Licenciatura en ciencia de la computación

Algoritmo de Strassen

Complejidad

Profesor:
Nicolas Thériault

Autor:
Sergio Salinas
Danilo Abellá

Contents

1 Introducción

2 Estrategias utilizadas

2.1 Arreglo ordenado con Mergesort

Esta estrategia consiste en crear un arreglo de dos dimensiones, la primera dimensión tiene largo 21000 que es una estimación de la cantidad de palabras que tiene al archivo y cada arreglo asociado a este tiene un largo de 30, que también es una estimación de la cantidad de letras que tiene cada palabra. por lo que en total este método gasta 63000 espacios de memoria cada vez que se ejecuta.

2.2 Árbol AVL

Esta estrategia consiste en usar un árbol binario balanceado conocido como Árbol AVL, para la implementación se le hizo una ligera modificación que es agregar a la estructura de un nodo la frecuencia que este aparece, de esta forma en vez de apilar los elementos repetidos en nodos, solo aumenta el contador de frecuencia de este logrando una notoria mejora en memoria, ya que el árbol implementado solo va a ocupar tantos nodos como **palabras únicas** haya en el archivo. De está forma la estructura del árbol es la que se muestra en la tabla 1.

Table 1: Estructura de un nodo de un árbol AVL

Nodo
Data
Altura
Frecuencia
Puntero Izquierdo
Puntero Derecho

2.3 Tabla Hash

Para implementar el algoritmo con funciones de hash se utilizar un arreglo de largo fijo para almacenar las cabeceras de las listas enlazadas que contienen las palabras que se desean almacenar, este arreglo en un principio solo almacena elementos nulos, cuando se quiere ingresar una palabra se pasa esta palabra por una función hash y luego el resultado es utilizado como incide y se agrega a la lista que corresponde a ese indice, cada elemento es ingresado al inicio de la lista.

La función hash que utiliza el algoritmo es simplemente sumar cada letra del string y calcular su mod MAX, donde MAX, es un número que puede ser modificado por el usuario, MAX también es el largo del arreglo de punteros.

La tabla hash siempre va a guardar todas las palabras que tenga el archivo, pero se puede hacer un intercambio de tiempo memoria en el largo del arreglo de punteros, entre más largo sea más memoria utilizara pero habrán menos colisiones lo que se traduce en menos tiempo en operaciones de búsqueda (la búsqueda llega a ser de costo 1 en este caso si no hay colisión) y conteo, entre más corto menos memoria utilizara pero se gasta más tiempo en hacer las operaciones de búsqueda y conteo dentro de las listas enlazadas.

Para propósitos de la medición de tiempo se decidió que el largo del arreglo de punteros sea de largo 800 debido a que más allá de 800 no hay crecimiento en los indices únicos que

se pueden obtener por lo que es un desperdicio de memoria.

3 Soluciones a cada problema y tiempos

3.1 Generar la estructura

3.1.1 Mergesort

Tiempo: 0.084478

3.1.2 Árbol AVL

Tiempo:

3.1.3 Tabla Hash

Tiempo:

3.2 Encontrar una palabra

3.2.1 Mergesort

Tiempo si la palabra está:

Tiempo si la no palabra está:

3.2.2 Árbol AVL

Tiempo si la palabra está:

Tiempo si la no palabra está:

3.2.3 Tabla Hash

Tiempo si la palabra está:

Tiempo si la no palabra está:

3.3 Contar la cantidad de palabras distintas

3.3.1 Mergesort

Tiempo: 0.084478

3.3.2 Árbol AVL

Tiempo:

3.3.3 Tabla Hash

Tiempo:

3.4 Encontrar la palabra más utilizada

3.4.1 Mergesort

Tiempo: 0.084478

3.4.2 Árbol AVL

Tiempo:

3.4.3 Tabla Hash

Tiempo: