UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Estructura de Datos y Análisis de Algoritmos

Informe Laboratorio 2

Ariel Ignacio Tirado Maturana

Profesor:
Jacqueline Kohler Casasempere
Alejandro Cisterna Villalobos

Ayudante:
Gerardo Zuñiga Leal

Santiago - Chile
2-2016

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

Desde la creación de la escritura, las personas han tratado de ocultar información a terceros, es por esto que se han creado algoritmos de cifrado que permiten que un texto se vuelva difícil de entender si no se conoce el algoritmo de descifrado. Un ejemplo de esto, es el cifrado Cesar, el cual consiste en recorrer el texto e intercambiar cada letra por otra, según un desplazamiento previamente definido.

Para el presente laboratorio, se les ha solicitado a los alumnos del curso de Análisis de Algoritmos y Estructuras de Datos realizar la implementación de este algoritmo, mediante el uso de Listas enlazadas, y que el programa abarque tanto el caso de cifrado como el de descifrado. Esto será programado en el lenguaje de programación C, el cual es uno de los mayores referentes de la programación imperativa.

Se espera que al finalizar la presente experiencia, se logré conseguir una mayor comprensión sobre el TDA Listas enlazadas, en proyección a buscar su utilidad en el área de la programación y ampliar la cantidad de herramientas disponibles al realizar algoritmos que se caractericen por trabajar con gran cantidad de datos.

A lo largo de la presente entrega, se le presentará al lector el problema actual y como fue diseñada la solución para este de una manera didáctica y precisa.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

2.1 ANÁLISIS DEL PROBLEMA

Para el presente laboratorio, se ha propuesto realizar un algoritmo de cifrado y descifrado de un texto, el cual será ingresado en un archivo de entrada llamado "*Entrada.in*". Este algoritmo entregará un resultado en un nuevo archivo llamado "*Salida.out*", donde se entregará la palabra cifrada o descifrada, según sea el caso. Además, se tiene la restricción de que la implementación del algoritmo debe ser realizada mediante el uso de dos Listas enlazadas, una llamada *letras* y otra llamada *codificacion*, en donde cada lista tendrá las letras, sin ser repetidas, del texto ingresado y en orden alfabético. El procedimiento se consistirá en ir desplazando la lista *codificacion* hacia la izquierda o derecha, en función de si la letra buscada es una vocal o consonante. Esto último hace que la elección del tipo de Lista enlazada sea la variante denominada *Lista doblemente enlazada circular*. A continuación se explica, mediante el uso de una imagen como funcionará este tipo de lista.

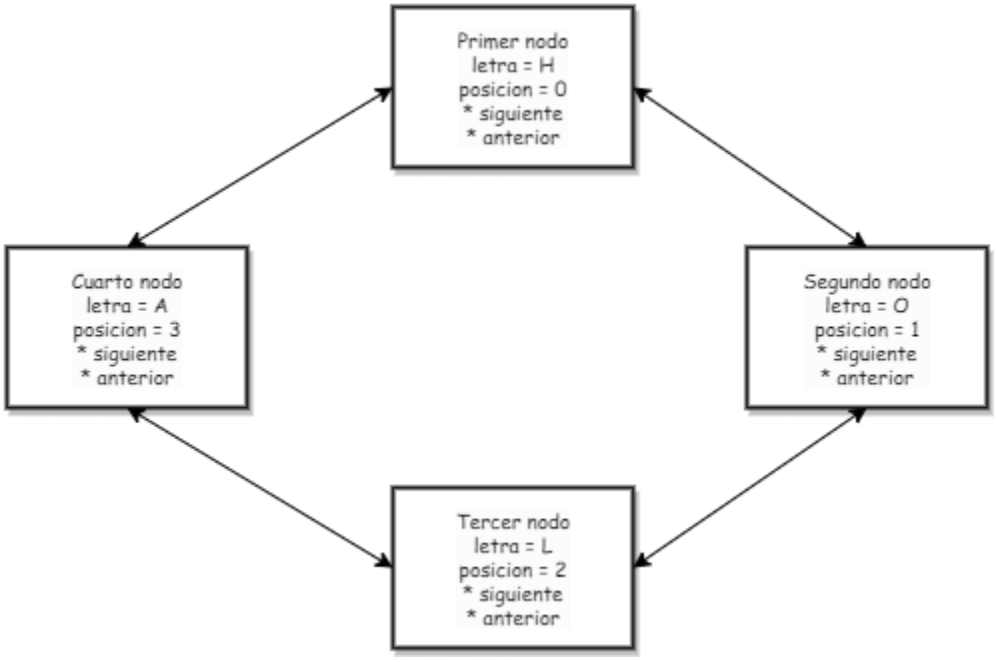

Imagen 2-1: Ejemplo lista doblemente enlazada circular con la palabra "HOLA"

La lista doblemente enlazada circular utiliza el mismo concepto de la lista enlazada, en que cada nodo tiene un puntero al siguiente nodo, con la diferencia de que se agrega por cada nodo un nuevo

puntero hacia el nodo anterior y además el último nodo apunta hacia el primer nodo y este a su vez apunta hacia el último, generando la conexión entre inicio y fin.

2.1.1 Cifrado

Para realizar el cifrado de un texto, este es almacenado en dos listas enlazadas distintas, una llamada *letras* y otra llamda *codificacion*, evitando que existan nodos con letras repetidas, además de que estas letras se encuentran almacenadas alfabéticamente. Una vez se encuentren los datos almacenados, se procede a realizar el proceso de cifrado, el cual consiste en leer letra a letra el texto, buscar esa letra en la lista *letras*, encontrar su posición en la lista y luego buscar en esa misma posición en la lista *codificacion*. Además, la lista *codificacion* se debe desplazar un espacio hacia la derecha si la letra buscada fue una consonante o tres espacios hacia la izquierda si la letra buscada era una vocal o un espacio. Finalmente se almacena la letra encontrada en la segunda lista en un arreglo de caracteres, el cual una vez que finalice el procedimiento, será escrito en el archivo de salida.

Realizando el desglose de lo dicho anteriormente, obtenemos los siguientes sub-problemas.

- Se debe permitir la lectura de archivos de texto en el formato solicitado.
- Se deben ordenar alfabéticamente las letras del texto leído y almacenar estas en las listas.
- Se debe recorrer la palabra original letra por letra y buscarlas en la lista *letras* y luego buscar en esa misma posición en la lista *codificacion,* para posteriormente almacenar la letra encontrada en un nuevo arreglo, para así generar finalmente el archivo de salida con el resultado obtenido.

2.1.2 Descifrado

Para el proceso de descifrado, se procede de manera similar al cifrado, pero realizando cada proceso de manera inversa. Para esto, se recibe como entrada un texto previamente cifrado y su correspondiente abecedario, cual se ordena alfabéticamente y se almacena en las listas *letras* y *codificacion.* Luego de esto, se busca letra por letra en el texto original y se verifica si la letra es una vocal, espacio o una consonante. Una vez encontrado que tipo de letra es,

se realiza el desplazamiento de la lista *codificacion* (un espacio hacia izquierda si es consonante o tres espacios hacia la derecha si es una vocal o un espacio) y se busca esta letra en la lista *codificacion*, luego se busca en esa misma posición la letra almacenada en la lista *letras* y se almacena en un arreglo, el cual al terminar el procedimiento será la palabra descifrada. Finalmente se escribe este arreglo en el archivo de salida.

Realizando el desglose de lo dicho anteriormente, se obtienen los siguientes sub-problemas.

- Leer el texto cifrado y el abecedario correspondiente a este, desde un archivo de entrada.
- Ordenar alfabéticamente el abecedario y almacenarlo en las listas.
- Recorrer el texto original, buscarlo en la lista *codificacion*, buscar en esa misma posición la letra almacenada en la lista *letras* y almacenar este dato en el arreglo que será escrito al finalizar el procedimiento en el archivo de salida.

2.2 DESCRIPCIÓNDE LA SOLUCIÓN

En base a los sub-problemas obtenidos en la sección anterior, es que se planteó una solución para cada uno de ellos.

2.2.1 Cifrado

A continuación se explica cómo se solucionaron los sub-problemas obtenidos.

2.2.1.1 *Almacenamiento de datos*

Para esto, se consideró crear una cadena de caracteres, el cual almacenará el texto encontrado en el archivo de entrada. Una vez obtenido el texto, se procede a ordenar alfabéticamente, para esto se utilizó el método de ordenamiento *Quicksort*, el cual es caracterizado por ser uno de los más eficientes métodos de ordenamiento interno.

Una vez ordenado el texto, se procede a agregar cada letra en un nodo de las listas enlazadas, donde cada nodo tenía como característica poseer una variable que almacenara la letra, la posición del nodo dentro de la lista y el largo de esta, además de tener los punteros hacia el nodo siguiente y el anterior.

2.2.1.2 *Procedimiento de cifrado*

Para realizar el procedimiento de cifrado, se procede a leer el texto original letra por letra, y buscar cada una de estas en la lista *letras* para luego devolver la posición del nodo donde se encontraba esta letra y buscar en dicha posición en la lista *codificacion.* Luego de almacenar la letra encontrada en el arreglo de caracteres, se realiza el desplazamiento de la lista *codificacion* según los criterios que fueron mencionados anteriormente. Finalmente se procede a escribir en el archivo de salida los resultados. A continuación se muestra el pseudocódigo de este procedimiento.

```
Función cifrar(letras, codificacion, palabra, largo):

        letrasAux = letras
        codAux = codificacion

        Para i = 0 hasta largo – 1, hacer:

                Pos = buscarPorLetra(letrasAux, palabra[i])
                Letra = buscarPorPosicion(codAux, pos)
                PalabraCifrada[i] = Letra

                Si verificarLetra(Letra) == 0, hacer:

                        codAux = desplazarIzquierdaTresEspacios(codAux, &desfase)

                Si verificarLetra(Letra) == 1, hacer:

                        codAux = desplazarDerechaTresEspacios(codAux, &desfase)

        Escribir(Salida, PalabraCifrada)
        Escribir(Salida, desfase)
```

Imagen 2-2: Pseudocódigo función cifrar

2.2.2 Descifrado

Para realizar el proceso de descifrado, se procede a leer el texto cifrado letra por letra, realizar los respectivos desplazamientos mencionados anteriormente y buscar cada una de estas letras en la lista *codificacion,* devolver la posición alojada en el nodo correspondiente y buscar en esa misma posición en la lista *letras*, para así recoger la letra almacenada en ese nodo y agregarla en el arreglo que contendrá la palabra descifrada. Finalmente se escriben en el archivo de salida los resultados obtenidos.

A continuación se muestra el pseudocódigo que representa lo mencionado anteriormente.

```
Función descifrar(letras, codificacion, palabra, largo, desfase):

        letrasAux = letras
        codAux = codificacion

        Para i = 0 hasta largo – 1, hacer:

                Letra = palabra[largo - i - 1]

                Si verificarLetra(Letra) == 0, hacer:

                        codAux = desplazarIzquierdaTresEspacios(codAux, &desfase)
                        Pos = buscarPorLetra(letrasAux, palabra[largo - i - 1])
                        Letra = buscarPorPosicion(codAux, pos)
                        PalabraCifrada[largo - i - 1] = Letra

                Si verificarLetra(Letra) == 1, hacer:

                        codAux = desplazarDerechaTresEspacios(codAux, &desfase)
                        Pos = buscarPorLetra(letrasAux, palabra[largo - i - 1])
                        Letra = buscarPorPosicion(codAux, pos)
                        PalabraCifrada[largo - i - 1] = Letra

        Escribir(Salida, PalabraCifrada)
        Escribir(Salida, desfase)
```

Imagen 2-3: Pseudocódigo de la función descifrar

CAPÍTULO 3. ANÁLISIS DE RESULTADOS

Luego de realizar completamente el algoritmo, se procede a medir la eficiencia de este, mediante los conceptos adquiridos en clases. A continuación se muestra el cálculo de la función que representa el tiempo de ejecución del algoritmo.

- *Agregar( ):*

$$T_{agregar}(n) = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + n(1) + 1 + 1$$

$$T_{agregar}(n) = n + O(1)$$

- *Borrar( ):*

$$T_{borrar}(n) = O(1) + n(1) + O(1) + n(1 + 1 + 1 + 1 + 1) + O(1)$$

$$T_{borrar}(n) = 6n + O(1)$$

- *BuscarPorLetra( ):*

$$T_{buscarPorLetra}(n) = O(1) + n(1 + 1 + 1) + 1$$

$$T_{buscarPorLetra}(n) = 3n + O(1)$$

- *BuscarPorPosicion( ):* $T_{buscarPorPosicion}(n) = O(1) + n(1) + 1 + 1$

$$T_{buscarPorPosicion}(n) = n + O(1)$$

- *Funciones de desplazamiento:*

$$T(n) = O(1) + n\big(1 + 1 + 1 + 1 + T_{agregar}(n) + 1 + 1\big) + n(1 + 1)$$

$$T_{desplazamientos}(n) = n^2 + O(n) + O(1)$$

- *Cifrar y Descifrar:*

$$T_{cifrarYDescifrar}(n) = O(1) + n\left(O(1) + T_{desplazamientos}(n)\right)$$

$$T_{cifrarYDescifrar}(n) = n^3 + O(n^2) + O(n) + O(1)$$

- *Procedimiento( ):*

$$T_{procedimiento}(n) = O(1) + T_{quicksort}(n) + n\left(2 * T_{agregar}(n)\right) + n\big(O(1)\big) + T_{cifrarYDescifrar}(n)$$

$$T_{procedimiento}(n) = n^3 + O(n^2) + O\big(n * log(n)\big) + O(n) + O(1)$$

Finalmente se obtiene que el *T(n)* del algoritmo corresponde al *T(n)* de la función *procedimiento( )*, por lo tanto:

$$T(n) = n^3 + O(n^2) + O\big(n * log(n)\big) + O(n) + O(1)$$

$$\Rightarrow T(n) \sim O(n^3)$$

CAPÍTULO 4. CONCLUSIONES

Para concluir con el presente informe, se deben tomar en cuenta dos parámetros. El primero de ellos es la eficacia del algoritmo, o dicho de otra forma, que el algoritmo cumpla con su objetivo. En base a este parámetro, se podría considerar medianamente exitoso, ya que si bien logra cifrar y descifrar palabras, no logra hacer estos procedimientos en frases (palabras separadas por espacios), ya que por falta de conocimientos sobre el lenguaje de programación C, no se supo generar la correcta lectura de los archivos de entrada que cumplían con las características mencionadas anteriormente. El otro parámetro a considerar es la eficiencia del algoritmo, la que se puede cuantificar mediante la función que expresa el tiempo de ejecución, la que resultó ser del orden de $O(n^3)$. Esto puede ser considerado eficiente para una pequeña cantidad de entradas al problema, pero del conocimiento matemático se puede apreciar que a medida que se ingresan más datos, el programa funcionará de manera más lenta, lo que en la vida real puede generar muchos problemas, más específicamente relacionado al problema planteado para este laboratorio, si se utiliza este algoritmo con fines, por ejemplo, de espionaje, este algoritmo podría ser fácilmente descartado, debido a que los textos que son enviados son por lo general muy grandes, y el tiempo perdido en cifrar y descifrar puede ser crucial.

Se espera para futuras entregas poder seguir buscando nuevos métodos más eficientes que el actual, y poder utilizar los conocimientos adquiridos en la presente entrega a futuro de manera ágil y eficaz.

CAPÍTULO 5. REEFERENCIAS

- Enunciado Laboratorio 1 Análisis de Algoritmos y Estructuras de Datos (Septiembre de 2016). *UdeSantiagoVirtual.* Obtenido de UdeSantiago Virtual Moodle: *http://www.udesantiagovirtual.cl/*
- StackOverflow (Agosto de 2008), creado por Jeff Attwood. Utilizado para dudas varias. *http://stackoverflow.com/*