PROGRAMACIÓN III INFORME DE LA RÁCTICA 3 (PROGRAMACIÓN DINÁMICA)

1. **Nombre de los participantes:**

- Marcos Jesús Santana Pérez
- Chetani Mesa Guzmán
- Ían Marrero Martín

2. **Descripción del problema a resolver:**

El problema consiste en calcular la cantidad mínima de operaciones necesarias para multiplicar una serie de matrices, para esto seguimos como criterio el tamaño de las matrices, por tanto ignoraremos el resultado y nos centraremos en cuantas operaciones van a realizarse.

3. **Lenguajes de programación elegidos:**

- Java v1.8.0_191
- Python v2.7
- C

4. **Enlace a la página web de *geeksforgeek* que describe el problema:**

   https://www.geeksforgeeks.org/matrix-chain-multiplication-dp-8/

5. **Estrategia implementada: *Memoization***
6. **Nombre de los ficheros que contienen el algoritmo implementado**

   Java:

   - P3/Java/src/P3_entrega_3/Memoization.java

   Python:

   - P3/Python/Memoization.java

   C:

   - P3/C/Memoization.java

7. **Copia de pantalla del fragmento de código que implementa el algoritmo**

   Python:

```
def MatrixChainOrder(i, j):

    key = str(i) + "," + str(j)

    if mem[key] < inf: return mem[key]
    else:
        for k in range(i, j):

            q = MatrixChainOrder(i, k) + MatrixChainOrder(k + 1, j) + mat[i - 1] *
mat[k] * mat[j]

            if q < mem[key]: mem[key] = q

    return mem[key]
```

Java:

```
  public long MatrixChainOrder(int i, int j){

    String key = i + "," + j;

    if(mem.get(key) < inf) return mem.get(key);

    else{

      for (int k = i; k < j ; k++) {

        long q = q = MatrixChainOrder(i, k) + MatrixChainOrder(k + 1, j) + matrix[i
- 1] * matrix[k] * matrix[j];

        if (q < mem.get(key)) mem.put(key,q);

      }
    }
    return mem.get(key);
  }
```

C:

```

```

8. **Formato del fichero de entrada del programa**

   El fichero contendrá cada elemento de un conjunto separado por una coma. Cada conjunto de elementos se pondrá en una línea nueva. **Formato:**

```
n1,n2.n3,n4
n1,n2,n3,n4,n5
```

   En la entrega adjuntamos una serie de ficheros para probar el código

9. **Copia de pantalla que muestre el uso del programa desde consola activando la opción que muestra el tiempo consumido en la ejecución del programa**

   En la entrega hemos adjuntado una serie de ficheros que contienen la salida que se obtiene al usar distintos ficheros.

   A continuación pantallazo de la salida al usar el archivo /ficheros/Contenido-Random/100-4000.txt

   Java:

   Python:

   C:

Información adicional

Ejecución:

Java:

javac -sourcepath src -d build src/**/*.java

java -cp .:build:**/*.class P3_entrega_3.Main

Añadimos los parámetros que queramos.

Python

python main.py

Añadimos los parámetros que queramos.

C

gcc *.c -o nombre_del_programa

./nombre_del_programa

Añadimos los parámetros que queramos.

Parámetros admitidos:

- **-f file**: Indica al programa el input del que se van a obtener los conjuntos de elementos.
- **-t:** Muestra el tiempo que tarda en resolverse el problema.
- **-do:** Muestra el resultado al resolver el problema.
- **-di:** Muestra el nombre del archivo y el contenido del mismo.
- **-mem:** Indica que el problema se va a resolver mediante memoization.
- **-tab:** Indica que el problema se va a resolver mediante tabulation.

> **Nota**: Si se utiliza -t o -do ha de indicarse -mem, -tab o ambas.
>
> Si se utiliza -f ha de indicarse a continuación el archivo.