Informe Algoritmo Voraz

## Jairo González, Alvaro López, Nicolas Rodrigo

# Especificación del problema

//**PRE**: longitud(n) > 2, n[i-1] x n[i] dimensiones de la matriz Mi, siendo las dimensiones de las matrices M1: n[0] x n[1], M2: n[1] x n[2]…, q contiene los elementos de n dando prioridad al máximo

**funcion** num\_multiplicaciones(n: Vector de enteros, q: Cola de Prioridad de Enteros) devuelve nm: entero

//**POST**: nm = número de operaciones totales para multiplicar las matrices de dimensiones n,

# Pseudocódigo

**Num\_multiplicaciones(**n, q**) retorna** nm:

dims\_size = longitud(n) + 1

nm = 0

**Mientras** dims\_size > 3 **hacer:**

Max\_dim = q.retiraMaximo()

Encontrado = Falso

**Para** j desde 1 hasta dims\_size – 1 **hacer:**

**Si** n[j] = max\_dim **y** j < dims\_size – 1 **entonces:**

nm = nm + dims[j-1] \* dims[j] \* dims[j+1]

encontrado = Cierto

dims\_size = dims\_size – 1

**FSi**

**Si** encontrado **entonces:**

dims[j] = dims[j+1]

**Fsi**

**FPara**

**FMientras**

nm = dims[0] \* dims[1] \* dims[2]

**retorna** nm

# Demostración de Optimalidad

No es posible demostrar la optimalidad del algoritmo voraz puesto que no es óptimo, ya que no siempre llega a la mejor solución. Sin embargo, para demostrar que el algoritmo no es óptimo, vale con una demostración experimental, ejecutando el código de la clase MainMix podemos ver que algunas veces el algoritmo implementado en el apartado 1 consigue mejores resultados que el algoritmo voraz, por lo que este segundo no es óptimo.

# Tabla de Experimentos

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 3 | 4 | 5 | 6 | 7 | 8 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |