Tarea 3 - 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

Para la solución se utilizan las siguientes estructuras de datos: TDA grafos para representar el tablero, y un arreglo 2D de caracteres para almacenar las palabras a buscar.

La abstracción general de mi solución propuesta es un código en C que primero pida al usuario los nombres con extensión de los archivos a utilizar en el programa; para que lea el archivo de las palabras a buscar y cree un arreglo de que en cada elemento contenga una palabra (arreglo de arreglos de caracteres) llamado arrayPalabras; luego lea el archivo de texto del tablero y cree un grafo no dirigido no ponderado para su representación, de modo que cada letra/elemento del tablero sea un vértice del grafo, los que además están conectados mediante aristas con todas las demás letras/vértices adyacentes a este, es decir, cada letra del tablero (vértice) está unida con todas las letras alrededor de ella dentro del tablero; esta representación utiliza una matriz de adyacencia para modelizar el grafo, por lo que, si n es la cantidad de filas del tablero, la matriz de adyacencia es de tamaño $n^2 \cdot n^2$; este grafo es llamado g.

Ya creadas estas estructuras de datos, se procede a la búsqueda. Mediante una función llamada

buscarPalabras, se buscan las palabras existentes en arrayPalabras mediante un ciclo for, ya que esta función solo busca una única palabra dentro del tablero (grafo). La misma función de búsqueda es la encargada de escribir los archivos de salida de las palabras con sus posiciones y los tableros con las palabras destacadas; por lo que, si es que encuentra una palabra dentro del tablero, automáticamente se escribe la posición de esta palabra en el archivo de salida lista.out, y se escribe el tablero destacado con asteriscos en el archivo de salida tablero.out.

Posterior, se cuenta la cantidad de palabras encontradas y este dato se escribe en la primera línea del archivo de salida de lista.out. Finalmente se muestra en consola la cantidad de palabras encontradas de las buscadas, y se libera la memoria utilizada por el programa para terminar el proceso.

Para desarrollar la tarea, he definido 5 funciones auxiliares aparte de la principal; donde la que posee el algoritmo de búsqueda y será explicada en mayor detalle. Estas funciones se mencionarán a continuación:

- LeerPalabras: Lee el archivo de palabras a buscar y retorna un arreglo de cadenas de caracteres donde cada elemento es una palabra; además mediante paso por referencia indica la cantidad de palabras.

- LeerTablero: Lee el archivo del tablero y retorna una representación de este mediante un grafo no dirigido no ponderado, además mediante paso por referencia indica la cantidad de filas del tablero.

- EscribirTablero: 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 entero indicando si se encontró la palabra (1) o no (0). Además, modifica los textos de salida; escribe la posición de la palabra encontrada si es que la hay en lista.out, y escribe el tablero con la palabra destacada con asteriscos en tablero.out. Sus parámetros son la palabra buscada (arreglo de caracteres), el tablero (grafo no dirigido no ponderado con matriz de adyacencia), la cantidad de filas del tablero (número entero), el flujo del archivo de salida de listas, 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 el primer vértice del grafo (letra) hasta el último. 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 una variable llamada palabraEncontrada como 1, posteriormente se comparada las letras siguientes dentro del grafo (tablero) en una dirección determinada con las letras siguientes de la palabra; si se cumple que las letras son distintas, la variable cambia a 0; si las letras son iguales la variable se mantiene como 1 retornándola, determinando que se encontró 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, retornando un 0.

Dentro de esta función, existe el proceso de escribir el tablero en el archivo de salida, el que se ejecuta cuando se encuentra una palabra. Consiste en cambiar cada letra de la palabra por asteriscos en el grafo, escribirlo en el archivo tablero.out con la función escribirTablero, y volver a cambiar los asteriscos por la palabra original en el tablero. A continuación de este, se ejecuta otro proceso llamado imprimir posición, el que escribir la posición de la palabra encontrada en el archivo lista.out. Una vez realizados ambos procesos, se retorna un 1 y termina la ejecución de la función.

Se destaca que en el código es indispensable el uso de un archivo externo para la ejecución, el que es la definición del TDA grafo no dirigido no ponderado utilizando matriz de adyacencia. El beneficio de usar este TDA es que pose funciones de uso que facilitan la ejecución del código.

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

```
BuscarPalabras(grafo, palabra, num nFilas, archivo salidaLista,
archivo salidaTablero): num

largoPalabra = obtenerLargo(palabra)                          o(1)
PARA i←0 HASTA grafo->n
   SI palabra[0] = grafo->V[i] ENTONCES
      … búsqueda horizontal izq der
      palabraEncontrada ←1
      SI i+largoPalabra-1 < grafo->n ENTONCES
         … se cumple condición palabra horizon

         PARA j←1 HASTA largoPalabra
            … analizar adyacencia de vertices horizon          o(n²)
            auxBus ←sonVerticesAdya(grafo,i+j-1,i+j)
                                                               Se recorren
                                                               los n2 vértices
            SI auxBus==1 ENTONCES
               …condicion letra palabra distinta grafo
               SI palabra[j] <> grafo->V[i+j] ENTONCES
                  palabraEncontrada←0
            SINO
               palabraEncontrada←0

         SI palabraEncontrada==1 ENTONCES
            … proceso escribir tablero
            PARA c←0 HASTA largoPalabra
               grafo->V[i+c] ← '*'
            escribirTablero(grafo, salidaTablero, nFila)
            PARA c←0 HASTA largoPalabra
               grafo->V[i+c] ← palabra[c]

            … proceso imprimir posición
            ESCRIBIR(salidaLista)
            DEVOLVER (palabraEncontrada)

      * búsqueda vertical
      * búsqueda diagonal \
      * búsqueda diagonal /

palabra ← invertirPalabra(palabra)                             o(1)
PARA i←0 HASTA grafo->n
   SI palabra[0] = grafo->V[i] ENTONCES
      * búsqueda horizontal
      *  búsqueda vertical                                     o(n²)
      *  búsqueda diagonal \
      *  búsqueda diagonal /

palabra ← invertirPalabra(palabra)                             o(1)
DEVOLVER(0)
```

Figura 1: 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, retornando un 1. También, en cada búsqueda de acuerdo a la dirección, se destaca dentro del seudocódigo el proceso escribir tablero e imprimir posición; los que tienen pequeñas variaciones en los cuatro procesos de búsqueda dependiendo de la dirección de la palabra encontrada, en el cómo se escribe posición de la palabra y cómo se reemplazan las letras por asteriscos para posteriormente retornar de los asteriscos a las letras originales.

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 $O(n^2)$ si consideramos a n como la cantidad de letras en cada fila del tablero. Esto se comprueba también al destacar que existen n^2 vértices en el grafo, y se recorren todos los vértices comparándolos con la primera letra de palabra. Luego, tenemos la siguiente de toma de datos del tiempo de ejecución (ver Tabla 1).

N Cantidad letras fila	Tiempo (segundos)
20	0,000
50	0,002
100	0,051
200	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 recorrer todos los vértices del grafo.

Al graficar tenemos lo siguiente (ver Figura 2).

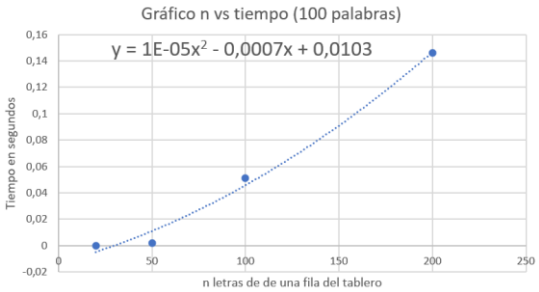

Figura 2: Gráfico N vs Tiempo.

Al realizar el ajuste lineal, podemos representar su línea de tendencia de forma polinómica de grado 2. Lo que comprueba de forma experimental pero reduccionista 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 adecuadas para el código, el uso de TDA grafo para guardar el tablero, y las palabras en arreglos; cumplen su cometido al ser estructuras ordenadas. Sin embargo, creo que utilizar grafos para esta representación del tablero no es óptima, ya que es más difícil de implementar a la hora del código. Lo que no ocurre al utilizar listas enlazadas o arreglos matriciales, como lo realizado en las tareas 1 y 2.

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.