25.03 Algoritmos y Estructuras de Datos

Level 3: EOAversi

Introducción

En esta práctica vamos a implementar un motor de inteligencia artificial para el juego Reversi.

Puedes encontrar el reglamento del juego junto a este enunciado.

Parte 1: Genera los movimientos válidos y realiza la jugada

En la primer etapa de esta práctica deberás generar la lista de movimientos válidos posibles que un jugador puede realizar para un estado específico del tablero.

Parte del starter code que te damos junto con el enunciado, y modifica la función `getValidMoves()` del módulo `model.cpp`.

Para acceder al estado del juego, usa las funciones de acceso `getCurrentPlayer()` y `getBoardPiece()`; evita acceder directamente a los campos de `GameModel`.

Para verificar si una posición se encuentra dentro de los límites del tablero, aprovecha la función `isSquareValid()`.

Luego, modifica `playMove()` para actualizar el tablero cuando se haga una jugada.

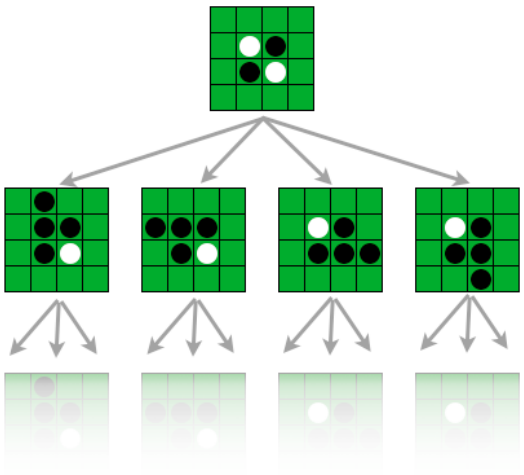
Una vez que hayas implementado `getValidMoves()` y `playMove()`, podrás jugar al Reversi con una inteligencia artificial que simplemente selecciona un movimiento válido al azar.

Para garantizar la calidad de tu generador de movimientos válidos y tu algoritmo de jugada, es fundamental que lo sometas a pruebas exhaustivas, considerando especialmente los casos límite. Documenta tus pruebas en `README.md`.

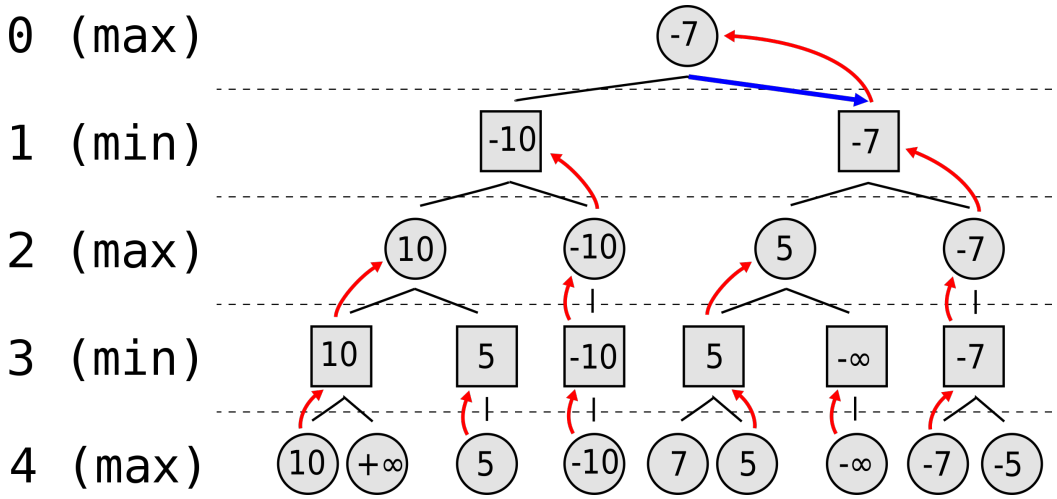
Parte 2: Implementa la inteligencia artificial

Para determinar el movimiento óptimo, vamos a utilizar el algoritmo Minimax que involucra los siguientes pasos:

- **Generación del árbol de juego:** genera todos los movimientos posibles de juego hasta alcanzar un estado terminal (el fin del juego).



- **Evaluación de los nodos hoja:** evalúa los nodos hoja a través de una *función de evaluación del estado del tablero*. Por ejemplo: +1 para victoria, 0 para empate, o -1 para derrota.
- **Evaluación de los nodos internos:** evalúa los nodos internos, quedándote con el máximo de los nodos hijos si la jugada es tuya, o con el mínimo, si la jugada es del oponente. Este paso es el que le da el nombre al algoritmo "minimax".



- **Evaluación del nodo raíz:** en el nodo raíz, selecciona el movimiento que maximiza el valor de los nodos hijo.

Implementa el algoritmo minimax en la función `getBestMove()` del módulo `ai.cpp`.

Puedes utilizar copias del estado de juego `gameModel` para que el usuario no vea lo que la inteligencia artificial hace. Aprovecha la función `getValidMoves()` para determinar los nodos hijo.

Piensa en cómo optimizar el uso de memoria y cálculo.

Parte 3: Poda el árbol

Habrás notado que la función anterior no finaliza, incluso aunque esperes mucho tiempo. Justifica en `README.md` por qué sucede esto.

Para resolver este problema, debes aplicar dos técnicas de poda del árbol:

- **Poda por profundidad:** recorta el árbol cuando superas cierta profundidad en la búsqueda.
- **Poda por cantidad de nodos:** detén la búsqueda cuando hayas explorado una cantidad específica de nodos. ¡El número de nodos debe poder ser configurable!

Además de aplicar estas técnicas de poda, también deberás ajustar la función de evaluación, dado que ya no evalúas el tablero al final del juego. Investiga en Internet qué funciones de evaluación son óptimas para el juego Reversi.

Tips

- **Llama periódicamente desde tu inteligencia artificial a `drawView()`** para que la interfaz gráfica siga respondiendo mientras tu algoritmo hace sus cálculos. Cuidado: `drawView()` espera al próximo fotograma, por lo que tampoco debes llamarlo demasiado seguido.

Bonus Points

Más cosas para hacer:

- **Compara diferentes enfoques:** hazlos competir entre sí con *auto-play* y quédate con el mejor.
- **Investiga e implementa la poda alpha-beta:** ¡para osados!