Comparación entre Merge Sort y Trie 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) |
| 5000 | 7 | 10 |
| 10000 | 22 | 16 |
| 15000 | 54 | 29 |
| 20000 | 20 | 12 |
| 25000 | 13 | 17 |
| 30000 | 19 | 26 |
| 35000 | 17 | 51 |
| 40000 | 22 | 51 |
| 45000 | 22 | 123 |
| 50000 | 25 | 101 |
| 55000 | 30 | 45 |
| 60000 | 28 | 59 |
| 65000 | 34 | 36 |
| 70000 | 35 | 40 |
| 75000 | 51 | 90 |
| 80000 | 62 | 68 |
| 85000 | 43 | 65 |
| 90000 | 61 | 78 |
| 95000 | 66 | 75 |

Tabla 1. Comparación de Merge Sort y Trie Sort.

Como puede verse en las pruebas, el Merge Sort fue mejor; sólo hubo algunos casos en los que no. Quizás esto se deba a que el Merge Sort si pueda variar en cuanto a cómo estén acomodadas las palabras, y el Trie no. El Trie en general se tarda lo mismo, pues de todas formas construye el árbol. La única ventaja que tendría es si las palabras son de una menor longitud, pero esta ventaja también sería aprovechada por el Merge Sort al realizar menos comparaciones de caracteres con el uso del comparteTo de la clase String.

Aún así, recordamos que Merge Sort tenía una gran supremacía sobre los otros algoritmos de ordenamiento, así que el Trie Sort, al estar tan cercano en tiempo y utilizar menos memoria auxiliar, también es un gran algoritmo.

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

El problema del Trie es que no puede usarse tan universalmente como los otros algoritmos que podían usarse siempre que se tratara de objetos comparables. Este sólo puede utilizarse cuando los objetos son construidos a partir de una misma base de *mini-objetos*, como en este caso todas las palabras están hechas de letras. El hecho de que estos *mini-objetos* puedan compararse o no es menos problemático, pues nosotros podemos asignarles un orden en función de lo que necesitemos.

Podemos concluir que el Trie es un gran algoritmo siempre que se trate de colección de objetos hechos a base de *“mini-objetos”*. Además, esta estructura no sólo es útil para ordenar, sino para encontrar prefijos comunes y usar de manera más eficiente la memoria.