**LABORATORIO NO. 2: LISTAS Y ORDENAMIENTOS BÁSICOS**

Integrantes grupo 6:

* Sebastian Heredia Vargas | [s.herediav@uniandes.edu.co](mailto:s.herediav@uniandes.edu.co) | 202012346
* Andres Felipe Mendoza Silva | [af.mendoza@uniandes.edu.co](mailto:af.mendoza@uniandes.edu.co) | 202012676
* Felix Samuel Rojas Casadiego | [fs.rojas@uniandes.edu.co](mailto:fs.rojas@uniandes.edu.co) | 202013414

|  |  |  |
| --- | --- | --- |
| Fuente de datos | Arraylist [s] | Singlelinkedlist [s] |
| Películas (MoviesDetailsCleaned) | 9.959375 | 10.746875 |
| Elenco (MoviesCastingRaw) | 5.778125 | 6.05625 |

* Tabla 1. Tabla de rendimiento para cargar archivos como lista enlazada y arreglo.

|  |  |  |
| --- | --- | --- |
| Tiempo (s) | Elenco | Películas |
| t1 | 6.015625 | 11.28125 |
| t2 | 6.078125 | 10.328125 |
| t3 | 6.046875 | 10.71875 |
| t4 | 5.96875 | 11.171875 |
| t5 | 6.171875 | 10.234375 |
| Promedio | 6.05625 | 10.746875 |

Tabla 2. Tabla de rendimiento para cargar archivos con lista enlazada

|  |  |  |
| --- | --- | --- |
| Tiempos | Elenco | Películas |
| t1 | 5.890625 | 10.953125 |
| t2 | 5.671875 | 9.59375 |
| t3 | 5.65625 | 9.953125 |
| t4 | 6.109375 | 9.765625 |
| t5 | 5.5625 | 9.53125 |
| Promedio | 5.778125 | 9.959375 |

Tabla 3. Tabla de rendimiento para cargar archivos con arraylist

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Shellsort | |  |  | |
| Tiempos | | Arraylist | SingleLinked | |
| t1 | | 0.125 | 0.206775 | |
| t2 | | 0.109375 | 0.18092813 |
| t3 | 0.125 | | 0.206775 | |
| t4 | | 0.15625 | 0.25846875 | |
| t5 | | 0.127 | 0.2100834 | |
| Promedio | | 0.128525 | 0.21260606 | |
| |  |  |  | | --- | --- | --- | | Selectionsort | |  | | Tiempos | Arraylist | SingleLinked | | t1 | 7.4375 | 9.83573675 | | t2 | 7.03125 | 9.29849063 | | t3 | 7.609375 | 10.0630332 | | t4 | 7.875 | 10.4143095 | | t5 | 7.765625 | 10.2696663 | | Promedio | 7.54375 | 9.97624728 |  |  |  |  | | --- | --- | --- | | Insertionsort | |  | | Tiempos | Arraylist | SingleLinked | | t1 | 3.75 | 6.95146125 | | t2 | 3.546875 | 6.57492377 | | t3 | 3.734375 | 6.92249683 | | t4 | 3.765625 | 6.98042567 | | t5 | 3.796875 | 7.03835452 | | Promedio | 3.71875 | 6.89353241 | | |  |  | |

(Tuvimos que hacer los test con los archivos small, los archivos large demoraban mucho. Los test de arriba con arraylist y singlelinked list si están hechos con archivos -large)

|  |  |  |
| --- | --- | --- |
| Algoritmo | Arraylist [s] | Singlelinkedlist [s] |
| SelectionSort | 7.54375 | 9.97624728 |
| InsertionSort | 3.71875 | 6.89353241 |
| ShellSort | 0.128525 | 0.21260606 |

Tabla 4. Tabla de rendimiento para los distintos algoritmos de ordenamiento en listas enlazadas y arreglos.

## Analizar los resultados de listas y ordenamientos básicos:

Con los resultados de la Tabla 1. y Tabla 2 reflexione y responda las siguientes preguntas con respecto a las estructuras y algoritmos.

* **¿Qué diferencias se observan en desempeño de la carga de datos entre arreglo (*Arraylist*) y lista sencillamente encadenada (*Singlelinkedlist*)?**

La variable por diferenciar entre las dos listas es el tiempo que demoran en cargar los datos. Las dos listas tienen tiempos diferentes debido a los procesos realizados al añadir elementos.

* **¿Cuál de las dos implementaciones (*Arraylist* y *Singlelinkedlist*) tiene mejor desempeño? Y ¿Por qué?**

La implementación del ArrayList tiene un mejor desempeño que el SingleLinkedList, esto es debido a la manera de añadir elementos al final de una lista. El primero tiene una complejidad de o(1) y el segundo una complejidad de o(n).

* **¿Qué diferencias existen entre cargar los archivos de películas (*MoviesDetailsCleaned*) y elenco (*MoviesCastingRaw*)?, ¿Por qué se presentan estas diferencias?**

Al cargar los archivos de películas (MoviesDetailsCleaned) se demora más tiempo de procesamiento, en comparación con los archivos del elenco (MoviesCastingRaw). Estos cambios se dan ya que a pesar de tener la misma cantidad de películas la cantidad de datos por elemento es diferente.

* **¿Qué diferencias en el desempeño se observan entre los tres algoritmos de ordenamiento?**

**ShellSort:** Este algoritmo de ordenamiento es usado muy ocasionalmente para problemas concretos con buenos tiempos. Debido a esto los cambios no varían significativamente.

**SelectionSort:** Este algoritmo a pesar de su simplicidad de implementación es el que tiene los tiempos más altos. Los cambios de tiempo varían escasamente entre el ArrayList y el SingleLinkedList.

**InsertionSort:** Los tiempos de este algoritmo son más bajos que de SelectionSort, y notamos que los cambios de tiempo varían significativamente entre el ArrayList y el SingleLinkedList.

* **¿Qué efectos tienen los dos tipos de lista en los tres algoritmos de ordenamiento?**

El recorrido de la lista en el ArrayList permite mantener mejores tiempos en todos los casos de ejecución, en comparación con el SingleLinkedList.