**DISEÑO DE EXPERIMENTO**

1. **Análisis y Realización**

**1.1. Resumen**

En una estructura de datos, el ordenamiento de elementos suele ser un problema el cual se ha llegado a estudiar de una manera estricta en la forma de cómo manejar esta problemática. Hasta la actualidad, se han desarrollado varios algoritmos donde cada uno funciona mejor para una situación adecuada. Para este trabajo, se escogió el algoritmo *selectionSort* Y *insertionSort* y se identificará cuál de estos dos algoritmos es más eficientes para los experimentos que se realizarán donde se comparará su eficiencia.

**1.2. Introducción**

*InsertionSort* es una algoritmo el cual itera sobre las posiciones del arreglo,donde se obtiene un valor como clave el cual se comparará con el resto de valores en el arreglo, insertandolo en el lugar donde corresponde. Este algoritmo tiene una complejidad computacional para el mejor caso O(n) y tanto para el caso promedio como para el peor caso, su complejidad es de O(n^2).

En cuanto a *SelectionSort,* consiste en encontrar el dato menor de todos los elementos que se encuentran en el arreglo y cambiarlo con el que se encuentra en la primer posición hasta ordenar todo el contenedor. Su complejidad computacional es igual al de InsertionSort para el mejor de los casos (O(n)) y para el peor de los casos es O(n^2).

**1.3 Análisis y Caracterización**

Primero, se procede a hacer un análisis de la complejidad temporal del algoritmo *SelectionSort* y del algoritmo *InsertionSort,* y de cómo son sus comportamientos en el mejor y peor de los casos, utilizando técnicas de conteo de líneas para algoritmos iterativos.

Segundo, y, después de los estudios del paso anterior, se establece nuestra hipótesis de cómo se espera que se comporte cada algoritmo de ordenamiento, teniendo en cuenta su tamaño de entrada para determinar el mejor método de acuerdo a los factores que influyen en su comportamiento.

Tercero, se realiza la ejecución de las pruebas para cada tamaño de entrada, repitiendolas 1000 veces. Después, se toma el tiempo de ejecución y se promedia en cada variante de la prueba.

Cuarto, y, con los datos recolectados y los promedios calculados, se procede a evaluar la hipótesis utilizando la prueba para comparar la media de 3 o más poblaciones ANOVA.

**1.4 Marco Procedimental**

**1.4.1. Materiales**

* Visual Studio 2019.
* Computador.

**1.4.2. Procedimiento**

Se hallarán los resultados de las ecuaciones de la complejidad espacial de los algoritmos elegidos, los cuales describen el comportamiento de dichos algoritmos en relación tiempo-tamaño acompañado con su notación asintótica.

Haciendo uso del entorno de desarrollo de visual studio, se realizará la implementación de los algoritmos de ordenamiento *InsertionSort* y *SelectionSort,* creándose pruebas unitarias automatizadas para verificar su correcta ejecución para los test del experimento.

Se establecerán los escenarios para las pruebas unitarias creando varias entradas, para los algoritmos de ordenamiento, de tamaños que van desde 10^1 hasta 10^5 junto con tres variaciones en cómo están organizadas dichas entradas: orden ascendente, orden descendente y desordenada.

**1.5. Unidad Experimental, Variables y Factores**

**1.5.1. Unidad Experimental**

Durante el experimento, se estarán realizando varias pruebas a los algoritmos de ordenamiento (InsertionSort y SelectionSort) con la utilización de arreglos, los cuales varían en tamaño que van desde 10^1 hasta 10^5 datos. Lo anterior, con el fin de medir el tiempo en que los métodos se demoran ordenando los arreglos anteriormente mencionados. Por lo tanto, se establecen a los algoritmos de ordenamiento como la unidad experimental.

**1.5.2. Variables**

**1.5.2.1. Variables de respuesta**

La complejidad de cada algoritmo de ordenamiento, que determina cuál tiene mejor o más eficiencia debido a que determina quien tiene menor tiempo de ejecución, nos indicó, para nuestro problema, cual seria mejor. Por lo tanto, la variable de respuesta sería la complejidad algorítmica mencionada anteriormente.

**1.5.3. Factores**

**1.5.3.1. Factores controlables**

Teniendo en cuenta el experimento a realizar, se definieron dos factores controlables:

* El tamaño del arreglo el cual, como se ha dicho anteriormente, variará desde 10^1 hasta 10^5 datos.
* Estado de los valores en el arreglo (ordenados ascendentemente, descendentemente o aleatorios).
* Los algoritmo de ordenamiento que se utilizarán durante las pruebas del experimento.
* Tiempo de ejecución de los algoritmos de ordenamiento, puesto que siempre se buscará que tarde el menor tiempo posible.

**1.5.3.2. Factores no controlables**

Según el alcance del experimento y, de su objetivo, se han identificado dos factores no controlables:

* Las especificaciones del o de los equipos donde se realizarán las pruebas. Lo anterior, puesto que cada computador varía en sus características internas (procesador, memoria RAM, velocidad de lectura y escritura del disco duro, tamaño del disco duro etc).
* Temperatura del ambiente, puesto que cada equipo de trabajo se ve influenciado internamente por las temperaturas que el mismo equipo produce o de las que lo rodean, haciendo que sus componentes varíen en cuanto a potencia se refiere.

**1.5.3.3. Factores estudiados**

* El tamaño del arreglo (de 10^1 hasta 10^5 datos).
* Estado de los valores en el arreglo (ordenados ascendentemente, descendentemente o aleatorios).

**1.5.4. Niveles y Tratamientos**

|  |  |  |  |
| --- | --- | --- | --- |
| No. de prueba | Algoritmo | Estado del arreglo | Tratamiento |
| 1 | Selection | Random | 1 |
| 2 | Selection | Ascendente | 2 |
| 3 | Selection | Descendente | 3 |
| 4 | Insertion | Random | 4 |
| 5 | Insertion | Ascendente | 5 |
| 6 | Insertion | Descendente | 6 |

***Tabla 1. Niveles y tratamientos.*** *Muestra los factores de entrada para cada tratamiento*

**1.5.5. Análisis de varianza de un Factor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resumen |  |  |  |  |
| Grupos | Cuenta | Suma | Promedio | Varianza |
| 1 | 5 | 75667,9692 | 3133,59384 | 46919424,59 |
| 2 | 5 | 16056,0394 | 3211,20788 | 49103314,38 |
| 3 | 5 | 15921,9531 | 3184,39062 | 48417289,2 |
| 4 | 5 | 12018,244 | 2409,6488 | 27879158,05 |
| 5 | 5 | 0,6538 | 0.13076 | 0,060152988 |
| 6 | 5 | 22071,163 | 4414,2326 | 93076040,85 |

***Tabla 2. Datos de las medias de los tratamientos***.

**1.5.6. Análisis de ANOVA**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Método** | **Tiempos** | | | | | | |
|  |  | **10^1** | **10^2** | **10^3** | **10^4** | **10^5** |  |
| **Selection** | Random | 0,2645 | 0,0224 | 2,6443 | 280,101 | 15384,937 | ms |
|  | Ascendente | 0,0011 | 0,1395 | 2,4114 | 309,3833 | 15744,104 | ms |
|  | Descendente | 0,0014 | 0,0271 | 3,1506 | 289,0812 | 15629,693 | ms |
| **Insertion** | Random | 0,2899 | 0,0136 | 2,2606 | 167,6352 | 11848,045 | ms |
|  | Ascendente | 0,0011 | 0,0019 | 0,0089 | 0,076 | 0,5659 | ms |
|  | Descendente | 0,0032 | 0,0318 | 5,4404 | 396,0187 | 21669,669 | ms |

***Tabla 3. Análisis de la varianza con la ANOVA***

**Método Selection Sort**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Selection |  |
|  | Random | Ascendente | Descendente |
| 10^1 | 0,2645 | 0,0011 | 0,0014 |
| 10^2 | 0,0224 | 0,1395 | 0,0271 |
| 10^3 | 2,6443 | 2,4114 | 3,1506 |
| 10^4 | 280,101 | 309,3833 | 289,0812 |
| 10^5 | 15384,937 | 15744,1041 | 15629,6928 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESUMEN |  |  |  |  |
| *Grupos* | *Cuenta* | *Suma* | *Promedio* | *Varianza* |
| 10^1 | 3 | 0,267 | 0,089 | 0,02310021 |
| 10^2 | 3 | 0,189 | 0,063 | 0,00439471 |
| 10^3 | 3 | 8,2063 | 2,735433333 | 0,142833123 |
| 10^4 | 3 | 878,5655 | 292,8551667 | 225,0453916 |
| 10^5 | 3 | 46758,7339 | 15586,24463 | 33666,05882 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ANÁLISIS DE VARIANZA | |  |  |  |  |  |
| *Origen de las*  *variaciones* | *Suma de cuadrados* | *Grados de libertad* | *Promedio de los cuadrados* | *F* | *Probabilidad* | *Valor crítico para F* |
| Entre grupos | 577707869,2 | 4 | 144426967,3 | 21307,39685 | 1,33E-19 | 3,478049691 |
| Dentro de los grupos | 67782,54908 | 10 | 6778,254908 |  |  |  |
|  |  |  |  |  |  |  |
| Total | 577775651,7 | 14 |  |  |  |  |

**Interpretación**

In valor de F de 21307,4 nos indica que hay una fuerte variación entre las medias de cada tamaño del arreglo ordenado con respeto a las formas en los que se encuentran los datos a ordenar con el método de ordenamiento Selection es muy marcada, es decir, el método no resulta ser muy eficiente, dado que a medida que se incrementa el tamaño del arreglo más tiempo le toma ordenarlo.

**Método InsertionSort**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Insertion |  |
|  | Random | Ascendente | Descendente |
| 10^1 | 0,2899 | 0,0011 | 0,0032 |
| 10^2 | 0,0136 | 0,0019 | 0,0318 |
| 10^3 | 2,2606 | 0,0089 | 5,4404 |
| 10^4 | 167,6352 | 0,076 | 396,0187 |
| 10^5 | 11848,0447 | 0,5659 | 21669,6689 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RESUMEN |  |  |  |  |
| *Grupos* | *Cuenta* | *Suma* | *Promedio* | *Varianza* |
| 10^1 | 3 | 0,2942 | 0,098066667 | 0,027601123 |
| 10^2 | 3 | 0,0473 | 0,015766667 | 0,000227023 |
| 10^3 | 3 | 7,7099 | 2,569966667 | 7,447078863 |
| 10^4 | 3 | 563,7299 | 187,9099667 | 39500,95504 |
| 10^5 | 3 | 33518,2795 | 11172,75983 | 117729513,4 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ANÁLISIS DE VARIANZA |  |  |  |  |  |  |
| *Origen de las variaciones* | *Suma de cuadrados* | *Grados de libertad* | *Promedio de los cuadrados* | *F* | *Probabilidad* | *Valor crítico para F* |
| Entre grupos | 297122154,2 | 4 | 74280538,54 | 3,15365354 | 0,064059466 | 3,478049691 |
| Dentro de los grupos | 235538043,7 | 10 | 23553804,37 |  |  |  |
|  |  |  |  |  |  |  |
| Total | 532660197,9 | 14 |  |  |  |  |

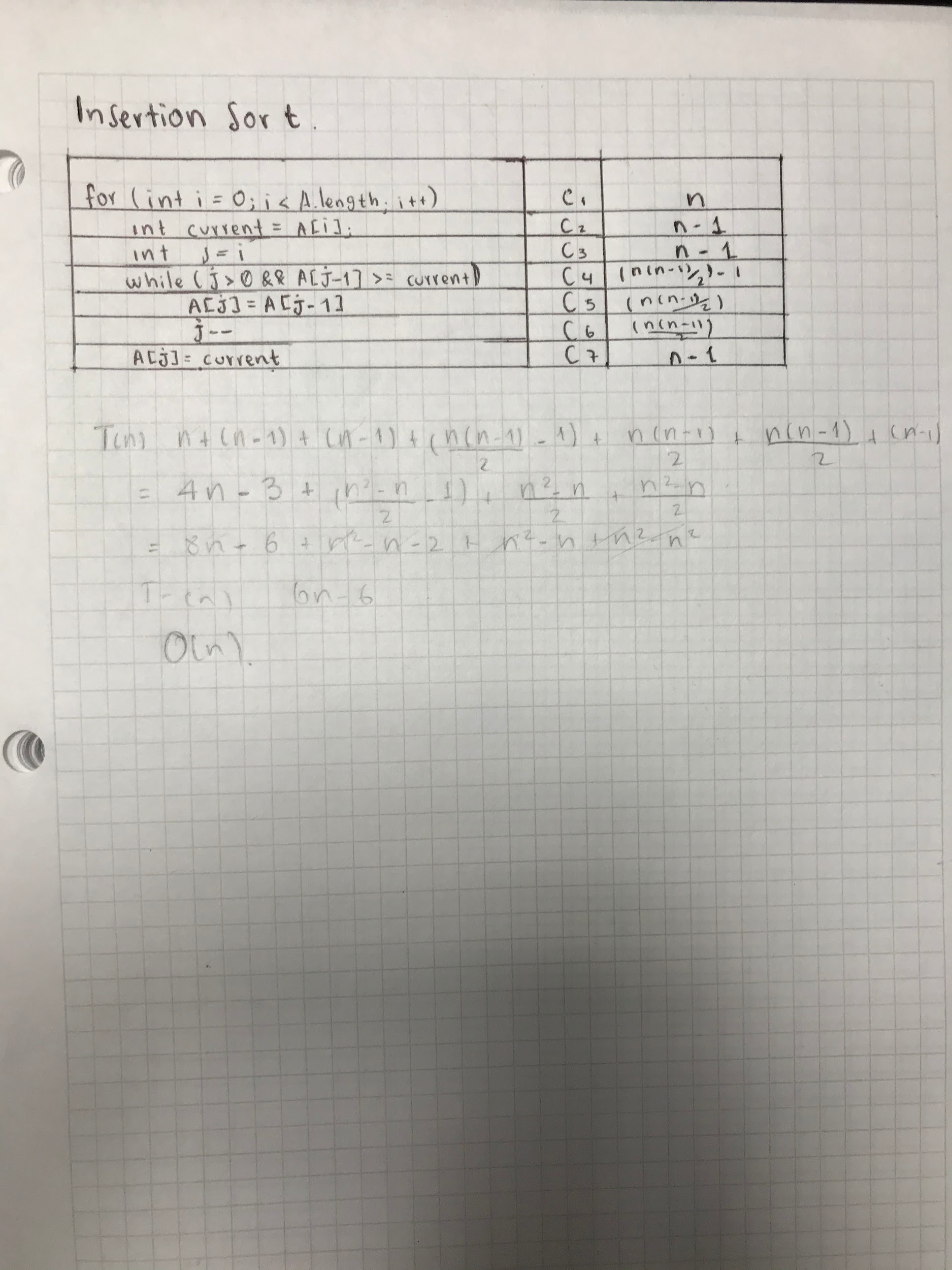
**Interpretación**

El valor de F de 3,2 nos indica que la variación entre las medias de cada tamaño del arreglo ordenado con respecto a las formas en los que se encuentran los datos a ordenar con el método de ordenamiento Selection no es tan marcada, es decir, el método resulta ser categóricamente eficiente sin importar el tamaño del arreglo a ordenar.

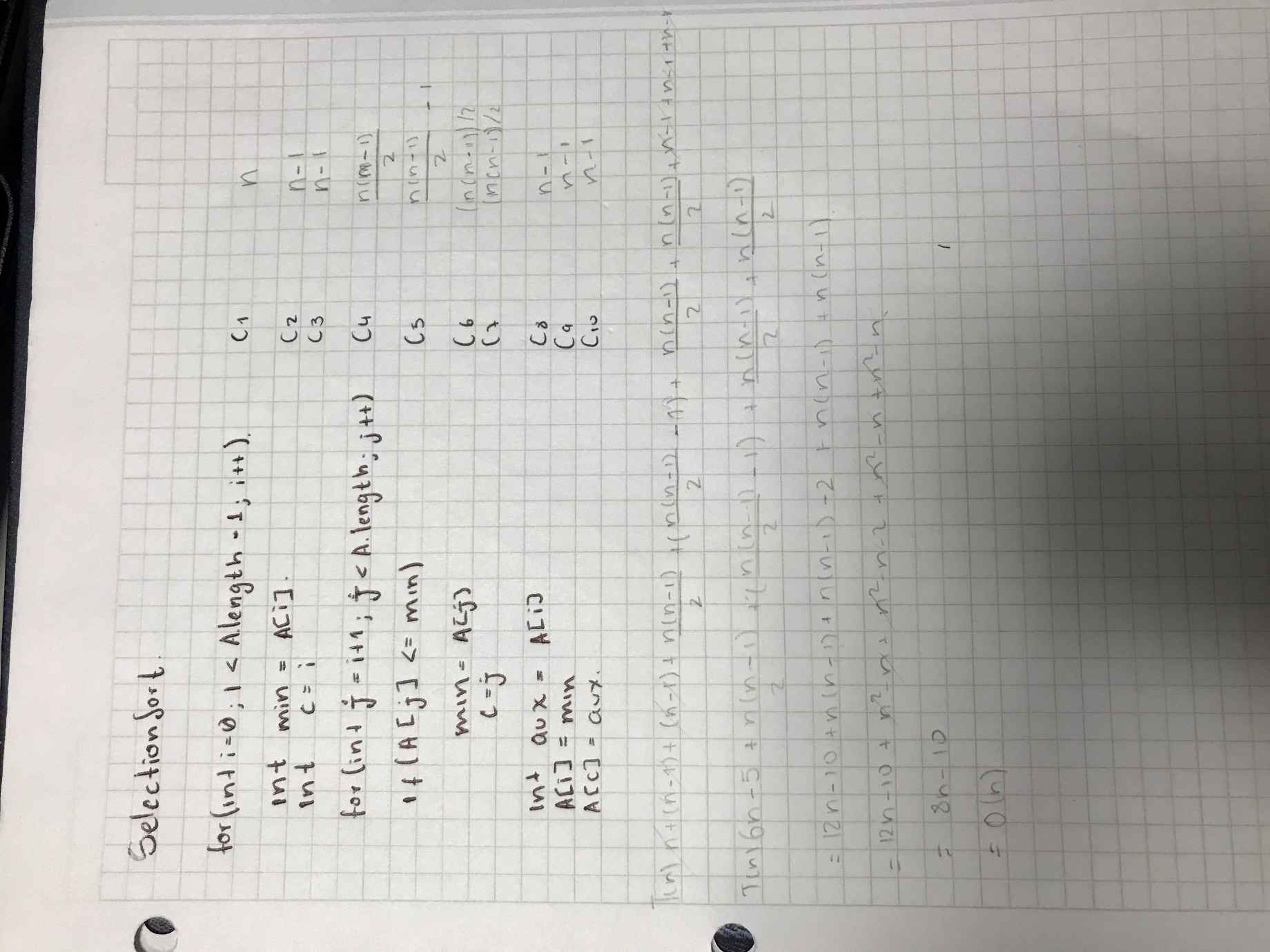
**Conclusión**

A partir de los resultados obtenidos anteriormente se puede concluir que ambos métodos son útiles para ordenar una colección de datos, sin embargo, el método InsertionSort resulta ser más eficiente que el método SelectionSort. Y esto dependiendo del tamaño de cada arreglo

a ordenar.

****

La complejidad del método *insertionSort* es O(n) para el mejor de los casos.



Complejidad temporal de *selectionSort* es O(n) para el mejor de los casos

**Medición del Tiempo**

**Elías Estupiñan**

https://wakatime.com/@0a4f4c88-dd7b-4556-bc42-067cc8e8c07d/projects/qtmjwedvzp?start=2020-02-11&end=2020-02-17