# Análisis Segundo Punto

El Segundo punto del proyecto fue solucionado realizando el ordenamiento de las fechas de finalización de los rodales con el algoritmo Merge Sort.

Posteriormente se obtiene el primer elementó en finalizar y se hace una búsqueda hasta encontrar un segundo elemento que permitiera su tala de árboles sin solaparse con el rodal anterior, Al ya haber realizado el ordenamiento de las fechas de finalización se garantiza que el segundo elemento encontrado tiene una fecha de finalización menor al resto elementos que se encuentra en el arreglo después de el.

Esto se realiza hasta terminar con todos los rodales disponibles dando como resultado la tala del mayor número de rodales con sus respectivos tiempos..

# Pasos utilizados:

* Ordenamiento con MergeSort. 🡪 **O(n.lgn)**
* Recorrido del arreglo encontrando el siguiente 🡪 **O(n)**

De esta forma determino que la complejidad del algoritmo es de **O(n.lgn)**

# 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 n de esta forma determina que nuestra solución tiene una complejidad total de **O(nlgn).**

Uso de aplicativo

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.

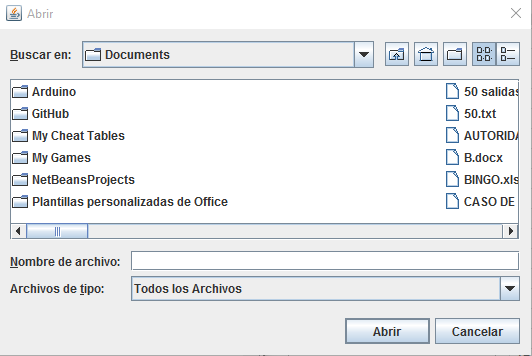


Ilustración 1

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

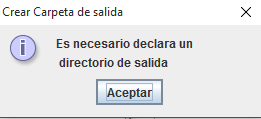


Ilustración 2

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

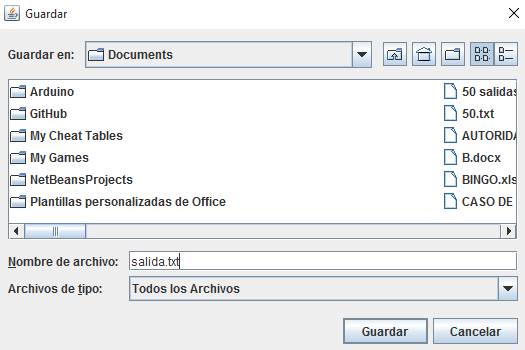


Ilustración 3

Al finalizar todos los procedimientos Retorna un mensaje de finalización.

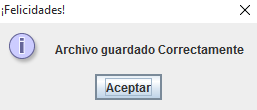


Ilustración 4

# 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** | **Numero de Iteraciones** |
| 50 registros | 50 | 54 | 85 |
| 150 registros | 150 | 41 | 1097 |
| 500 registros | 500 | 88 | 3107 |
| 1600 registros | 1600 | 172 | 11804 |
| 2999 registros | 2999 | 215 | 24010 |
| 5200 registros | 5200 | 666 | 44493 |

Ilustración 1

No. De Registros

No. De Iteraciones

Ilustración 2

Entorno de pruebas

Procesador: Amd E1-2500 1.40Ghz.

Memoria: 4GB Ddr3.

Disco duro: 500 Gb 5400 rpm.

Sistema Operativo: Windows 10 Pro 64bit.