Tarea 2 - Análisis de algoritmo y estructura de datos

Isaac Alejandro Espinoza Barría
Universidad de Santiago de Chile
2-2023

I. INTRODUCCIÓN

A la edad de 6 años, correspondiente al curso escolar de primer año básico en Chile, se espera que los niños comiencen el camino hacia aprender a leer, por lo que una empresa de juegos online busca motivar y acercar el desarrollo del aprendizaje lector/escritor en los niños. Para realizar esta ayuda, la empresa decide crear el famoso juego lingüístico "sopa de letras" de formar online, donde los niños además de ejercitar el reconocimiento de palabras puedan familiarizarse con el uso de recursos tecnológicos como lo es un computador o celular móvil. Estos juegos lingüísticos desarrollan la autoestima de los niños al lograr desafíos, el pensamiento lógico y mecánico, la rapidez mental, además asocian las palabras con la diversión.

El juego sería llamado DerechoRevés, consiste en buscar palabras dentro de un tablero cuadrado, tal que estas puedan estar horizontales, verticales, o en ambas diagonales; incluyendo también que las palabras puedan estar en sentido inverso. De modo que el tablero se encuentre lleno de letras.

Asimismo, se propone como tarea la simulación del proceso de solución en el juego DerechoRevés. La tarea debe ser realizada en código para el lenguaje de programación C, donde se lea un archivo de extensión .ini contenedor del tablero y otro archivo de extensión .lst que posea las palabras a buscar dentro del tablero; posterior a ello, se ejecute un algoritmo de búsqueda de palabras dentro del tablero, para que después se generen dos archivos de salida; uno que contenga la cantidad de palabras encontradas, cada una de ellas y la posición (fila/columna) dentro del tablero; y otro archivo que tenga las palabras encontradas resaltándolas con asteriscos en el tablero una vez cada una. Como requisitos, los nombres de los archivos a leer se deben indicar mediante teclado, en la consola se debe indicar el estado de lectura y creación de los archivos usados en el programa e indicar el número de palabras encontradas y de palabras buscadas.

El objetivo de la tarea es diseñar una estrategia de búsqueda de palabras dentro de un tablero lleno de letras en español.

Para realizar la tarea utilizaré un computador con lenguaje C, que pueda leer, compilar e interpretar este lenguaje; además de la ayuda de un editor de código para agilizar la escritura del programa.

II. SOLUCIÓN PROPUESTA

La abstracción general de mi solución propuesta es un código en C que primero pida al usuario los nombres con su extensión de los archivos a utilizar en el programa; para que lea el archivo de las palabras a buscar y cree una lista enlazada de estas (TDA lista enlazada con listas enlazadas de caracteres) llamada listaTablero; luego lea el archivo de texto del tablero y cree una lista enlazada de las filas del tablero, dado que en cada fila se encuentren los caracteres de las columnas de forma ordenada, logrando una representación de una matriz cuadrada NxN llamada tablero. Ya teniendo creadas estas estructuras de datos, ahora se define una tercera lista enlazada que guarde dos datos de posición (fila y columna) de las posibles palabras a encontrar, llamada listaCoordenadas; donde la posición dentro de esta lista se relaciona con la posición de la palabra dentro de listaPalabras. Posterior a esto, mediante una función de búsqueda, se buscan las palabras de listaPalabras dentro de tablero y los datos de las posiciones se guardan dentro de listaCoordenadas; si se encuentra la palabra se guardarán la fila y columna correspondiente, si no se encuentran, las posiciones guardadas serán ambas el entero -1. En este punto sólo queda la creación y escritura de los archivos de salida; todas las palabras que no tenga como posición (-1,-1) se

escriben en el texto de salida lista.out, y se destacan en el texto tablero.out mediante asteriscos. Finalmente se muestra en consola la cantidad de palabras encontradas de las buscadas, terminando el proceso.

Para desarrollar la tarea, definí 5 funciones aparte de la principal main, de las que una es el algoritmo de búsqueda y será el que explicaré en mayor detalle. Estas funciones se mencionarán a continuación:

- LecturaPalabras: Lee el archivo de palabras a buscar y retorna una lista enlazada de listas enlazadas de caracteres, donde cada elemento es una palabra; además mediante paso por referencia indica la cantidad de palabras.

- LecturaTablero: Lee el archivo del tablero y retorna una representación matricial NxN de este mediante lista enlazada de listas enlazadas de caracteres, además mediante paso por referencia indica la cantidad de filas y columnas.

- EscribirTableroLista2D: Escribe el tablero dentro de un archivo de salida, se utiliza dentro de la función BuscarPalabras.

- CambiarExtension3char: Cambia los últimos tres caracteres de una cadena de texto, esto para cambiar la extensión del nombre de los archivos.

- BuscarPalabras: Busca una palabra dentro del tablero, derecha o invertida, y retorna un booleano indicando si se encontró la palabra; además, mediante referencia, agrega un nuevo elemento a listaCoordenadas, donde se guarda la fila y columna de la palabra encontrada; en el caso de no entrar la palabra, la fila y columna son -1. Sus parámetros son la palabra buscada (lista enlazada de caracteres), el tablero (lista enlazada de listas enlazadas de caracteres), la cantidad de filas (número entero), la cantidad de columnas (número entero), un puntero a lista de coordenadas (lista enlazada de dos enteros cada nodo), y el flujo del archivo de salida de tableros.

Esta última función es la más importante del código, ya que de ella depende la finalidad de este. Se basa en recorrer de forma ordenada todas las letras del tablero, analizando desde la primera hasta la última columna de cada fila. Luego, si en el recorrido se cumple que la letra analizada es la misma que la primera letra de la palabra buscada, se ejecutarán la siguiente serie de condiciones para determinar si se trata de esta o no: primero se define un booleano verdadero, posteriormente se comparada las letras siguientes dentro del tablero con las letras siguientes de la palabra; si se cumple que las letras son distintas, el booleno cambia a falso; si las letras son iguales el booleano se mantiene en verdadero retornándolo, determinando que se encontró la palabra y agregando por referencia a listaCoordenada la fila y columna de la palabra. Este proceso se realiza para las cuatro direcciones posibles (horizontal, vertical y las dos diagonales). Luego, si no se encontraron resultados, se vuelve a realizar la misma búsqueda, pero con la palabra invertida. En este punto si todavía no hay resultado significa que la palabra no está en el tablero, por lo que se agrega por referencia a la listaCoordenadas los valores -1 para la fila y columna, y termina el algoritmo retornando falso.

Dentro de esta función, existe el proceso de escribir el tablero en el archivo de salido, el que se ejecuta cuando se encuentra una palabra. Consiste en cambiar la palabra por asteriscos en el tablero, escribirlo en el archivo tablero.out con la función escribirTableroLista2D, y volver a cambiar los asteriscos por la palabra original en el tablero.

Se destaca que en el código es indispensable el uso de dos archivos externos para la ejecución, los que son definiciones de dos TDAs: TDA lista Char, y TDA lista Coordenadas. Ambos, se sustentan en listas enlazadas que utilizan nodos donde se almacena datos y un puntero (dirección de memoria) de otro nodo/elemento, esto hasta que un nodo apunte a NULO y se termine la lista. Como se representa en la figura 1.

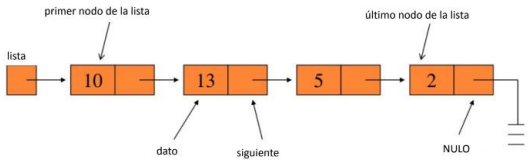

Figura 1: Representación lista enlazada simple

El beneficio de usar estos TDAs es que poseen funciones para trabajarlos, las que facilitan la ejecución del código.

Explicando los TDAs, el TDA lista Char no sólo es una lista enlazada de caracteres, también posee la definición de una lista enlazada de TDAs lista Char, lo que es práctico para representar matricialmente el tablero y almacenar las palabras a buscar; el TDA lista Coordenadas simplemente es una lista enlazada donde en cada nodo posee dos datos enteros, donde se busca almacenar la fila y columna de las palabras.

A continuación, se presenta el seudocódigo de la función BuscarPalabras (ver Figura 2).

```
BuscarPalabras(tablero, palabra, num Mfila, num Ncolumnas,
lista Coordenadas, archivo archivoSalida): booleano

    largoPalabra = obtenerLargo(palabra)
    auxiliar ← tablero                                        o(1)
    fila ← 0
    auxPalabra ← palabra
    MIENTRAS auxiliar <> NULO HACER
        aux2 ← auxiliar->dato
        columna ← 0
        MIENTRAS aux2 <> NULO HACER
            … condición letra 1 palabra es correcta
            SI aux2->dato = palabra->dato ENTONCES
                * PROCESO BUSQUEDA HORIZONTAL *               o(n^2)
                * PROCESO BUSQUEDA VERTICAL *
                * PROCESO BUSQUEDA DIAGONAL IZQUIERDA *
                * PROCESO BUSQUEDA DIAGONAL DERECHA *
            columna ← columna + 1
            aux2 ← aux2->puntero
        fila ← fila + 1
        auxiliar ← auxiliar + 1

    palabraReves ← invertirPalabra(palabra)                   o(1)
    * SE REALIZA MISMA BÚSQUEDA ANTERIOR *                    o(n^2)

    insertar (fila -1 y columna -1) a lista Coordenadas       o(1)
    DEVOLVER(falso)
```

Figura 2: Seudocódigo función BuscarPalabras

El seudocódigo tiene cuatro procesos de búsqueda de palabras en las cuatro direcciones, los que comparan las letras de la palabra con las del tablero y determinan si se encuentra en el tablero; si es así, se agregan por referencia la posición (fila, columna) a lista Coordenadas, y se retorna un booleano verdadero. También, en cada proceso de búsqueda se destaca dentro del seudocódigo el proceso "palabra encontrada", el que tiene pequeñas variaciones en los cuatro procesos de búsqueda dependiendo de la dirección de la palabra encontrada, en el cómo se guardan la fila y columna en listaCoordenadas. A continuación, el seudocódigo del proceso de búsqueda horizontal:

```
* PROCESO BUSQUEDA HORIZONTAL *
auxPalabra ← palabra
SI columna + largoPalabra - 1 < Ncolumna ENTONCES
    auxTablero1D ← aux2
    palabraEncontrada ← verdadero
    MIENTRAS auxPalabra <> NULO HACER
        SI auxTablero1D->dato <> auxPalabra->dato ENTONCES
            palabraEncontrada ← falso
        auxTablero1D ← auxTablero1D->puntero
        auxPalabra ←auxPalabra->puntero

    … proceso palabra encontrada
    SI palabraEncontrada = verdadero
        insetarFilaColumnaPalabra(lista Coordenadas)

        * cambiar palabra encontrada por asteriscos en tablero
        * escribir tablero en archivoSalida
        * cambiar asteriscos por palabra encontrada en tablero

        DEVOLVER(palabraEncontrada)
```

Figura 3: Seudocódigo búsqueda horizontal

III. RESULTADOS Y ANÁLISIS

Como resultado, tenemos un programa que cumple con lo solicitado. De acuerdo con los archivos ingresados, busca palabras en ambos sentidos de lectura dentro del tablero en las cuatro direcciones. Indicando en el archivo de salida las posiciones de las palabras encontradas, y resaltando con asteriscos las letras de la palabra encontrada dentro del tablero en otro archivo. El programa posee varias funciones para lograr su cometido, junto con algoritmo iterativo de búsqueda de palabras.

Al estudiar la complejidad del algoritmo principal, obtenemos que es $O(n^2)$; ahora, considerando a N cantidad de letras del tablero $(fila \cdot columna)$, obtenemos la siguiente de toma de datos del tiempo de ejecución (ver Tabla 1).

N Cantidad elementos	Tiempo (segundos)
400 (20x20)	0,001
2500 (50x50)	0,003
10000 (100x100)	0,016
40000 (200x200)	0,075

Tabla 1: Datos experimentales tiempo.

Destaco que en la ejecución de la función main con la búsqueda, el tiempo aumenta si se buscan palabras que no estén en el tablero. Por este motivo, los tiempos consideran la búsqueda de 100 palabras inexistentes en el tablero, para notar una verdadera diferencia de tiempos.

Al graficar tenemos lo siguiente (ver Figura 4).

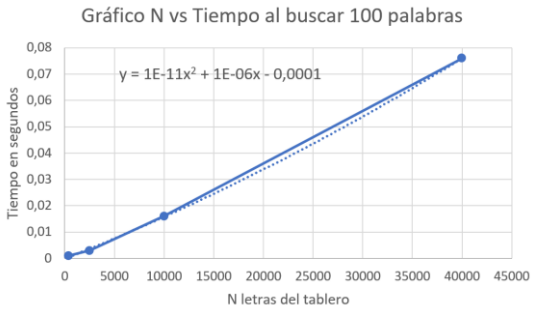

Figura 4: Gráfico N vs Tiempo.

Al realizar el ajuste lineal, podemos representar su línea de tendencia de forma polinómica de grado 2. Lo que comprueba experimentalmente que la complejidad del algoritmo corresponde a $O(n^2)$.

IV. CONCLUSIONES

Se logra un programa que cumple con los requisitos estipulados, diseñando una estrategia de búsqueda de palabras dentro de un tablero compuesto por letras en español. Por lo que se cumple el objetivo de la tarea.

La ventaja de este algoritmo de búsqueda es la capacidad de buscar una palabra en ambos sentidos para las cuatro direcciones posibles. La desventaja es que utiliza muchas líneas de código, lo que dificulta el entendimiento de este; además, si se busca una palabra que no está en el código, se recorre dos veces el tablero, lo que aumenta el tiempo de ejecución.

Considero que las estructuras de datos utilizados son óptimas para el código, el uso de TDAs de listas enlazadas para guardar el tablero, las palabras y las posiciones de las palabras; cumplen su cometido al ser estructuras ordenadas y sin límite, a menos que se llene la memoria, como lo son los arrays. Este orden de estructura nos beneficia, porque se puede asociar las listas por posición, lo que es práctico para vincular listaPalabras con listaCoordenadas.

ANEXO MANUAL DE USUARIO

Para seguir este manual, se requiere que el usuario tenga conocimientos previos de cómo compilar un archivo C y cómo ejecutarlo desde la consola de comandos. A continuación, se enumerarán una serie de pasos para ejecutar el programa:

0.- Como paso previo, asegurarse de que los dos archivos de texto utilizados se encuentren en la misma carpeta del código fuente.

1.- Compilar y ejecutar el archivo C en la consola de comandos.

2.- Introducir por teclado el nombre con extensión del archivo que contiene la lista de palabras, y luego el nombre con extensión del archivo que contiene el tablero. Los que son solicitados al usuario mediante la consola.

3.- Observar en la consola el resultado de la lectura y escritura de los archivos, así como la cantidad de palabras encontradas y buscadas.

4.- Abrir los archivos de salida que se han generado en la misma carpeta contenedora del código, y analizar los resultados obtenidos.

En este punto, el programa se ejecutó exitosamente.

En el transcurso del proceso, pueden existir errores tales como: escribir mal los nombres de los archivos a utilizar, que los archivos de entradas no estén en la carpeta contenedora, o que el computador no posea memoria disponible para ejecutar el código. Todas estas situaciones provocarán que el programa no funcione. Para solucionarlo, se debe volver a ejecutar el código procurando no cometer los errores antes mencionados.