Integrante:

Stalin Franco 27.922.928

**Planteamiento del problema:**

Se desea implementar un diccionario de palabras en ingles haciendo uso de árboles trie, los trie son arboles de búsqueda, donde la clave de búsqueda es seccionada en claves mas pequeñas, de forma que estas sean usadas para armar la estructura del árbol de búsqueda.

Este método es particularmente conveniente con un diccionario, ya que las palabras se pueden seccionar en letras y cada una de ellas se puede usar de indice para crear una estructura.

**Diseño de la solución:**

Un árbol es una estructura auto-referenciada, por lo que lógicamente necesita de tener en su estructura uno o mas apuntadores a si mismo, en el caso de tener uno seria solo una lista enlazada, con dos en cambio se puede hacer un árbol, en el que cada uno de los apuntadores son dos hijos del nodo actual. Ahora, teniendo en cuenta estas características el árbol trie puede tener una cantidad grande de apuntadores, donde cada uno de ellos representa una parte de la clave inicial.

Los arboles trie en el caso de nuestro diccionario puede tener dos implementaciones principales, una con listas enlazadas con apuntadores y otra con arreglos, cada una con sus ventajas particulares.

En implementación con arreglos el acceso a memoria es mas rápido, lógicamente por la característica de los arreglos, en cambio con listas es al contrario, mayor eficiencia con respecto al ahorro de memoria, pero lógicamente el acceso de forma secuencial, que lo hará mas lento.

El punto a considerar es que tanta memoria habrá de diferencia entre ambas, teniendo en cuenta solo un nodo, en la estructura con arreglos se reservan 26 espacios de apuntadores, cada puntero en 64 bits ocupa 8 bytes, teniendo 208 bytes de memoria solo en apuntadores, por otro lado los significados considerando 200 caracteres máximos serian 200 bytes por significado, teniendo al rededor de unos 400 bytes en memoria, también siendo esta implementación la mas sencilla y fácil de corregir

Por otro lado la estructura basada en listas no tendría problemas con la memoria, ya que cada nodo solo tendría dos apuntadores, uno a la lista de los demás elementos y otro con los hijos, siendo estos 16 bytes, mas el significado que son 200 bytes, así que el ahorro de memoria es de la mitad, la velocidad de acceso en vez de ser de O(1) como en la implementación de arreglos, pasa a ser O(n) por la naturaleza de las listas enlazadas.