AJEDREZ

AUTORES:

- Ángel Alfonso Jiménez Martínez
  - Uvus = angjimmar1
  - Correo = angjimmar1@alum.us.es
- Antonio Jesús Caro Vela
  - Uvus = antcarvel
  - Correo = antoniojesuscarovela@gmail.com

Contenido

INTRODUCCIÓN AL PROBLEMA

En este trabajo no resolveremos un "problema" como tal, a no ser que se nos presente el aburrimiento o queramos practicar este deporte y nuestra mente. Hemos decidido realizar un juego de ajedrez con interacción de entrada y salida, basándonos en el lenguaje de programación declarativa Haskell.

Como añadido hemos implementado una pequeña IA basada en el algoritmo Minimax, que nos servirá de contrincante.

COMPONENTES

Está compuesto de 2 archivos, el primero (main.hs) funcionará de programa principal que se podrá ejecutar desde la consola, teniendo mayoritariamente funciones encargadas de realizar el tratamiento de entrada y salida. El segundo archivo (tablero.hs) contiene las funciones necesarias para su correcto funcionamiento, siendo el tablero un módulo que se importa en el main.

El código del tablero presenta un orden en concreto. Empezamos por las importaciones de las librerías necesarias para su uso, seguido de los tipos nuevos creados.

Después de esto, las funciones referentes a los movimientos y al tablero, de forma general.

A continuación, los detalles y funciones necesarias para llevar a cabo la partida.

Por último, la IA. En esta parte del código están las funciones referentes a la inteligencia artificial y a su correcto funcionamiento para determinar las evaluaciones correspondientes para realizar los movimientos que estime necesario.

Aun con esta explicación en el documento, dentro del código de cada archivo se disponen de comentarios que aclaran el funcionamiento de las funciones.

TABLERO Y PIEZAS

El ajedrez está compuesto por una matriz 8x8 de la librería Matrix vista en la asignatura, pero a la hora de mostrarlo por pantalla no se muestra solo esta matriz, también se muestran las piezas y las casillas vacías con el carácter "_".

Las piezas son letras mayúsculas que indican en función de su nombre qué pieza es, así como los signos positivo y negativo si son blancas o negras respectivamente:

- Peón (P): De salida puede moverse 1 o 2 casillas hacia adelante, si no es salida solo 1 casilla. Puede matar en diagonal hacia adelante.
- Caballo (C): Única pieza que puede saltar otras. Se mueve en forma de L.
- Alfil (A): Se mueve en diagonal hasta encontrar una pieza que, si es enemiga, puede matarla.

- Torre (T): Se mueve en vertical y horizontal hasta encontrar una pieza que lo obstaculice y, si es enemiga, puede comerla.
- Dama (D): Se desplaza en todas direcciones, vertical, horizontal y diagonal hasta encontrar una pieza que la obstaculice o matarla si es enemiga.
- Rey (R): Es la pieza más valiosa de la partida, si el rey está amenazado y no puede salvarse, se acaba la partida. Se mueve en todas direcciones, pero solo 1 casilla.

MOVIMIENTOS ESPECIALES

- Captura al paso: Cuando un peón sale moviéndose 2 casillas y un peón enemigo pudiese capturarlo si se hubiese movido solo 1, puede hacerlo.
- Enroque: Si el rey no se ha movido de su casilla en toda la partida, ni alguna de las torres, y no hay piezas intermedias ni casillas amenazadas o el rey amenazado, puede moverse dos casillas y posicionar su torre en el lado opuesto.
- Promoción: Al llegar un peón a la última fila del tablero, se puede convertir en la pieza de más valor que queramos, ya sea caballo, alfil, torre o dama.
- Jaque: Cuando el rey está amenazado, no se puede realizar ningún movimiento que no lo salve de estar en jaque.
- Jaque mate: Fin de la partida. El rey está amenazado y no puede salvarse.
- Rey ahogado: No se puede realizar ningún movimiento y el rey no está amenazado. Esto es tablas (empate).

PROGRAMACIÓN Y APLICACIONES DE LOS ELEMENTOS MÍNIMOS

En este apartado vamos a explicar cómo y dónde se han aplicado cada uno de los elementos mínimos exigidos.

Conceptos Básicos:

- En prácticamente cualquier función se usa alguna función básica de prelude, 2 ejemplos de este uso son la función verifica (uso de +) y comparaTableros (uso de /=). Las funciones del Data.List se usan bastante como elem o concat en las funciones movimiento_valido y posiblesMovimientos respectivamente.
- Funciones definidas recursivamente tenemos, entre otras, la función minimax y la función maximum'.
- Patrones usamos muchos, ya que cada vez que se recorre una lista se usa el patrón xs (por ejemplo), así como el uso del "@" o cualquier entrada de parámetros en una función. Por ejemplo, las funciones minimax y movimientos.

- Usamos tanto las guardas como el if else, que viene a ser lo mismo, pero con varias comprobaciones. Esto lo podemos ver en las funciones tablero_inicial y movimientos, entre otras.
- Usamos case of solamente en el tratamiento de la entrada y salida.
- Cualquier función referente a posiciones, como en cada uno de los movimientos de las piezas.
- El orden superior lo usamos mediante las funciones foldr o filter, en las funciones movimiento_con_chequeo_jaque y movimientos_posibles_peon.
- Todas las funciones tienen sus correspondientes declaraciones de tipo.
- La evaluación perezosa la usamos al calcular los números para las posiciones con take, y el takeWhile. Se usa en las funciones lista_pos_enemigos y PosicionesDisponiblesFiltro respectivamente.

Creación de un módulo:

Al principio del fichero "tablero.hs" hemos declarado con module las funciones necesarias así como los tipos necesarios que necesita nuestro fichero principal "main.hs". Éste se importa en el fichero principal que trata la entrada y salida del juego. Creemos que separar la entrada y la salida de las funciones necesarias para el juego facilitan la lectura de la interacción entre los jugadores y/o la inteligencia artificial.

Creación de tipos de datos nuevos:

Tablero → Este nuevo tipo es una matriz hecha de casillas. Se usa para jugar y tener en cuenta cómo estaba el tablero antes de un movimiento y cómo quedará después del mismo.

Casilla → Lo creamos para determinar si la posición en el tablero está ocupada o vacía y de qué color es la pieza.

ColorPieza → Con este nuevo tipo asociamos en una tupla la pieza con su color.

Pieza → Tipo de pieza.

Color → Si la pieza es Blanca o Negra.

Posicion → Par de dos valores enteros que designa la fila y la columna de la matriz tablero.

Direccion → Lo usamos para determinar si es la torre izquierda o derecha.

Uso de tipos de datos abstractos o librerías vistos en la asignatura:

En todo momento usamos el tipo Matrix, ya que el propio tablero es un tipo de dato nuevo creado con este tipo abstracto. Se accede a las posiciones mediante esta librería y creamos el tablero inicial o se generan los movimientos a través de ella.

En cambio, el otro tipo de datos abstractos que usamos solo se usa para una cosa en específico, y es el enroque. Este movimiento especial requiere de varias condiciones que deben de haberse cumplido durante toda la partida y movimientos anteriores a la realización de este. Por esto, decidimos inicializar un mapa con clave String, siendo si las piezas son Negras o Blancas, y con valores una lista de 3 Bool, inicialmente a True. Con esto tenemos que para la clave Negra (por ejemplo), sus valores indican si se han movido anteriormente en la partida la torre izquierda, el rey y la torre de la derecha. Así, si se mueve una torre, no se podría realizar el enroque con esa torre, pero sí con la otra. Si se mueve el rey directamente no podríamos hacer enroque.

INTELIGENCIA ARTIFICIAL

Hemos creado una pequeña inteligencia artificial para que un jugador pueda jugar contra la máquina. La idea es usar el algoritmo Minimax de tal forma que comenzando en un tablero inicial hagamos un árbol con todos los movimientos posibles y le aplicamos el máximo para nosotros y el mínimo para el adversario ya que intentamos que cometa el menor error posible. Para cada nodo terminal calculamos los valores en función del tablero. Esta evaluación no es más que la diferencia de material entre las piezas blancas y las piezas negras.

COMO EJECUTAR

Para ejecutar nuestro programa simplemente nos dirigimos al fichero “Main.hs” y le damos a “Run File” ya que directamente incluye una función main. Primero nos propondrá el numero de jugadores que queremos que jueguen. Debemos poner 1 si queremos jugar con la inteligencia artificial o 2 para jugar dos personas. Si no ponemos ninguno de esto pues nos pedirá que lo introduzcamos bien.

Si ponemos 2 ya podremos empezar a jugar. Las Blancas siempre comienzan. Podemos ver los movimientos de una ficha pasándole una posición, mover una ficha pasándole una posición de partida y una posición de destino o bien salir del juego. Ya podemos jugar de dos jugadores.

Si ponemos 1 entonces nos pedirá el color con el que queremos empezar. Tendremos que poner Blanca si queremos empezar. En cualquier otro caso empezaremos con las negras. Siempre por consola nos dice que debemos responder, así evitamos confusiones.

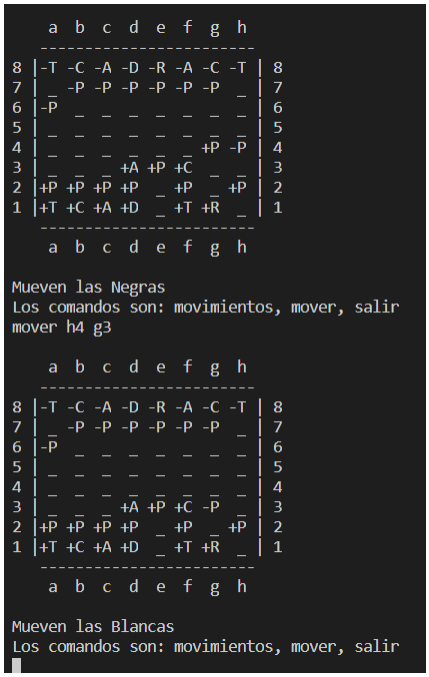
INSTALAR SPLIT

Para que vaya bien el juego se requiere de una librería que hemos usado proveniente del Data.List. Por tanto, si no funciona correctamente es porque no se ha instalado dicha librería anteriormente. Para ello, metemos en la consola del sistema lo siguiente:

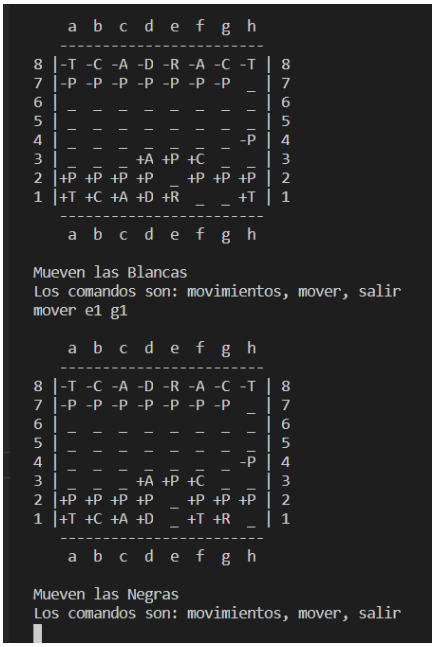
cabal install split

CAPTURAS CON EJEMPLOS

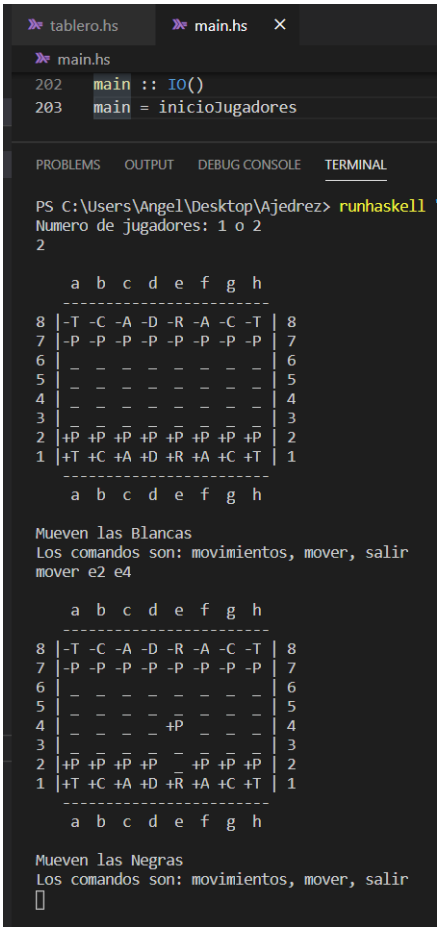
- Captura del peón al paso



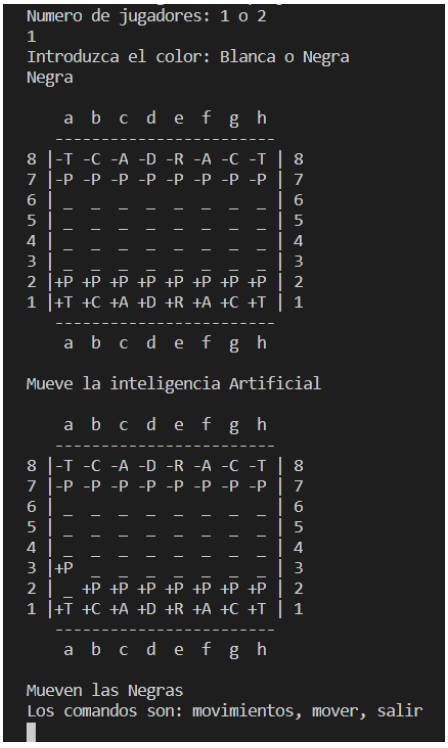
- Enroque



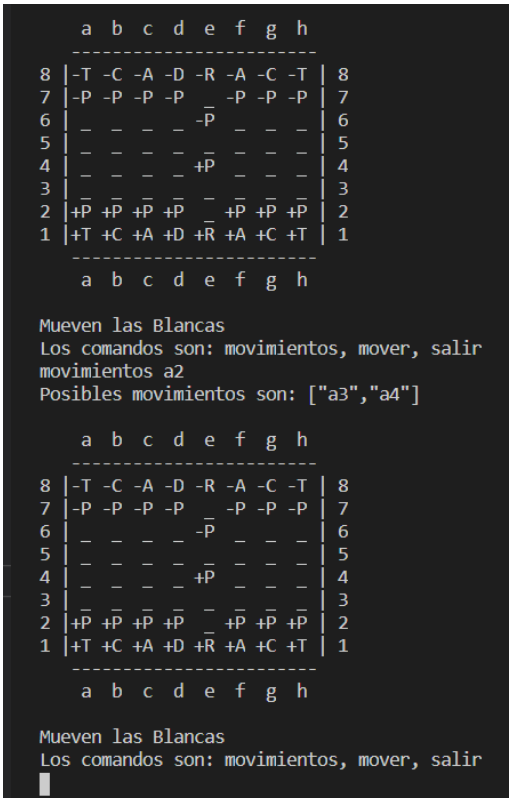
- Juego 2 jugadores



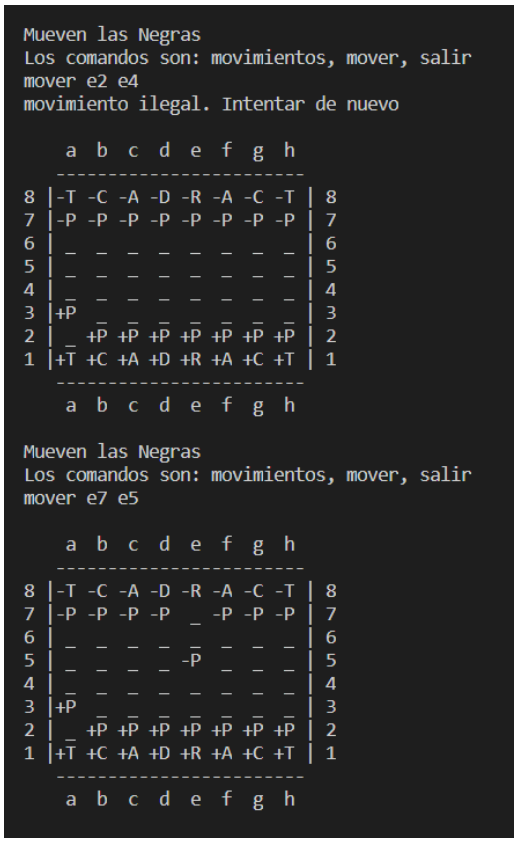
- Juego con Inteligencia Artificial.



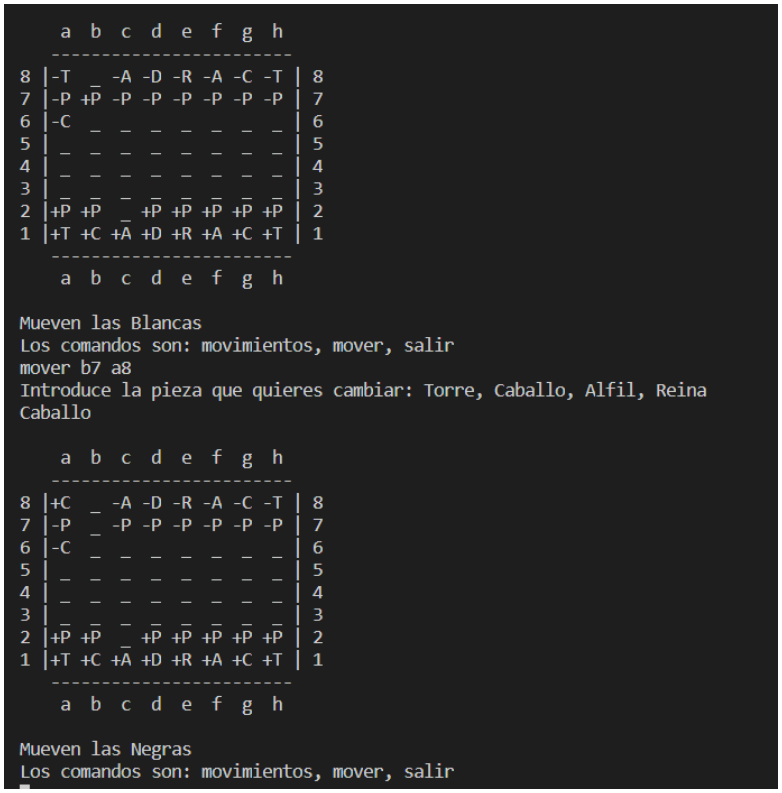
- Movimientos posibles



- Movimiento ilegal



- Promoción del peón



- Posición mal introducida.