Proyecto TicTacToe

Área de Ingeniería en Computadores

Instituto Tecnológico de Costa Rica

CE310 - Lenguajes, Compiladores e Intérpretes

Estudiantes
Allan Josué Calderón Quirós - 2018114634
Ronny Josué Santamaria Vargas - 2018109283
Antony Fabián Fallas Elizondo - 2018178906

Profesor
Marco Rivera Meneses

I Semestre 2020

Manual de Usuario

El programa es una versión del juego Tic-Tac-Toe el cual, consiste en completar ya sea, una columna, una fila o una diagonal, con el símbolo que se esté utilizando para jugar. En el juego normalmente hay dos jugadores, uno juega con “X” y otro con “O”.
En este caso el programa cuenta con la posibilidad de hacer un tablero de tamaño variable que llega hasta 10 filas y 10 columnas, además el tablero puede no necesariamente ser cuadrado, sino que se puede jugar en con todas las combinaciones de tamaño, siempre y cuando esté dentro del rango anterior.
El programa cuenta con un algoritmo que toma una “decisión” basado en la información que tiene en el momento, este algoritmo intentara ganarle al jugador, escogiendo la mejor posición disponible dentro del tablero, el jugador usará “X” como símbolo de juego y la PC usará “O”.

Requisitos para ejecutar el programa:
Tener instalado el intérprete DrRacket, y tenerlo configurado para el lenguaje “The Racket Lenguaje”. Para configurar esto se deben seguir estos pasos.

1. En la barra superior de DrRacket, seleccionar la pestaña que dice “Lenguaje”.
2. Posteriormente, elegir la opción “Seleccionar Lenguaje”.
3. Para finalizar, elegir “The Racket Language”.

¿Como jugar?
Se debe ejecutar el archivo (Grafic.rkt).
Posteriormente se va a abrir una ventana que le dara la opcion de escoger la cantidad de columnas y de filas que tendrá el tablero. Una vez seleccionado el tamaño del tablero, se presiona el botón aceptar.
Una vez que se ha abierto la ventana donde se muestra el tablero, se debe jugar seleccionando con el mouse, la casilla que se desea seleccionar como jugada. Una vez que el jugador ha seleccionado la casilla, la PC continúa con su movimiento. Posteriormente le corresponderá al jugador y así hasta que suceda uno de los tres casos.
El primer caso es el empate,en este caso tanto el jugador como la computadora no logran ganar la partida y acaba con todo el tablero lleno.
El segundo es el gane del jugador, en este caso el usuario logra completar una fila, una columna o una diagonal con su simbolo de juego.
El último caso es cuando la computadora logra ganar, para esta situación la computadora logra llenar una fila, una columna o una diagonal con su símbolo de juego.
Una vez terminada una partida, para volver a iniciar una partida se puede presionar el botón para volver a jugar o se puede volver a correr el archivo (Grafic.rkt).

1.Documentación

1.1. Descripción detallada de los algoritmos/arquitectura desarrollada:
greedyAlgorithm: El algoritmo funciona evaluando las posibilidades que tiene de ganar basado en la cantidad de filas, columnas y diagonales disponibles en el momento. Inicia verificando si en alguna de las posiciones disponibles hay un posible gane de la PC, en este caso, si hay una fila con todas las casillas iguales menos una la PC toma esta casilla como jugada. En el caso donde la PC no pueda ganar, pero el jugador si, evalúa cual es la primera posición en la que el jugador pueda ganar y mata la jugada. En el caso que ninguno pueda ganar, el algoritmo irá cambiando posiciones en busca de un posible, esto por medio de una tabla de probabilidad en la que conforme se sustituye un valor, calcula una valor donde verifica si hay un posible gane por fila, por columna o por diagonal, la posición que tenga la mayor probabilidad será la posición elegida.

1.2. Descripción de las todas las funciones implementadas:

1.2.1.Funciones de la lógica:
En el archivo “greedyAlgorithm.rkt”

greedyAlgorithm: Es el encargado de ejecutar los movimientos de la PC, basado en el estado que tiene la matriz que entra como parámetro, busca cual es la mejor jugada posible y retorna una matriz con el movimiento efectuado.

greedyAlgorithmAux: Se encarga de ejecutar el movimiento que retornó la PC.

wins?: Verifica si en una posición dada se puede dar un gane de la PC o del jugador, en caso de encontrar una posibilidad retorna un par ordenado con valores (x, y).

bestPos: Calcula cual es la mejor posición para marcar cuando no hay una posición en la que se pueda tener un ganador.

bestPosAux: Busca en lista de probabilidades y toma la probabilidad más negativa para el jugador, en este caso seria la mas positiva para la PC.

probabilityList: Retorna una lista con listas que tienen ((probabilidad) (coordenada x) (coordenada y)).

probabilityListAux: Sustituye posiciones vacías en la matriz y calcula las probabilidades que tiene cada movimiento y retorna una lista con listas que tienen ((probabilidad) (coordenada x) (coordenada y)).

En el archivo “validations.rkt”

win: Verifica si hay un gane ya sea por parte del jugador o de la PC.

horizontalValid: Evalúa que al menos una de las filas de la matriz está llena de un número ya sea 1 o 2.

verifyLineValid: Valida que toda una línea esté llena del mismo valor ya sean todos 1 o todos 2.

verticalValid: Evalúa que al menos una de las columnas de la matriz está llena de un número ya sea 1 o 2.

getColumnValid: Obtiene cada una de las columnas de la matriz.

deleteColumnValid: Elimina cada una de las columnas de la matriz.

diagonalValid: Evalúa que al menos una diagonal de la matriz está llena de un número ya sea 1 o 2.

diagonalDownValid: Verifica las diagonales inferiores a la diagonal principal de la matriz.

diagonalUpValid: Verifica las diagonales superiores a la diagonal principal y la diagonal principal de la matriz.

lenghtMatrixValid: Obtiene la longitud de una lista.

diagonalCheckerValid: Valida que los valores de una diagonal sean todos iguales 1 o 2.

diagonalUpwardValid: Verifica que los valores de las diagonales ascendentes sean iguales 1 o 2.

invertMatrixValid: Invierte la matriz de abajo hacia arriba.

En el archivo “matrix.rkt”

createMatrix: Crea una matriz de **nxm** dimensiones, donde n es cantidad de filas y m cantidad de columnas.

createMatrixAux: Retorna una matriz de **nxm** dimensiones, con valores establecidos en 0.

setValueMatrix: Cambia el valor de una posición en la matriz por el valor de entrada.

replaceSublist: Reemplaza en una sub-lista el valor definido por un valor de entrada.

getValue: Obtiene el valor de una posición de la matriz, en las coordenadas (x, y).

getValueAux: Obtiene el valor en una lista.

possibleWins: Retorna las posibles victorias que puede tener un número en una matriz dada.

horizontal: Verifica que cada fila de la matriz contenga ya sea 0 o el número que tiene por entrada.

verifyLine: Verifica que en una lista sean todos los valores de 0 o el número de entrada.

vertical: Verifica que cada columna de una matriz dada esté llena de un 0 o valor dado.

transposed: Transpone una matriz.

getColumn: Obtiene cada columna de una matriz.

deleteColumn: Elimina cada columna de una matriz.

diagonal: Verifica que cada diagonal de una matriz esté llena de 0 o un valor de entrada.

diagonalDown: Verifica que las diagonales inferiores a la diagonal principal de la matriz contenga solo 0 o un valor de entrada.

diagonalUp: Verifica que las diagonales superiores a la diagonal principal y la principal de la matriz contenga solo 0 o un valor de entrada.

lenghtMatrix: Obtiene la longitud de una matriz.

diagonalChecker: Valida que una diagonal contenga solamente 0 o un valor de entrada.

diagonalUpward: Valida que las diagonales ascendentes.

invertMatrix: Invierte una matriz de arriba hacia abajo.

matrixToList: Convierte una matriz en una lista.

matrixToListAux: Retorna cada uno de los elementos de una matriz en una sola lista.

1.2.2.Funciones de la GUI:

En el archivo “Grafic.rkt”

Ventana: define las ventanas sobre los cuales se pide y colocan datos.

Datos: se encarga de pedirle al jugador que escoja el tamaño de la matriz

Lanzar: le permite al jugador dirigirse a donde ingresará los valores de la matriz, y luego de esto se encarga de verificar verifica si es entre el rango de 3 y 10 e inicializa la pantalla de juego

Ventana-de-dialogo: en donde el usuario ingresa los datos

txt-C: función donde se ingresa el valor de las columnas

txt-F: función donde se ingresa el valor de las filas

click-canvas%: permite utilizar el clic izquierdo, además de que se le agrega las funciones a este click, por ejemplo seleccionar las posiciones de las “X” e imprimirlos

bloquear-matrix: bloquea la matrix cuando existe un empate, derrota o gane.
get-end-game-message: imprime en pantalla un mensaje según corresponda (derrota, gane o perdida)

Menu-bar: crea una barra de menú sobre la ventana de juego
File-menu: guarda dos opciones la de inicar el juego y la de quitar el juego, las cuales se muestran en la Ventana

New-game: permite iniciar el juego desde la ventan luego de haber finalizado

Quit-menu: cierra la ventana y finaliza el juego, se accede a ella desde la ventana de juego

Frame:define un “frame”, que es la ventana donde se muestra toda la GUI

Canvas:define un “canvas” o lienzo sobre el frame, sobre el cual se dibujarán los elementos

Create-matrix:crea la matriz que almacena los valores utilizados durante el juego. Tiene una función auxiliar fill_columns que llena cada fila de la matriz de ceros

Draw-vertical-lines : dibuja las líneas verticales de la matriz

Draw-horizontal-lines : dibuja las líneas horizontales de la matriz

Dc:permite dibujar encima del canvas

Change-at : permite cambiar un elemento en una matriz dada una posición

Draw_X: una función que dibujas las X colocadas por el usuario. Además, modifica la matriz cada vez que esto se realiza e invoca a la función draw_O que dibuja una O, cuya posición le es proporcionada por el greedyAlgorithm. Tiene una función auxiliar draw_X_aux que coloca la X en el tablero una vez indicada la posición

Draw_O: dibuja una O en el tablero (la PC lo hace).Tiene una función auxiliar draw_O_aux que coloca la O en el tablero una vez indicada la posición

1.3. Descripción de la ejemplificación de las estructuras de datos desarrolladas:
Lista de movimientos: Una lista con sub-listas que contienen probabilidad de gane si se hace un movimiento coordenadas donde va fila y después columna. Ejemplo: (probabilidad, fila, columna).

Matriz: Una matriz en la cual se va a jugar, al principio del juego se llena esta matriz con ceros en todas sus casillas, conforme avanza la partida se llena con 1 y con 2 para diferenciar al jugador (usa el número 1) y para diferenciar a la PC (usa el número 2).

Ejemplo: ((0 0 0) (0 0 0) (0 0 0)), cada sub-lista es una fila y el conjunto es el tablero donde se juega.

1.4. Problemas sin solución:
-En algunos casos la PC no lo logra determinar la mejor posición para jugar, esto debido a que el algoritmo se basa en tomar una decisión con la información que tiene en el momento y casi siempre toma la primera opción que tenga la más alta probabilidad de victoria.
-La última columna es ligeramente más delgada que las demás,sin importar las dimensiones de la matriz
-En ocasiones envía mensajes impredecibles al final de la partida(por ejemplo, que el jugador perdió cuando realmente ganó) o greedyAlgorithm retorna una matriz sin sentido y el programa falla. Este error se depuró hasta que no volvió a aparecer, pero no se puede asegurar que siempre será así (lo fue con todos los casos que se probaron)

1.5. Plan de Actividades realizadas por estudiante:

División de las actividades por estudiante:

Encargado	**Tiempo estimado**	**Actividad**	**Fecha de entrega**
Allan Calderón-Ronny Santamaría	15 h	Investigación e implementación del greedy algorithm, unión con lógica	27/06
Ronny Santamaría	10 h	Implementación de matrices y componentes lógicos	26/06
Fabian Fallas-Allan Calderón	15 h	Implementación de matrices gráficas y GUI	28/06
Allan Calderón-Ronny Santamaría-Fabian Fallas	13 h	Unión de GUI con lógica Finalización de la documentación	01/07

1.6. Problemas encontrados:
El greedy Algorithm en algunas ocasiones tomaba una casilla con un valor ya asignado (diferente de 0), y lo cambiaba, esto se debía a que se estaba tomando de forma incorrecta el par ordenado de la posición que retorna la función “bestPos”, el ajuste necesario fue verificar cual era el par ordenado y cuál era el orden de los parámetros que recibe la función “setValueMatrix”.
La función “probabilityList” retornaba los pares ordenados, pero no el valor de la probabilidad calculada, la razón por la cual no se estaba obteniendo la probabilidad de tener una victoria en una casilla del tablero es por que se estaba calculando las probabilidades para el mismo valor dos veces, ejemplo se calculaba la probabilidad de que ganara la PC y luego se le restaba la probabilidad de gane de la PC. Esto se resolvió evaluando la probabilidad de victoria del jugador menos la probabilidad de gane de la PC.
1.7. Conclusiones y Recomendaciones del proyecto:
El algoritmo goloso pedido en la especificación, consume muchos más recursos de la computadora en la cual se ejecuta el programa, debido a que toma muchos datos al mismo tiempo e intenta “decidir” por una solución óptima para el problema en cuestión.

Se recomienda buscar acerca de distintos algoritmos utilizados en el juego de tres en línea. Esto ayudará en la selección de un mejor algoritmo para que la computadora pueda tomar mejores decisiones a la hora de jugar y evitar que consuma tantos recursos.

1.8. Bibliografía consultada en todo el proyecto:
Racket Documentation. (2020). Retrieved 1 July 2020, from https://docs.racket-lang.org/
Greedy algorithm. (2020). Retrieved 1 July 2020, from https://en.wikipedia.org/wiki/Greedy_algorithm

Greedy Algorithms - GeeksforGeeks. (2020). Retrieved 1 July 2020, from https://www.geeksforgeeks.org/greedy-algorithms/
Traverse the matrix in Diagonally Bottum-Up fashion using Recursion - GeeksforGeeks. (2020). Retrieved 1 July 2020, from https://www.geeksforgeeks.org/traverse-the-matrix-in-diagonally-bottum-up-fashion-using-recursion/
Traverse a given Matrix using Recursion - GeeksforGeeks. (2020). Retrieved 1 July 2020, from https://www.geeksforgeeks.org/traverse-a-given-matrix-using-recursion/

Importing and Exporting: require and provide. (2020). Retrieved 1 July 2020, from https://docs.racket-lang.org/reference/require.html
Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI - Finding optimal move) - GeeksforGeeks. (2020). Retrieved 1 July 2020, from https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/

Stack Overflow - Where Developers Learn, Share, & Build Careers. (2020). Retrieved 29 June 2020, from https://stackoverflow.com/

2.Bitácoras digitales

Allan Calderón

Fecha	**Tiempo**	**Actividad**
19/06	1 h	Lectura de los requerimientos del programa, organización y división del trabajo
22/06-27/06	5 h 30 m	Investigación e implementación del greedy algorithm
28/06	4 h	Implementación de tamaño dinámico en matrices gráficas
29/06-01/07	9h 30m	Implementaciones gráficas, unión GUI con lógica

Ronny Santamaría

Fecha	**Tiempo**	**Actividad**
19/06	1h	Lectura de los requerimientos del programa. Organización y división del trabajo.
19/06 - 20/06	1h	Investigación sobre el algoritmo goloso requerido para el proyecto.
20/06 - 21/06	5h	Implementación de las funciones sobre el manejo de la matriz. Validaciones necesarias para determinar la victoria por

		parte de la PC o del jugador sobre la matriz de juego.
22/06 - 26/06	4h	Investigación sobre el greedy Algorithm. Implementación del greedy Algorithm.
30/06 - 01/07	6h	Creación de una interfaz alternativa, para la ejecución del código.
01/07	2h	Finalización de la documentación externa e interna del código.

Antony Fallas

Fecha	**Tiempo**	**Actividad**
19/06	1h 30	Lectura de los requerimientos del programa. Organización y división del trabajo.
19/06 - 20/06	4h	Investigación sobre métodos para crear la interfaz gráfica en racket
20/06 - 21/06	1h	Creación de la ventana de inicio
22/06 - 25/06	12h	Creación de las ventanas en donde el jugador ingresa los valores de matriz que desea. Creación de las las “X” y “O” que se dibujan sobre el tablero
27/28	3h	Conexión entre la logia y la interfaz
29/30	4h	Corrección de errores en la lógica
01/07	1h	Finalización de la documentación externa e interna del código.