**REPÚBLICA DE CHILE**

**UNIVERSIDAD DEL BIO-BIO**

**FACULTAD DE CIENCIAS EMPRESARIALES**

**INGENIERÍA CIVIL EN INFORMÁTICA**

**Tarea 2**

**NOMBRES: Camila Martínez**

**Fredy Moncada**

**Alan Moreno**

**ASIGNATURA: Análisis y diseño de algoritmo.**

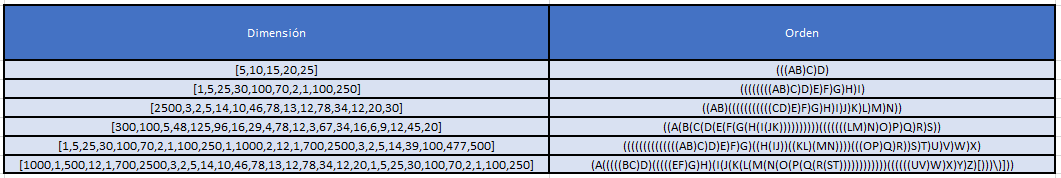
**PROFESOR: Gilberto Gutiérrez**

**Chillán, 8 de Noviembre del 2018**.

**Instrucciones para ejecutar los programas**

El archivo se entregará con el nombre de “matrices” en formato “.exe”, la ejecución es abriendo el archivo desde el escritorio de cualquier pc, este funcionará desde la ventana de comandos y entregará los resultados automáticamente.

**Resultados**





Complejidad

(Algoritmo de programación dinámica)

El problema entonces consiste en encontrar todas las parentizaciones posibles para M1, M2,..., Mn, evaluar la cantidad de productos necesarios, y obtener el menor entre todos ellos

La cantidad de parentizaciones posibles está deﬁnida por la recurrencia

T(n)

Con T (1) = 1

Los T(n) forman los llamados números de Catalán y se puede probar que por inducción

Luego el algoritmo directo toma tiempo de por lo que es inviable en la práctica para n mediano

El tiempo de ejecución, tomando como barómetro cualquiera de la sentencia del ciclo interno, es:

T(n) =

∈ Θ ()

Para obtener cuál es la mejor forma de multiplicar la matrices, es suﬁciente con recordar para cada (i, j) cuál es el k que determinó su menor valor.

Explicación

(Algoritmo de programación dinámica)

Programación Dinámica (PD) resuelve problemas a través de combinar soluciones a sub-problemas

PD comienza resolviendo las instancias más simples de los problemas y guardando sus resultados en alguna estructura de datos especial

Para construir soluciones de instancias más complejas, se divide la instancia en sub-problemas más simples y se recuperan los resultados ya calculados de la estructura de datos

PD se aplica cuando los sub-problemas no son independientes entre sí, es decir los sub-problemas tienen sub-sub-problemas en común. Esto se denomina superposición de sub-problemas

El problema de multiplicación de matrices en cadena satisface el principio de optimalidad y tiene también superposición de instancias

La función buscará la cantidad óptima de productos reales necesarios para multiplicar una secuencia de matrices, este valor depende de la cantidad de productos necesarios para multiplicar sub-secuencias de matrices

Existes 3 casos posibles para la resolución de este problema teniendo una matriz A y B (i: fila, j: columna)

1. Si i = j significa que solo tenemos una matriz existente, por lo tanto no hay que realizar ninguna multiplicación y el resultado dado es 0
2. Si j = i+1 tenemos que multiplicar el tamaño de la fila de A por el tamaño de la columna de B y a eso multiplicar el tamaño de la columna de A que tiene el mismo valor de la fila de B
3. El último caso es cuando j es mayor que i por almenos 2 valores (ej.: i:1 y j:3) en este caso se recurre a la asistencia de un arreglo con dimensiones de las matrices existentes, en resumen se usa la siguiente formula:

mij = min (mik + m(k+1)j + di-1dkdj)

Siendo:

min: El valor mínimo de los k posibles resultados entregados.

k: Valores extraídos mayor o igual que el índice i y estrictamente menor que j

i: El índice de la fila

j: El índice de la columna

d: Arreglo con las dimensiones de las matrices