# Análisis Tercer Punto

Para solucionar el primer punto inicialmente utilizo el algoritmo de ordenamiento Mergesort para obtener un arreglo ordenado de las fechas de finalización posteriormente obtengo un elemento y comparo si no se solapa con los elementos siguientes del arreglo para generar una lista con todos los elementos compatibles. Y de forma inversa obtengo todos los elementos antes de él que no se solapan.

Además cada vez que genero una lista de rodales compatible obtengo su beneficio en números de árboles que almaceno en una variable.

Para determinar cuál de las listas tiene el mayor número de beneficio me dirijo a la variable temporal y creo una condición que pregunta si la nueva lista tiene un beneficio mayor a la anterior si es **verdadero** remplazo la lista antigua si no la mantengo hasta que llegue una con mayor benéfico de arboles

# Los pasos que utilice para resolverlos

* Ordenamiento con MergeSort. 🡪 **O(nlgn)**
* For(recorrer arreglo de elementos) 🡪Compljida total **O(n2)**
  + funcionalidades.MaximoNumeroRodales 🡪 **O(n)**
  + funcionalidades.MaximoNumeroRodalesReverse 🡪 **O(n)**
* comparación para saber si es el mayor elemento.
* Comparación() 🡪 **O(n)**

De esta forma determino que la complejidad del algoritmo es de **O(n2)**

# La complejidad de esta solución

Es determinado por la complejidad del algoritmo de ordenamiento MergeSort la cual es de **O(nlgn).** Sumado a la complejidad de recorrer los elementos previamente organizados para su selección que obtenemos una complejidad de **O(n)** encada selección pero al realizarlo en un ciclo **for** se convierte en **O(n2)** elemento de esta forma determina que nuestra solución tiene una complejidad total de **O(n2).**

Uso de aplicativo

Es un proyecto de Netbeans para su ejecución solo es necesario construir el ejecutable e iniciar la ejecución. **”Triángulo de reproducción”**



Al ejecutarse despliega una pantalla para la selección del archivo que se va a tomar como entrada, la estructura del archivo de entrada cumple con las especificaciones del documento del proyecto.

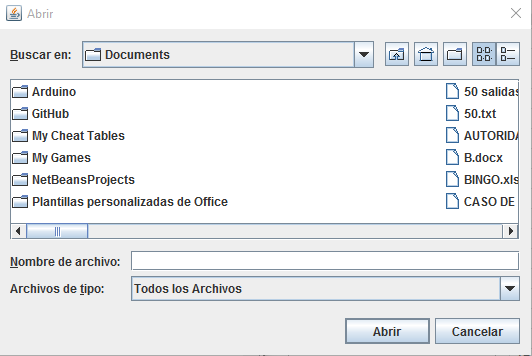


Ilustración 1

Posteriormente pide que se le indique un directorio para el almacenamiento de los resultados

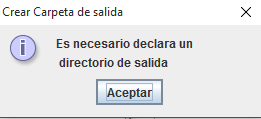


Ilustración 2

Se selecciona la ruta donde se va a realizar el almacenamiento de los resultados.

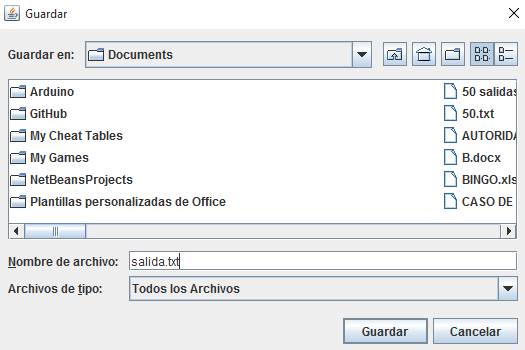


Ilustración 3

Al finalizar todos los procedimientos Retorna un mensaje de finalización.

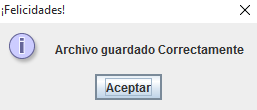


Ilustración 4

# Análisis de complejidad.

Se realizaron pruebas de rendimientos con múltiples entradas que fueron obtenidas con la construcción de un algoritmo de generación de entradas de forma aleatoria, Este aplicativo está disponible con el nombre **“Generador de prueba de entrada “**como proyecto de netBeans los archivos de prueba se encuentra en la carpeta” **PruebasInput”.**

# Tabla de rendimiento

|  |  |  |
| --- | --- | --- |
| **Nombre de Archivo** | **Numero de registro de entradas** | **Tiempo Milisegundos** |
| 50 registros | 50 | 532 |
| 150 registros | 150 | 672 |
| 500 registros | 500 | 5485 |
| 1600 registros | 1600 | 14891 |
| 2999 registros | 2999 | 33579 |
| 5200 registros | 5200 | 33157 |

Ilustración 5

Entorno de pruebas

Procesador: Amd E1-2500 1.40Ghz.

Memoria: 4GB Ddr3.

Disco duro: 500 Gb 5400 rpm.

Sistema Operativo: Windows 10 Pro 64bit.