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

Estructura de Datos y Análisis de Algoritmos

Laboratorio 1 : Buscaminas

Bastián Gonzalo Vera Palacios

Profesor: Jacqueline Köhler

Ayudantes: Nicole Henríquez

Sebastián Vallejos

Javiera Torres

Santiago - Chile
2-2017

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

A través de los años, la industria de los videojuegos ha ido en crecimiento de tal manera que hay varios de estos que nosotros y todas las generaciones ya los concideramos como grandes clasicos de la historia de los juegos. En esto nos encontramos con dos tipos de juegos, los de consola, tales como la playstation o nintendo, y los juegos de computadora, los cuales tienen grandes exponentes como uno de los que todos han jugado alguna vez, el Buscaminas.

El famoso juego inventado en 1989, el cual venia preinstalado en el sistema operativo Windows y gracias a este, logro gran popularidad entre adultos y jovenes que buscaban entretenerse o hacer algo distinto en su computador. El juego consistia en poder despejar un campo de minas sin detonar ninguna.

Durante este laboratorio se solicita que implemente el juego del buscaminas. El objetivo en el laboratorio por desarrollar es poder aprender a utilizar e internalizarnos un poco mas con la programación en el lenguaje C y a la vez poder utilizar todas las habilidades adquiridas, tanto en años anteriores, como en este curso, las cuales presentará nuevas herramientas para poder desarrollar las distintas problemáticas que surgen durante el desarrollo de este laboratorio.

En este informe desarrollaremos de maneras mas amplia como se puede abarcar el problema, las herramientas que se usaran para esto, los algoritmos y estructuras utilizados para el desarrollo, como poder plantear una solución a este para lograr un desarrollo completo de el juego y las conclusiones respectivas sobre los resultados obtenidos.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN

2.1 MARCO TEÓRICO

Para poder desarrollar de manera mas facil y ordenada nuestro codigo, se crean funciones para cada tipo de algoritmo generado en el codigo, las cuales seran llamadas posteriormente en un metodo encargado de la iniciación de nuestro juego.

La primera deducción que se puede hacer respecto al juego, es que al tratarse de un tablero de N x M dimensiones, tenemos que trabajar con arreglos bidimensionales, con el fin de poder acceder y trabajar cada ubicación como una posición, con un valor en el eje X y otro en el eje Y. A la vez, ya que no se tiene las dimensiones de el tablero de manera estatica, se debe asignar la memoria para cada arreglo de manera dinamica, a través de un grupo de funciones en la biblioteca estandar de C.
Una vez logrado esto, se generaran dos matrices, en las cuales una tendrá las jugadas ingresadas por el usuario y otra tendra el tablero con la respectiva solucion de el juego.
Ambas matrices se van comparando entre si para poder obtener resultados como la posición donde el jugador realizo la jugada, hacer la verificación con respecto a las bombas, verificar la victoria, entre otros.
Una vez que ya tenemos claro en como manejar la información, se le solicita al usuario que ingrese la cantidad de filas y columnas de el tablero, junto con la cantidad de bombas que desea que tenga el juego. Posterior a estas primeras opciones se inicializan todas las variables, tales como las filas y columnas ingresadas, la jugada ingresada, etc.
Despues de realizado lo anterior, se le muestra al usuario el tablero inicializado sin ningun valor con una función encargada de mostrar por pantalla el tablero y se le solicita que ingrese las coordenadas de su primera jugada junto con un valor que indique la jugada que desea realizar.
Luego de que esta primera jugada se haya realizado, se procede a insertar las bombas de tal manera que el jugador no pierda en la primera jugada, para la cual solo utilizamos la posición actual y así solo verificamos que no se coloque una bomba sobre esta. Para poder colocar las bombas, lo optimo a utilizar fue una función que permite manejar valores aleatorios usando la hora como referencia. Junto con iniciar las bombas también se generan los numeros alrededor de las bombas, los cuales quedan almacenados en una matriz solución.

A medida que continua el juego,el usuario va insertando jugadas las cuales se pueden considerar como jugadas marcadas o jugadas seleccionadas. A partir de cada una de las opciones, se hacen las respectivas verificaciones de que se cumpla que la casilla selecionada no sea bomba, lo que significaria que el juego continua, o si es una bomba que el juego marque la derrota respectiva, entre otras posibles opciones. En caso que no sea una jugada valida, el juego va a repetirle la opcion a ingresar la jugada por una valida.

En caso de que la jugada sea en un espacio sin valor alguno, lo optimo es usar un algoritmo recursivo que logre avanzar y ampliar el tablero hasta que llegue a alguna casilla con numero. Cuando llegamos al final de las verificaciones solo faltaria entregar el resultado de el codigo y a la vez liberar la memoria solicitada para los arreglos bidimensionales.

2.2 HERRAMIENTAS Y TÉCNICAS

Dentro de las herramientas utilizadas para el desarrollo de este laboratorio, se encuentran las librerías de :

- stdio.h: Para la entrada, salida de datos y manejo de archivos.
- stdlib.h: Utilizada para el manejo de la memoria dinámica.
- time.h: Utilizada para generar un número aleatorio con la función srand( ).
- ctype.h: Usada en la función tolower( ) para ajustar el carácter recibido por el usuario.

Ya que para desarrollar este juego se necesitaba emplear matrices sobre las cuales las dimensiones dependen de la entrada del usuario, se utilizó para esto arreglos dinámicos para poder trabajarlos óptimamente. Para esto usamos la función malloc( ) la cual es la encargada de solicitar la memoria según las dimensiones solicitadas, y la función free( ), la cual es la que se utiliza al final del desarrollo del programa para poder liberar la memoria utilizada anteriormente.

Para el desarrollo de la función expandirMatriz( ) se utilizó la técnica de la recursión ya que fue la manera efectiva de realizar la expansión del tablero, ya que cada vez que se revisa una posición, se llama a la misma función logrando así volver a realizar la revisión de todas las casillas pendientes. Para esto también se utilizo la función verificarMargen( ) con el fin de poder restringir los limites de la revisión al expandir la matriz.

Para poder insertar las bombas de tal manera que la posición de estas sea aleatoria, se usan las funciones srand( ) y rand( ), las cuales se encargan de generar un número aleatorio, el cual es tomado como valor de coordenadas para posicionar una nueva bomba.

Para poder retornar al usuario la solución de la partida, se utiliza el manejo de archivos para crear un archivo de salida el cual contiene la matriz con la solución.

2.3 ALGORITMOS Y ESTRUCTURAS DE DATOS

verificarVictoria(jugadas, filas, columnas,cantMinas): Función encargada de hacer la verificación si se ha llegado al final de el juego con una victoria. Recibe como parametro la matriz jugadas, las dimensiones del tablero y la cantidad de minas ingresada por el usuario.

verificarDerrota(tablero, jugadas, filas, columnas): Función encargada de verificar si se seleccionó una casilla con bomba, es decir, que ha perdido la partida. Recibe como parametro las matrices (jugadas y tablero) y las filas y columnas del tablero.

insertarMinas(tablero, filas, columnas, cantMinas, x, y): Función que genera y posiciona en el tablero la cantidad de bombas ingresadas por el usuario de manera aleatoria. Recibe como parametro parametro las matriz tablero, las filas y columnas del tablero, la cantidad de minas y la posición de la primera jugada.

crearArchivo(tablero,filas,columnas): Función encargada de generar un archivo de salida con la respectiva solución de la partida. Recibe de parametro la matriz tablero y las dimensiones de la matriz.

printMatriz(matriz,filas,columnas): Función que genera el formato del tablero visible para el usuario tomando la matriz tablero. Recibe de parametro la matriz a imprimir y las dimensiones de el tablero.

VerificarMargen(i,j,filas,columnas): Función encargada de verificar que la jugada o posición a revisar, se encuentre dentro de los margenes de la matriz, de ser correcto retorna un 1, de lo contrario, retorna un 0. Recibe de parametro la posición a verificar y las dimensiones de la matriz.

insertarNumeros(matriz,filas,columnas): Función encargada de insertar los números que indican cuantas bombas posee cada casilla alrededor. Recibe de parametro la matriz tablero a modificar, junto con las dimensiones del tablero.

expandirMatriz(tablero,jugadas,filas,columnas,posX,posY): Función encargada de expandir el tablero cuando se seleccione una casilla en blanco, sin valor de bomba alrededor. Esta función se desarrolla con un algoritmo recursivo el cual despues de verificar una casilla, vuelve a ser llamada asi misma para realizar una nueva inspección a los elementos nuevos.En caso de encontrarse con un valor numero, la funcion retorna ese valor; y en caso de encontrar una bomba, el algoritmo prosigue para hacer la respectiva verificación en el siguiente paso. Recibe de parametro las matrices (tablero y jugadas) junto con las dimensiones de el tablero y la posición a verificar.

verificarFinal(tablero,filas,columnas,ganar): Función encargada de realizar la ultima revisión y de entregar el resultado de la partida. En caso de que la variable ganar sea TRUE, se genera la victoria, se caso contrario, este retorna la solución de la partida junto con un

aviso de derrota. Recibe de parametro la matriz con el tablero, las dimensiones de la matriz y el booleano ganar.

liberarMemoria(matriz,filas): Función encargada de liberar la memoria solicitada para los arreglos dinamicos sobre los cuales se encuentran montadas nuestras matrices. Recibe de parametro la matriz a limpiar junto a la cantidad de filas de la matriz.

iniciar( ): Función encargada de iniciar todas las variables, obtener los valores de el usuario y usar las funciones explicadas anteriormente.

Main( ): Función que realiza el llamado a la función iniciar.

CAPÍTULO 3. ANÁLISIS DE LOS RESULTADOS

Estos son los T(n) calculados del codigo:

Tabla 1.1: Tabla de tiempos de ejecución

Funciones	T(n)
verificarDerrota ( )	3
verificarVictoria ( )	$n^2 + 4$
insertarMinas ( )	$9n^2 - 15$
crearArchivo ( )	$5n^2 + n + 2$
printMatriz ( )	$7n^2 + 20n + 12$
VerificarMargen ( )	5
insertarNumeros ( )	$64n^2$
expandirMatriz ( )	8^n
verificacionFinal ( )	$7n^2 + 20n + 15$
liberarMemoria ( )	$n + 1$
Iniciar ( )	$8^n + 102n^2 + 80n + 56$
Main ( )	$8^n + 102n^2 + 80n + 56$

El orden de complejidad de este codigo es O(n) = 8^n

Dentro del codigo podemos apreciar que se mantuvo en general un nivel alto en tiempos de ejecución, debido a que siempre nos encontramos trabajando con matrices, por lo que en la mayoria de nuestras funciones nos encontramos con ciclos anidados logrando un tiempo promedio de en la mayoria de las funciones. La función expandirMatriz ( ) es la de mayor costo, debido a que es una función recursiva, lo que implica que cada vez que la función realiza una acción (8 en este caso particular) esta vuelve a llamarse y cada vez con una recursión menos por ejecutar. Dentro de el codigo nos encontramos con un caso particular el cual se concidera como falencia, que consiste en que la funcion expandirMatriz ( ) al empezar a expandirse y lograr avanzar en las posiciones sin valor, esta no entrega los valores numericos alrededor de las bombas, si no que solo retorna las casillas las cuales no poseian valor alguno. Esto es debido a que en el ciclo de la recursión, cuando la función debe entregar ese valor, no entra en esa sentencia y por ende, no retorna el valor. Por temas de tiempo la función no se termino del todo, pero se encuentra funcional para el desarrollo del juego ya que se puede asumir que toda la orilla del area son valores a los cuales se puede acceder sin problema de que sean bombas.

2. CONCLUSIONES

Durante el desarrollo de este laboratorio se pudo empezar a interactuar de frente con la programación en el lenguaje C de tal modo de poder familiarizarse con los conceptos de tiempos de ejecución de nuestros codigos y a la vez con el concepto de optimización, ya que de a poco se logra entender de mejor manera como funciona nuestro codigo y el costo de cada una de las funciones que implementamos. Este proyecto fue bastante util para poder entender como se utiliza el lenguaje de programación C, ya que se utilizaron la mayoria de las funciones basicas para poder implementar las funciones en nuestro codigo, tales como las lecturas de datos por usuario, trabajos con archivos para escribir información, manejo de memoria estatica y dinamica, uso de librerias, etc. Como resultado de lo realizado, se logro completar el enunciado en su mayoria, logrando una implementación idonea de las funciones y de todas las funciones solicitadas.

CAPÍTULO 5. REFERNCIAS

Buscaminas. (2017, 28) de agosto. Wikipedia, La enciclopedia libre. Fecha de consulta: 11:43, agosto 30, 2017 from https://es.wikipedia.org/wiki/Buscaminas