Tarea 1 - 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 será llamado DerechoRevés, consiste en buscar dentro de un tablero cuadrado palabras 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 ella 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 de las 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 resolver 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 hacer más amena la escritura de este.

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 un array de estas (array de cadenas de caracteres) llamado palabrasTablero; luego lea el archivo de texto del tablero y cree un array de las filas del tablero, dado que en cada fila se encuentren los caracteres de las columnas de forma ordenada, para lograr una representación de una matriz cuadrada NxN llamada tablero. Ya teniendo creadas estas estructuras de datos, ahora se define un struct llamados datosPalab donde se definen las coordenadas de las posibles palabras a encontrar (fila y columna); y a partir de esto, se crea un array de elementos datosPalab llamado datosPalabras, donde la posición dentro de este array se relaciona con la posición de la palabra dentro de palabrasTablero. Posterior a esto, se buscan las palabras del array palabrasTablero dentro de tablero y los datos se guardan dentro del array datosPalabras; si se encuentra la palabra se guardarán sus coordenadas, si no se encuentran, las coordenadas 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 coordenadas (-1,-1) se escriben en el texto de salida lista, y se destacan en el texto del

tablero mediante asteriscos. Finalmente se muestra mediante consola la cantidad de palabras encontradas del total de buscadas, para terminar el proceso.

Para desarrollar la tarea, definí 5 funciones externas aparte de la función principal main, de las que una tiene mayor relevancia que las otras y será la que explicaré en mayor detalle. Estas funciones se mencionarán a continuación:

- LecturaPalabras: Lee el archivo de palabras a buscar y retorna un array de cadenas de caracteres donde cada elemento es una palabra, además mediante paso por referencia indica la cantidad de estas. Sus parámetros son el nombre del archivo y un puntero que apunta a la variable donde se almacena la cantidad de palabras.

- LecturaTablero: Lee el archivo del tablero y retorna una representación matricial NxN de este, además mediante paso por referencia indica la cantidad de filas y columnas. Sus parámetros son el nombre del archivo, un puntero a una variable para almacenar las filas y otro para almacenar las columnas del tablero.

- EscribirTablero: Escribe el tablero dentro de un archivo de salida, se utiliza dentro de la función BuscarPalabras. Sus parámetros son el array de cadenas de caracteres del tablero, la cantidad de filas, la cantidad de columnas, y el flujo de texto del archivo.

- CambiarExtension3char: Cambia los últimos tres caracteres de una cadena de texto, esto para cambiar la extensión del nombre de los archivos a leer y crear. Sus parámetros son el nombre del archivo, y la extensión a cambiar de 3 caracteres.

- BuscarPalabra: Busca una palabra dentro del tablero, y dependiendo está invertida o no, retorna un booleano indicando si se encontró la palabra; además modificar mediante referencia las coordenadas de la primera letra de la palabra encontrada al leerse correctamente. Sus parámetros son la palabra buscada, el tablero, la cantidad de filas, la cantidad de columnas, un puntero al array de los datos de las palabras, un booleano que indica si la palabra está invertida, y el flujo del archivo de salida de tableros.

Esta última función es la más importante dentro del código, ya que de ella depende la finalidad de la tarea. Su fundamento se basa en recorrer de forma ordenada todas las letras del tablero, tal que se analice desde la primera hasta la última columna de la primera fila y así sucesivamente hasta completar todas las filas. Luego, si en medio del recorrido se cumple que la letra analizada es la misma que la primera letra de la palabra buscada, se ejecutarán una serie de condicionales 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 determinando que se encontró la palabra.

Antes de terminar la ejecución de la función, se cambia por asteriscos la palabra dentro del tablero, esto para ejecutar la función EscribirTablero que escribe el archivo de salida, y luego se vuelve a cambiar los asteriscos por la palabra original en el tablero. Ahora dependiendo si la palabra buscada estaba invertida o no, se modificarán mediante referencia las coordenadas de fila y columna de la primera letra de la palabra buscada al leerse de forma correcta. Este proceso se hace para los cuatro sentidos, horizontal, vertical y las dos diagonales. En este momento se retorna un verdadero si la palabra fue encontrada; si se recorrió todo el tablero sin éxito de búsqueda se retorna un falso y las coordenadas tanto fila como columnas se cambian a -1.

Se destaca que dentro de la función main al buscar la palabra en sentido correcto y no encontrarla, se realiza el mismo proceso, pero con la palabra invertida. Si no se encuentra la palabra en ambos sentidos, se procede a buscar la palabra siguiente.

A continuación, se presenta el seudocódigo de la función BuscarPalabra (ver Figura 1).

```
BuscarPalabra(array de caracteres palabra, matrizNxN tablero, num
fila, num columnas, coordenadas palabAct, booleano palabraInversa,
archivo archivoSalida): booleano


   largoPalabra = obtener el largo de palabra

   PARA i ← 1 HASTA filas
      PARA j ← 1 HASTA columnas
         … condición letra 1 de la palabra es correcta
         SI tablero[i][j] = palabra[0] ENTONCES

            … búsqueda palabra horizontal
            SI j-1+largoPalabra < columnas ENTONCES
               palabraEncontrada ← verdadero
               PARA u ←2 HASTA largoPalabra
                  … iteramos desde letra 2 de la palabra
                  SI tablero[i][j+u] <> palabra[u]
                     palabraEncontrada ← falso
               Proceso palabra encontrada

            … búsqueda palabra vertical
            SI i-1+largoPalabra < filas ENTONCES
               palabraEncontrada ← verdadero
               PARA u ←2 HASTA largoPalabra
                  SI tablero[i+u][j] <> palabra[u]
                     palabraEncontrada ← falso
               Proceso palabra encontrada

            … búsqueda palabra diagonal ↙
            SI i-1+largoPalabra<filas Y j+1-largoPalabra>=0 ENTONCES
               palabraEncontrada ← verdadero
               PARA u ←2 HASTA largoPalabra
                  SI tablero[i+u][j-u] <> palabra[u]
                     palabraEncontrada ← falso
               Proceso palabra encontrada

            … búsqueda palabra diagonal ↘
            SI i-1+largoPalabra<filas Y j-1+largoPalabra ENTONCES
               palabraEncontrada ← verdadero
               PARA u ←2 HASTA largoPalabra
                  SI tablero[i+u][j+u] <> palabra[u]
                     palabraEncontrada ← falso
               Proceso palabra encontrada

   Guardar por referencia las coordenadas (-1,-1) en palaAct
   RETORNAR falso
```

Figura 1: Seudocódigo función BuscarPalabra

El proceso palabra encontrada, está presente cuatro veces en el algoritmo BuscarPalabra con pequeñas variaciones cada uno. La única diferencia entre ellos es cómo guarda las coordenadas en palabAct, la que depende de la dirección de la palabra encontrada en el tablero (ver figura 2).

```
SI palabraEncontrada = verdadero
   Proceso escribir texto de salida
   SI palabraInversa = falso ENTONCES
      Guardar por referencia las coordenadas de la letra
inicial en palabAct
      RETORNAR verdadero
   SINO  … ultima letra porque la palabra esta invertida
      Guardar por referencia las coordenadas de la letra
final en palabAct
      RETORNAR falso
```

Figura 2: Seudocódigo palabra encontrada

El proceso escribir texto de salida, cambia los elementos de la palabra encontrada con asteriscos en el tablero, llama a una función externa para escribir el tablero en el archivo de salida, y vuelve a cambiar los asteriscos por la palabra original dentro del tablero.

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 posibles. 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. El programa posee varias funciones para lograr su cometido, junto con una función con un algoritmo iterativo de búsqueda de palabras.

Al estudiar la complejidad del algoritmo principal, obtenemos que esta es $O(n^2)$; ahora, considerando a N como el producto entre la cantidad de filas y las columnas (cantidad de letras del tablero), obtenemos la siguiente información experimental de toma de datos (ver Tabla 1).

N Cantidad elementos	Tiempo (segundos)
400 (20x20)	0,002
2500 (50x50)	0,003
10000 (100x100)	0,011
40000 (200x200)	0,061

Tabla 1: Datos experimentales tiempo.

Destaco que en la ejecución de la función main, la complejidad del algoritmo de búsqueda puede aumentar si se consideran un mayor número de palabras buscadas comparado con N. Por este motivo todos los tiempos fueron tomados considerando una búsqueda de 100 palabras, para notar una verdadera diferencia de tiempos.

Al graficar tenemos lo siguiente (ver Figura 3).

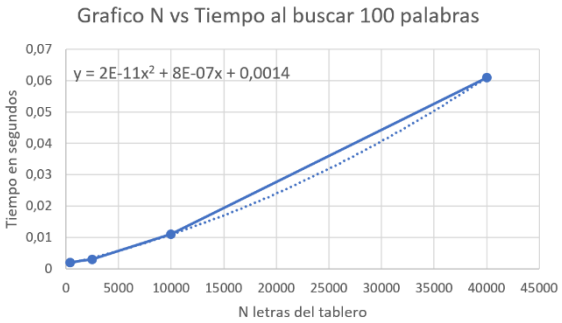

Figura 3: Grafico N vs Tiempo.

Notamos que, al realizar el ajuste lineal en los datos podemos representar su línea de tendencia de forma polinómica de grado 2. Por lo que se comprueba experimentalmente que la complejidad del algoritmo corresponde a $O(n^2)$.

IV. CONCLUSIONES

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

La ventaja del algoritmo de búsqueda creado es que se utiliza el mismo para buscar las palabras en ambos sentidos, lo que reduce las líneas de código y es más fácil de entender. Pero esto también se puede ver como una desventaja, ya que, si buscamos una palabra que no está en el tablero, el programa lo recorrerá dos veces. Esto debido a que primero lo hace con la palabra normal y luego con la palabra invertida. Lo que no pasaría si al recorrer una sola vez el tablero, consideremos todas las posibles dirección y sentidos de la palabra. Esta solución hubiera evitado recorrer el tablero de forma innecesaria pero el algoritmo sería más extenso.

Considero que las estructuras de datos utilizados son óptimas para el desarrollo del código, el uso de un struct para las coordenadas de cada palabra, el array de arrays de caracteres para el tablero, y otro para las palabras; cumplen su cometido al ser estructuras ordenadas. Esto nos beneficia, porque se puede asociar los arrays por posición, lo que es necesario para vincular el array de palabras con el array de datos de palabras (coordenadas) por ejemplo.

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 y la extensión del archivo que contiene la lista de palabras, y luego el nombre y la 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.