**ALGORITMOS DE ORDENAMIENTO**

Cristian Cobo , Joe Hernandez, Kliver Daniel Girón, Christian Tamayo

Universidad ICESI, Facultad de ingeniería, Proyecto Integrador **I**

**Santiago de Cali, febrero 11 del 2019.**

1. **RESUMEN**

Este informe, se realizó con el objetivo de analizar el comportamiento de dos (***2***)algoritmos de ordenamiento con igual complejidad temporal ante el cambio o modificación de factores que alteran su naturaleza. Para poder llevar a cabalidad este experimento fue necesario aplicar diferentes conocimientos teóricos y prácticos que ayudaron a comprender los procedimientos y conceptos en el desarrollo de ésta práctica.

**2. INTRODUCCIÓN**

Una de las prácticas más comunes en las ciencias de la computación es la medición de distintos aspectos de los algoritmos, como lo son el tiempo y espacio en memoria. Desde que nació el término ***software*** unas de las miles de preguntas que surgieron fue que te tan eficientes eran los algoritmos para resolver problemas del mundo real. Fue así como cada vez los científicos informáticos se preocupaban más sobre la eficiencia de sus algoritmos y ver la necesidad de realizarles estudios exhaustivos con ayuda de herramientas formales como lo son las matemáticas. Diseñar algoritmos bien estructurados y eficientes es una tarea compleja y es por ello que antes de utilizarlos se deben realizar análisis que verifiquen no solo su solución al problema sino la ***eficiencia*** en que la realiza. Probar algoritmos es una tarea diaria de los diferentes departamentos de pruebas en todas las casas de desarrollo de software y son ellos los encargados de certificar o no el uso de éstos. Los algoritmos son probados desde distintas perspectivas, identificando así factores, variables y demás con el objetivo de tener un universo de tratamientos e inferir sobre su eficiencia.

**3. MARCO TEÓRICO**

**Factores contables:**

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

**Factores no contables:**

Son variables o características de materiales y métodos que no se pueden controlar durante el experimento o la operación normal del proceso.

**Factores estudiados:**

Son las variables que se investigan en el experimento, respecto de cómo influyen o afectan a la(s) variable(s) de respuesta.

**Niveles y tratamientos**

Los diferentes valores que se asignan a cada factor estudiado en un diseño experimental se llaman *niveles*. Una combinación de niveles de todos los factores estudiados se llama *tratamiento* o *punto de diseño*.

**Unidad experimental:**

La *unidad experimental* es la pieza(s) o muestra(s) que se utiliza para generar un valor que sea representativo del resultado del experimento o prueba

**Error aleatorio y experimental:**

Siempre que se realiza un estudio experimental, parte de la variabilidad observada en la respuesta no se podrá explicar por los factores estudiados. Esto es, siempre habrá un remanente de variabilidad que se debe a causas comunes o aleatorias, que generan la variabilidad natural del proceso. Esta variabilidad constituye el llamado *error aleatorio*. (Sin autor, s.f.)

**Algoritmo:**

Se denomina algoritmo a un grupo finito de operaciones organizadas de manera lógica y ordenada que permite solucionar un determinado problema. Se trata de una serie de instrucciones o reglas establecidas que, por medio de una sucesión de pasos, permiten arribar a un resultado o solución. (Merino, 2010)

**Ordenar:**

El orden implica colocar las cosas en el lugar que les corresponde con la misión de organizarlas y ganar en cuanto a su sencilla localización, es decir, cuanto más ordenado tengamos

un espacio mucho más simple será encontrar en él las cosas. (Ucha, s.f.)

**Eficiencia algorítmica:**

En Ciencias de la Computación, el término eficiencia algorítmica es usado para describir aquellas propiedades de los algoritmos que están relacionadas con la cantidad de recursos utilizados por el algoritmo. Un algoritmo debe ser analizado para determinar el uso de los recursos que realiza. La eficiencia algorítmica puede ser vista como análogo a la ingeniería de productividad de un proceso repetitivo o continuo. (Wikipedia, s.f.)

**Algoritmo de ordenamiento:**

En computación y matemáticas un algoritmo de ordenamiento es un algoritmo que pone elementos de una lista o un vector en una secuencia dada por una relación de orden, es decir, el resultado de salida ha de ser una permutación o reordenamiento de la entrada que satisfaga la relación de orden dada. Las relaciones de orden más usadas son el orden numérico y el orden lexicográfico. Ordenamientos eficientes son importantes para optimizar el uso de otros algoritmos (como los de búsqueda y fusión) que requieren listas ordenadas para una ejecución rápida. También es útil para poner datos en forma canónica y para generar resultados legibles por humanos. (Wikipedia, s.f.)

**Java:**

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. (ictea, s.f.)

**C#:**

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. (Wikipedia, WIKIPEDIA, s.f.)

**RAM:**

RAM es el acrónimo del concepto inglés de Random Access Memory (Memoria de Acceso Aleatorio). Se trata de la memoria que, en un equipo informático, es utilizada por un procesador para recibir instrucciones y guardar los resultados. (Gardey, 2010)

**Procesador:**

El procesador es el cerebro del sistema, encargado de procesar toda la información. Básicamente, es el “cerebro” de la computadora. Prácticamente, todo pasa por él, ya que es el responsable de ejecutar todas las instrucciones existentes. Mientras más rápido vaya el procesador, más rápido serán ejecutadas las instrucciones. (Informatica, s.f.)

**Sistema operativo:**

Un sistema operativo (OS) es el programa que, después de ser inicialmente cargado en la computadora por un programa de arranque, gestiona todos los otros programas en una computadora. Los otros programas se llaman aplicaciones o programas de aplicación. Los programas de aplicación usan el sistema operativo haciendo peticiones de servicios a través de una interfaz de programa de aplicación (API) definida. (SearchDataCenter, s.f.)

**4. MARCO PROCEDIMENTAL**

**4.1 Materiales:**

**Computadoras utilizadas:**

**Tabla 01.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Computadora** | **ROM** | **RAM** | **Disco Duro** | **Tarjeta de grafica** | **OS** | **Procesador** |
| **LENOVO Y700** | **1T** | **16 GB** | **250 GB** | **Nvidia gtx 960m** | **Windows 10** | **Intel I7** |
|  | **1T** | **16 GB** | **240 GB** | **Nvidia gtx 1050m** | **Windows 10** | **Intel I7** |

Como se mencionó anteriormente, para ésta práctica se realizará pruebas de **eficiencia algorítmica** a dos (2) famosos algoritmos de ordenamiento. Es por ello que se han identificado

algunos **factores** que no son propiamente materiales tangibles pero sin embargo son de gran utilidad al momento de probar algoritmos. Estos factores son:

* Memoria RAM
* Lenguajes de programación
* Arreglos o contenedoras unidimensionales
* Procesador de máquina
* Sistema operativo
* Nivel de fragmentación DD
* Algoritmo de ordenamiento Merge Sort
* Algoritmo de ordenamiento Heap Sort
* Estado de un arreglo unidimensional
* Tamaño de arreglo

**Procedimiento:**

* En primer lugar se realizó un pequeño software en dos lenguajes de programación diferentes, encargado de registrar el tiempo en que los algoritmos tardan en ordenar arreglos unidimensionales.
* En segundo lugar se toma registros del estado de la memoria RAM del dispositivo o computador.
* En tercer lugar, se establecen criterios del estado de los arreglos a probar, es decir, su comportamiento actual como por ejemplo su longitud, orden y tipo de contención de datos.
* Posteriormente, se inicializan arreglos de distintas longitudes con el fin de tener variabilidad en los resultados y así mismo con el orden de sus elementos.
* Más adelante, se prueban cada una de las contenedoras anteriormente inicializadas en el software y se registran los respectivos tiempos en hojas electrónicas

**5. RESULTADOS Y ANÁLISIS**

Se obtuvieron los siguientes datos experimentales:

**Tabla 02.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tratamiento** | **Repeticiones** | | | | |
| **#** | **1 (en ns)** | **2 (en ns)** | **3 (en ns)** | **4 (en ns)** | **5 (en ns)** |
| **1** | 360500 | 8500 | 12300 | 8600 | 8800 |
| **2** | 1335200 | 1023300 | 951200 | 929600 | 845300 |
| **3** | 117805400 | 117109200 | 111225900 | 111948300 | 112549400 |
| **4** | 123700 | 6500 | 5600 | 5800 | 58600 |
| **5** | 1049100 | 968600 | 855300 | 947700 | 967200 |
| **6** | 165370600 | 159338300 | 159142600 | 159237200 | 155322900 |
| **7** | 80198 | 72296 | 80592 | 69926 | 43457 |
| **8** | 3418469 | 2453334 | 1909333 | 2042865 | 2256987 |
| **9** | 233266173 | 222912790 | 178874470 | 184155654 | 243145086 |
| **10** | 177382 | 54914 | 21333 | 19753 | 18963 |
| **11** | 2821926 | 1848494 | 1664000 | 2571852 | 1806618 |
| **12** | 225172148 | 202061432 | 208702025 | 191844741 | 225632000 |

La tabla No. 2 refleja la cantidad de **repeticiones** de cada uno de los respectivos **tratamientos** y es evidente que los resultados varían por diversos f**actores** como lo son el tipo de máquina, longitud de contenedor, lenguaje utilizado, etc.

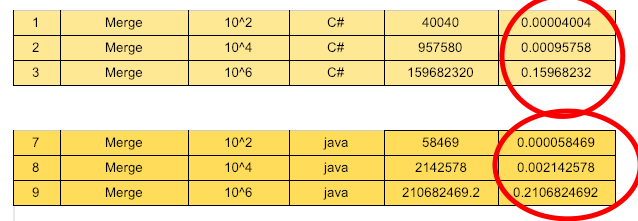
**Tabla 03.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tratamientos** | | | | | |
| **#** | **Algoritmos de ordenamiento** | **Tamaño del arreglo** | **Lenguaje de programación** | **Variable Esperada (Tiempo, NanoSegundos)** | **Variables (Segundos)**  **(promedio)** |
| 1 | Merge | 10^2 | C# | 40040 | 0.00004004 |
| 2 | Merge | 10^4 | C# | 957580 | 0.00095758 |
| 3 | Merge | 10^6 | C# | 159682320 | 0.15968232 |
| 4 | Heap | 10^2 | C# | 79740 | 0.00007974 |
| 5 | Heap | 10^4 | C# | 1016920 | 0.00101692 |
| 6 | Heap | 10^6 | C# | 114127640 | 0.11412764 |
| 7 | Merge | 10^2 | java | 58469 | 0.000058469 |
| 8 | Merge | 10^4 | java | 2142578 | 0.002142578 |
| 9 | Merge | 10^6 | java | 210682469.2 | 0.2106824692 |
| 10 | Heap | 10^2 | java | 69293.8 | 0.000069294 |
| 11 | Heap | 10^4 | java | 2416197.6 | 0.0024161976 |
| 12 | Heap | 10^6 | java | 212470834.6 | 0.2124708346 |

En la tabla anterior se logra evidenciar los algoritmos que se evaluaron. Así mismo, el tamaño de la entrada de prueba que estos tuvieron junto con sus resultados **“variables” esperadas** dadas en nanosegundo y segundos. Lo anterior son datos obtenidos y evaluados por medio de los lenguajes de programación: C# y Java.

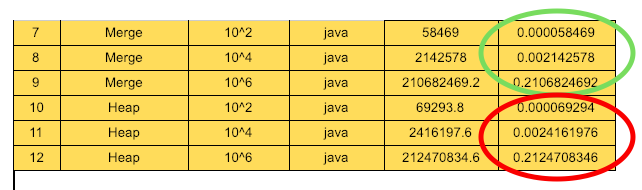
De la tabla No.3 es posible observar que el promedio de tiempo de los algoritmos en ordenar contenedoras de longitudes pequeñas no varía mucho. En el lenguaje de programación c# se observa que el algoritmo que ordena de forma más eficiente es el algoritmo Merge ante el Heap y esto lo hacen tanto para arreglos pequeños como de gran tamaño, no obstante su diferencia no incurre significativamente con longitudes menores a 10^4. En el caso de Java ocurre exactamente lo mismo en cuanto a la forma que responde un algoritmo frente al otro sin embargo los tiempos promedios son superiores en éste lenguaje.

**Imagen 01.**

****

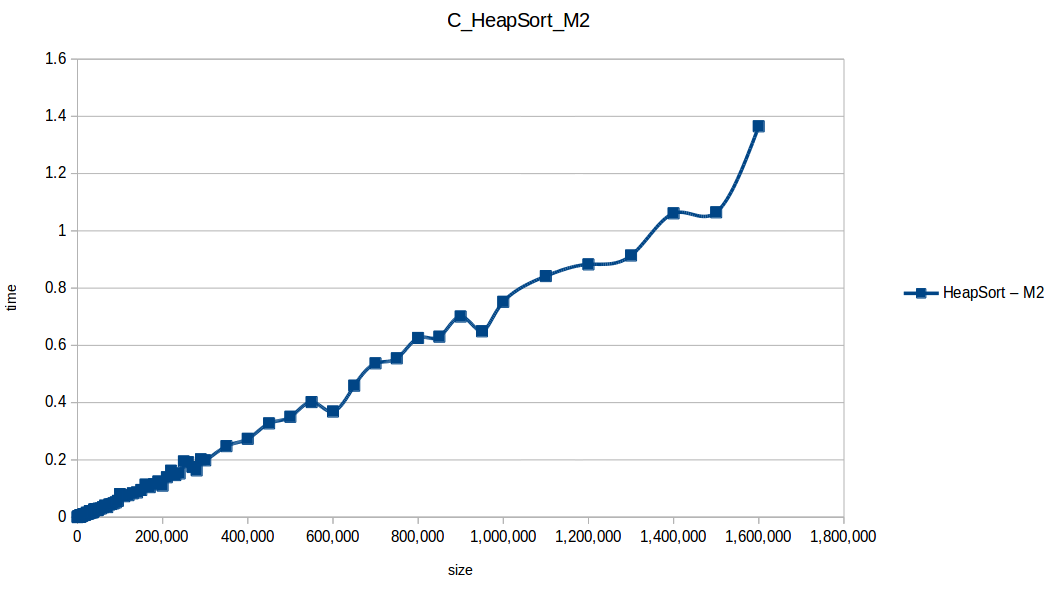
La imagen no. 1 es de gran ayuda para así observar de manera más clara lo anteriormente dicho.

**Imagen 02.**

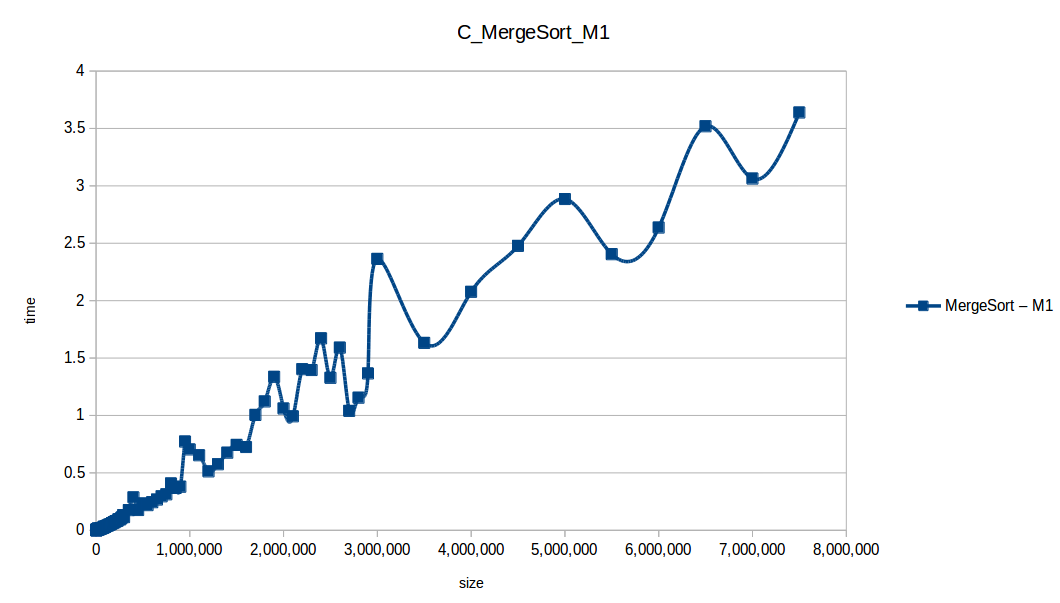


En la imagen no. 02 el círculo verde refleja el tiempo promedio que el algoritmo de ordenamiento merge se toma en ordenar contenedores -arreglos- de diferentes longitudes y es éste quien lo hace mejor frente al ordenamiento heap o por montículos.

**Gráfica 01.** Comportamiento HeapSort



**Gráfica 02.** Comportamiento MergeSort

****

Respecto a la gráfica no. 01 y 02 se puede decir que el algoritmo que ordena de forma más eficiente (en cuanto a tiempo) es el MergeSort frente al Heapsort. A simple vista, basándose en la forma de la gráfica se podría decir que el algoritmo de ordenamiento heapsort es mucho mejor que el merge sort. Sin embargo, al observar mucho más a profundidad el algoritmo heapsort es eficiente con un tamaño de arreglo (contenedor) pequeño. Por ejemplo, en este experimento se evalúa el algoritmo de ordenamiento heapsort y mergesort hasta un tamaño de arreglo como máximo de 10^6 posiciones por ordenar y se observa que la diferencia entre el tiempo de ordenamiento de estos dos algoritmos es muy estrecha. Sin embargo, al examinar las 2 gráficas anteriores, el algoritmo de ordenamiento mergesort puede ordenar hasta casi 8’000.000 de elementos en un tiempo mucho más considerable que el algoritmo heap o por montículos. Cabe resaltar que al comparar las gráficas, el heapsort se evaluó hasta un máximo de 1’800.000 posiciones por ordenar a comparación del mergesort (que como ya se mencionó anteriormente) fue de 8’000.000 de posiciones por ordenar. Sin embargo, al limitar la observación de la gráfica del merge al tamaño máximo de los arreglos del heap se puede ver que el algoritmo de ordenamiento mergesort sigue siendo mucho mejor y eficiente. Por otro lado, es evidente que mientras el tamaño de la longitud de contenedores aumenta la diferencia entre el tiempo que toma ordenar los elementos que éste contiene es cada vez más grande.

**6. CONCLUSIONES**

Ahora bien, de acuerdo con los resultados obtenidos:

* En primer lugar, teniendo en cuenta los lenguajes de programación se logra observar que al ejecutar los algoritmos de ordenamiento mergesort y heapsort en el lenguaje C# es mucho más rápido a comparación de Java a pesar de que los algoritmos evaluados son los mismos.
* En segundo lugar, de acuerdo con los tiempos obtenidos (resultados) de cada uno de los algoritmos al ejecutar se da certeza de que el mergesort y heapsort con tamaños de contenedores pequeños no varía mucho su complejidad temporal pues es muy estrecha. Sin embargo, al evaluar estos dos mismos algoritmos con tamaños de arreglos mucho más grande se logra observar que el mergesort es mucho más eficiente. Por lo tanto, su estrategia de solución se ejecuta mucho mejor para datos de gran magnitud.
* Por último, basándonos en el número de repeticiones con los que se llevaron a cabo todos los tratamientos se puede decir que los resultados varían por varios factores esenciales y estos son factores estudiados como los algoritmos mencionados, el tamaño de los arreglo o contenedores y el lenguaje de programación. Además, existen otros factores no controlables como lo son variables ambientales cantidad de procesos de fondo. Estos dos últimos no son evaluados en el presente experimento pero cabe mencionarlos, ya que, son factores que pueden afectar los resultados obtenidos.

**7. BIBLIOGRAFÍA**

autor, S. (s.f.). Obtenido de https://www.icesi.edu.co/moodle/pluginfile.php?file=%2F45492%2Fmod\_resource%2Fcontent%2F1%2FCapitulo1-IntroduccionAlDisenoDeExperimentos.pdf

Gardey, J. P. (2010). *Definicion.DE*. Obtenido de https://definicion.de/ram/

ictea. (s.f.). *ICTEA*. Obtenido de http://www.ictea.com/cs/knowledgebase.php?action=displayarticle&id=8790

Informatica, T. y. (s.f.). *TECNOLOGIA Y INFORMATICA*. Obtenido de https://tecnologia-informatica.com/el-procesador-de-la-computadora/

Merino, J. P. (2010). *Definicion.De*. Obtenido de https://definicion.de/algoritmo/

SearchDataCenter. (s.f.). *SearchDataCenter*. Obtenido de https://searchdatacenter.techtarget.com/es/definicion/Sistema-operativo

Ucha, F. (s.f.). *DefinicionABC*. Obtenido de https://www.definicionabc.com/general/ordenar.php

Wikipedia. (s.f.). Obtenido de https://es.wikipedia.org/wiki/Eficiencia\_algor%C3%ADtmica

Wikipedia. (s.f.). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Algoritmo\_de\_ordenamiento

Wikipedia. (s.f.). *WIKIPEDIA*. Obtenido de https://es.wikipedia.org/wiki/C\_Sharp