

**IMPLEMENTACIÓN DE HEAPS**

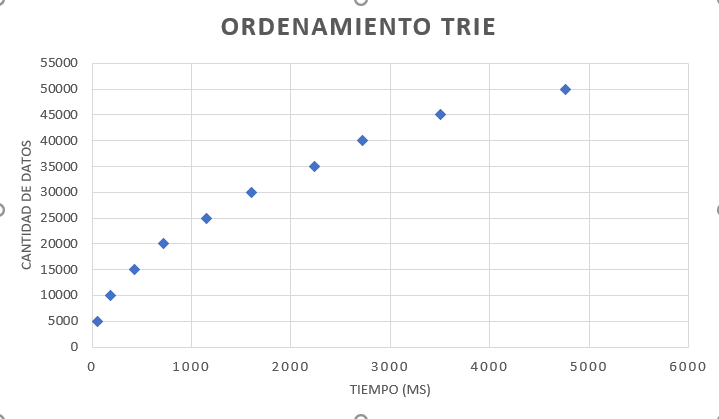
Montserrat Olivares Ruiz 179905

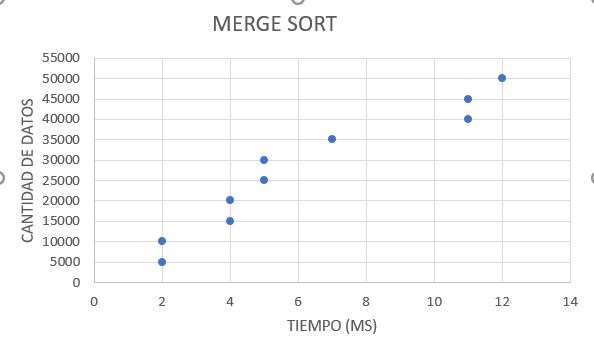
Estructuras de datos avanzadas

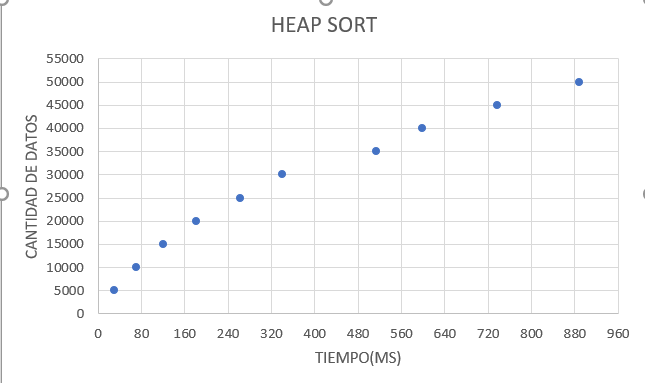
12/diciembre/2019

COMPARACIÓN DE TIEMPO (MS)

|  |  |  |  |
| --- | --- | --- | --- |
| **Cantidad de datos** | **Ordenamiento Trie** | **Merge Sort** | **Heap Sort** |
| 5000 | 52.00 | 2.00 | 30.00 |
| 10000 | 188.00 | 2.00 | 70.00 |
| 15000 | 426.00 | 4.00 | 120.00 |
| 20000 | 723.00 | 4.00 | 181.00 |
| 25000 | 1150.00 | 5.00 | 262.00 |
| 30000 | 1607.00 | 5.00 | 341.00 |
| 35000 | 2240.00 | 7.00 | 513.00 |
| 40000 | 2716.00 | 11.00 | 598.00 |
| 45000 | 3510.00 | 11.00 | 737.00 |
| 50000 | 4765.00 | 12.00 | 888.00 |







El objetivo de esta tarea era programar la funcionalidad de la estructura de montículo o heap (en este caso, con el menor elemento). Después había que programar el HeapSort u ordenamiento por montículos, donde primero se debían almacenar y ordenar los elementos recibidos en un arreglo en un montículo. Posteriormente se van extrayendo los nodos que van quedando como raíz del montículo en sucesivas iteraciones para finalmente obtener el conjunto ordenado. Se hizo un registro del tiempo que le tomaba realizar sus operaciones.

Se realizó una comparación entre los métodos Merge Sort, Heap Sort y el ordenamiento de Trie, realizando pruebas desde 5000 a 50000 elementos. En las tablas arriba presentadas se exhiben los resultados de los experimentos.

Sin importar las comparaciones, el método de Merge Sort resultó ser el más veloz, con el menor tiempo incluso conforme aumentaba el número de datos pasados como parámetros, teniendo un comportamiento asintótico de nlong(n). Por otro lado, el tiempo que le toma al Heap Sort y al Ordenamiento Trie va incrementando considerablemente conforme se incrementa el número de datos a insertar. Sin embargo, el Heap Sort resulta ser mucho más veloz que el Ordenamiento Trie, ya que con 50000 datos al Heap Sort le toma 888 ms y al Ordenamiento Trie le toma 4765 ms. Además, en algunos casos, el MergeSort llegó a tener un tiempo constante con incrementos de 5000 datos.

De las comparaciones podemos concluir que el Heap Sort es más eficiente que el Ordenamiento Trie, pero el Merge Sort sigue siendo más eficiente que los dos, ya que les sigue tomando más tiempo realizar las operaciones.