Reporte Proyecto 1

*Apórtela Hernandez Víctor Hugo 201440035, Ortiz Ortega José Ignacio 201413674***Facultad de Ciencias de la Computación.**

Ciudad Universitaria, Edif. CCO1-103, Col. San Manuel, C.P. 72570, Puebla, Pue., México.

[hugo.aportela.20@gmail.com](mailto:hugo.aportela.20@gmail.com) , ig\_ortiz\_ort@hotmail.com

*Abstract*— The investigation, experimentation and application of the operation of the ordering algorithms is carried out.

***Keywords—Sorting, Array, Index***

*Resumen*—Se realiza la investigación , experimentaciòn y aplicación del funcionamiento de los Algoritmos de Ordenamiento

*Índices*— Ordenamiento, Arreglo, indice.

# Nomenclatura

Ordenamiento, Algoritmo, Arreglo, Ìndice

# Marco Teórico (reparación del Trabajo)

## Algoritmo de Ordenación

Debido a que las estructuras de datos son utilizadas para almacenar información, se debe poder recuperar dichos datos de manera eficiente, por lo que resulta imperativo que la información esté ordenada.

Los Algoritmos de ordenamiento nos permiten, como su nombre lo dice, ordenar información de una manera especial basándonos en un criterio de ordenamiento.

En la Computación, el Ordenamiento de Datos cumple un papel muy importante, ya sea como un fin en sí o como parte de otros procedimientos más complejos. Se han desarrollado muchas técnicas en éste ámbito, cada una con características específicas, con ventajas y desventajas sobre las demás.

## Ejemplos de Algoritmos de Ordenamiento

* Burbuja
  + Es un Sencillo algoritmo de ordenamiento. Funciona revisando cada elemento de la lista que va a ser ordenada con el siguiente, intercambiándolos de posión si están en el orden equivocado.
* Quick Sort
  + El ordenamiento rápido (*Quicksort* en inglés) es un algoritmo basado en la técnica de **divide y vencerás**, que permite, en promedio, ordenar n elementos en un tiempo proporcional a **n log n**. Esta es **la técnica** de ordenamiento **más rápida conocida**. Fue desarrollada por C. Antony R. Hoare en 1960. El algoritmo original es recursivo, pero se utilizan versiones iterativas para mejorar su rendimiento (los algoritmos recursivos son en general más lentos que los iterativos, y consumen más recursos)
* Comb Sort
  + El algoritmo *CombSort* **mejora el algoritmo de ordenamiento de burbuja** y **rivaliza en velocidad con** algoritmos más complejos como **el *Quicksort***. La idea básica es eliminar *tortugas*, o pequeños valores cerca del final de la lista, ya que en el algoritmo de ordenamiento de burbuja esto reduce la velocidad de ordenamiento tremendamente.
* Merge Sort
  + El algoritmo de **ordenamiento por mezcla** (*MergeSort* en inglés) es un algoritmo de ordenamiento externo estable **basado en la técnica divide y vencerás**. Es de complejidad O (*n* log *n*).

# Desarrollo

Las puerbas se realizaron en dos diferentes equipos en diferentes sistemas operativos, los cuales se detallan a continuación:

## Especificación Técnica de los Equipos

Computadora de escritorio AMD

* Procesador: AMD A8 7600 de 4 Núcleos.
* Frecuencia del Procesador: 3.1GHz.
* Memoria RAM: 4 GB
* Sistemas Operativos:
  + Windows 10 64 Bits
  + Debian 9.8

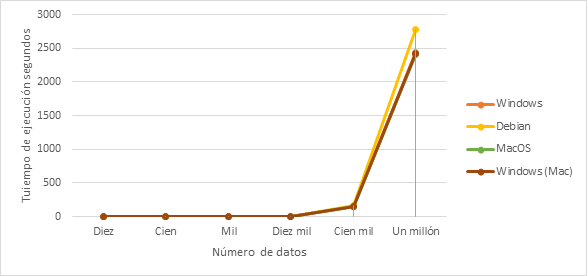
Computadora MacBook Pro 2012

* Procesador: Intel Core i5 de 2 Núcleos.
* Frecuencia del Procesador: 2.5 GHz
* Memoria RAM: 10 GB
* Sistemas Operativos:
  + MacOS Mojave 10.14.03
  + Windows 10 64 Bits (Bootcamp)

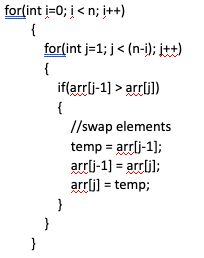
## Resultados de las Pruebas

* Burbuja

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sistema operativo | Numero de datos | | | | | |
| Diez | Cien | Mil | Diez mil | Cien mil | Un millón |
| Windows | 0.0000166222 | 0.0022008800 | 0.0096050056 | 0.5780208378 | 150.0332734608 | 2436.960532 |
| Debian | 0.0000161844 | 0.002891705 | 0.013343029 | 0.701033274 | 167.9581275 | 2772.191071 |
| MacOS | 0.000014701 | 0.00189838 | 0.014453084 | 0.652948977 | 144.5729916 | 2401.474062 |
| Windows (Mac) | 0.000015354 | 0.001855588 | 0.007500683 | 0.565626785 | 150.8918597 | 2415.537935 |

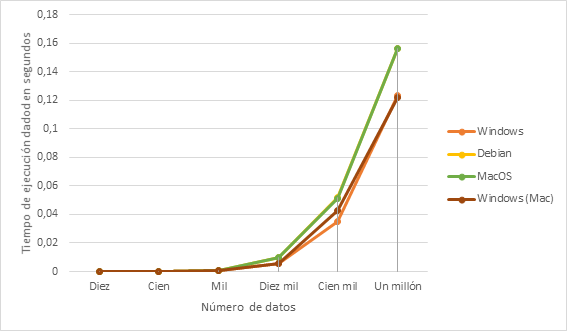


* + Código

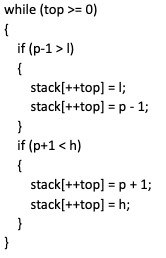


* QuickSort

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sistema operativo |  | | | | | |
| Diez | Cien | Mil | Diez mil | Cien mil | Un millón |
| Windows | 0.0000141778 | 0.0001586934 | 0.0009775824 | 0.0055017604 | 0.0349053024 | 0.123515838 |
| Debian | 0.0000152474 | 0.000201109 | 0.001068972 | 0.010163202 | 0.051754803 | 0.156282101 |
| MacOS | 0.0000167302 | 0.000189025 | 0.000974811 | 0.009737339 | 0.050833171 | 0.156637934 |
| Windows (Mac) | 0.0000161746 | 0.000184902 | 0.000645925 | 0.005391388 | 0.042699635 | 0.121998084 |

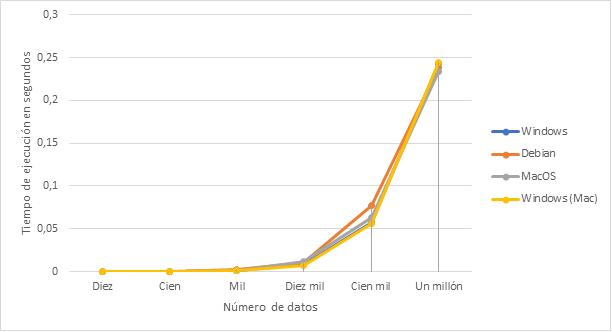


* + Código

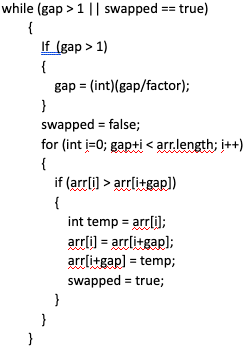


* CombSort

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sistema operativo | Numero de datos | | | | | |
| Diez | Cien | Mil | Diez mil | Cien mil | Un millón |
| Windows | 0.0000125158 | 0.000217458 | 0.0016659378 | 0.0080265788 | 0.0573472608 | 0.243336653 |
| Debian | 0.000013115 | 0.000295514 | 0.001965011 | 0.010686161 | 0.076825973 | 0.238763476 |
| MacOS | 0.000018669 | 0.000345378 | 0.001158007 | 0.011649916 | 0.063166325 | 0.233659171 |
| Windows (Mac) | 0.000014286 | 0.000211833 | 0.001113271 | 0.006722567 | 0.056336572 | 0.244554307 |

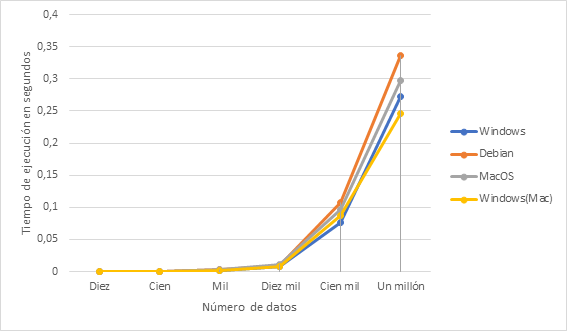


* + Código

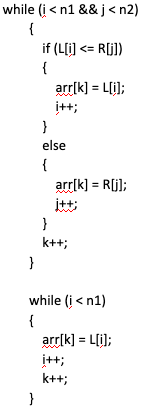


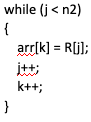
* MergeSort

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sistema operativo | Numero de datos | | | | | |
| Diez | Cien | Mil | Diez mil | Cien mil | Un millón |
| Windows | 0.0001005156 | 0.0005286846 | 0.0013131556 | 0.0081175118 | 0.0769274588 | 0.272131235 |
| Debian | 0.0000700378 | 0.000596387 | 0.002718321 | 0.010153939 | 0.10675137 | 0.337159655 |
| MacOS | 0.000112543 | 0.000582661 | 0.002388178 | 0.010950988 | 0.096968048 | 0.297264961 |
| Windows (Mac) | 0.0000807924 | 0.000501255 | 0.001297598 | 0.007053371 | 0.087544445 | 0.245949747 |



* + Código





# Conclusiones

Los métodos de ordenamiento de datos son muy útiles, ya que la forma de arreglar los registros de una tabla en algún orden secuencial de acuerdo a un criterio de ordenamiento, el cual puede ser numérico, alfabético o alfanumérico, ascendente o descendente. Nos facilita las búsquedas de cantidades de registros en un moderado tiempo, en modelo de eficiencia.

Después de realizar el presente proyecto, los conocimientos adquiridos acerca de los algoritmos de ordenamiento aumentaron y se aclararon muchas dudas con respecto a los nombrados anteriorimente. Existieron algunas dificultades durante el proceso de desarrollo, sin embargo, se resolvieron de forma deseada.

Con los resultados obtenidos es fácil deducir que QuickSort es un de los algoritmos más rápidos para una mayor cantidad de datos a ordenar, por otra parte la maquina que ordeno de manera más rápida fue la computadora MacBook Pro destacando el sistema operativo Windows corriendo en BootCamp que ejecuto en menor tiempo el ordenamiento en la mayoría de los casos, a pesar de que sus características son menores a las de la computadora de escritorio.

# Agradecimientos

Agredecemos de forma especial al Profesor Miguel Ángel Vargas Lomeli.

# Referencias

1. https://www.abc.es/local-comunidad-valenciana/20121218/abci-blanco-201212181131.html
2. https://es.wikipedia.org/wiki/Led#Led\_blanco\_y\_evoluci%C3%B3n
3. https://repositorio.unican.es/xmlui/bitstream/handle/10902/7111/Daniel%20Garcia%20Diaz.pdf;jsessionid=66AADE7B422A0099F8CE63B3160E5A42?sequence=1