UADE

Programación lll
Cuatrimestre 2 de 2022

TRABAJO PRÁCTICO OBLIGATORIO

Turno: **Viernes, Noche**

GRUPO Nº: 3

1145452	Yoo, Juan Pablo
1148226	Giraudo, Ariel

Fecha de entrega: 25/11/2022

Introducción

Este informe tratará acerca del desarrollo del juego “TATETI”. Para el mismo utilizaremos y aplicaremos, entre otras herramientas, la técnica del backtracking. Este consiste, como vimos en clase, en la exploración directa de todas las posibilidades (fuerza bruta). Implica un método de búsqueda de soluciones exhaustivo sobre grafos acíclicos.

En primera instancia, daremos una breve explicación acerca del algoritmo MinMax que utilizamos para resolver el problema y la implementación del mismo. Luego presentaremos la interfaz y la estrategia utilizada para implementar el juego junto a las líneas del código.

Algoritmo MinMax

El algoritmo MinMax es un algoritmo de backtracking en el cual tendremos dos jugadores hipotéticos Min y Max que harán una movida por turno. El jugador Min buscará minimizar el resultado del juego mientras el jugador Max intentará maximizarlo. En nuestro caso en particular, habrá tres posibles resultados:

- 10 si gana Max
- 1 si gana Min
- 5 si empatan

La implementación es como veremos a continuación:

```
private int MinMax(int[][] tablero, int turno) {
    //MinMax devuelve 1 si gana X, 10 si gana O y 5 si empata
    int valor;
    int valorHoja;
    //Verificacion si alguien gano (caso base 1)
    if(victoria1(tablero)) {
        return 1;
    }
    else if (victoria10(tablero)){
        return 10;
    }
    else {
        //Si nadie gana se evaluan las posibles jugadas
        if(turno == 10) {
            valor = 1;
        }
        else {
            valor = 10;
```

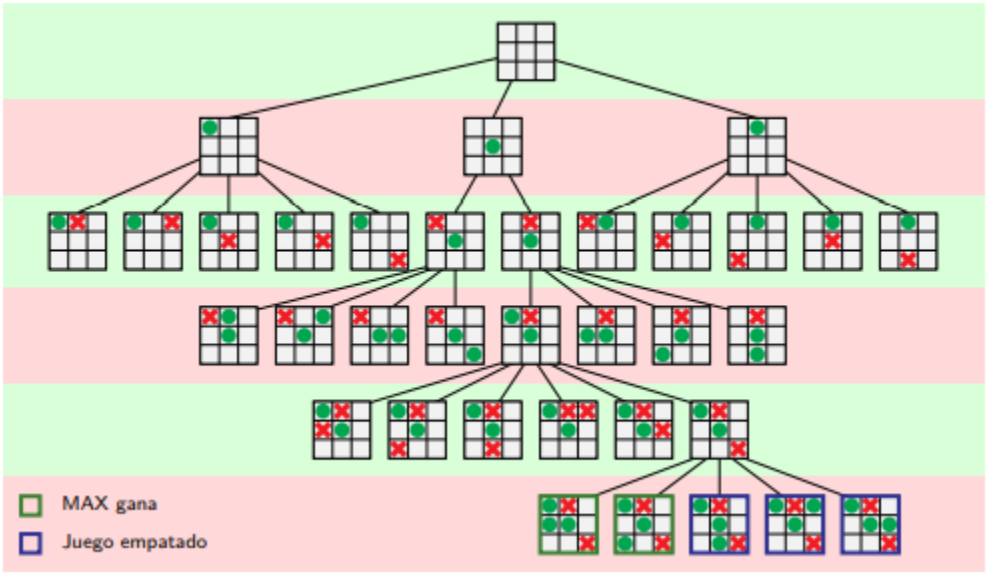
```
            */
            for (int i = 0;i<3;i++) {
                for (int j = 0;j<3;j++) {
                    if (tablero[i][j]==0) {
                        if (turno == 10) {
                            tablero[i][j]=10;
                            valorHoja = MinMax(tablero,1);
                            tablero[i][j]=0;
                            if (valor<valorHoja) {
                                valor=valorHoja;
                            }
                        }
                        else {
                            tablero[i][j]=1;
                            valorHoja = MinMax(tablero,10);
                            tablero[i][j]=0;
                            if (valor>valorHoja) {
                                valor=valorHoja;
                            }
                        }
                    }
                }
            }
        }
        //Verifica si el tablero esta lleno, si lo esta quiere decir
        //que hubo empate (caso base 2)
        int suma = 0;
        for (int i = 0;i<3;i++) {
            for (int j = 0;j<3;j++) {
                suma += tablero[i][j];
            }
        }
        if(suma==54||suma==45) {
            valor = 5;
        }
        return valor;
    }
}
```

Nuestro método MinMax recibirá como parámetros la matriz que contiene al juego y el turno correspondiente (10 si le toca a Max y 1 si le toca a Min). Como en cualquier método recursivo, empezaremos definiendo los casos base. En primer lugar, tendremos la posibilidad de que alguno de los jugadores haya ganado, en cuyo caso devolveremos el resultado correspondiente. En segundo lugar, tendremos la posibilidad de que el tablero esté completo y este haya culminado en empate, es decir que devolveremos un 5.

En nuestro caso recursivo, evaluaremos todas las posibles jugadas, alternando a cada jugador. Empezaremos asignándole a una variable “valor” un 1 si juega Max y un 10 si juega Min, es decir los peores posibles escenarios para cada respectivo caso. Luego, el jugador al que le toque probará colocar su ficha en la primera casilla disponible, a partir de la cual se evaluará (recursiva y nuevamente) todas las diferentes posibles jugadas. Así, probará también colocando la ficha en la segunda casilla disponible, en la tercera y en todas las demás, evaluando los posibles

resultados y quedándose con el mejor. Nótese que una vez simuladas las futuras jugadas, revertiremos el tablero a su versión original, únicamente agregando la jugada que se realiza.

Una simulación será como veremos a continuación:



Interfaz TatetiTDA

```
public interface TatetiTDA {
    void Inicializar();     // sin precondiciones
    void Turno(int x);          // tateti inicializado
    int Jugar(int x, int y);            // tateti inicializado
}
```

Para implementar el juego, contaremos con los siguientes métodos:

- Inicializar: inicializará el tablero a utilizar en el juego
- Turno: permitirá indicar quién jugará primero
- Jugar: recibiendo como parámetro dos valores correspondientes a la fila y columna, permitirá al jugador indicar la posición del tablero en la que desea colocar la pieza, colocará la pieza de la máquina e imprimirá el tablero.

Implementación Tateti

```
public class Tateti implements TatetiTDA {
    int[][] matriz;

    public void Inicializar() {
        matriz = new int[3][3];
    }


    public void Turno(int x) {
        if(x == 0) {
            matriz[1][1]=1;
            imprimirTablero();
        }
        else {
            imprimirTablero();
        }
    }
```

Para implementar el tablero, utilizaremos una matriz de 3 filas y 3 columnas, la cual inicializaremos en el método Inicializar.

En el método Turno, tomando 0 como turno de la máquina y 1 como turno del jugador, si este corresponde al primero colocaremos su ficha en el medio (posición que se considera por muchos la mejor para empezar) e imprimiremos el tablero. En caso contrario, simplemente imprimiremos el tablero (que estará vacío).

```
public int Jugar(int x, int y) {
    if(matriz[y][x] == 0) {
        matriz[y][x] = 10;
        int i = 0;
        int j = 0;
        int resultado = 0;
        int auxi = 0;
        int auxj = 0;
        int auxres = 0;
        boolean empateAnterior = false;
        boolean victoriaAnterior = false;
        boolean continuar = true;
```

Para implementar el método Jugar, empezaremos ingresando la ficha del jugador en la posición ingresada siempre y cuando corresponda a una casilla válida (no ocupada). Luego inicializaremos las siguientes variables:

- i: indicará la fila del tablero que estamos evaluando
- j: indicará la columna del tablero que estamos evaluando

- resultado: indicará el resultado de la posición que estamos evaluando. 10 si es una posición en la que gana Max, 1 si es una posición en la que gana Min o un 5 si es una posición en la que empatan.
- auxi y auxj: son los mismos parámetros que i y j, que utilizaremos como variables auxiliares para guardar una posición antes de buscar, si es que existen, mejores jugadas.
- auxres: es el mismo parámetro que resultado, que utilizaremos como variable auxiliar para guardar el resultado correspondiente a la jugada en la posición auxi y auxj.
- empateAnterior y victoriaAnterior: variables booleanas que nos indicarán si hubieron jugadas evaluadas anteriormente que correspondan a una victoria o a un empate-
- continuar: variable booleana que se tornará falsa únicamente cuando encuentre una ficha que terminé el juego.

```
while(i<3 & continuar) {
    while(j<3 & continuar) {
        if (matriz[i][j]==0) {
            matriz[i][j]=1;
            resultado =  MinMax(matriz,10);
            if (resultado == 10) {
                matriz[i][j]=0;
            }
            else if(resultado == 1 && victoria1(matriz)) {
                empateAnterior = false;
                victoriaAnterior = false;
                continuar = false;
            }
            else if(resultado == 1 && !victoria1(matriz) && !victoriaAnterior) {
                matriz[i][j] = 0;
                auxi = i;
                auxj = j;
                victoriaAnterior = true;
                auxres = 1;
            }
            else if(resultado == 5 && !empateAnterior && !victoriaAnterior) {
                matriz[i][j] = 0;
                auxi = i;
                auxj = j;
                empateAnterior = true;
            }
            else
                matriz[i][j] = 0;
        }
        j++;
    }
    j = 0;
    i++;
}
```

Una vez definidas las variables, empezaremos a evaluar todas las posibles jugadas. Iremos recorriendo, a través de los ciclos while, la matriz e iremos colocando la ficha de la máquina en

todas las posibles casillas libres, fijándonos en cuál le conviene. Para esto, utilizaremos el método MinMax descrito anteriormente. Como prioridad, buscará una posición en la que gane terminando el juego, en cuyo caso colocará la ficha, tornará la variable continuar en falso y saldrá del ciclo. Como segunda prioridad, buscará una posición en la que se asegure la victoria si bien no en el turno inmediato. Como tercera prioridad, buscará una posición en la que empaten. Si como resultado obtiene una victoria del oponente, simplemente seguirá buscando otras alternativas (recordemos que en este juego no hay manera de perder si ambos juegan correctamente y, como la máquina siempre juega correctamente, nunca perderá). Notemos que, si encuentra una posición en la que empata o en la que gana, pero no en el turno inmediato, se guardará la posición en las variables auxiliares a las cuales recurrirá si no encuentra una mejor opción.

```
            if(empateAnterior || victoriaAnterior) {
                matriz[auxi][auxj] = 1;
            }
            imprimirTablero();
            if(auxres == 1)
                return 1;
            if(victoria1(matriz))
                return 1;
            else if(victoria10(matriz))
                return 10;
            int suma = 0;
            for (int ii = 0;ii<3;ii++) {
                for (int jj = 0;jj<3;jj++) {
                    suma += matriz[ii][jj];
                }
            }
            if((suma==54||suma==45)) {
                return 0;
            }
            return 5;
        }
        else {
            System.out.println("Ingrese una casilla valida");
            return 5;
        }
    }
```

Una vez que salgamos del ciclo, evaluaremos si existe una victoria o un empate que hayamos guardado en las variables auxiliares, es decir que no hayamos encontrado una mejor alternativa. Si existe, colocaremos la ficha (que aún no habremos colocado) en la posición correspondiente.

Luego imprimiremos el tablero y devolveremos el resultado según corresponda a una victoria, a un empate o aún falte jugar.

Notemos, por último, que si bien el método MinMax devuelve un 5 cuando corresponde a un empate, este método utilizará el 5 para indicar que el juego aún no ha terminado y devolverá un 0 cuando se trate de un empate.

Métodos auxiliares

Como habrán notado en la implementación del juego, utilizaremos algunos métodos auxiliares como veremos a continuación:

```
private void imprimirTablero() {

    for(int i = 0; i<3; i++) {
        System.out.println("-------------");
        System.out.print("|");
        for(int j = 0; j<3; j++) {
            if(matriz[i][j] == 1)
                System.out.print(" X |");
            else if(matriz[i][j] == 10)
                System.out.print(" O |");
            else {
                System.out.print(" - |");
            }
            if(j == 2)
                System.out.println();
        }
    }
    System.out.println("-------------");
}
```

El método imprimirTablero recorrerá la matriz llenada por números y los convertirá a una X o una O según corresponda.

```
private boolean victoria1(int [][] tablero) {
    if(tablero[0][0]+tablero[0][1]+tablero[0][2]==3||
               tablero[1][0]+tablero[1][1]+tablero[1][2]==3||
               tablero[2][0]+tablero[2][1]+tablero[2][2]==3||
               tablero[0][0]+tablero[1][0]+tablero[2][0]==3||
               tablero[0][1]+tablero[1][1]+tablero[2][1]==3||
               tablero[0][2]+tablero[1][2]+tablero[2][2]==3||
               tablero[0][0]+tablero[1][1]+tablero[2][2]==3||
               tablero[0][2]+tablero[1][1]+tablero[2][0]==3)
        return true;
    return false;
}

private boolean victoria10(int [][] tablero) {
    if (tablero[0][0]+tablero[0][1]+tablero[0][2]==30||
             tablero[1][0]+tablero[1][1]+tablero[1][2]==30||
             tablero[2][0]+tablero[2][1]+tablero[2][2]==30||
             tablero[0][0]+tablero[1][0]+tablero[2][0]==30||
             tablero[0][1]+tablero[1][1]+tablero[2][1]==30||
             tablero[0][2]+tablero[1][2]+tablero[2][2]==30||
             tablero[0][0]+tablero[1][1]+tablero[2][2]==30||
             tablero[0][2]+tablero[1][1]+tablero[2][0]==30)
        return true;
    return false;
}
```

Los métodos victoria1 y victoria10 recibirán como parámetro el tablero y evaluarán si existe un ganador.

Implementación de la jugabilidad

```
package principal;
import java.util.Scanner;
import imp.Tateti;

public class Juego {

    public static void main(String[] args) {

        Tateti juego = new Tateti();
        juego.Inicializar();
        Scanner in = new Scanner(System.in);
        System.out.println("TA TE TI");
        System.out.println("Quien juega primero? (ordenador 0, Jugador 1)");
        juego.Turno(in.nextInt());
        int resultado = 5;
        int x;
        int y;
        System.out.print("Elegir posicion X: ");
        x = in.nextInt();
        while(x > 2 || x < 0) {
            System.out.print("Elija una posicion valida de X");
            x = in.nextInt();
        }
        System.out.print("Elegir posicion Y: ");
        y = in.nextInt();
        while(y > 2 || y < 0) {
            System.out.print("Elija una posicion valida de Y");
            y = in.nextInt();
        }
        resultado = juego.Jugar(x, y);
```

```
        while(resultado==5) {
            System.out.print("Elegir posicion X: ");
            x = in.nextInt();
            while(x > 2 || x < 0) {
                System.out.print("Elija una posicion valida de X");
                x = in.nextInt();
            }
            System.out.print("Elegir posicion Y: ");
            y = in.nextInt();
            while(y > 2 || y < 0) {
                System.out.print("Elija una posicion valida de Y");
                y = in.nextInt();
            }
            resultado = juego.Jugar(x, y);
        }
        if (resultado == 1) {
            System.out.println("Gana X");
        }
        else if (resultado == 10) {
            System.out.println("Gana O");
        }
        else if (resultado == 0){
            System.out.println("Empate");
        }
        in.close();;
    }
```

Respecto a la implementación de la jugabilidad del juego, se solicita al jugador que ingrese por consola quien jugará primero, se llama al método Turno y luego se le pide ingresar la posición X e Y de la ficha que quiere colocar. Con los datos de la posición de la ficha del jugador se llama al método Jugar y se almacena el resultado en una variable. Posteriormente se ingresa en un ciclo en el que se pide la posición para la siguiente ficha del jugador y se ejecuta el método Jugar. Se saldrá del ciclo cuando Jugar devuelva una victoria o un empate (resultado igual a 1, 10 o 0). Finalmente, tenemos un condicional que imprime en consola el resultado del juego.