TALLER DISEÑO DE EXPERIMENTOS

Universidad icesi

Miguel sarasti-Santiago Hurtado-Sebastian Morales- Sebastian Barrera

Integrantes del grupo:

Sebastián Barrera A00358271
Santiago Hurtado A00362570
Miguel Sarasti A00364978
Sebastián Morales A00365920

A continuación, se encontrará con una breve introducción al experimento de forma resumida, posteriormente se encuentra cada fase explicada con toda la información.

Fenómeno, sistema o proceso sobre el cual experimentará:
BUBBLE SORT:
La **Ordenación de burbuja** (**BubbleSort** en inglés) es un sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos 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. Este algoritmo obtiene su nombre de la forma con la que suben por la lista los elementos durante los intercambios, como si fueran pequeñas "burbujas". También es conocido como el **método del intercambio directo**. Dado que solo usa comparaciones para operar elementos, se lo considera un algoritmo de comparación, siendo uno de los más sencillos de implementar.
Tomado de: Ordenamiento de burbuja

Características:

- Los valores grandes siempre se ordenan primero.
- Solo se necesita una iteración para detectar que una colección ya está ordenada.
- La mejor complejidad de tiempo para Bubble Sort es O (n). La complejidad de tiempo promedio y peor es O (n^2).
- La complejidad del espacio para Bubble Sort es O (1), porque solo se requiere un espacio de memoria adicional.

INSERTION SORT
La inserción es uno de los métodos de ordenamiento más semejantes a los utilizados en forma natural al del ser humano. El método consiste en insertar elementos en forma ordenada en una lista directamente en el orden que le corresponde. Aunque esta operación en forma humana podría consistir en una única operación, debido a nuestras capacidades cognitivas, para una computadora implica el uso de un algoritmo que consiste en tomar un elemento y colocarlo en un arreglo nuevo en la posición que le corresponde.
Tomado de: Ordenamiento por inserción

Complejidad: O(n²)

Objetivo del experimento: Determinar el algoritmo más eficiente entre bubblesort e insertión sort para ordenar arreglos de valores donde estos son diferentes tamaños.

Unidad Experimental: La unidad experimental de este experimento son los algoritmos de ordenamiento, Bubblesort e insertionsort. Estos serán desarrollados en c# para evaluar su rendimiento, junto a sus pruebas que tendrán factores como el orden del arreglo que se va a ordenar y el tamaño. Con el fin de determinar cuál tiene mejor desempeño.

Variable(s) de Respuesta: La variable de respuesta que determinamos para realizar el experimento es el tiempo del ordenamiento por cada algoritmo al ejecutarse, cuando se ingresen entradas de diferentes tamaños y de orden diferente, ya que con estos factores el tiempo del ordenamiento va a tener cambios.

Factores controlables: Son variables de proceso o características de los materiales experimentales que se pueden fijar en un nivel dado.

- Tamaño del arreglo (10^1, 10^2, 10^3, 10^4, etc).
- Estado de los valores en el arreglo (en orden aleatorio, ordenado ascendente, ordenado descendente).
- El tipo de algoritmo de ordenamiento.
- La RAM del computador donde se ejecuta el algoritmo (2GB, 4GB, 8GB, 16GB, etc) es un factor que podemos controlar porque podemos escoger computadores con un tamaño de ram específico o incluso limitar el uso de esta, para el proceso del algoritmo.
- Procesador del computador donde se ejecuta el algoritmo.
- Lenguaje de programación (Java, C#, Python, etc).
- Sistema Operativo.
- Cantidad de aplicaciones que se están ejecutando en el computador mientras se ejecuta el algoritmo.
- Tipo de variable de datos a ordenar (números enteros, números reales, letras, etc).

Factores no controlables: Son factores que no podemos controlar en el experimento.

- Limitaciones del algoritmo a la hora de ingresar cantidades extremadamente grandes datos.

- Cantidad de procesos que se están ejecutando en el computador mientras se ejecuta el algoritmo.
- La cantidad de RAM que utiliza el algoritmo en su ejecución.

Factores estudiados:
Los factores que vamos a estudiar en este experimento son los que influyen directamente en la variación del tiempo al momento de llevar a cabo el algoritmo.

- Orden de los elementos por arreglo
- Cantidad de valores por arreglo
- Los algoritmos para estudiar

Niveles
Tipo de ordenamiento

- Burbuja
- Insertion

Tamaño del arreglo:

- Pequeño
- Mediano
- Grande

Estado de los valores en el arreglo:

- Aleatorio
- Ordenado ascendente
- Ordenado descedente

Tamaño del arreglo	Nivel de elementos
10^2	1
10^3	2
10^4	3

Orden inicial del arreglo	Niveles de orden
Ascendente	1
Descente	2
Aleatorio	3

Algoritmo de ordenamiento	Nivel del algoritmo
Bubble Sort	1
Insertion Sort	2

Tratamientos

Algoritmo de ordenamiento	Número de elementos de entrada	Orden de los elementos	Tratamiento	Repeticion	Tiempo promedio (ms)
Bubble Sort	100	Ascendente	1	100	0,0
Bubble Sort	100	Descendente	2	100	0,0
Bubble Sort	100	Aleatorio	3	100	0,0
Bubble Sort	10^3	Ascendente	4	100	5,3
Bubble Sort	10^3	Descendente	5	100	5,1

Bubble Sort	10^3	Aleatorio	6	100	5,1
Bubble Sort	10^4	Ascendente	7	100	602,3
Bubble Sort	10^4	Descendente	8	100	602,2
Bubble Sort	10^4	Aleatorio	9	100	602,8
Insertion Sort	100	Ascendente	10	100	0,0
Insertion Sort	100	Descendente	11	100	0,0
Insertion Sort	100	Aleatorio	12	100	0,0

Insertion Sort	10^3	Ascendente	13	100	5,0
Insertion Sort	10^3	Descendente	14	100	5,0
Insertion Sort	10^3	Aleatorio	15	100	5,0
Insertion Sort	10^4	Ascendente	16	100	581,7
Insertion Sort	10^4	Descendente	17	100	582,0
Insertion Sort	10^4	Aleatorio	18	100	582,4

Etapas del experimento

Planeación y Realización

El Objetivo de este experimento es ver que pasa con el tiempo de ejecución de dos algoritmos de ordenamiento la misma complejidad temporal, a medida que le vamos cambiando otros factores y determinar cuál, de los dos algoritmos en mejor, para determinados casos. Para realizar este experimento, escogimos los algoritmos de ordenamiento de burbuja (Bubble Sort en inglés) y de ordenamiento por inserción (Insertion Sort en inglés), los cuales tienen una complejidad temporal exponencial de n^2 en notación BigO la cual nos indica cómo se comporta el algoritmo en el peor escenario, es decir, el máximo tiempo en la mayor cantidad de repeticiones que el algoritmo tiene que ejecutar.

Los factores que vamos a modificar para realizar nuestro experimento se les conoce también como factores de estudio, estos los escogimos a partir de una lista de todos los posibles factores que se puedan modificar a nuestro gusto y que sepamos que durante todo el experimento estos factores se mantendrán como nosotros deseamos que estén. Todos estos factores son los siguientes:

- Los Algoritmos de ordenamiento.
- Tamaño del arreglo a ordenar.
- Estado de los valores del arreglo (aleatorio, ordenados ascendente, ordenados descendente).
- La cantidad de memoria RAM.
- El procesador que cuenta el computador donde se ejecuta.

- Lenguaje de programación.
- Sistema operativo
- Cantidad de aplicaciones en ejecución mientras se corre el algoritmo
- Tipo de variables a ordenar (números, letras, etc.).

Dado a que son muchos factores, escogimos solo unos cuantos los cuales nos interesaba experimentar, los cuales son: tamaño del arreglo, estado de los valores del arreglo y los algoritmos de ordenamiento. Los niveles de estos factores son muy importantes ya que nos ayudan a ver de antemano cuantas combinaciones se nos puede llevar realizar el experimento, por lo que decidimos que el tamaño del arreglo tendría 3 niveles: grande con 10000 datos, mediano con 1000 y pequeño con 100; los estados de los valores del arreglo también tendrán 3 niveles: aleatorio, ordenados ascendente, ordenados descendente; y el algoritmo de ordenamiento serán 2: Bubble Sort e Insertion Sort. Como para el experimento necesitamos todos los casos, al hacer una combinación de estos niveles, nos da un total de 18 combinaciones diferentes o mejor conocidos como tratamientos y cada tratamiento lo repetiremos 150 veces para evitar el sesgo.

Análisis

Como ya tenemos los resultados del proceso anterior lo que sigue será analizar los resultados, para esto utilizamos principalmente la herramienta de Excel y la técnica de estadística inferencial de análisis de varianza ANOVA (acrónimo en inglés). Para realizar un análisis de varianza, se deben tener como mínimo dos grupos a comparar los cuales sus datos deben estar en una distribución normal o aproximadamente normal, aunque esto se puede ignorar si la cantidad de datos es suficientemente grande (30+). También se deben tener dos hipótesis una nula y otra alterna, la nula siempre afirma que el factor

estudiado en los grupos es igual mientras que la alterna lo contrario, y por último debe tener un nivel de significancia el cual nos indica la posibilidad de equivocarse al momento de dar un resultado, por lo cual entre más bajo sea este valor más confiable será la interpretación. El resultado del análisis de varianza son dos valores, uno llamado F calculado y el otro Valor crítico, aunque es mejor conocido como F de tabla. Las condiciones del análisis es que si F calculado es menor al F de tabla aceptamos la hipótesis nula, de lo contrario la debemos rechazar y acepar la alterna.

El primer análisis que realizamos es el de cual algoritmo tenia los mejores tiempos estadísticamente, así que tomamos todos los datos de los tratamientos que tuvieran que ver con Bubble Sort y los unimos en uno solo, lo mismo para los tratamientos de Insertion Sort. Dejando un análisis para dos grupos con una hipótesis nula de que los tiempos de ejecución para cantidades de datos variadas (grandes, pequeñas y medianas) de los algoritmos Bubble Sort e Insertion Sort son iguales, mientras que la alterna dice que son diferentes.

En nuestro segundo análisis repetimos el mismo proceso que el anterior, pero en este caso solo dejamos los tratamientos que tuviera un tamaño de ordenamiento grande, y comparando de nuevo el Bubble Sort e Insertion Sort, como hipótesis nula tenemos que los tiempos de ejecución en grandes cantidades de datos de los algoritmos Bubble Sort e Insertion Sort son iguales y como alterna que son diferentes.

Y nuestro último análisis es sobre si el orden de los valores en el arreglo afecta el tiempo de ejecución del algoritmo, por lo que juntamos todos los tratamientos en 3 grupos, uno donde estuvieran los tratamientos con orden aleatorio, otro con orden ascendente y otro con orden descendente, y como hipótesis nula tenemos que el estado de los valores del

arreglo antes de ordenarlo NO altera el resultado del tiempo de ejecución del algoritmo, mientras que la alterna dice que si lo afecta.

Interpretación

Antes de dar interpretación a cada uno de los análisis, cabe aclarar que la codificación de cada algoritmo se realizo teniendo en cuenta la implementación base de estos algoritmos, es decir que esta versión no es la mas optimizada o la mejor versión de todas, aunque claro tampoco la peor.

Iniciando con la interpretación del primer análisis tenemos que el valor de F calculado nos dio 0,410461917 y el de F de tabla 6,66318854, por lo cual podemos interpretar con un nivel de significancia del 1% que para cantidades de datos variadas los tiempos de ejecución de los algoritmos de ordenamiento Bubble Sort e Insertion Sort son iguales, este resultado ya era esperado ya que ambos tienen la misma complejidad entonces no era extraño que sus tiempos estadísticamente fueran iguales.

Dado el resultado anterior, es que decidimos realizar este análisis ya que la diferencia de tiempos entre los arreglos de tamaño pequeños y medianos era gigante comparados con el tiempo de los arreglos de tamaño grande, por esto repetimos el análisis anterior pero solo con arreglos de tamaño grande. Entonces nuestro segundo análisis nos dio que el valor de F calculado es de 2561,604393 y el de F de tabla es de 6,663188541, con lo que al contrario de la anterior esta vez si podemos concluir que hay diferencia entre los datos y para escoger cual es mejor solo debemos mirar el promedio, el cual nos indica que el tiempo de ejecución del algoritmo de Insertion Sort es menor que el del Bubble Sort.

Y por último para interpretar el análisis del estado de los arreglos tenemos que el valor de nuestro F calculado es de 0,000105682 y el de F de tabla 4,613042546, con lo que podemos decir que con un nivel de significancia del 1% podemos concluir que el orden de los algoritmos no afecta el tiempo de ejecución de los algoritmos.

Conclusión

En conclusión, hay varios factores que pueden afectar el resultado en tiempo de ejecución del algoritmo, y no solo los factores como tal sino también sus niveles, como se vio reflejado en el segundo análisis, que con tan solo tener en cuanta 1 de los 3 niveles del tamaño del arreglo, pudimos hacer que el resultado cambiase significativamente. Podemos decir que el algoritmo de Insertion Sort es mejor en tiempo de ejecución que el algoritmo de Bubble Sort aunque esta diferencia solo se ve reflejada en tamaños de arreglos suficientemente grandes, también tenemos que el no importa el orden que estén los datos del arreglo, cualquiera de los algoritmos seguirá tomando la misma cantidad de tiempo en ordenarlos, teniendo en cuanta que no esta optimizado ya que hay una versión en la que el algoritmo termina su ejecución si el arreglo ya se encuentra ordenado.

Análisis de complejidad espacial de los algoritmos

¿Cuánto espacio usa el algoritmo?

Algoritmo Bubble Sort

Tipo	Variable	Cantidad de valores atómicos
Entrada	T[] array	n
Auxiliar	int size	1
	int i	1
	int j	1
	boolean change	1
	T temp	1
Salida	T[] Bubble	n
Resultado		5+2n = O(n)

Algoritmo Insertion Sort

Tipo	Variable	Cantidad de valores atómicos
Entrada	T[] array	n
Auxiliar	int size	1
	int i	1
	int j	1
	boolean change	1
	T temp	1
Salida	T[] Insert	n
Resultado		5+2n = O(n)

Las implementaciones de los algoritmos de ordenamiento analizados tienen la misma complejidad espacial, siendo esta O(n), lo que traduciría a una función lineal y a mayor cantidad de datos en el arreglo a ordenar, el espacio en memoria ocupado aumentará en el mismo factor de crecimiento

Análisis de complejidad temporal

Bubble Sort

ALGORITMO	# veces que se repite
public T[] BubbleSort(T[] array)	n = tamaño de array
{	
int size = array.Length;	1
T[] Bubble = array;	1
for (int i = 0; i < size - 1; i++)	n-1
{	
for (int j = 0; j < size - i - 1; j++)	n(n-1)/2
{	
bool change = ascendencing ? Bubble[j].CompareTo(Bubble[j + 1]) > 0 : Bubble[j].CompareTo(Bubble[j + 1]) < 0	(n(n-1)/2) - (n-2)
if (change)	(n(n-1)/2) - (n-2)
{	
T temp = Bubble[j];	(n(n-1)/2) - (n-2)
Bubble[j] = Bubble[j + 1];	(n(n-1)/2) - (n-2)
Bubble[j + 1] = temp;	(n(n-1)/2) - (n-2)
}	
}	
}	
return Bubble;	1
}	
COMPLEJIDAD TEMPORAL DEL ALGORITMO BubbleSort	(5(n^2 - n)/2) - 4n + 12 O(n^2)

Insertion sort

ALGORITMO	**# veces que se repite**
public T[] InsertionSort(T[] array)	**n = tamaño de array**
{	
int size = array.Length;	**1**
T[] Insert = array;	**1**
for (int i = 0; i < array.Length - 1; i++)	**n-1**
{	
for (int j = i + 1; j > 0; j--)	**n(n-1)/2**
{	
bool change = ascendencing ? Insert[j].CompareTo(Insert[j - 1]) < 0 : Insert[j].CompareTo(Insert[j - 1]) >	**n(n-1)/2 - n-2**
if (change)	**n(n-1)/2 - n-2**
{	
T temp = array[j - 1];	**n(n-1)/2 - n-2**
array[j - 1] = array[j];	**n(n-1)/2 - n-2**
array[j] = temp;	**n(n-1)/2 - n-2**
}	
}	
}	
return Insert;	**1**
}	
COMPLEJIDAD TEMPORAL DEL ALGORITMO InsertionSort	**(5(n^2 - n)/2) - 4n + 12** **O(n^2)**