

TAREA HEAP

ANTONIO TOLEDO GALINDO

CU: 184166

PROFESOR: FERNANDO SPONDA DARLINGTON

MATERIA: ESTRUCTURAS DE DATOS AVANZADAS

**Implementación Heap Sort**

Creamos una estructura de datos minHeap con funciones para insertar, eliminar el mínimo y también obtenerlo. Para implementar esta estructura cree el método heapSort que lo que hace simplemente es introducir todos los elementos y sacarlos, pero al sacar los elementos saca el menor cada vez ordenándolos de esta manera. Utilicé las mismas palabras con las mismas cantidades que en la tarea de la implementación de Tries y también la misma clase Trie y merge sort.

Resultados en milisegundos contra la cantidad de palabras

|  |  |  |  |
| --- | --- | --- | --- |
| Palabras | Heap(ms) | Trie(ms) | Merge(ms) |
| 500 | 0.4 | 1.4 | 0.2 |
| 1000 | 1.6 | 2.7 | 1.2 |
| 10000 | 7.1 | 13.6 | 3.5 |
| 20000 | 18 | 27.8 | 7 |
| 50000 | 51.3 | 62 | 16.4 |

Podemos observar que el merge sort sigue siendo el más efectivo por bastante tiempo, aunque el método del Heap si es más rápido que el del Trie. Aunque en la teoría el más rápido de ellos debería de ser el Trie con tiempo constante o muy cercano a constante, el problema se encuentra en que solo es así si se implementa con tablas de hash, cosa que no hicimos. La complejidad del Heap es 0(2n) por lo que en realidad no debería de distar de una manera tan estrepitosa del merge sort óptimo con su 0(n).

Esto nos demuestra la importancia del merge sort y su metodología y también las del log(n)