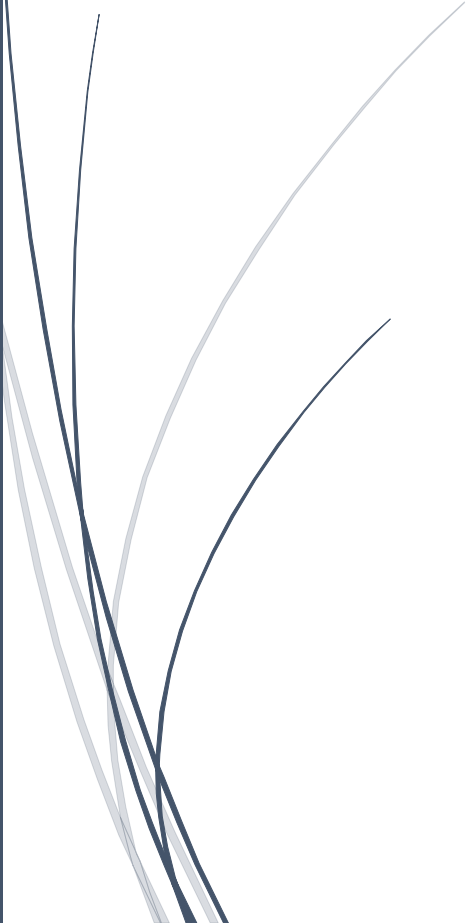
A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

24-1-2022

Mini-Proyecto PD

Tic Tac Toe

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Germán Blanco Pérez-Victoria
Francisco de Asís Bermúdez Campuzano

Contenido

Integrantes del proyecto	2
Temática elegida y problema a resolver	2
Elementos mínimos exigidos.....	3
1. Funciones básicas de prelude	3
2. Data.List.....	3
3. Funciones recursivas	3
4. Funciones por patrones.....	3
5. Uso de guardas.....	3
6. Uso de <i>case of</i>	3
7. Listas por comprensión	3
8. Funciones de orden superior	3
9. Módulos	3
10. Datos algebraicos	4
11. Datos abstractos o librerías.....	4
Compilado y uso del programa	4
Bibliografía	4
Minimax.....	4
TicTacToe.....	4

Integrantes del proyecto

Este proyecto está realizado por los alumnos de Programación Declarativa de la Universidad de Sevilla Germán Blanco Pérez-Victoria y Francisco de Asís Bermúdez Campuzano. Nuestros UVUS son gerblaper y frabercam, nuestros correos de la US son gerblaper@alum.us.es y frabercam@alum.us.es y nuestros email son gblancoperezvictoria@gmail.com y franbercam@hotmail.es

Temática elegida y problema a resolver

Hemos decidido implementar el clásico juego TicTacToe o 3 en raya, al que se podrá jugar tanto en modo JcJ (jugador contra jugador) como en modo JcE (jugador contra entorno).

Al arrancar el juego, la máquina te pregunta a qué modo de juego quieres jugar. Si escribes JcJ, el programa te mostrará el tablero de juego y el jugador 1 comenzará colocando la primera ficha.

Si eliges el modo JcE, tendrás a continuación la opción de elegir la dificultad fácil o difícil. Para conseguir dos dificultades distintas, hemos implementado dos módulos con algoritmos de inteligencia artificial distintos.

Para el modo de juego JcE fácil usamos un algoritmo basado únicamente en aleatoriedad, en el que el programa revisa las posiciones libres del tablero y selecciona una al azar.

Para el modo difícil usamos el algoritmo minimax optimizado con poda. Para implementarlo hemos seguido los siguientes pasos:

1. Generación de un árbol de juego: El árbol está compuesto por nodos cuya etiqueta es un tablero de juego. El nodo raíz representa el estado actual del tablero, y los siguientes nodos generados corresponden a los posibles futuros estados de este.
2. Valoración de los nodos: En nuestro caso, usamos una función maximiza que devuelve 0 si el nodo es una hoja sin ganador, -1 si el nodo es una hoja con ganador o el máximo valor que devuelva la función minimiza en el siguiente nivel de profundidad si no es una hoja; y la función minimiza que devuelve 0 si el nodo es una hoja sin ganador, 1 si es una hoja con ganador y el mínimo valor que devuelva la función maximiza en el nivel de profundidad inferior si no es una hoja.
3. Algoritmo de poda: Nuestro algoritmo simplemente convierte todos los nodos de una profundidad dada en hojas, evitando así que la valoración de nodos tarde demasiado tiempo.
4. Selección de la jugada: Una vez que quedan valorados todos los nodos hasta el nodo raíz, se selecciona como siguiente jugada aquella que genera el tablero cuya valoración es igual a la del nodo raíz.

Estructura del programa

El programa esta estructurado en un módulo principal que importa otros tres módulos correspondientes a cada modo de juego: JcJ, JcE fácil y JcE difícil.

Todos los módulos tienen en común las funciones del módulo JcJ, usadas e resumen para crear el tablero y seguir el curso del juego. Al módulo JcE_F se le añaden varias líneas de comando en la función ia (línea 113) para crear un número aleatorio en el rango de 0 al número de posiciones libres del tablero y que la IA use ese número para hacer un número aleatorio.

Por último, en el módulo JcE_D se añaden todas las funciones correspondientes al uso del algoritmo minimax, explicado en el apartado anterior.

Elementos mínimos exigidos.

1. Funciones básicas de prelude

- 1.1. Línea (60), función(tienGanador), módulo (JcJ) // uso de listas.
- 1.2. Línea (78), función (muestratablero), módulo (JcJ) //uso de string.

2. Data.List

- 2.1. Línea (33), función (turno), módulo (JcJ) // uso de length
- 2.2. Línea (89), función (main), módulo (JcE_F) // uso de head

3. Funciones recursivas

- 3.1. Línea (43), función (elementoEnLista), módulo (JcJ)
- 3.2. Línea (146), función (poda), módulo (JcE_D)

4. Funciones por patrones

- 4.1. Línea (43), función (elementoEnLista), módulo (JcJ)
- 4.2. Línea (146), función (poda), módulo (JcE_D)

5. Uso de guardas

- 5.1. Línea (54), función (continuaJuego), módulo (JcJ)
- 5.2. Línea (67), función (muestraPosicion), módulo (JcJ)

6. Uso de *case of*

- 6.1. Línea (132), función (maximiza), módulo (JcE_D)
- 6.2. Línea (139), función (minimiza), módulo (JcE_D)

7. Listas por comprensión

- 7.1. Línea (151), función (seleccionaTablero), módulo (JcE_D)
- 7.2. Línea (50), función (posicionesLibres), módulo (JcJ)

8. Funciones de orden superior

- 8.1. Línea (60), función (tieneGanador), módulo (JcJ) // uso de `elem`
- 8.2. Línea (155), función (jugadaIA), módulo (JcE_D) // uso de composición

9. Módulos

- 9.1. Un módulo principal llamado TicTacToe
- 9.2. Tres submódulos llamados:
 - 9.2.1.JcJ
 - 9.2.2.JcE_F

9.2.3.JcE_D

10. Datos algebraicos

- 10.1. Línea (98), dato Arbol, módulo (JcE_D)
- 10.2. Línea (24), dato Tablero, módulo (JcJ *todos)

11. Datos abstractos o librerías

- 11.1. Uso librería Data.List
- 11.2. Línea (74), función (muestraLinea) // uso de map

Compilado y uso del programa

Para el compilado de nuestro programa hemos utilizado el editor de código Visual Studio Code donde se ha instalado y configurado un interprete haskell. Para ello disponemos de las extensiones:

- Haskell Syntax Highlighting
- Haskell Runner

Para usar el programa únicamente será necesario correr desde la terminal el comando ticTacToe, para ello será necesario cargar el módulo TicTacToe.hs mediante GHCi

```
*Main> :l "c:\\Users\\franb\\Desktop\\Universidad\\Primer Cuatrimestre\\3º - Programación Declarativa\\Estudio\\TicTacToe\\TicTacToe.hs"
[1 of 4] Compiling TicTacToe.JcE_D ( TicTacToe\\JcE_D.hs, interpreted )
[2 of 4] Compiling TicTacToe.JcE_F ( TicTacToe\\JcE_F.hs, interpreted )
[3 of 4] Compiling TicTacToe.JcJ ( TicTacToe\\JcJ.hs, interpreted )
[4 of 4] Compiling Main ( C:\\Users\\franb\\Desktop\\Universidad\\Primer Cuatrimestre\\3º - Programación Declarativa\\Estudio\\TicTacToe\\TicTacToe.hs, interpreted )
Ok, four modules loaded.
*Main> ticTacToe
Seleccione un modo de juego -> JcJ / JcE
JcE
Seleccione una dificultad -> Facil / Dificil
Dificil
Tres en raya
1|2|3
--+--
4|5|6
--+--
7|8|9
¿Desea ser la primera jugada? (s/n) █
```

Bibliografía

Minimax

<http://www.cs.us.es/~fsancho/?e=107>

<https://es.wikipedia.org/wiki/Minimax>

TicTacToe

<https://www.youtube.com/watch?v=0-pOaa0dpko>

<https://www.glc.us.es/~jalonso/vestigium/el-juego-de-tres-en-raya-en-haskell/>