
Facultad de
Ciencias Exactas
Físicas y Naturales

# Algoritmos y Estructura de Datos

Segundo Parcial 2019

*Profesor:*
*Wolffman, Gustavo.*
*Aime, Ruben.*

*Alumnos:*
*Lujan, Martin - 39448179*

***- 2019-***

# Introducción

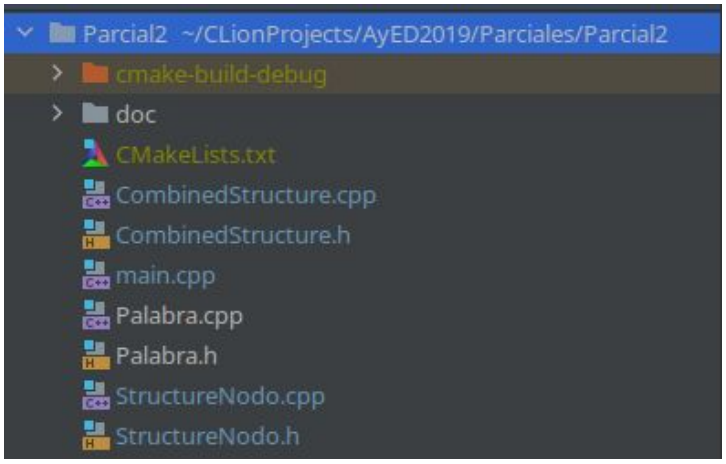
En el siguiente trabajo se deberá crear un programa con la capacidad de contar cuantas veces se repite cada palabra, leídas desde un texto con una cantidad mínima de 500. Se implementará una estructura combinada de listas y árbol binario de búsqueda. Se tendrá un campo para llevar la cuenta de la cantidad de veces que está repetida dicha palabra, para ordenar la lista por número de repeticiones (se deberá agregar un cuarto puntero). Para cada palabra se generará un nodo, que formará parte de una lista y de un árbol binario de búsqueda. Al finalizar la lectura del archivo, se deberá ordenar por el método de QuickSort sobre por cantidad de repeticiones, sin alterar el orden alfabético con la que fue armada la lista/ABB. Una vez finalizado el ordenamiento se mostraran la cantidad de comparaciones realizadas por el metodo de insercion frente al QuickSort. Por último, se generará un archivo con el listado ordenado por la cantidad de repeticiones y el orden alfabético.
Se realizará el proyecto en CLion con Archlinux como sistema operativo, finalmente el lenguaje de programación utilizado será C++.

# Desarrollo

En primera instancia se desgloso el problema en partes para poder tener una idea clara y por donde comenzar a escribir código, para el el proceso fuese siempre continuo y no tuviese que retroceder a corregir o refactorizar código.
Las clases que se implementaron son las siguientes:

- StructureNodo
- Palabra
- CombinedStructure
- main

Se adjunta imagen del proyecto:




Se optó por un número pequeño de clases debido a que cada una tiene una participación importante pero justa. Separar el código en más clases no se consideró necesario.

La implementación se realizó de la siguiente manera: la clase main se encarga de crear un objeto de la estructura combinada (CombinedStructure) para después proceder a abrir y leer el texto que se desea ordenar. Main extrae palabra por palabra y hace un chequeo de la existencia de mayúsculas (se evaluaran solo en minúscula), comas y puntos para almacenarlos en la estructura combinada.
Una vez insertados todos los elementos, main llama al método “QuickSort” de la clase CombinedStructure para realizar un ordenamiento mediante número de repeticiones de cada palabra creando así una nueva lista.
Finalmente, esta clase creó un archivo llamado “Texto-Ordenado” en el que se escribirán las palabras almacenadas alfabéticamente y por número de repeticiones. A esto se suma la muestra de los números de comparaciones que se utiliza para insertar las palabras alfabéticamente en la lista, en el árbol binario y por QuickSort.
La clase CombinedStructure contiene los métodos necesarios para insertar, ordenar en la Lista y en el ABB. Además contiene el método Quicksort que posteriormente será ejecutado.
StructureNodo contiene los campos y métodos necesarios para que la estructura funcione como Lista y Árbol Binario de búsqueda al mismo tiempo.
Por último se cuentra Palabra que es la clase necesaria para almacenar la palabra extraída del texto y llevar un conteo de las repeticiones que tiene dicha palabra.

El programa finaliza una vez que se ordenan (alfabéticamente y repetidas) todas las palabras del texto y son escritas en el nuevo texto antes mencionado. Nos muestra por pantalla algunos datos importantes, su salida es la siguiente:

1. Todas las palabras extraídas del texto:

```
/home/mlujan/CLionProjects/AyED2019/Parciales/Parcial2/cmake-build-debug/Parcial2
--> Se procede a leer el texto, se imprimiran por pantalla en el orden que se van

definiciÓn de drogadicciÓn  es una enfermedad que tiene su origen en el cerebro de
su progresiva y las recaidas  es el uso indebido de cualquier tipo de drogas con c
una dependencia síquica  cuyo individuo siente una imperiosa necesidad de tomar dr
terribles síntomas de abstinencia al no ingerirla  la drogadicción causa problemas
entre un organismo vivo y una droga  caracterizado por modificaciones del comporta
periódica con el fin de experimentar sus efectos síquicos y a veces para evitar el
uso compulsivo de este pero hay que diferenciar la dependencia física y síquica  e
dicho síndrome no se presenta  se debe entender que el adicto seguirá siendo un ad
droga y  de allí en adelante  éste será un adicto en remisión  no estará usando la
la guardia  vivimos en una cultura de la droga  desde la mañana cuando tomamos caf
casa  con un aperitivo alcohólico  o un inductor del sueño con un somnífero  recet
estas peripecias de la cotidianeidad  muchos además nos activamos a medida que tra
o depresión del sistema nervioso central  o que dan como resultado un trastorno en
de alterar el organismo y su acción psíquica la ejerce sobre la conducta  la perce
de toda sustancia tóxica  el término drogas visto desde un punto de vista estricta
farmacología y dentro de la medicina con un fármaco  es decir que droga y fármaco
prevención de enfermedades  los fármacos pueden elaborarse a partir de plantas  mi
```

2. Número de Comparaciones entre Lista, Árbol Binario y QuickSort

```
-->Por ultimo se genera un nuevo texto con las palabras ordenaras alfabeticamente y por repeticiones
-Comparaciones:
Insercion por lista: 69493 / Insercsion ABB: 4198 / Quicksort: 1983
Process finished with exit code 0
```

# Conclusión

El trabajo fue realizado con éxito y correcto funcionamiento. La idea de combinar lista con árbol en un mismo nodo genera una notable confusión al principio hasta entender que solo era un simple movimiento de punteros. QuickSort resultó relativamente fácil de aplicar una vez obtenida el método *Swap* funcionando correctamente.
Finalmente, por los datos arrojados por el programa, podemos observar la eficiencia de Quicksort para ordenar los datos pero además, la eficiencia del Árbol Binario de búsqueda frente a las listas, dando un rendimiento notablemente mejor para el almacenamiento de datos, dato clave que nos servirá para el desarrollo en el futuro.