

TAREA TRIE

ANTONIO TOLEDO GALINDO

CU: 184166

PROFESOR: FERNANDO SPONDA DARLINGTON

MATERIA: ESTRUCTURAS DE DATOS AVANZADAS

|  |  |  |
| --- | --- | --- |
| Datos | Merge(ms) | Trie(ms) |
| 500 | 1 | 2 |
| 1000 | 1 | 3 |
| 10000 | 9 | 12 |
| 20000 | 15 | 25 |
| 50000 | 79 | 55 |

Previo.

Utilizamos el Trie con arreglos en el que aún no habíamos implementado las HashTable para cambiar la complejidad.

Análisis

Complejidad Ordenar (Teórico)

Merge: O(n)= log(n)

Trie: O(n)= 1 (constante)

Pero como el Trie no tiene Hash esa no es su verdadera complejidad

El merge es mejor en cada uno de los casos excepto en el de 50 mil, eso demuestra que con más palabras puede volverse más eficiente el Trie por la manera en que puede guardar varias palabras con prefijos iguales en el mismo lugar e imprimirlos de manera más rápido.

Hace falta ver cómo cambia el resultado al implementar el Trie con Hash.

**No se puede utilizar en todos los casos que los otros Sorts**

Solo se puede utilizar si los elementos están conformados por cosas que se puedan guardar en un arreglo (como por ejemplo formados de caracteres).

Este método es útil principalmente cuando tienes elementos con prefijos iguales, y solo si son muchos elementos, con una cantidad de datos chica son preferibles los otros métodos.

También es muy útil cuando tienes tantos datos que se utiliza la memoria secundaria, entonces puedes utilizar menos llamadas al traer los datos en menos llamadas.