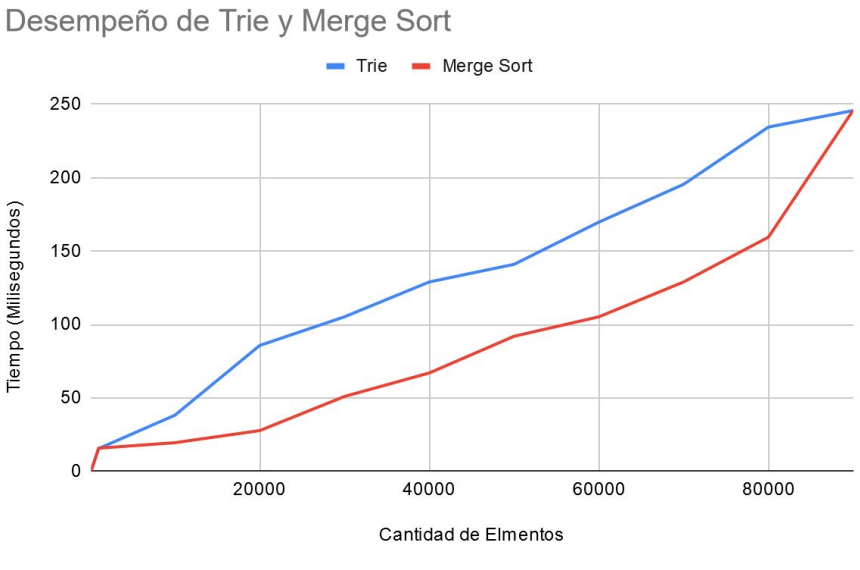
Jorge Edgar Rodríguez Ortiz Loyola 181334

Estructuras de Datos Avanzadas

Tarea Trie

La tarea consistió en programar la clase Trie implementando las operaciones de buscar, insertar y borrar. Una vez que se tiene programada la clase Trie se realizó un experimento para probar su desempeño. Se leyeron 90 mil palabras de un archivo de texto, se guardaron en un arreglo (pasando todo a minúsculas e ignorando palabras con caracteres especiales para que fueran compatibles con nuestra lista de símbolos de Trie) y se insertaron distintas longitudes de este arreglo en un Trie o en un nuevo arreglo auxiliar. A continuación se ordenó el arreglo auxiliar con Merge Sort (previamente programado en clase) y se mandó a llamar al método Ordenamiento Lexicográfico en Trie. Se tomó el tiempo de ambos procesos, los resultados son los siguientes:

	Tiempo (Milisegundos)	
# Elementos	Trie	Merge Sort
100	0,25	0
1000	15,5	15,75
10000	38,25	19,5
20000	85,75	27,75
30000	105,25	51
40000	129	67
50000	141	92
60000	169,75	105,25
70000	195,5	129
80000	234,5	159,5
90000	245,75	246



Los resultados indican que el algoritmo de Merge Sort es más eficiente, aunque el Trie presenta un crecimiento más lineal lo que podría ser útil en algunos casos. Sería posible mejorar el Trie utilizando tablas de hash para mejorar la busqueda del siguiente caracter, pues en nodos con pocos elementos de su hijo ocupados se pierde eficiencia.

Lamentablemente no es posible utilizar un Trie como sustituto a los algoritmos de ordenación previamente vistos, pues solo puede ordenar objetos compuestos. Tal es el caso de un String, compuesto por caracteres, en los que para realizar un CompareTo hay que irse fijando en cada uno de los subelementos del String.