Trabajo Práctico Especial
Estructura de Datos y Algoritmos

1Q 2018

Objetivo

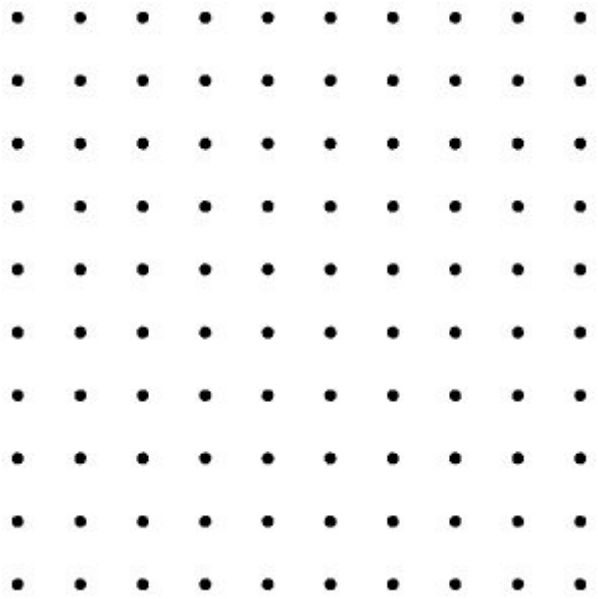
El objetivo del trabajo es desarrollar un algoritmo *minimax* para el juego **Timbiriche.**

Condiciones

- **Los grupos deberán ser conformados por 3 integrantes.**
- **La fecha límite de entrega es el 19 de Junio a las 23:59hs.**

Juego

El juego inicia con el siguiente tablero vacío:

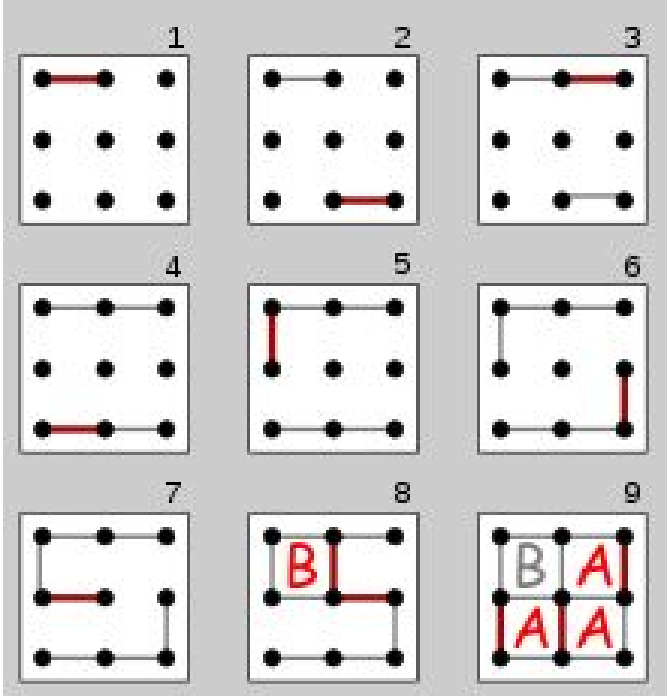


El tablero anterior es de 10x10, pero podría tener otras dimensiones (siempre siendo cuadrado).

Para este trabajo, el juego va a ser únicamente de 2 personas (aunque se podría jugar con más personas).

A continuación, la partida se maneja por turnos. En cada turno, un jugador debe unir dos vértices contiguos con una línea (de una manera horizontal o vertical). Si un jugador cierra un cuadrado (de longitud 1), entonces gana un punto y entonces puedo volver a dibujar otra línea (si en esta última línea cierra otro cuadrado, vuelve a jugar y así sucesivamente hasta que no complete ningún cuadrado).

Ejemplo de ejecución:



La partida termina cuando no queden más lugares donde poner un lado. Entonces gana el jugador con más puntos (es decir, aquel que ha encerrado una mayor cantidad de cuadrados).

Para más información del juego, proponemos el siguiente link:

- https://es.wikipedia.org/wiki/Timbiriche_(juego)

En el siguiente link, se puede jugar al juego propuesto:

- http://es.dotsgame.co/games

Implementación

Se pide implementar una aplicación en **Java** para jugar al juego. 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
- 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):

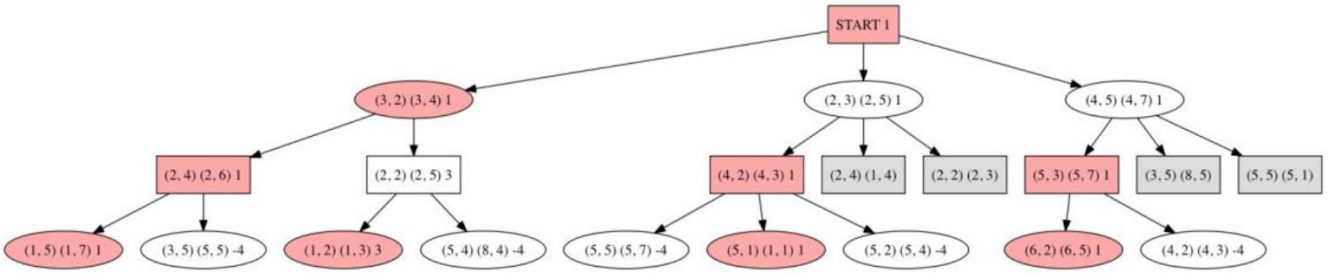
- Un opción de *undo*: deshace la última jugada. Se puede multiples veces, obviamente.
- Una opción de generar un archivo en formato **dot** (link) con el árbol que fue explorado por el algoritmo en la última jugada de la AI.

[Opcional]

- *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
    - 3: AI vs AI
- -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. Implementarlo en el caso de implementar la opcion de guardado. Este parámetro debe cargar la partida previamente guardada. “file” es una referencia al archivo donde se guardó el tablero.

Ejemplos:

```
$ java -jar tpe.jar -size 7 -ai 1 -mode time -param 30 -prune on
```

```
$ java -jar tpe.jar -size 5 -ai 3 -mode depth -param 10 -prune off -load saved.txt
```

Entrega

La entrega deberá realizarse a través de un repositorio *git* provisto por la cátedra. (Este repositorio puede utilizarlo para el desarrollo del código). En dia de la entrega, el repositorio debe contener:

- Un ejecutable del programa.
- 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 así como: “*Abrir eclipse, importar el proyecto y ejecutar Run”.*

Acerca del informe

El informe debe contener:

- Justificación de la elección de estructuras de datos basadas en métricas de tiempo, espacio, etc.
- Detalle de la(s) heurística(s)
- Decisiones de diseño para la implementación del algoritmo *minimax.*
- Problemas encontrados durante la implementación del algoritmo.
- Posibles extensiones del trabajo y como las desarrollarían.

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