**Planeación y Realización**

**Objeto de estudio:**

El objeto de estudio en este experimento, va a ser el tiempo de ORDENAMIENTO en arreglos de elementos numéricos haciendo uso de algoritmos de ordenamiento.

**Delimitación del experimento:**

Al analizar el objeto de estudio del experimento, nos damos cuenta de que hay distintos tipos de algoritmos en la actualidad que nos permiten ordenar una secuencia de valores, procedemos entonces a definir qué es un algoritmo de ordenamiento.

- Se denomina algoritmo de ordenamiento a un algoritmo que al aplicarse sobre un conjunto de elementos A= { a1, a2, a3… an } con una propiedad de orden, genera una permutación del conjunto A tal que los elementos ahora cumplen con la propiedad de orden { a1 ≤ a2 ≤ a3… ≤ an } o en su defecto {a1≥ a2≥ a3…≥an}. (Thomas H. Cormen, 2009)

Ahora que sabemos qué es un algoritmo de ordenamiento, tenemos que saber también que existen distintos tipos de algoritmos para ordenar, pero en este experimento nos centraremos solamente en dos:

* Burbuja
* QuickSort

Cabe recalcar que la complejidad temporal de ambos algoritmos es la misma para el peor caso y es O(n^2).

Se procede entonces a definir qué es la complejidad temporal de un algoritmo:

* Se denomina complejidad temporal a la función T(n) que mide el número de instrucciones realizadas por el algoritmo para procesar los n elementos de entrada. (Bijit, 2010)

Se procede ahora a definir la secuencia que sigue el algoritmo QuickSort para su posterior análisis espacial y temporal:

El algoritmo consta de los siguientes pasos:

1. Elegir un elemento de la lista de elementos a ordenar, al que llamaremos pivote.

2. Resituar los demás elementos de la lista a cada lado del pivote, de manera que a un lado queden todos los menores que él, y al otro los mayores. Los elementos iguales al pivote pueden ser colocados tanto a su derecha como a su izquierda, dependiendo de la implementación deseada. En este momento, el pivote ocupa exactamente el lugar que le corresponderá en la lista ordenada.

3. La lista queda separada en dos sublistas, una formada por los elementos a la izquierda del pivote, y otra por los elementos a su derecha.

4. Repetir este proceso de forma recursiva para cada sublista mientras éstas contengan más de un elemento. Una vez terminado este proceso todos los elementos estarán ordenados. (contributors, QuickSort, 2019)

Luego se define la secuencia que sigue el algoritmo Burbuja para su posterior análisis:

Este algoritmo funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiandolos de posición si están en el orden equivocado. Es necesario revisar varias veces toda la lista hasta que no se necesiten más intercambios, lo cual significa que la lista está ordenada (contributors, Ordenamiento de burbuja, 2019).

**Unidad experimental:**

Dicho esto, se puede definir la unidad experimental como el arreglo de elementos numéricos.

**Factores controlables:**

Ahora bien, ya que se definió y se delimitó bien cuál va a ser el objeto de estudio, se tendrán en cuenta las siguientes variables controlables para el seguimiento del experimento:

- El algoritmo de ordenamiento (QuickSort, Burbuja).

- El tamaño del arreglo (10^2,10^4,10^8).

- La cantidad de memoria RAM del computador donde se ejecuta el Algoritmo(4GB,8GB,12GB).

- El estado inicial de los valores en el arreglo (orden ascendente, descendente o aleatorio).

- Número de procesos activos en el computador donde se ejecuta.

- Sistema operativo del computador donde se ejecuta.

- Lenguaje de programación usado para implementar los algoritmos.

- Nivel de fragmentación del disco duro donde se ejecuta.

**Factores no controlables:**

Así como se tiene factores controlables en el experimento, y se puede poner su disposición al antojo, se tiene una serie de variables que no son tan fáciles de cambiar y son las siguientes:

- Compilador con el cual vamos a traducir el algoritmo a lenguaje máquina.

- Procesador de la máquina donde vamos a llevar a cabo el experimento.

- Humedad del equipo donde se va a realizar el experimento.

- Temperatura del equipo donde se va a realizar el experimento.

**Factores estudiados:**

Aunque se tiene muchos factores controlables y no controlables para el experimento, en esta ocasión solo se va a centrar en algunos porque se considera, son los de mayor importancia a la hora de cambiar el resultado del experimento y son los siguientes:

- El algoritmo de ordenamiento.

- El tamaño del arreglo a ordenar.

- Cantidad de memoria RAM donde se ejecuta el algoritmo.

- Estado inicial de los valores en el arreglo a ordenar.

**Niveles:**

Igualmente, se tendrán en cuenta los siguientes niveles para cada factor:

Para el algoritmo de ordenamiento se tendrán dos opciones: QuickSort o Burbuja.

El tamaño del arreglo podrá ser de 10^2, 10^4 o 10^6.

La cantidad de memoria RAM podrá ser de 4GB, 8GB o 12GB.

Por último, el estado inicial de los valores en el arreglo podrá ser de 3 formas, ascendente, descendente u orden aleatorio.

**Tratamientos:**

Para este experimento, se tendrá en cuenta todas las posibles combinaciones de los distintos niveles de cada factor, es decir, en la práctica se llevarán a cabo todas las combinaciones de cada nivel y se harán por lo menos 10 ejecuciones para cada tratamiento.

**Variables de respuesta:**

En este experimento se centró en una única variable de respuesta, la cual es el tiempo, dado en los milisegundos que tarda la ejecución de un tratamiento

**Análisis**

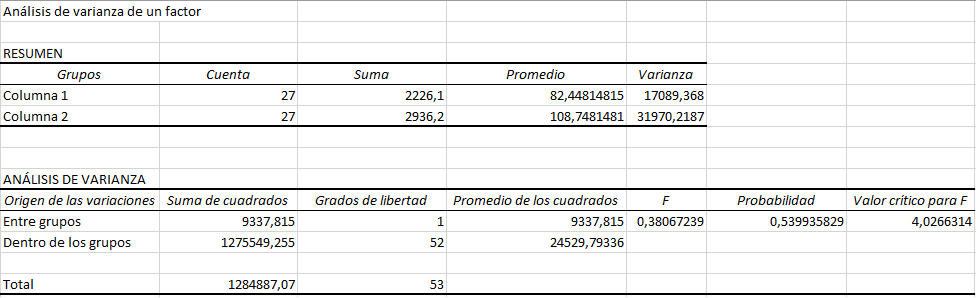
Un análisis requiere un proceso arduo para poder emprender un tema específico, mediante el uso del diseño experimental se arroja unos valores los cuales se pueden utilizar para referirse en un sentido propio hacía el objetivo, en base de los resultados obtenidos; además se muestra detalladamente el progreso de la etapa experimental asimilando las ideas y los resultados, por lo que se genera posteriormente un campo a la interpretación concisa hacía las hipótesis planteadas. Por consiguiente, se usó un análisis estadístico llamado ANOVA (**AN**alysis **O**f **VA**riance) lo cual consiste en el análisis de las varianzas de los factores incluidos, para así generar una hipótesis evaluando un valor denominado F, que se calcula con la varianza del factor frente a un valor crítico que se calcula mediante la distribución de F.

Cabe recalcar que por cada factor se genera una hipótesis, pero el proceso estadístico se encarga de definir si verdaderamente la hipótesis planteada se puede reafirmar o descartar, por eso se tiene en cuenta las siguientes condiciones:

* Si el valor F es mayor que el valor crítico, se dice que la hipótesis es falsa, debido a un proceso estadístico.
* Si el valor F es menor que el valor crítico, se dice que la hipótesis es verdadera, debido a un proceso estadístico.

Como primera instancia se tomó el factor del algoritmo de ordenamiento, el cual mediante la experimentación arrojó diferentes datos, descrito en la siguiente tabla:

*ANOVA: Algoritmos de ordenamiento. tabla 1.0.*

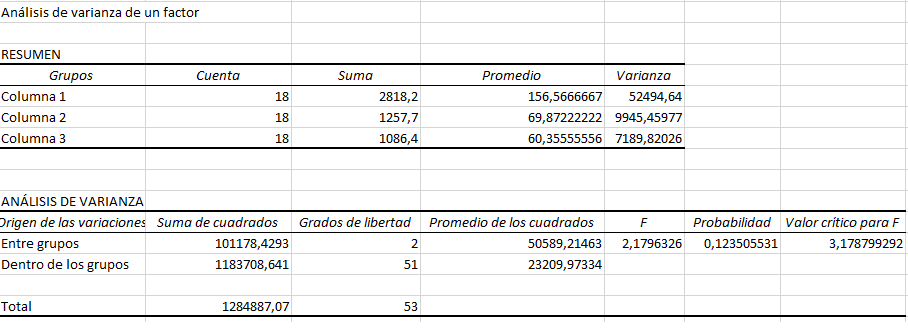


En contextualización, la tabla 1.0 es un diseño resumido que ofrece el sistema ANOVA para inducir los datos tomados en el proceso experimental.

Ahora bien, se tiene en cuenta que la columna uno y dos consta de su promedio y varianza dados en tiempo de ejecución, recalcando que la columna uno está dado por el algoritmo de Quick y la columna dos por el algoritmo de Burbuja.

Es claro y conciso que el promedio y la varianza del algoritmo Quick es mucho menor que el del algoritmo Burbuja. Por consiguiente, se puede afirmar el algoritmo QuickSort es más eficiente que el algoritmo de Burbuja. No obstante, el factor presenta un valor crítico mayor a su valor F, por tanto el proceso estadístico apoya la anterior hipótesis.

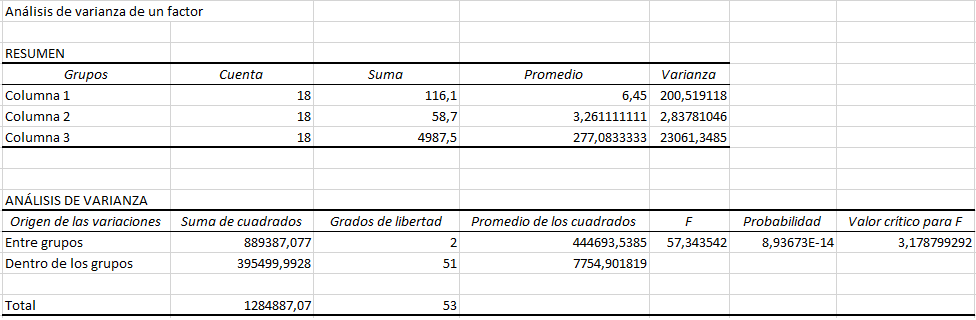
*ANOVA: RAM tabla 2.0*



Como segunda instancia, se tuvo en cuenta los datos arrojados por el experimento, dado que se evidenciaron diferentes RAM se asignó a la columna 1 una RAM de 4 GB, para la columna 2 una RAM de 8 GB, por último, para la columna 3 una RAM de 12 GB.

Por medio de la varianza y el promedio se nota la diferencia de tiempo entre cada una, pero sí se puede apreciar que entre mayor tamaño tenga la memoria RAM mejor es su rendimiento de tiempo de ejecución, por ende la RAM que se utilice para ejecutar el algoritmo es fundamental en lo que tarde un arreglo en ordenarse. Lo cual es apoyado por su escala de superioridad del valor crítico frente a su valor F.

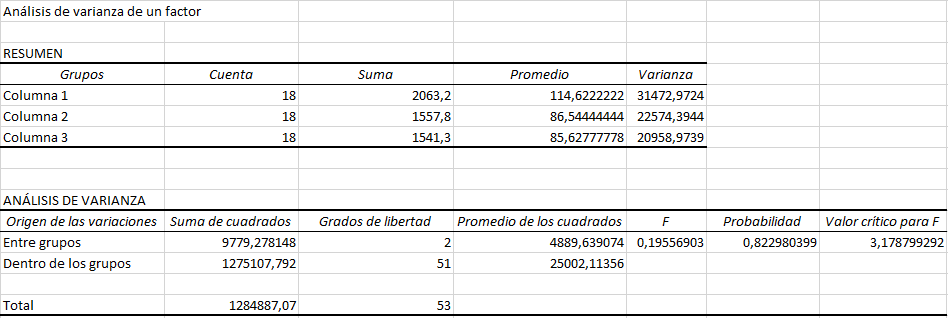
*ANOVA: Tamaño del arreglo tabla 3.0*



**Hipótesis**: El tamaño del arreglo influye en lo que tarda el método en ser ordenado, por lo tanto un arreglo de tamaño 10^4 tarda menos que uno de 10^5

Como tercera instancia, no puede faltar en cuanto al análisis estadístico experimental el tamaño del arreglo, el cual es vital en el tiempo de ejecución, se usó diferentes tamaños en el Array designados por: 10^2, 10^4 y 10^5. Sin embargo, estos tamaños pueden aumentar, pero se implementó solo 3 para el experimento. Si bien se nota que es incierto afirmar que un arreglo de tamaño 10^4 tarda menos que uno de 10^5, ya que en él influyen más factores, como vimos anteriormente la RAM del dispositivo donde se esté ejecutando hace parte de su rendimiento. Por un momento se llegase a pensar que sí determina contundentemente el tamaño del arreglo el tiempo de ejecución pero en detrás de él hay más procesos que lo ayudan a optimizar. Por consiguiente, el valor crítico rechaza la hipótesis que un arreglo de mayor tamaño se tarda menos que un arreglo de menor tamaño, ya que su valor es menor que el valor F.

*ANOVA: Estado del arreglo tabla 4.0*



Por último, el factor de estado se refiere a cómo ingresa el arreglo al momento de ser ordenado por alguno de los dos algoritmos mencionados anteriormente, puede tomar 3 valores: que ingrese de manera ordenada ascendentemente (columna 3), descendentemente (columna 2) o totalmente desordenado (columna 3). Si bien es claro, un arreglo que ya está ordenado ascendentemente y descendentemente gasta menos tiempo en cuanto a uno que está totalmente desordenado, y este dato lo ofrece la tabla 3.0 cuando analizamos el promedio de cada estado. Por tanto, el estado inicial del arreglo influye con respecto a la cantidad de tiempo que tarde el método de ordenamiento. Como dato estadístico se tiene que, el valor crítico es superior al valor F por lo que se afirma la hipótesis mediante un proceso estadístico.

**Interpretación:**

Primeramente se fijaron los factores a estudiar los cuales fueron, la memoria RAM, el estado de los arreglos, el tamaño de los mismos, y por último el tipo de algoritmo de ordenamiento. Basados en estas características se realizaron tratamientos, en total 54 , cantidad dada por el producto de todos los niveles por cada factor.Lo anterior fue de gran ayuda para poder, bajo pruebas rigurosas de ejecución de cada tratamiento, notar los aspectos sobresalientes en el experimento y así poder definir cuál era la mejor opción basada en el tiempo de ejecución de un algoritmo de ordenamiento.

Por un lado, al observar los resultados, se pudo ver como la cantidad de memoria RAM donde se ejecuta el programa afecta de manera significativa el tiempo de ejecución tanto para el algoritmo QuickSort tanto para BubbleSort. En ambos, variando los demás factores controlables, se logró notar que para todos los tratamientos que se ejecutaron en un computador de 4GB de RAM, se obtuvo los mayores tiempos de ejecución. Esta variable de respuesta fue ligada directamente a las conjeturas iniciales que se dieron en el proceso de planeación. Debido a que, como Ingenieros de Sistemas, es claro que entre más cantidad de memoria se tenga, más eficiente se vuelve el manejo de información, por ende, en el manejo de los números de un arreglo se vuelve más fácil su ordenamiento en una computadora con más memoria RAM. Por lo tanto, el mejor nivel fue el de 12GB.

Cabe aclarar, que primeramente se pensaba en la memoria RAM como un factor de carácter netamente influyente en el experimento realizado. Sin embargo, pudo notarse que para conjuntos de datos pequeños, el tiempo de ejecución era el mismo para los tres niveles de memoria estudiados, cuyo tiempo era de 1 milisegundo aproximadamente.

Por consiguiente, se vio involucrado como factor principal, el tamaño del arreglo que se va a ordenar. Y es prácticamente claro ver que entre más cantidad de números, mayor será el costo de realización del algoritmo. Dicho esto, cabe aclarar que si se ve desde un punto de vista general, hoy en dia, es poco probable que grandes empresas, industrias, etc, utilicen algoritmos de ordenamiento para conjuntos de datos pequeños. Por el contrario, los usan para datos significativamente grandes, por ende, se optó por la opción de dividir los datos en grupos más pequeños de tal forma de poder escoger los tratamientos con menor nivel en el factor tamaño, en este caso 10^2, y así mejorar el tiempo de ejecución tanto para conjuntos de datos pequeños como para grandes.

Por otro lado, analizando el estado inicial del arreglo, se notó a partir de los resultados arrojados, que este factor no influye en la variable de respuesta de todos los tratamientos, puesto que en cada combinación en los que este factor era la único factor que cambiaba, los tiempos de ejecución eran muy similares. Con esto se puede plantear que un algoritmo de ordenamiento, sin importar cómo llega el arreglo inicial, cumple con todas las especificaciones y funciones dadas dentro del algoritmo. Por ejemplo, aún cuando el arreglo está completamente ordenado, el algoritmo lleva a cabo todas sus funciones y se puede decir que ordena nuevamente el arreglo.

Por lo tanto, se optó por tener los dos algoritmos tanto QuickSort como BubbleSort como parte de los factores más convenientes. Sin embargo, debido a una mínima diferencia entre los dos algoritmos debido a su complejidad temporal similar, el tratamiento más conveniente para llevar a cabo nuestro objetivo de estudio de tener el tiempo de ejecución más corto posible para un algoritmo de ordenamiento, es el Tratamiento #37, el cual cumple con los siguientes factores:

Tamaño del arreglo: 10^2

Estado Inicial del arreglo: Aleatorio

Algoritmo de ordenamiento: QuickSort

Memoria RAM: 12 GB

Tiempo de ejecución: 20,8 ms

**Conclusión**

Se puede concluir que una máquina que cuente con 12GB de RAM tiene la posibilidad de hacer más eficiente,en cuanto a tiempo, para cualquiera de los casos que se le presente, sin embargo está afirmación es parcialmente verdadera, ya que se realizó el experimento con tamaños de arreglos más grandes de 10^8 y en todos los niveles del factor de RAM los resultados no concluían, el tiempo era tan extenso que lo más rentable era no organizar toda la colección. De manera general, como lo afirman los datos obtenidos en el experimento,se puede establecer que lo más eficiente es dividir los datos en grupos pequeños, por ejemplo colecciones de un tamaño máximo 10^7 y posteriormente ordenar cada colección, en este caso no importa la RAM o el método que se esté aplicando, los tiempos van a ser mucho más cortos, un aspecto a tener en cuenta, si lo que se busca es aun mayor eficiencia es, si los datos cuentan con demasiada dispersión es más eficiente aún filtrarlos por intervalos más reducidos, ya que como muestra los datos de los arreglos cuyos valores eran aleatorios, tardan mucho más por lo que el algoritmo necesita realizar más trabajo para obtener el resultado deseado. Otro elemento a resaltar es que al realizar el estudio estadístico se logró llegar a afirmaciones más precisas, por lo tanto es de gran utilidad para el caso, a pesar de las limitaciones con las que se encontraron como la reducida capacidad con la que cuenta el equipo del que disponemos, se logra especificar una regla que es aplicable bajo cualquier tipo de situación que se presente.

# **Bibliografía**

Bijit, L. S. (2010). Estructuras de Datos y Algoritmos. En L. S. Bijit, *Estructuras de Datos y Algoritmos* (págs. 1-2).

contributors, E. (9 de 02 de 2019). *Ordenamiento de burbuja*. Obtenido de https://www.ecured.cu/index.php?title=Ordenamiento\_de\_burbuja&oldid=1326492

contributors, E. (9 de 02 de 2019). *QuickSort*. Obtenido de https://www.ecured.cu/index.php?title=QuickSort&oldid=2134213

Thomas H. Cormen, C. E. (2009). *Introduction to Algorithms.* Massachusetts: Massachusetts Institute of Technology.