Materia: Estructura de datos y algoritmos

Docentes:

- Frias Marcelo Fabian
- Itzcovich Ivan
- Sal-lari Julieta
- Liberman Daniella

Trabajo práctico especial:.

Alumnos:

- Bossi, Agustín - legajo: 57068
- Dorado, Tomás - legajo: 56594
- Martín, Fernando - legajo: 57025

Estructuras

Tablero

El tablero está compuesto por un entero que guarda el tamaño, un vector de enteros que guarda referencia a los cuadrados del tablero y dos HashSets, uno para los movimientos realizados y uno para los movimientos aún posibles en el juego. Dado que los movimientos iban a ser una de las colecciones más consultadas a lo largo del código, se pensó que lo mejor era utilizar una estructura que permitiera la consulta en orden constante.

Juego

El juego está conformado por dos jugadores (clase Player), un tablero, una variable entera para el manejo de los turnos, otra variable entera para el tipo de AI y un Stack de movimientos realizados (clase MoveDone) que almacena todos los movimientos hechos durante la partida. Se eligió un Stack debido a que las características del mismo son ideales para la funcionalidad undo.

Jugador

La clase jugador consiste en un entero que guarda puntaje y una referencia al juego, a través de la cual se pasan los movimientos realizados y se accedía al tablero para el manejo de los Sets de movimientos.
También se tiene la clase AIPlayer que extiende de Player, como tiene que usar los mismos métodos, para realizar los movimientos, una vez que los elige, y también tiene que tener un puntaje.

Movimiento

Move

Clase sencilla, getters, equals,toString y hashCode como métodos. Las jugadas están compuestas por cuatro enteros, rowFrom, rowTo, colFrom y colTo.

Move done

Un Move y una variable Player que indica quien hizo el movimiento.

Guardado y cargado de partidas

Está desarrollada la serialización del juego, de manera que cuando se guarde una partida se va a guardar en un archivo binario, lo mismo al cargarlo.

Parte visual

En cuanto a la parte visual se tuvieron grandes problemas debido a que ninguno de los integrantes poseían ningún tipo de conocimiento sobre algún framework visual. En cuanto a lo desarrollado, generamos una grilla con círculos referenciando a los nodos del juego, y las aristas las colocamos con rectángulos. Estas últimas se diferencian con un color de acuerdo al jugador que haya realizado el movimiento.

Hay un botón de undo move, que deshace el último movimiento, un botón make dot file, que guarda en la carpeta target el archivo dot llamado “jugada.dot” correspondiente a la jugada anterior de la computadora, y un botón save game que guarda la partida en un archivo serializado en la carpeta target, llamado “partida.game”. También hay un boton next turn que si es jugador, toma los clics hechos con los cuales se seleccionó una arista y genera ese movimiento, o si es el turno de la computadora, calcula su movimiento y lo hace.

Para los turnos, se decidió hacerlo simple siempre se debe presionar el botón next turn, para hacerlo más simple, que realiza la acción correcta si el turno es de una computadora o de un jugador. Luego de cada vez que se presione next turn se va a actualizar las aristas del tablero.

MinMax y heurística

El problema con el minimax fue que no eran pasos de simplemente un movimiento, sino que pueden ser de varios, ya que al ganar un punto el jugador debe seguir jugando, entonces lo que hicimos fue que se agreguen los movimientos a una lista, y cuando cambia el turno, recien ahi llama de nuevo a minimax para obtener su valor.

Para cada recursión del minimax al principio lo hicimos de manera que cree una copia de la instancia del tablero, pero esto no solo consume mucha memoria, sino que tardaba mucho más tiempo ya que tenia que hacer una copia profunda. Lo elegido finalmente fue que después de cada recursión se haga la función deshacer de los movimientos realizados correctamente.

La función heurística fue simple, en caso de que con ese movimiento la partida termine, si hizo que termine siendo ganador, retorna un valor alpha que es 10000, si

hizo que termine siendo perdedor, retorna un valor beta que es -10000. Si no es ninguno de los anteriores, retorna el puntaje suyo - puntaje del jugador contrario.

Como se debe poder hacer un archivo .dot con todos los nodos en el formato pedido, lo que se hizo fue una clase que guarde cada movimiento y las siguientes instancias de ese movimiento. Esto se va guardando siempre a medida que se hace el minimax de manera que siempre quede guardado como se eligió el movimiento anterior de la computadora.