METODOS DE ORDENAMIENTO

Lina María Muñoz Ospina

Fausto Antonio Andrade vera

[Linita26.jl@gmail.com](mailto:Linita26.jl@gmail.com) [ayf1207@gmail.com](mailto:ayf1207@gmail.com)

**Resumen**

**Debido a que las estructuras de datos son utilizadas para almacenar información, para poder recuperar esa información de manera eficiente es deseable que aquella esté ordenada. Existen varios métodos para ordenar las diferentes estructuras de datos básicas.**

**En general los métodos de ordenamiento no son utilizados con frecuencia, en algunos casos sólo una vez. Hay métodos muy simples de implementar que son útiles en los casos en dónde el número de elementos a ordenar no es muy grande (ejemplo, menos de 500 elementos). Por otro lado hay métodos sofisticados, más difíciles de implementar pero que son más eficientes en cuestión de tiempo de ejecución.**

**Los métodos sencillos por lo general requieren de aproximadamente n x n pasos para ordenar n elementos.**

**Los métodos simples son: insertionsort (o por inserción directa) selectionsort, bubblesort, y shellsort, en dónde el último es una extensón al insertionsort, siendo más rápido. Los métodos más complejos son el quick-sort, el heapsort, radix y addresscalculationsort. El ordenar un grupo de datos significa mover los datos o sus referencias para que queden en una secuencia tal que represente un orden, el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente.**

**Palabras Claves**

**Estructuras, métodos, sofisticados, tiempo de ejecución, referencias, secuencia.**

**Abstrac**

**Becausethe data structures are used to store information, In order to retrieve this information efficiently it is desirable that the information be ordered. There are several methods for sorting the different basic data structures.**

**In general, sortingmethods are notusedfrequently, in some cases only once.**

**There are very simple methods to implement that are useful in cases where the number of items to order is not very large (example, Less than 500 elements). On the other hand, there are sophisticated methods that are more difficult to implement but are more efficient in terms of execution time.**

**Simple methods usually require approximately n x n steps to sort n elements.**

**The simple methods are: Insertion sort (or by direct insertion) selection sort, bubble sort, and shellsort, where the last is an extension to insert sort,   
Beingfaster. Los métodos más complejos son el quick-sort, el heapsort, radix y addresscalculationsort.**

**Sorting a data group means moving the data or its references so that they are in a sequence that represents an order, which can be numeric, alphabetical or even alphanumeric, ascending or descending.**

**Keywords**

**Structures, methods, sophisticated, runtime, references, sequence**

1. **Introducción**

Los algoritmos de ordenamiento nos permiten, como su nombre lo dice, ordenar. En este caso, nos servirán para ordenar vectores o matrices con valores asignados aleatoriamente. Nos centraremos en los métodos más populares, analizando la cantidad de comparaciones que suceden, el tiempo que demora y revisando el código, escrito en Java, de cada algoritmo. Este informe nos permitirá conocer más a fondo cada método distinto de ordenamiento, desde uno simple hasta el más complejo. Se realizaran comparaciones en tiempo de ejecución, pre-requisitos de cada algoritmo, funcionalidad, alcance, etc

1. **Metodología Utilizada**

Se realizaron consultas a través de la internet sobre los métodos de ordenamiento para realizar la implementación a través de un le guaje de programación (python) y a si obtener los resultados de cada uno de los métodos de ordenamiento investigados por el grupo de trabajo.

1. **Algoritmos usados**

El ordenar un grupo de datos significa mover los datos o sus referencias para que queden en una secuencia tal que represente un **orden**, el cual puede ser numérico, alfabético o incluso alfanumérico, ascendente o descendente.

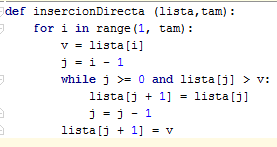
* Ordenamiento inserción
* Ordenamiento Mezcla
* Ordenamiento HeapSort
* Ordenamiento quicksort
* Ordenamiento countingsort
* Ordenamiento Radixsort

**Lenguaje de programación usado**

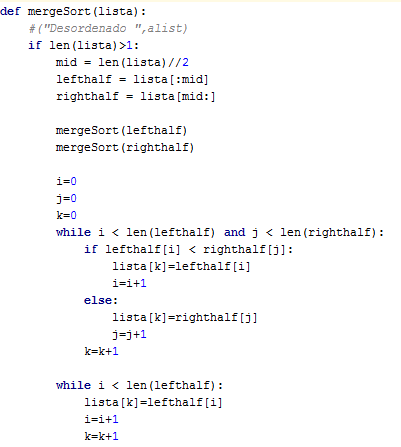
Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

**Código de los algoritmos implementados**

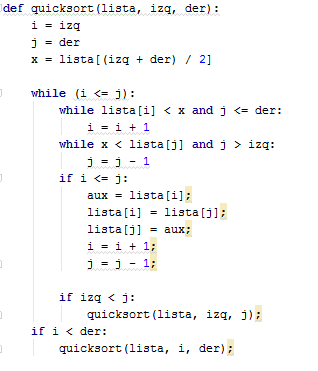
**Inserción**

****

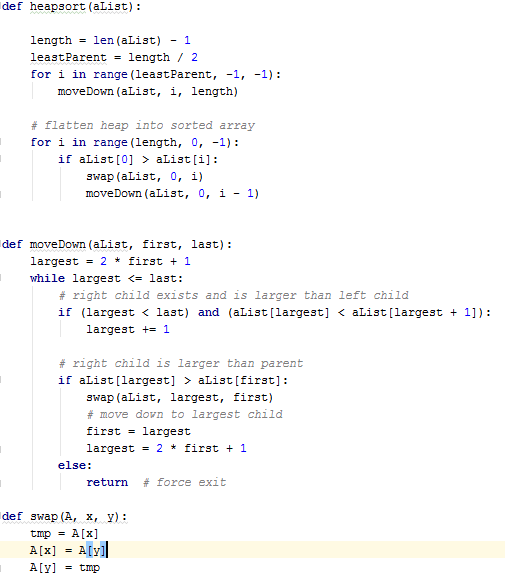
**Mezcla**



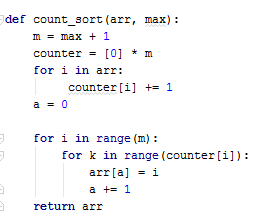
**Quick Sort**

****

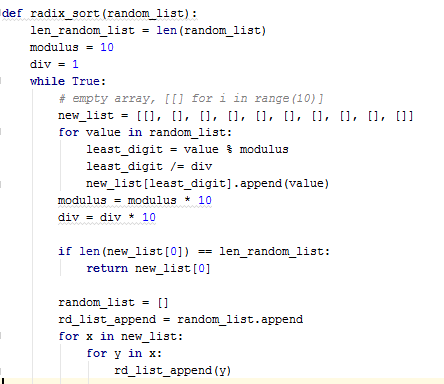
**HeapSort**

****

**Counting Sort**



**Radix Sort**



**Conclusiones**

Podemos decir que los resultados obtenidos por medio de los métodos de ordenamiento son muy inestables (variación) ya que los datos a ordenar son aleatorios, y también debemos tener en cuenta la participación de otras tareas realizadas por el ordenador al momento de ejecutar la consulta de los datos aleatorios.

**Evaluación de los algoritmos implementados.**

La evaluación de los resultados no nos dan una idea en los tiempos de ejecución, no podemos determinar las diferencias entre tiempo y tiempo para asegurar que método es más eficiente con certeza.

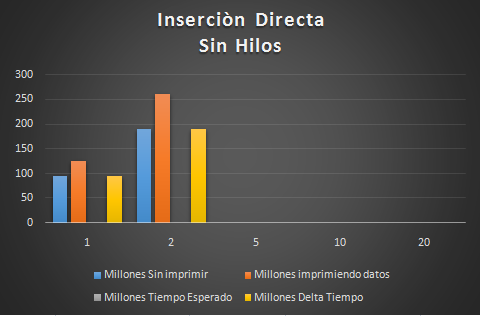
**Recomendaciones**

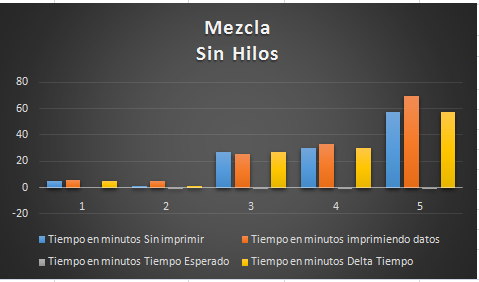
Establecer una cantidad para que los resultados sean más coherentes con los tiempos y métodos de ordenamiento.

**Resultados**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Inserción directa Sin Hilos | | | | |
| Complejidad = O(n2) - inserciòn | | | | |
| Número de datos | Millones | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 94 | 126 | 4,957E-09 | 94 |
| 2 | 190,4 | 260,2 | 1,377E-08 | 190 |
| 5 | Sin Respuesta | Sin Respuesta | 8,603E-08 | - |
| 10 | Sin Respuesta | Sin Respuesta | 3,442E-07 | - |
| 20 | Sin Respuesta | Sin Respuesta | 0,000001377 | - |

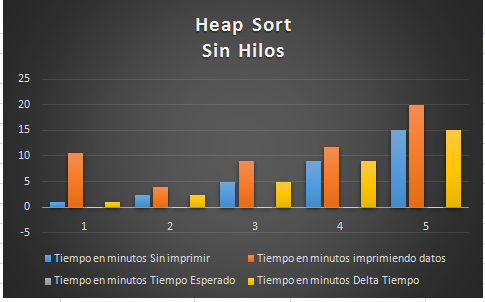
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Mezcla | | | | |
| Complejidad = O(n log n) - Mezcla | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 5,1 | 5,7 | 6 | 0,9 |
| 2 | 1,3 | 4,8 | 6,301029996 | 5,0010 |
| 5 | 27,3 | 25,3 | 6,698970004 | 5,3997 |
| 10 | 30,3 | 33,1 | 7 | 30,3105 |
| 20 | 57,6 | 69,7 | 7,301029996 | 57,6185 |

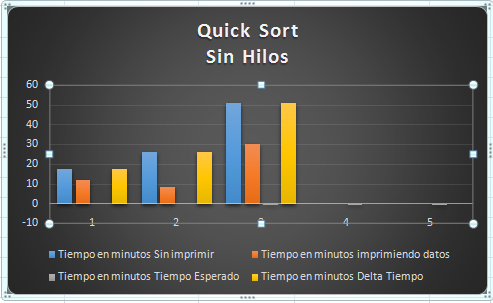




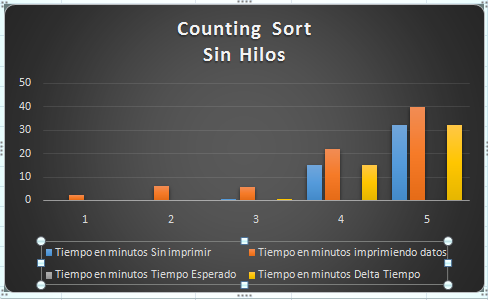
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Heap Sort | | | | |
| Complejidad = O(n log n) - Selecciòn | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 1 | 10,5 | -0,00002531 | 1,0000 |
| 2 | 2,3 | 4 | -0,00004607 | 2,3000 |
| 5 | 5 | 9 | 0,006006 | 5,006 |
| 10 | 9 | 11,74 | 0,0105 | 9,0105 |
| 20 | 15 | 20 | 0,0185 | 15,0185 |

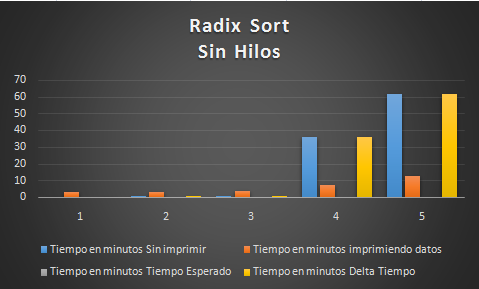
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Quick Sort | | | | |
| Complejidad = O(n log(n)) - No estable | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 17,4 | 12 | -0,00002531 | 17,40002531 |
| 2 | 26 | 8,4 | -0,00004607 | 26,00004607 |
| 5 | 51 | 30 | -0,006006 | 51,006006 |
| 10 | NO REG TIEMPO | NO REG TIEMPO | -0,0105 |  |
| 20 | NO REG TIEMPO | NO REG TIEMPO | -0,0185 |  |





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Radix Sort | | | | |
| Complejidad = O(n¨3) - No Comparativo | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 0,5 | 3,3 | 9,388E-11 | 0,50000 |
| 2 | 0,6 | 3,5 | 7,51E-10 | 0,60000 |
| 5 | 0,85 | 4 | 1,172E-08 | 0,85000 |
| 10 | 36 | 7,5 | 9,388E-08 | 36,00000 |
| 20 | 61,5 | 13 | 0,000000751 | 61,50000 |

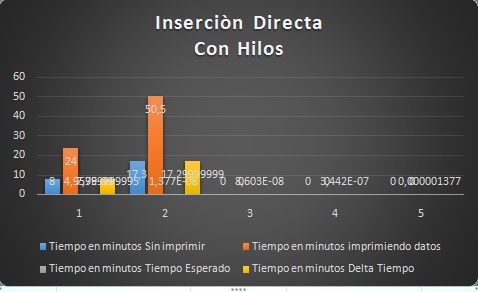
****

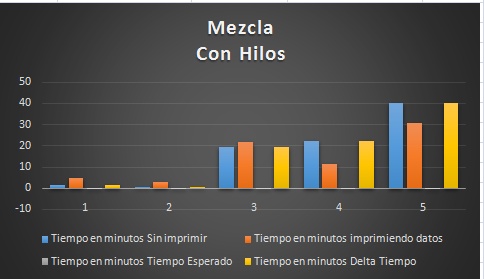


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Counting Sort | | | | |
| Complejidad = O(n+k) - No Compartivo | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 0,3 | 2,1 | 0,000007575 | 0,299992425 |
| 2 | 0,3 | 6 | 0,00001515 | 0,29998485 |
| 5 | 0,6 | 5,6 | 0,002272 | 0,597728 |
| 10 | 15,2 | 22 | 0,00003786 | 15,19996214 |
| 20 | 32 | 40 | 0,0001515 | 31,9998485 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - inserción directa Con Hilos | | | | |
| Complejidad = O(n2) - inserciòn | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 8 | 24 | 4,957E-09 | 7,999999995 |
| 2 | 17,3 | 50,5 | 1,377E-08 | 17,29999999 |
| 5 | NO REGISTRA | NO REGISTRA | 8,603E-08 |  |
| 10 | NO REGISTRA | NO REGISTRA | 3,442E-07 |  |
| 20 | NO REGISTRA | NO REGISTRA | 0,000001377 |  |

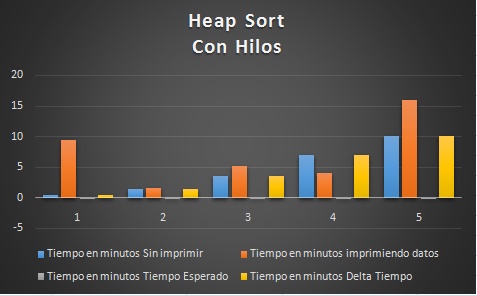
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Mezcla | | | | |
| Complejidad = O(n log n) - Mezcla | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 1,7 | 4,8 | -0,00002531 | 1,70002531 |
| 2 | 0,4 | 2,9 | -0,00004607 | 0,40004607 |
| 5 | 19,6 | 22 | -0,006006 | 19,606006 |
| 10 | 22,2 | 11,4 | -0,0105 | 22,2105 |
| 20 | 40,3 | 30,8 | -0,0185 | 40,3185 |

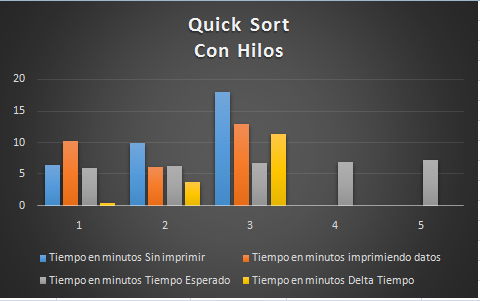




|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Heap Sort Con hilos | | | | |
| Complejidad = O(n log n) - Selecciòn | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 0,4 | 9,5 | -0,00002531 | 0,40002531 |
| 2 | 1,4 | 1,6 | -0,00004607 | 1,40004607 |
| 5 | 3,5 | 5,2 | -0,006006 | 3,506006 |
| 10 | 7 | 4 | -0,0105 | 7,0105 |
| 20 | 10 | 16 | -0,0185 | 10,0185 |

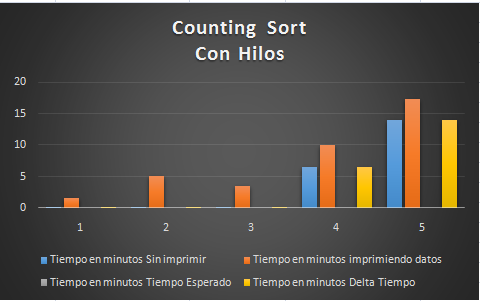
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Quick Sort Con Hilos | | | | |
| Complejidad = O(n log(n)) - No estable | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 6,5 | 10,2 | 6 | 0,5 |
| 2 | 10 | 6,2 | 6,301029996 | 3,698970004 |
| 5 | 18 | 13 | 6,698970004 | 11,30103 |
| 10 | NO REGISTRA | NO REGISTRA | 7 |  |
| 20 | NO REGISTRA | NO REGISTRA | 7,301029996 |  |

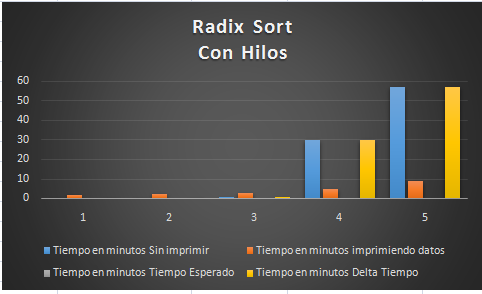




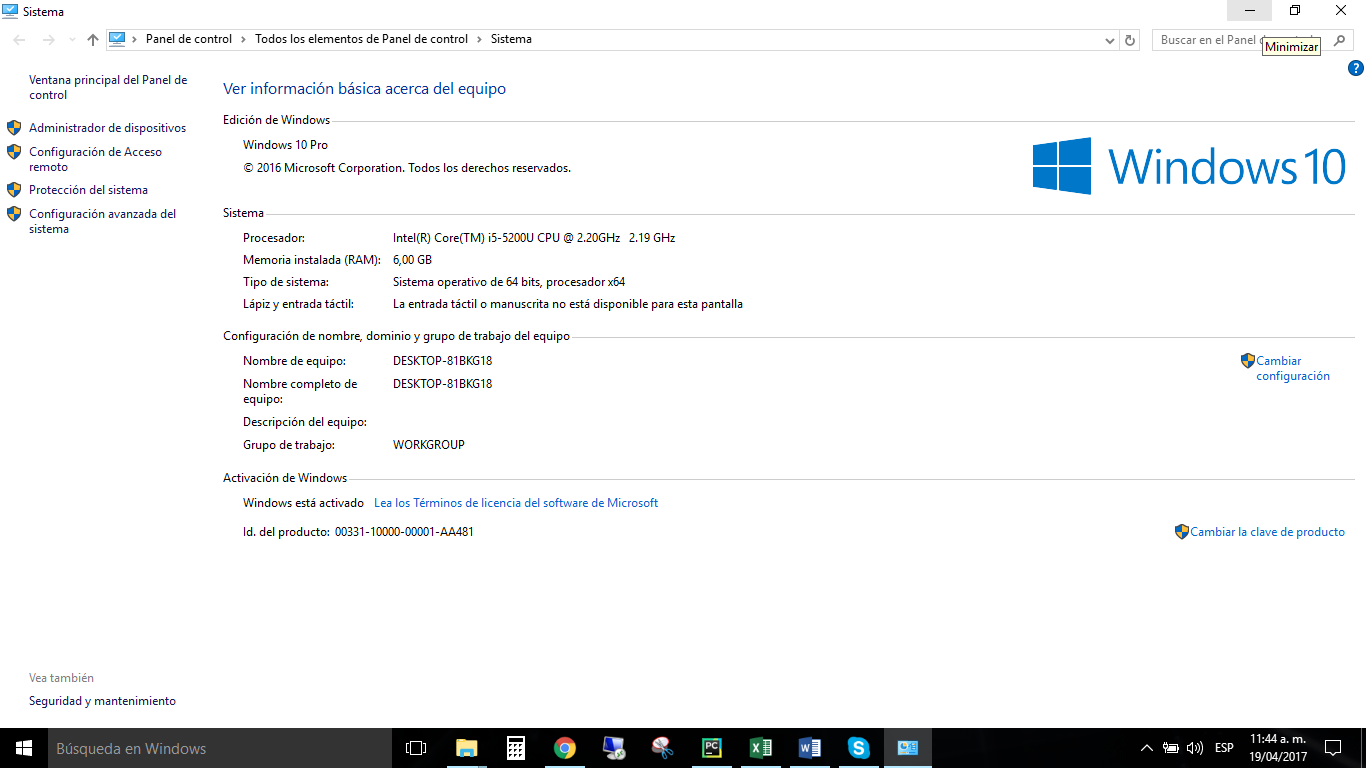
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Counting Sort | | | | |
| Complejidad = O(n+k) - No Compartivo | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 0,1 | 1,6 | 0,000007575 | 0,099992425 |
| 2 | 0,1 | 5 | 0,00001515 | 0,09998485 |
| 5 | 0,1 | 3,5 | 0,002272 | 0,097728 |
| 10 | 6,5 | 10 | 0,00003786 | 6,49996214 |
| 20 | 14 | 17,4 | 0,0001515 | 13,9998485 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algoritmo - Radix Sort | | | | |
| Complejidad = O(n¨3) - No Comparativo | | | | |
| Número de datos | Tiempo en minutos | | | |
| Millones | Sin imprimir | imprimiendo datos | Tiempo Esperado | Delta Tiempo |
| 1 | 0,3 | 1,8 | 9,388E-11 | 0,3 |
| 2 | 0,3 | 2,1 | 7,51E-10 | 0,299999999 |
| 5 | 0,6 | 2,7 | 1,172E-08 | 0,599999988 |
| 10 | 30 | 5 | 9,388E-08 | 29,99999991 |
| 20 | 57 | 9 | 0,000000751 | 56,99999925 |





**Computador Usad**o



**References Bibliograficas**

<http://iutprogramacion.blogspot.com.co/2013/02/metodos-de-ordenamiento.html>

<https://blog.zerial.org/ficheros/Informe_Ordenamiento.pdf>

<http://c.conclase.net/orden/>

<https://desarrolloweb.com/articulos/1325.php>