decir, aquellos que llegan a un resultado a través de una iteración mediante un ciclo definido o indefinido.

Eficiencia
Medida del uso de los recursos computacionales requeridos por la ejecución de un algoritmo en función del tamaño de las entradas.

Complejidad temporal
Función que describe el comportamiento (en tiempo) de un algoritmo conforme se incrementa el tamaño de la entrada.

Complejidad espacial
Uso de recursos de espacio que requiere un algoritmo en cuestión.

Notación asintótica
Son aquellas notaciones utilizadas para describir el tiempo de ejecución asintótico de un algoritmo. Se conoce la notación O para el pero caso, Ω para el mejor caso y θ para el caso promedio.

Complejidad	Terminología
O(l)	Complejidad constante
O(log n)	Complejidad logarítmica
O(n)	Complejidad lineal
O(n log n)	Complejidad n log n
O(n^k)	Complejidad polinómica
O(K^n)	Complejidad exponencial
O(n!)	Complejidad factorial

Matriz
Se puede definir una matriz, como un conjunto de elementos (números) ordenados en filas y columnas. Para designar una matriz se emplean letras mayúsculas. Cada uno de los elementos de la matriz (**a_{ij}**) tiene dos subíndices. El primero **i** indica la fila a la que pertenece y el segundo **j** la columna.
Esta es una matriz de **m** filas y ***n*** columnas, es decir, de **dimensión *m x n***. Esta matriz también se puede representar de la forma siguiente: **$A = (a_{ij})_{m \times n}$.** Si el número de filas y de columnas es igual (**m = n**), entonces se dice que la matriz es de **orden n**.

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}$$

Sean *A*, *B* matrices y k escalar, la multiplicación de matrices se da de la siguiente manera:

Multiplicación de Matrices
Dos matrices A y B son multiplicables si el **número de columnas de A** coincide con el **número de filas de B**. **$M_{m \times n} \times M_{n \times p} = M_{m \times p}$**
El elemento **c_{ij}** de la matriz producto se obtiene **multiplicando** cada elemento de la **fila i** de la matriz A por cada elemento de la **columna j** de la matriz B y **sumándolos**.

$$C = AB = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & \cdots & b_{1p} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{np} \end{pmatrix} = \begin{pmatrix} a_{11}b_{11} + \cdots + a_{1n}b_{n1} & \cdots & a_{11}b_{1p} + \cdots + a_{1n}b_{np} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{11} + \cdots + a_{mn}b_{n1} & \cdots & a_{m1}b_{1p} + \cdots + a_{mn}b_{np} \end{pmatrix}$$

Multiplicación de Matrices por un Escalar

$$kA = (kI)A = \begin{pmatrix} k & 0 & \cdots & 0 \\ 0 & k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k \end{pmatrix} \cdot \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} ka_{11} & \cdots & ka_{1n} \\ \vdots & \ddots & \vdots \\ ka_{n1} & \cdots & ka_{nn} \end{pmatrix}.$$

Multiplicación Producto Punto

Sea:

$$\vec{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \text{y} \quad \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

dos vectores del mismo número de componentes.

El producto escalar de $\vec{a}$ y $\vec{b}$, denotado $\vec{a} \cdot \vec{b}$, se define por:

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n$$

Modelo RAM

Es el modelo que se usa de manera implícita en la practica para evaluar la complejidad de los algoritmos.

PASO 3. BÚSQUEDA DE SOLUCIONES CREATIVAS

ALTERNATIVA 1: MULTIPLICACIÓN DE MATRICES

Supongamos el problema de multiplicar dos matrices cuadradas A, B de tamaños nxn. C = AxB
C(i, j) = ∑ A(i, k)·B(k, j); Para todo i, j= 1..n k=1..n

• *Método clásico de multiplicación:*

```
for i:= 1 to N do
  for j:= 1 to N do
    suma:= 0
      for k:= 1 to N do
         suma:= suma + a[i,k]*b[k,j]
      end
      c[i, j]:= suma
  end
end
```

El método clásico de multiplicación requiere Θ(n^3).

ALTERNATIVA 2: TÉCNICA DE DISEÑO "DIVIDE Y VENCERÁS"

- Descomponer el problema a resolver en un cierto número de subproblemas más pequeños del mismo problema.
- Resolver independientemente cada subproblema.
- Combinar los resultados obtenidos para construir la solución del ejemplar original.

→ Aplicar esta técnica recursivamente

Al aplicar "divide y vencerás":
Cada matriz de **n**x**n** es dividida en cuatro submatrices de tamaño (**n**/2)x(**n**/2): Aij, Bij y Cij.

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

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

- Es necesario resolver 8 problemas de tamaño **n**/2.
- La combinación de los resultados requiere un O(n^2).
  t(n) = 8·t(n/2) + a·n^2.
- Resolviéndolo obtenemos que t(n) es O(n^3).
- Podríamos obtener una mejora si hiciéramos 7 multiplicaciones (o menos)...

ALTERNATIVA 3: ALGORITMO DE STRASSEN

El algoritmo de Strassen mejora tan sólo levemente la complejidad del primer algoritmo recursivo para multiplicación de matrices. La combinación de las sub-soluciones las lleva a cabo de exactamente igual forma, y ya veremos que en esencia lleva a cabo exactamente el mismo procedimiento a la hora de hallar las soluciones de cada uno de los subproblemas en cada nivel de recursividad. La clave reside en que Strassen se las arregló para tener que llevar a cabo tan solo siete productos de matrices para resolver cada subproblema, mientras que el primer algoritmo recursivo debía realizar ocho productos.

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} P_5+P_4-P_2+P_6 & P_1+P_2 \\ P_3+P_4 & P_1+P_5-P_3-P_7 \end{pmatrix}$$

En principio, la definición de estas matrices implica cada una un producto de submatrices desde la A hasta la H.

P1 = A * (F - H)
P2 = (A + B) * H
P3 = (C + D) * E
P4 = D * (G - E)
P5 = (A + D) * (E + H)
P6 = (B - D) * (G + H)
P7 = (A - C) * (E + F)

que luego se utilizan para expresar $C_{i,j}$ en términos de M_k. Debido a nuestra definición de la M_k podemos eliminar una multiplicación de matrices y reducir el número de multiplicaciones a 7 (una multiplicación por cada M_k) y expresar $C_{i,j}$ como

C1,1 = P1 + P4 – P5 + P7
C1,2 = P3 + P5
C2,1 = P2 + P4
C2,2 = P1 – P2 + P3 + P6

Iteramos n-veces el proceso de división hasta que las submatrices degeneran en números (elementos del anillo *R*).

Las implementaciones prácticas del algoritmo de Strassen, permiten cambiar a métodos estándar de multiplicación de matrices para submatrices lo suficientemente pequeñas, para las cuales son más eficientes. El punto a partir del cual el algoritmo de Strassen es más eficiente depende de la implementación específica y del hardware. Se ha estimado que el algoritmo de Strassen es más rápido para matrices con anchura desde 32 a 128 para implementaciones optimizadas,[1] y 60.000 o más para implementaciones básicas.

• El tiempo de ejecución será:

$t(n) = 7 \cdot t(n/2) + a \cdot n^2$

• Resolviéndolo, tenemos que:

$t(n) \in O(n^{log_2 7}) \approx O(n^{2.807})$

Aunque el algoritmo es más complejo e inadecuado para tamaños pequeños, se demuestra que **la cota de complejidad** del problema es menor que $O(n^3)$.

PASO 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES

ALTERNATIVA 1: MULTIPLICACIÓN DE MATRICES

- Es la forma común utilizada para multiplicar matrices
- El tiempo de complejidad es $\Theta(n^3)$.

ALTERNATIVA 2: TÉCNICA DE DISEÑO “DIVIDE Y VENCERÁS”

- El algoritmo divide y vencerás trabaja de manera recursiva dividiendo en problema en subproblemas, hasta que es lo suficientemente pequeño para resolverlo de manera directa; , de manera recursiva resuelve los subproblemas y, por último, combina las soluciones de los subproblemas para resolver el problema original.
- El tiempo de complejidad es $O(n^3)$.

ALTERNATIVA 3: ALGORITMO DE STRASSEN

- La idea de Strassen es reducir el número de multiplicaciones a costa de aumentar el número de sumas y restas e introducir variables auxiliares (como en la multiplicación de enteros grandes).
- El tiempo de complejidad es $O(n^{log_2 7}) \approx O(n^{2.807})$

PASO 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN

CRITERIO 1: MEDICIÓN

[1]: Precisa

[2]: Aproximada

[3]: Inexacta

CRITERIO 2: COMPLEJIDAD

[1]: Constante

[2]: Logarítmica

[3]: Lineal

[4]: n log n

[5]: Polinómica

[6]: Exponencial

[7]: Factorial

[8]: n/p

CRITERIO 3: COMPLETITUD

[1]: Todas las soluciones

[2]: Algunas soluciones

[3]: 1 o ninguna solución

	Criterio 1	Criterio 2	Criterio 3	Total
Alternativa 1	1	5	1	5
Alternativa 2	1	5	2	10
Alternativa 3	1	8	2	16

De acuerdo con la tabla anterior, la mejor solución es la **ALTERNATIVA 3**, el algoritmo de Strassen, seguido de la estrategia de “divide y vencerás” que hace parte de su estructura y la multiplicación de matrices.

PASO 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES

De acuerdo con los pasos elaborados anteriormente se procederá a realizar el análisis de requerimientos y el diagrama de clases. Este documento será adjuntado en la próxima entrega.

PASO 7: IMPLEMENTACIÓN DEL DISEÑO

Una vez se realice la especificación de requerimientos y el diagrama de clases, se implementará la solución en lenguaje de programación Java.

BIBLIOGRAFÍA

RESOLUCIÓN DE PROBLEMAS

https://www.cs.upc.edu/~bejar/ia/transpas/teoria-n/2-BH1-introduccion_busqueda-n.pdf

ALGORITMO DE STRASSEN

https://es.wikipedia.org/wiki/Algoritmo_de_Strassen

ALGORITMO DE STRASSEN

https://alu0100881677.github.io/DAA_L2_1_Strassen/Strassen.html

TECNICA DIVIDE Y VENCERAS

http://dis.um.es/~ginesgm/files/doc/tema2-2.pdf