Comparación entre Merge, Trie y Heap Sort

Para comparar al Merge Sort y al Trie Sort se hicieron pruebas sobre conjuntos de miles de palabras, como se observará en la tabla y en la gráfica, y se midió el tiempo que cada uno tomaba.

|  |  |  |  |
| --- | --- | --- | --- |
| Cantidad de palabras | Merge (ms) | Trie (ms) | Heap (ms) |
| 5000 | 7 | 10 | 18 |
| 10000 | 22 | 16 | 11 |
| 15000 | 54 | 29 | 26 |
| 20000 | 20 | 12 | 22 |
| 25000 | 13 | 17 | 26 |
| 30000 | 19 | 26 | 29 |
| 35000 | 17 | 51 | 33 |
| 40000 | 22 | 51 | 30 |
| 45000 | 22 | 123 | 31 |
| 50000 | 25 | 101 | 41 |
| 55000 | 30 | 45 | 46 |
| 60000 | 28 | 59 | 50 |
| 65000 | 34 | 36 | 50 |
| 70000 | 35 | 40 | 65 |
| 75000 | 51 | 90 | 59 |
| 80000 | 62 | 68 | 72 |
| 85000 | 43 | 65 | 77 |
| 90000 | 61 | 78 | 81 |
| 95000 | 66 | 75 | 88 |
| Tiempo total | 631 | 992 | 855 |

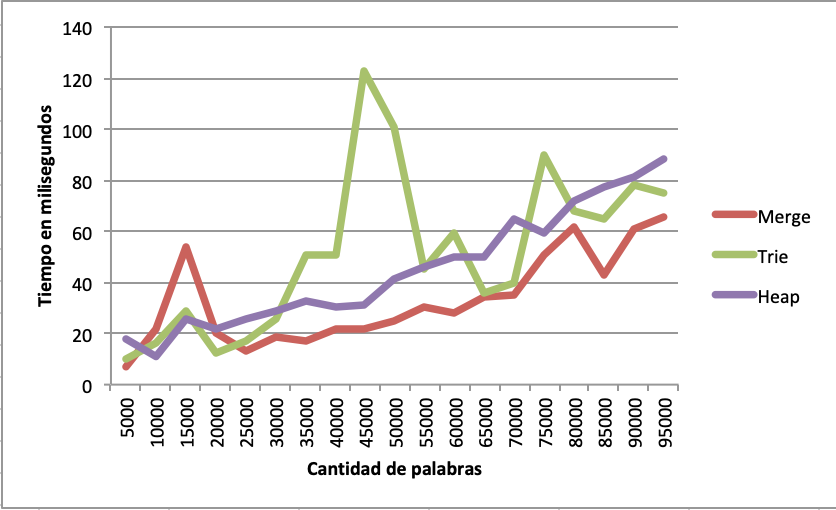
Tabla 1. Comparación de Merge Sort, Trie Sort y Heap Sort.

Como puede verse en las pruebas, el Merge Sort fue mejor; sólo hubo algunos casos en los que no. Por otro lado Trie Sort y Heap Sort se alternaron. Trie Sort fue más constante, pero, aún así, hubo casos en los que se tardó muchísimo más que los otros dos, quizás porque Merge Sort y Heap Sort pueden variar más de acuerdo a como sean dados los datos, mientras que el Trie Sort no varía realmente, más que en la longitud de las palabras, que es algo que los otros también pueden aprovechar.

Lo que sí es cierto es que el Merge Sort es mucho más rápido de codificar, pues hay muy pocos problemas que tienen que resolverse cuando ya se tiene la estructura. Por otro lado, Merge Sort y Trie deben tomar en cuenta algunos detalles que pueden hacer la codificación más tardada. Claro, una vez programado, se acabó y funcionan mejor.

Estas tareas ya nos hablan de cuán eficiente es el Merge Sort, y la verdad, creo que Trie Sort debería ser mucho mejor, pero no sé qué cosas podrían ser mucho más eficientes en mi programación.

En cuanto a memoria, Merge Sort usa en su programación óptima con listas ligas tiene una complejidad de O(n) de memoria, Heap Sort de O(2n), mientras que Trie sort usa en el peor de los casos O(tamañoDelAlfabetoUsado\*longitudMáxima\*n).



Gráfica 1. Gráfica hecha a partir de los datos de la Tabla 1.

Mientras que el Trie no puede ser usado en todas las ocasiones, Heap Sort y Merge Sort son buenos algoritmos. Su complejidad aproximada es O(nlogn) que es ideal para muchas situaciones.