Trabajo Práctico Especial
Estructura de Datos y Algoritmos

2Q 2018

Objetivo

El objetivo del trabajo es desarrollar un algoritmo *minimax* para el juego **Reversi.**

Condiciones

- **Los grupos deberán ser conformados por 3 integrantes.**
- **La fecha límite de entrega es el 5 de Noviembre a las 23:59hs.**
- Existe la opción de entregar en fecha tardía, el Lunes 12 de Noviembre a las 23:59hs. En tal caso, la nota de aprobación será de 4 puntos.

Juego

- Para conocer las reglas del juego, https://es.wikipedia.org/wiki/Reversi

- Para tener la experiencia de cómo se juega, pueden recurrir al siguiente sitio: http://www.webgamesonline.com/reversi/

Implementación

Se pide implementar una aplicación en **Java (> 8)** para jugar al juego, valga la redundancia. Para esto, deben desarrollar una interfaz gráfica para interactuar con el programa. La interfaz debe ser únicamente un mecanismo fácil para evitar usar la consola. **No se va a evaluar la interfaz gráfica**, alcanza con algo simple y funcional.

Inicializando el juego

El juego debe poder soportar como parámetros:

- El tamaño del tablero (de 4x4, 6x6, 8x8 o 10x10)
- El rol de la AI
- Ejecutar el algoritmo minimax:
    - Por tiempo máximo
    - Por profundidad máxima del árbol
- Habilitar poda alfa-beta

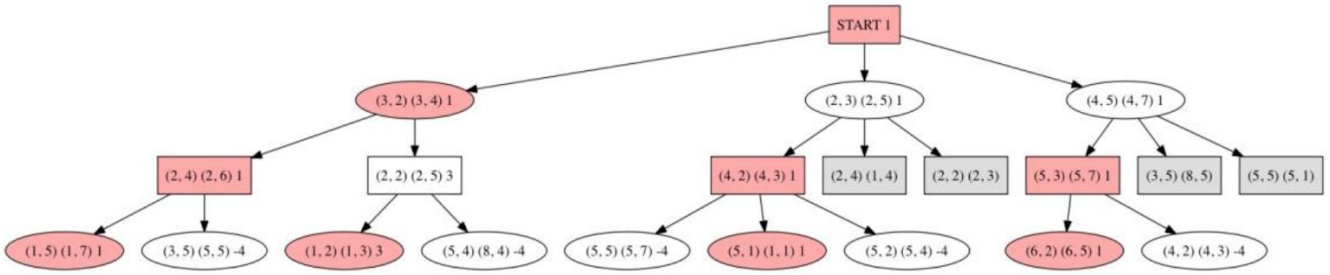
Opciones de GUI

La interfaz gráfica debe soportar (además de la funcionalidad del juego):

- Un opción de *undo*: deshace la última jugada. Se puede ejecutar múltiples veces, obviamente.
- Una opción de generar un archivo en formato **dot** (link) con el árbol que fue explorado por el algoritmo en la última jugada de la AI.
- Poder guardar el estado actual del tablero para luego inicializar un juego con el estado guardado. Esto servirá si se quiere mostrar algunos ejemplos ejecutables.

Acerca del árbol dot:

En el árbol generado debe aparecer claramente el arbol de estados donde se distingan las jugadas de cada nodo con su respectivo puntaje, y se deben poder distinguir los nodos elegidos en cada decisión (ya sea por *min* o por *max*) y también los nodos que pudieron ser podados (en el caso que esté activada la poda). A continuación se muestra un ejemplo:



Ejecutando el programa

El programa va a ser ejecutado de la línea de comandos. El ejecutable debe ser un *.tar* (existen múltiples herramientas, como *ant* o *maven,* para poder generar este archivo). La sintaxis será la siguiente:

```
$ java -jar tpe.jar -size [n] -ai [m] -mode [time|depth] -param [k] -prune [on|off] -load [file]
```

Donde:

- -size [n]: determina el tamaño del tablero. “n” debe ser un número entero.
- -ai [m]: determina el rol de la AI. “m” es un número que significa:
  - 0: no hay AI. Juegan dos jugadores humanos
  - 1: AI mueve primero
  - 2: AI mueve segundo

- -mode [time|depth]: determina si el algoritmo minimax se corre por tiempo o por profundidad.
- -param [k]: acompaña al parámetro anterior. En el caso de “time”, k deben ser los segundos. En el caso de “depth”, debe ser la profundidad del árbol.
- -prune [on|off]: activa o desactiva la poda.
- -load [file]: *opcional.* Este parámetro debe cargar la partida previamente guardada. “file” es una referencia al archivo donde se guardó el tablero. **En el caso de utilizar esta opción, no es necesario especificar el tamaño del tablero, pues este dato se debe interpretar directamente sobre el tablero cargado.**

Ejemplos:

En este ejemplo, se ejecuta el programa con un tablero de 6x6, donde la computadora mueve primero. *Minimax* corre por tiempo (30 segundos) y hay poda alfa-beta.

```
$ java -jar tpe.jar -size 6 -ai 1 -mode time -param 30 -prune on
```

En este segundo ejemplo, se ejecuta el programa con un tablero guardado en el archivo *saved.txt*. Minimax corre por profundidad (llegando máximo a 10 niveles) y no hay poda alfa-beta. La computadora mueve segundo.

```
$ java -jar tpe.jar -ai 2 -mode depth -param 10 -prune off -load saved.txt
```

Entrega

El dia de la entrega, deberan mandar un *email* (con copia a todos los profesores), en el cual el proyecto puede estar en alguno de los siguientes formatos:

- Como archivo adjunto en el *email.*
- Proporcionando un link a un repositorio *git* donde este alojado el proyecto.

Cualquiera sea el formato de entrega, el proyecto debe contener:

- Un ejecutable del programa (testeado en múltiples versiones de Java y en múltiples sistemas operativos, para evitar conflictos de ejecución).
- El código fuente.
- Un archivo README **donde figuren todas las instrucciones para poder generar los ejecutables.**
- El informe en formato PDF.

IMPORTANTE. Deberán utilizar alguna herramiento tipo Ant o Maven para la generación del ejecutable. No es válido indicar en el README algo del estilo: “*Abrir Eclipse, importar el proyecto y ejecutar Run”.*

Acerca del informe

El informe debe contener:

- Justificación de la elección de estructuras de datos.
- Detalle de la(s) heurística(s) elegidas y justificación.
- Decisiones de diseño para la implementación del algoritmo *minimax.*
- Problemas encontrados durante la implementación del algoritmo.
- Métricas de tiempo para distintas configuraciones de minimax (distintas profundidades, con/sin poda alfa-beta, etc) y de heurísticas.
- Interpretación de las heurísticas basándose en las métricas de tiempo. En este juego, sirven heurísticas más complejas o aquellas que requieren muy poco computo?

El informe no debe superar las 4 hojas de longitud. Incluyan en el informe todo lo que consideren necesario ser considerado para la evaluación.

Criterios de evaluación

Los criterios de evaluación son:

- Correcto funcionamiento.
- Detalles de implementación del algoritmo *minimax*
- Elección de estructura de datos.
- Modularización y capacidad de extensión.
- Informe