INFORME DE INGENIERIA

LABORATORIO UNIDAD 1

Presentado por:

Marisol Giraldo Cobo

Presentado al Profesor:

Andres Aristizábal

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*

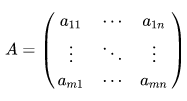
Son aquellas notaciones utilizadas para describir el tiempo de ejecución asintótico de un algoritmo. Se conoce la notación O para el pero caso, Ω para el mejor caso y θ para el caso promedio.

|  |  |
| --- | --- |
| Complejidad | Terminología |
| O(l) | Complejidad constante |
| O(log n) | Complejidad logarítmica |
| O(n) | Complejidad lineal |
| O(n log n) | Complejidad n log n |
| O() | Complejidad polinómica |
| O() | 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 (**aij)** 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 = (aij)m x n.** Si el número de filas y de columnas es igual (**m = n**), entonces se dice que la matriz es de **orden n**.

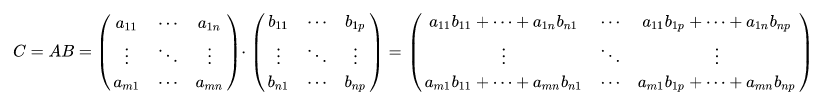


Sean *A*, *B* matrices y k escalar, la multiplicación de matrices se da de la siguiente manera:

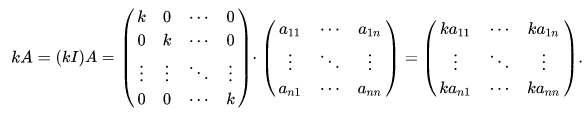
*Multiplicación de Matrices*

Dos matrices A y B son multiplicables si el **número de columnas de A** coincide con el **número de filas de B**. **Mm x n x Mn x p= M m x p**

El elemento**cij** 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**.

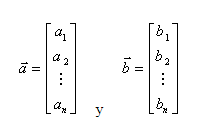


*Multiplicación de Matrices por un Escalar*



*Multiplicación Producto Punto*

Sea:



dos vectores del mismo número de componentes.

 El producto escalar de  http://oceanologia.ens.uabc.mx/~matematicas/algebralineal/II%20Matrices/promat_files/image006.gif  y  http://oceanologia.ens.uabc.mx/~matematicas/algebralineal/II%20Matrices/promat_files/image008.gif , denotado  http://oceanologia.ens.uabc.mx/~matematicas/algebralineal/II%20Matrices/promat_files/image010.gif , se define por:



*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().

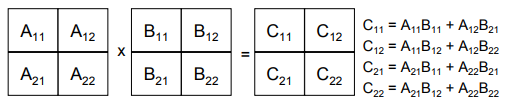
**ALTERNATIVA 2: TÉCNICA DE DISEÑO “DIVIDE Y VENCERÁS”**

* 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.



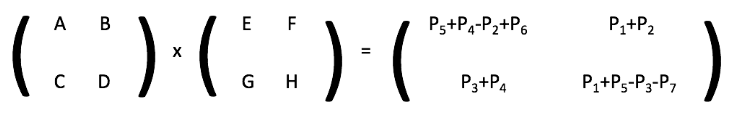
* Es necesario resolver 8 problemas de tamaño **n**/2.
* La combinación de los resultados requiere un O().

t(n) = 8·t(n/2) + a·.

* Resolviéndolo obtenemos que t(n) es O().
* 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.



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{\displaystyle \mathbf {C} \_{1,1}=\mathbf {M} \_{1}+\mathbf {M} \_{4}-\mathbf {M} \_{5}+\mathbf {M} \_{7}}

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](https://es.wikipedia.org/wiki/Algoritmo_de_Strassen#cite_note-1)​ y 60.000 o más para implementaciones básicas.

• El tiempo de ejecución será:

t(n) = 7·t(n/2) + a·

• Resolviéndolo, tenemos que:

t(n) ∈ O() ≈ O()

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

**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 Θ().

**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().

**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() ≈ O()

**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**

|  |  |
| --- | --- |
|  | Laboratorio I  Algoritmos y Estructuras de Datos  Requerimientos Funcionales |
| Nombre del Proyecto: | Venus Attack |
| Presentado por: | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | Andres Aristizábal |
| Fecha: | Agosto 25 del 2019 |

|  |  |
| --- | --- |
| Nombre | R1 – Generar una matriz aleatoria |
| Resumen | Generar aleatoriamente los valores de una matriz de una dimensión determinada e indicando si los números a generar deben ser todos diferentes o pueden haber repetidos. |
| Entradas | |
| - rows :Un valor entero que representa el número de filas de la matriz a generar.  - cols :Un valor entero que representa el número de columnas de la matriz a generar.  - repeatedNumbers :Un booleano que representa el valor de verdad acerca de si los números de la matriz están repetidos o no. | |
| Resultados | |
| Una matriz de dimensiones rows\*cols. | |

|  |  |
| --- | --- |
| Nombre | R2 – Multiplicación de Matrices |
| Resumen | Multiplicar la matriz A y la matriz B mediante de dimensiones correctas para realizar la operación seleccionando tres algoritmos de multiplicación de matrices: Algoritmo de Multiplicación Standard, Algoritmo de Multiplicación bajo la técnica de Dividir y Conquistar y el Algoritmo de Multiplicación Strassen . |
| Entradas | |
| - A : Una Matriz que representa las posiciones de las naves en las batallas pasadas de Marte.  - B : Una matriz que representa la matriz descubierta por el servicio secreto de Venus. | |
| Resultados | |
| C Una matriz que representa la posición actual de las tropas Marcianas según el Algoritmo escogido. | |

|  |  |
| --- | --- |
|  | Proyecto Final  Algoritmos y Estructuras de Datos  Pruebas Unitarias Automáticas |
| Nombre del Proyecto: | Venus Attack |
| Presentado por: | Marisol Giraldo Cobo – Código: A00246380 |
| Presentado a: | Andres Aristizábal |
| Fecha: | Agosto 25 del 2019 |

**CONFIGURACIÓN DE LOS ESCENARIOS**

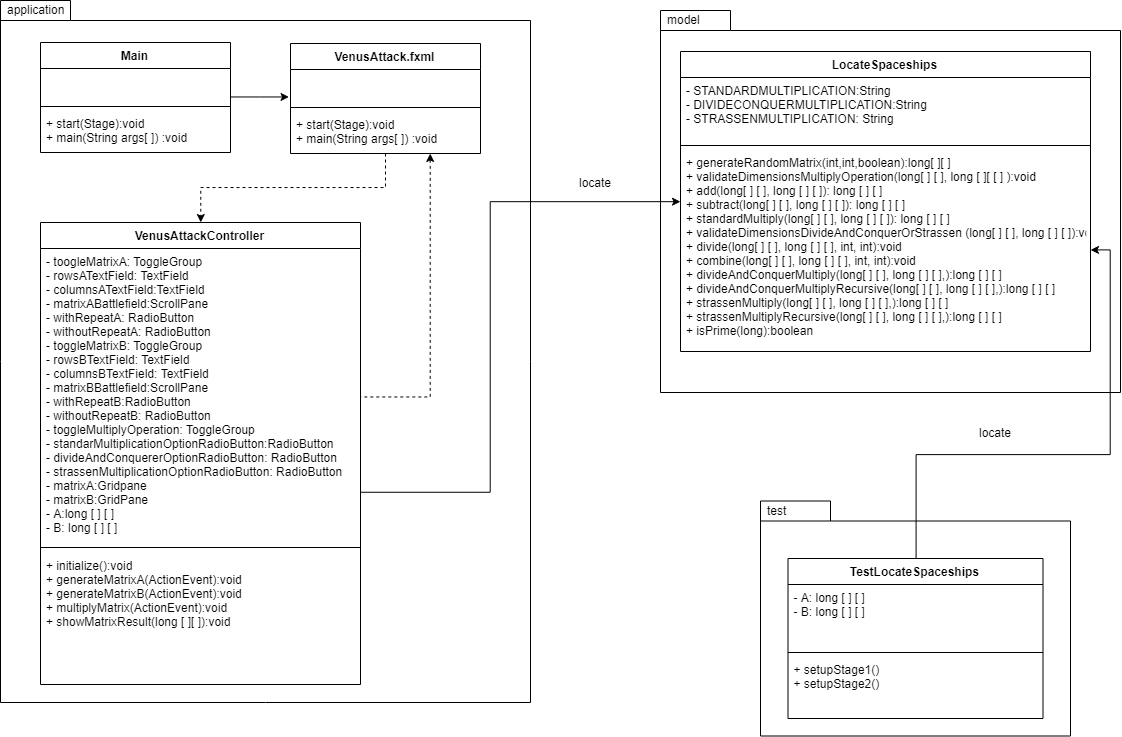
|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupStage2 | TestLocateSpaceships | : LocateSpaceships |
| setupStage2 | TestLocateSpaceships | : LocateSpaceships |

**DISEÑO CASOS DE PRUEBA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba 1** | | | | |
| **Objetivo de la Prueba:** Verificar que el método generar matriz aleatoria, genera de manera correcta una matriz con las dimensiones establecidas por el usuario. | | | | |
| **Clase** | **Método** | **Escenario** | **Entrada** | **Resultados** |
| LocateSpaceships | testGenerateRandomMatrix() | setupStage1 | rows= 8  columns=8  repeatNumbers= false | La prueba es exitosa. La matriz se genera de manera correcta. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba 2** | | | | |
| **Objetivo de la Prueba:** Verificar que el método multiplica matrices de manera correcta según el algoritmo de multiplicación Stándar. | | | | |
| **Clase** | **Método** | **Escenario** | **Entrada** | **Resultados** |
| LocateSpaceships | testStandardMultiply () | setupStage1 | A = (2 0)  1 3  B=(-1 -1)  5 6 | La prueba es exitosa. La multiplicación de matrices de manera Estándar funciona correctamente.  C= (-2 -2)  14 17 |

**PASO 7: IMPLEMENTACIÓN DEL DISEÑO**



**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>

#### [Algoritmo – Divide y conquista la multiplicación de matrices](https://codeday.me/es/qa/20190402/401543.html)

<https://codeday.me/es/qa/20190402/401543.html>